Automatica 108 (2019) 108460

Contents lists available at ScienceDirect

Automatica

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

Brief paper

On scalable supervisory control of multi-agent discrete-event systems[☆]

Yingying Liu [a], Kai Cai [b,*], Zhiwu Li [a,c,**]

[a] School of Electro-Mechanical Engineering, Xidian University, Xi'an, 710071, China
[b] Department of Electrical & Information Engineering, Osaka City University, Osaka, 558-8585, Japan
[c] Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau

ARTICLE INFO

Article history:
Received 11 August 2017
Received in revised form 6 June 2019
Accepted 14 June 2019
Available online 15 July 2019

ABSTRACT

In this paper we study multi-agent discrete-event systems where the agents can be divided into several groups, and within each group the agents have similar or identical state transition structures. We employ a *relabeling* map to generate a "template structure" for each group, and synthesize a *scalable* supervisor whose state size and computational process are independent of the number of agents. This scalability allows the supervisor to remain invariant (no recomputation or reconfiguration needed) if and when there are agents removed due to failure or added for increasing productivity. The constant computational effort for synthesizing the scalable supervisor also makes our method promising for handling large-scale multi-agent systems. Moreover, based on the scalable supervisor we design scalable local controllers, one for each component agent, to establish a purely distributed control architecture.

1. Introduction

Multi-agent systems have found increasing applications in large-scale engineering practice where tasks are difficult to be accomplished by a single entity. Examples include multiple machines in factories, robots in manufacturing cells, and AGVs in logistic systems (ElMaraghy, 2005; Wu & Zhou, 2007). Although not always the case, multi-agent systems typically can be divided into several groups, according to different roles, functions, or capabilities. For instance, machines are grouped to process different types of workpieces, robots to manufacture different parts of a product, AGVs to transport items of distinct sizes, shapes and weights. Agents in the same group often have similar or even identical state transition structures, i.e. dynamics. This we shall refer to as a *modular* characteristic.

In this paper we study multi-agent systems with such a modular characteristic, and consider individual agents modeled by discrete-event systems (DES). Given a control specification, one may in principle apply supervisory control theory (Wonham & Cai, 2019; Wonham & Ramadge, 1987) to synthesize a monolithic (i.e. centralized) supervisor for the entire multi-agent system. While the supervisor computed by this method is optimal (i.e. maximally permissive) and nonblocking, there are two main problems. First, the state size of the supervisor increases (exponentially) as the number of agents increases (Gohari & Wonham, 2000); consequently, the supervisor synthesis will become computationally infeasible for large numbers of agents. Second, whenever the number of agents changes (increases when more agents are added into the system to enhance productivity or to improve redundancy for the sake of reliability; or decreases when some agents malfunction and are removed from the system), the supervisor must be recomputed or reconfigured (e.g. (Nooruldeen & Schmidt, 2014)) in order to adapt to the change.

In this paper we solve both problems mentioned above by exploiting the modular characteristic of multi-agent systems, and thereby designing a *scalable* supervisor whose state number and computational process are *independent* of the number of agents. First, owing to similar/identical transition structures of agents in the same group, we employ a *relabeling map* to generate a "template structure" for each group. The template structures thus generated are independent of the agent numbers. Then we design a supervisor based on these template structures, and

[☆] This work was partially supported by the National Key Research and Development Program of China under Grant 2018YFB1700104; the JSPS KAKENHI, Japan under Grant no. JP16K18122; Doctoral Students' Short-Term Study Abroad Scholarship Fund of Xidian University. The material in this paper was partially presented at The 14th International Workshop on Discrete Event Systems (WODES 2018), May 30–June 1, 2018, Sorrento Coast, Italy. This paper was recommended for publication in revised form by Associate Editor Christoforos Hadjicostis under the direction of Editor Christos G. Cassandras.
[*] Corresponding author.
[**] Corresponding author at: School of Electro-Mechanical Engineering, Xidian University, Xi'an, 710071, China.
E-mail addresses: liu@c.info.eng.osaka-cu.ac.jp (Y. Liu), kai.cai@eng.osaka-cu.ac.jp (K. Cai), zhwli@xidian.edu.cn (Z. Li).

https://doi.org/10.1016/j.automatica.2019.06.012
0005-1098/© 2019 Elsevier Ltd. All rights reserved.

prove that it is a scalable supervisor for the multi-agent system under an easily-checkable condition. The controlled behavior of the designed scalable supervisor needs not be optimal, but is nonblocking.

While the designed scalable supervisor serves as a *centralized* controller for the multi-agent system, it may sometimes be natural, and even more desirable, to equip each individual agent with its own *local* controller. Hence we move on to design *scalable* local controllers whose state numbers and computational process are invariant with respect to the number of agents; for this design, we employ the method of supervisor localization (Cai & Wonham, 2010, 2016). Directly localizing the scalable supervisor may be computationally expensive, inasmuch as the localization method requires computing the overall plant model. To circumvent this problem, we localize the supervisor based on the template structures and thereby derive scalable local controllers without constructing the underlying plant model. It is proved that the collective controlled behavior of these local controllers is equivalent to that achieved by the scalable supervisor.

The contributions of our work are threefold. First, our designed centralized supervisor has scalability with respect to the number of agents in the system. This scalability is a desired feature of a supervisor for multi-agent systems, inasmuch as it allows the supervisor to remain invariant regardless of how many agents are added to or removed from the system (which may occur frequently due to productivity/reliability concerns or malfunction/repair). Second, the local controllers we design for individual agents have the same scalability feature, and are guaranteed to collectively achieve identical controlled behavior as the centralized supervisor does. With the local controllers 'built-in', the agents become autonomous and make their own local decisions; this is particularly useful in applications like multi-robot systems. Finally, the computation of the scalable supervisor and local controllers is based solely on template structures and is thus independent of agent numbers as well. As a result, the computation load remains the same even if the number of agents increases; this is advantageous as compared to the existing supervisory synthesis methods.

The work most related to ours is reported in (Jiao et al., 2017, 2018). Therein the same type of multi-agent systems is investigated and relabeling maps are used to generate template structures. Various properties of the relabeling map are presented which characterize relations between the relabeled system and the original one. Moreover, a supervisor is designed that is provably independent of agent numbers, when these numbers exceed a certain threshold value. The design of the supervisor is, however, based on first computing the synchronous product of all agents, which can be computationally expensive. This can be relieved by using *state tree structures* (Jiao et al., 2017), but the computation is still dependent on the agent numbers and thus the supervisor has to be recomputed or reconfigured whenever the number of agents changes. By contrast, our synthesis is based only on the template structures and thus independent of the agent numbers; furthermore the state size of our designed supervisor is *always* independent of the number of agents, with no threshold value required.

2. Preliminaries and problem formulation

2.1. Preliminaries

Let the DES plant to be controlled be modeled by a *generator* $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$, where $\Sigma = \Sigma_c \dot{\cup} \Sigma_u$ is a finite event set that is partitioned into a controllable event subset and an uncontrollable subset, Q is the finite state set, $q_0 \in Q$ the initial state, $Q_m \subseteq Q$ the set of marker states, and $\delta : Q \times \Sigma \rightarrow Q$ the (partial) transition function. Extend δ in the usual way such that $\delta : Q \times \Sigma^* \rightarrow Q$. The *closed behavior* of $\mathbf{G}$ is the language $L(\mathbf{G}) := \{s \in \Sigma^* \mid \delta(q_0, s)!\}$, where the notation $\delta(q_0, s)!$ means that $\delta(q_0, s)$ is defined. The *marked behavior* of $\mathbf{G}$ is $L_m(\mathbf{G}) := \{s \in L(\mathbf{G}) \mid \delta(q_0, s) \in Q_m\} \subseteq L(\mathbf{G})$. A string s_1 is a *prefix* of another string s, written $s_1 \leq s$, if there exists s_2 such that $s_1 s_2 = s$. The *prefix closure* of $L_m(\mathbf{G})$ is $\overline{L_m(\mathbf{G})} := \{s_1 \in \Sigma^* \mid (\exists s \in L_m(\mathbf{G})) s_1 \leq s\}$. We say that $\mathbf{G}$ is *nonblocking* if $\overline{L_m(\mathbf{G})} = L(\mathbf{G})$. A language $K \subseteq L_m(\mathbf{G})$ is *controllable* with respect to $L(\mathbf{G})$ provided $\overline{K}\Sigma_u \cap L(\mathbf{G}) \subseteq \overline{K}$. Let $E \subseteq L_m(\mathbf{G})$ be a specification language for $\mathbf{G}$, and define the set of all sublanguages of E that are controllable with respect to $L(\mathbf{G})$ by $\mathcal{C}(E) := \{K \subseteq E \mid \overline{K}\Sigma_u \cap L(\mathbf{G}) \subseteq \overline{K}\}$. Then $\mathcal{C}(E)$ has a unique supremal element $\sup\mathcal{C}(E) = \bigcup_{K \in \mathcal{C}(E)} K$ (Wonham & Cai, 2019).

For describing a modular structure of plant $\mathbf{G}$, we first introduce a relabeling map. Let T be a set of new events, i.e. $\Sigma \cap T = \emptyset$. Define a *relabeling* map $R : \Sigma \rightarrow T$ such that for every $\sigma \in \Sigma$, $R(\sigma) = \tau$ for some $\tau \in T$ (Jiao et al., 2018). In general R is surjective but need not be injective. For $\sigma \in \Sigma$, let $[\sigma]$ be the set of events in Σ that have the same R-image as σ, i.e. $[\sigma] := \{\sigma' \in \Sigma | R(\sigma') = R(\sigma)\}$. Then $\Sigma = [\sigma_1] \dot{\cup} [\sigma_2] \dot{\cup} \cdots \dot{\cup} [\sigma_k]$, for some $k \geq 1$, and T can be written as $T = \{R(\sigma_1), R(\sigma_2), \ldots, R(\sigma_k)\}$. We require that R preserves controllable/uncontrollable status of events in Σ; namely $R(\sigma)$ is a controllable event if and only if $\sigma \in \Sigma_c$. Thus $T_c := \{R(\sigma) | \sigma \in \Sigma_c\}$, $T_u := \{R(\sigma) | \sigma \in \Sigma_u\}$, and $T = T_c \dot{\cup} T_u$.

We extend R such that $R : \Sigma^* \rightarrow T^*$ according to

(i) $R(\varepsilon) = \varepsilon$, where ε denotes the empty string;

(ii) $R(s\sigma) = R(s)R(\sigma)$, $\sigma \in \Sigma$ and $s \in \Sigma^*$.

Note that $R(s) \neq \varepsilon$ for all $s \in \Sigma^* \setminus \{\varepsilon\}$. Further extend R for languages, i.e. $R : Pwr(\Sigma^*) \rightarrow Pwr(T^*)$, and define $R(L) = \{R(s) \in T^* | s \in L\}$, $L \subseteq \Sigma^*$. The *inverse-image function* R^{-1} of R is given by $R^{-1} : Pwr(T^*) \rightarrow Pwr(\Sigma^*)$: $R^{-1}(H) = \{s \in \Sigma^* | R(s) \in H\}$, $H \subseteq T^*$ (Jiao et al., 2018). Note that $RR^{-1}(H) = H$, $H \subseteq T^*$; while $R^{-1}R(L) \supseteq L$, $L \subseteq \Sigma^*$. We say that $L \subseteq \Sigma^*$ is $(\mathbf{G}, R)$-*normal* if $R^{-1}R(L) \cap L_m(\mathbf{G}) \subseteq L$. Several properties of R and R^{-1} are presented in the following lemma.

Lemma 1. *For $R : Pwr(\Sigma^*) \rightarrow Pwr(T^*)$ and $R^{-1} : Pwr(T^*) \rightarrow Pwr(\Sigma^*)$, the following statements are true.*

(i) $R(\overline{L}) = \overline{R(L)}$, $L \subseteq \Sigma^*$.

(ii) $R^{-1}(\overline{H}) = \overline{R^{-1}(H)}$, $H \subseteq T^*$.

The proof of Lemma 1 and all subsequent proofs (except for Theorem 1) are referred to (Liu et al., 2018a).

We now discuss computation of R, R^{-1} by generators. Let $R : \Sigma^* \rightarrow T^*$ be a relabeling map and $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$ a generator. First, relabel each transition of $\mathbf{G}$ to obtain $\mathbf{G}_T = (Q, T, \delta_T, q_0, Q_m)$, where $\delta_T : Q \times T \rightarrow Q$ is defined by

$$\delta_T(q_1, \tau) = q_2 \text{ iff } (\exists \sigma \in \Sigma) R(\sigma) = \tau \ \&\ \delta(q_1, \sigma) = q_2.$$

Hence $L_m(\mathbf{G}_T) = R(L_m(\mathbf{G}))$ and $L(\mathbf{G}_T) = R(L(\mathbf{G}))$. However, $\mathbf{G}_T$ as given above may be *nondeterministic*. Thus apply *subset construction* (Wonham & Cai, 2019) to convert $\mathbf{G}_T$ into a deterministic generator $\mathbf{H} = (Z, T, \zeta, z_0, Z_m)$, with $L_m(\mathbf{H}) = L_m(\mathbf{G}_T)$ and $L(\mathbf{H}) = L(\mathbf{G}_T)$.[1]

Lemma 2. *If $\mathbf{G}$ is nonblocking, then the relabeled generator $\mathbf{H}$ is also nonblocking.*

[1] Subset construction has exponential complexity in the worst case. Nevertheless, in the scalable supervisor synthesis presented in Section 3, there is no need to compute the overall plant $\mathbf{G}$, nor to relabel $\mathbf{G}$; indeed the generators that need to be relabeled in this paper have reasonably small state size, and hence their relabeled models may be feasibly computed.

Conversely, to inversely relabel **H**, simply replace each transition τ ($\in T$) of **H** by those σ ($\in \Sigma$) with $R(\sigma) = \tau$; thus one obtains $\mathbf{G}' = (Z, \Sigma, \zeta', z_0, Z_m)$, where $\zeta' : Z \times \Sigma \to Z$ is defined by

$$\zeta'(z_1, \sigma) = z_2 \text{ iff } (\exists \tau \in T) R(\sigma) = \tau \ \& \ \zeta(z_1, \tau) = z_2.$$

It is easily verified that $L_m(\mathbf{G}') = R^{-1} L_m(\mathbf{H})$ and $L(\mathbf{G}') = R^{-1} L(\mathbf{H})$. Note that $\mathbf{G}'$ as given above is deterministic (since **H** is), and has the same number of states as **H**; namely inverse-relabeling does not change state numbers. Note that $L_m(\mathbf{G}') \supseteq L_m(\mathbf{G})$ and $L(\mathbf{G}') \supseteq L(\mathbf{G})$. Henceforth we shall write $R(\mathbf{G}) := \mathbf{H}$ and $R^{-1}(\mathbf{H}) := \mathbf{G}'$.

2.2. Problem formulation

Let $R : \Sigma^* \to T^*$ be a relabeling map, and $\mathcal{G} = \{\mathbf{G}_1, \ldots, \mathbf{G}_k\}$ be a set of generators. We say that $\mathcal{G}$ is a *similar set* under R if there is a generator **H** such that

$$(\forall i \in \{1, \ldots, k\}) R(\mathbf{G}_i) = \mathbf{H}. \tag{1}$$

One may view **H** as a "template" for $\mathcal{G}$ in that each generator $\mathbf{G}_i$ in the set may be relabeled to **H**.

In this paper, the plant **G** is divided into l (≥ 1) groups of component agents, each group $\mathcal{G}_i$ ($i \in \{1, \ldots, l\}$) being a similar set of generators under a given relabeling map R, i.e. $\mathcal{G}_i = \{\mathbf{G}_{i1}, \ldots, \mathbf{G}_{i\,n_i}\}$ ($n_i \geq 1$) and there is a generator $\mathbf{H}_i$ such that

$$(\forall j \in \{1, \ldots, n_i\}) R(\mathbf{G}_{ij}) = \mathbf{H}_i. \tag{2}$$

Let $\mathbf{G}_{ij}$ be defined on Σ_{ij} and $\mathbf{H}_i$ on T_i. Then $R(\Sigma_{ij}) = T_i$ for all $j \in \{1, \ldots, n_i\}$.

(A1) All component agents are nonblocking and independent, i.e. their event sets are pairwise disjoint.[2]

(A2) The template generators $\mathbf{H}_i$ ($i \in \{1, \ldots, l\}$) have pairwise-disjoint event sets. (This assumption can be regarded as being imposed on the relabeling map R, since the event set T_i of $\mathbf{H}_i$ is obtained by relabeling those Σ_{ij} of $\mathbf{G}_{ij}$, $j \in \{1, \ldots, n_i\}$.)

As described above, the plant **G** represents a multi-agent DES with a *modular* structure, i.e. containing multiple groups of similar and independent agents. Although it would be more general to consider event sharing among agents, this modular structure is not uncommon in practical multi-agent systems (e.g. machines in factories, robots in warehouses, and vehicles at intersections). One example of this type of modular plant is given in Fig. 1.

Let Σ ($= \Sigma_c \dot\cup \Sigma_u$) be the event set of plant **G**, and $E \subseteq \Sigma^*$ a specification language that imposes behavioral constraints on **G** (thus the specification with respect to the plant is $E \cap L_m(\mathbf{G})$). We make the following assumption.

(A3) The specification language E can be represented by a (nonblocking) generator **E** (i.e. $L_m(\mathbf{E}) = E$) that satisfies $R^{-1}(R(\mathbf{E})) = \mathbf{E}$.

This assumption means that the specification is invariant under the composition $R^{-1} \circ R$ (Jiao et al., 2018). In particular, Assumption (A3) implies that E is $(\mathbf{G}, R)$-normal, i.e. $R^{-1}R(E) \cap L_m(\mathbf{G}) \subseteq E$, which is essential in the proof of our main result below. To check if (A3) holds, first compute $R^{-1}(R(\mathbf{E}))$ as described in Section 2.1, and then verify if the result is DES *isomorphic* (e.g. Cai & Wonham, 2016) to **E**. Since R^{-1} does not change state number, a consequence of (A3) is that R also does not change state number nor causes nondeterminism.

Now with plant **G** and specification E, the standard supervisory control design in Wonham & Cai, 2019 proceeds as follows. First compute the plant **G** by *synchronous product* (Wonham & Cai, 2019) of all component agents:

$$\mathbf{G} = \|_{i \in \{1,\ldots,l\}} \mathbf{G}_i, \quad \text{where } \mathbf{G}_i = \|_{j \in \{1,\ldots,n_i\}} \mathbf{G}_{ij}.$$

[2] Under (A1), $\mathbf{H}_i$ ($i \in \{1, \ldots, l\}$) computed from (2) are nonblocking by Lemma 2.

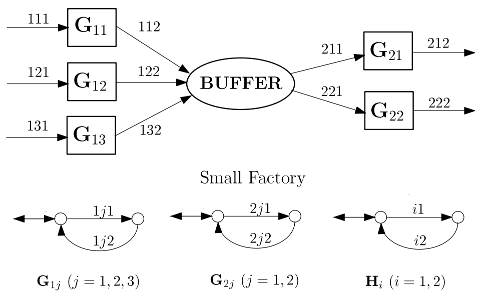


Fig. 1. Consider a small factory consisting of 3 input machines $\mathbf{G_{11}}, \mathbf{G_{12}}, \mathbf{G_{13}}$ and 2 output machines $\mathbf{G_{21}}, \mathbf{G_{22}}$, linked by a buffer in the middle. Events $1j1$ ($j \in \{1, 2, 3\}$) and $2j1$ ($j \in \{1, 2\}$) mean that machine $\mathbf{G}_{ij}$ starts to work by taking in a workpiece; events $1j2$ and $2j2$ mean that $\mathbf{G}_{ij}$ finishes work and outputs a workpiece. Let $\Sigma = \Sigma_c \dot\cup \Sigma_u = \{111, 121, 131, 211, 221\} \dot\cup \{112, 122, 132, 212, 222\}$, $T = \{i1, i2 \mid i \in \{1, 2\}\}$, and a relabeling map $R : \Sigma^* \to T^*$ with $R(ij1) = i1 \in T_c$, $R(ij2) = i2 \in T_u$ for all $i \in \{1, 2\}$. Hence, under R, the plant is divided into 2 similar groups $\{\mathbf{G}_{11}, \mathbf{G}_{12}, \mathbf{G}_{13}\}$ and $\{\mathbf{G}_{21}, \mathbf{G}_{22}\}$, with template generators $\mathbf{H}_1$ and $\mathbf{H}_2$ respectively. It is evident that Assumptions (A1) and (A2) hold. Convention: the initial state of a generator is labeled by a circle with an entering arrow, while a marker state is labeled by a circle with an exiting arrow. The same notation will be used in subsequent figures.

Under Assumption (A1), **G** is nonblocking. Then synthesize a supervisor **SUP** (a nonblocking generator) with

$$L_m(\mathbf{SUP}) = \sup \mathcal{C}(E \cap L_m(\mathbf{G})).$$

To rule out the trivial case, we assume the following.

(A4) $L_m(\mathbf{SUP}) \neq \emptyset$ for $n_i = 1$, $i \in \{1, \ldots, l\}$. Denote this special **SUP** by **SUP1** henceforth, which is the supervisor when plant **G** contains exactly one agent in each group.

By this synthesis method, the number of states of **SUP** increases (exponentially) as the number of agents (n_i, $i \in \{1, \ldots, l\}$) increases, and consequently the supervisor synthesis becomes computationally difficult (if not impossible). In addition, whenever the number n_i of agents changes (e.g. an operating agent malfunctions and is removed from the system, or a new agent is added to increase productivity), the supervisor **SUP** has to be recomputed or reconfigured. These two problems may be resolved if one can synthesize a supervisor whose state size, as well as the computational effort involved in its synthesis, is *independent* of the number n_i of agents, by exploiting the modular structure of the plant **G**. We will call such a supervisor *scalable*, where scalability is with respect to the number of agents in the plant.

With this motivation, we formulate the following Scalable Supervisory Control Synthesis Problem (SSCSP):

Design a scalable supervisor **SSUP** *(a nonblocking generator) such that*

(i) The number of states of **SSUP** *and its computation are independent of the number n_i of agents for all $i \in \{1, \ldots, l\}$;*

(ii) $L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G})$ satisfies $\emptyset \neq L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G}) \subseteq L_m(\mathbf{SUP})$.

Condition (ii) requires that $L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G})$ be nonempty and controllable with respect to $L(\mathbf{G})$. It would be ideal to have $L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G}) = L_m(\mathbf{SUP})$. Inasmuch as this requirement might be too strong to admit any solution to the problem, we shall consider (ii) above.

3. Scalable supervisory control

In this section we design a scalable supervisor to solve the Scalable Supervisory Control Synthesis Problem (SSCSP), under an easily-verifiable condition. Consider the plant **G** as described in Section 2.2. Let Σ ($= \Sigma_c \dot\cup \Sigma_u$) be the event set of **G**, and $R : \Sigma \to T$ a relabeling map. The procedure of designing a scalable supervisor is as follows, (P1)-(P4), which involves first synthesizing a supervisor for "relabeled system" under R and then inverse-relabeling the supervisor.

(P1) Let $k_i \in \{1, \dots, n_i\}$ denote the number of agents in group i allowed to work in parallel, and compute $\mathbf{M}_i := R(\|_{j=1,\dots,k_i}\mathbf{G}_{ij})$. Then compute the relabeled plant $\mathbf{M}$ as the synchronous product of the generators $\mathbf{M}_i$, i.e.

$$\mathbf{M} :=\|_{i\in\{1,\dots,l\}} \mathbf{M}_i. \tag{3}$$

We call $\mathbf{M}$ the *relabeled plant* under R; it is nonblocking if Assumptions (A1), (A2) hold. The event set of $\mathbf{M}$ is $T = T_c\dot{\cup}T_u$, where $T_c = R(\Sigma_c)$ and $T_u = R(\Sigma_u)$. For computational efficiency, one would choose k_i to be (much) smaller than n_i (the number of agents in group i). When all $k_i = 1$, we have the special case addressed in (Liu et al., 2018b). Note that once k_i are fixed, the state sizes of $\mathbf{M}_i$ and $\mathbf{M}$ are fixed as well, and thus independent of the number n_i of agents in group i.

(P2) Compute $F := R(E)$, where $E \subseteq \Sigma^*$ is the specification imposed on $\mathbf{G}$. We call $F \subseteq T^*$ the *relabeled specification* imposed on $\mathbf{M}$. If Assumption (A3) holds, then R does not cause nondeterminism and no subset construction is needed in computing a generator to represent F.

(P3) Synthesize a *relabeled supervisor* **RSUP** (a nonblocking generator) such that

$$L_m(\mathbf{RSUP}) = \sup\mathcal{C}(L_m(\mathbf{M}) \cap F) \subseteq T^*.$$

The number of states of **RSUP** is independent of the number of agents, since **M**'s state size is so.

(P4) Inverse-relabel **RSUP** to derive **SSUP**, i.e.

$$\mathbf{SSUP} := R^{-1}(\mathbf{RSUP}) \tag{4}$$

with the marked behavior

$$L_m(\mathbf{SSUP}) = R^{-1}(L_m(\mathbf{RSUP})) \subseteq \Sigma^*.$$

By the inverse-relabeling computation introduced in Section 2.1, **SSUP** computed in (4) has the same number of states as **RSUP**. It then follows that the state size of **SSUP** is independent of the number of agents in plant **G** (although dependent on k_i fixed in (P1)). Moreover, it is easily observed that **SSUP** is nonblocking (since **RSUP** is), and its computation does not depend on the number n_i of agents in each group i $(\in \{1, \dots, l\})$.

Our main result is the following.

Theorem 1. *Consider the plant* **G** *as described in Section 2.2 and suppose that Assumptions (A1), (A2), (A3), and (A4) hold. If $L_m(\mathbf{M})$ is controllable with respect to $R(L(\mathbf{G}))$, then* **SSUP** *in (4) is a scalable supervisor that solves SSCSP.*

Theorem 1 provides a sufficient condition under which **SSUP** in (4) is a solution to SSCSP. This condition is the controllability of $L_m(\mathbf{M})$ with respect to $R(L(\mathbf{G}))$, i.e. $\overline{L_m(\mathbf{M})}\Sigma_u \cap R(L(\mathbf{G})) \subseteq \overline{L_m(\mathbf{M})}$. This means that the relabeled plant should be controllable with respect to the relabeling of the original plant **G**; in other words, the relabeling operation should not remove uncontrollable events that are allowed by **G**. As we shall see below, this condition is essential in proving the controllability of $L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G})$ with respect to $L(\mathbf{G})$.

For the success of our scalable supervisory control synthesis, it is important to be able to efficiently verify this sufficient condition. At the appearance, however, this condition seems to require computing **G** which would be computationally infeasible for large systems. Nevertheless, we have the following result.

Proposition 1. *Consider the plant* **G** *as described in Section 2.2 and suppose that Assumptions (A1), (A2) hold. For each group $i \in \{1, \dots, l\}$ if $L_m(\mathbf{H}_i)$ is controllable with respect to $R(L(\mathbf{G}_{i1} \parallel \mathbf{G}_{i2}))$, then $L_m(\mathbf{M})$ is controllable with respect to $R(L(\mathbf{G}))$.*

Proposition 1 asserts that the controllability of $L_m(\mathbf{M})$ with respect to $R(L(\mathbf{G}))$ may be checked in a modular fashion: namely it is sufficient to check the controllability of $L_m(\mathbf{H}_i)$ ($\mathbf{H}_i$ in (2)) for each group with respect to only two component agents. As a result, the computational effort of checking the condition is low. Note that the condition in Proposition 1, $L_m(\mathbf{H}_i)$ being controllable with respect to $R(L(\mathbf{G}_{i1} \parallel \mathbf{G}_{i2}))$, does not always hold. An example where this condition fails is given in (Liu et al., 2018a).

Thus under the easily checkable-sufficient condition, Theorem 1 asserts that **SSUP** in (4) is a valid scalable supervisor whose state size is independent of the number of agents in the plant.

To prove Theorem 1 we need to the following lemmas.

Lemma 3. *Consider the plant* **G** *as described in Section 2.2 and suppose that Assumptions (A1), (A2) hold. Then* **M** *is nonblocking, and $L_m(\mathbf{M}) \subseteq R(L_m(\mathbf{G}))$.*

Lemma 4. *Consider the plant* **G** *as described in Section 2.2 and suppose that Assumptions (A1), (A2) hold. Then* **SSUP** *and* **G** *are nonconflicting, i.e.*

$$\overline{L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G})} = \overline{L_m(\mathbf{SSUP})} \cap \overline{L_m(\mathbf{G})}.$$

Now we are ready to provide the proof of Theorem 1.

Proof of Theorem 1. That the number of states of **SSUP** and its computation are independent of the number n_i of agents for all $i \in \{1, \dots, l\}$ has been asserted following (P4) of designing **SSUP**. Hence to prove that **SSUP** is a scalable supervisor that solves SSCSP, we will show that $\emptyset \neq L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G}) \subseteq L_m(\mathbf{SUP})$.

The fact that $L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G})$ is nonempty follows from Assumption (A4). To see this, suppose on the contrary that $L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G}) = \emptyset$, which implies that no behavior of the agents is legal; then the same implication also holds for the special case where there is only one agent in each group, namely $L_m(\mathbf{SUP1}) = \emptyset$. But this is in contradiction with Assumption (A4).

It remains to show that $L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G}) \subseteq L_m(\mathbf{SUP}) = \sup\mathcal{C}(E \cap L_m(\mathbf{G}))$. For this we will prove that (i) $L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G})$ is controllable with respect to $L(\mathbf{G})$, and (ii) $L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G}) \subseteq E \cap L_m(\mathbf{G})$. For (i) let $s \in \overline{L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G})}$, $\sigma \in \Sigma_u$, $s\sigma \in L(\mathbf{G})$. Then

$$\begin{aligned}
& s \in \overline{L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G})} \\
\Rightarrow & (\exists t) st \in L_m(\mathbf{SSUP}) \\
\Rightarrow & st \in R^{-1}(L_m(\mathbf{RSUP})) \quad \text{(by (P4))} \\
\Rightarrow & R(st) \in L_m(\mathbf{RSUP}) \subseteq L_m(\mathbf{M}) \\
\Rightarrow & R(s) \in \overline{L_m(\mathbf{RSUP})} \;\&\; R(s) \in \overline{L_m(\mathbf{M})}.
\end{aligned}$$

Since $s\sigma \in L(\mathbf{G})$, we have $R(s)R(\sigma) \in R(L(\mathbf{G}))$ where $R(\sigma) \in T_u$ (since $\sigma \in \Sigma_u$). It then follows from the controllability of $L_m(\mathbf{M})$ with respect to $R(L(\mathbf{G}))$ that $R(s)R(\sigma) \in \overline{L_m(\mathbf{M})} = L(\mathbf{M})$ (**M** is nonblocking by Lemma 3). Now use the controllability of $L_m(\mathbf{RSUP})$ with respect to $L(\mathbf{M})$ to derive $R(s)R(\sigma) \in \overline{L_m(\mathbf{RSUP})}$, and in turn

$$\begin{aligned}
& s\sigma \in R^{-1}R(s\sigma) \subseteq R^{-1}\overline{L_m(\mathbf{RSUP})} \\
\Rightarrow & s\sigma \in \overline{R^{-1}(L_m(\mathbf{RSUP}))} = \overline{L_m(\mathbf{SSUP})}.
\end{aligned}$$

In the derivation above, we have used Lemma 1(ii). In addition, since $s\sigma \in L(\mathbf{G}) = \overline{L_m(\mathbf{G})}$ (**G** is nonblocking by Assumption (A1)), we have $s\sigma \in \overline{L_m(\mathbf{SSUP})} \cap \overline{L_m(\mathbf{G})}$. Under Assumptions (A1), (A2), it follows from Lemma 4 that **SSUP** and **G** are nonconflicting, i.e. $\overline{L_m(\mathbf{SSUP})} \cap \overline{L_m(\mathbf{G})} = \overline{L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G})}$. Hence $s\sigma \in \overline{L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G})}$, which proves (i).

For (ii) let $s \in L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G})$. Then

$$\begin{aligned} & s \in R^{-1}L_m(\mathbf{RSUP}) \cap L_m(\mathbf{G}) \\ \Rightarrow & s \in L_m(\mathbf{G}) \ \& \ R(s) \in L_m(\mathbf{RSUP}) \subseteq F = R(E) \\ \Rightarrow & s \in L_m(\mathbf{G}) \ \& \ s \in R^{-1}R(s) \subseteq R^{-1}R(E). \end{aligned}$$

Since E is $(\mathbf{G}, R)$-normal (by Assumption (A3)), i.e. $R^{-1}R(E) \cap L_m(\mathbf{G}) \subseteq E$, we derive $s \in E \cap L_m(\mathbf{G})$, which proves (ii). The proof is now complete. □

From the proof above, note that if $R(L_m(\mathbf{SUP})) \subseteq L_m(\mathbf{RSUP})$, then we derive $L_m(\mathbf{SUP}) \subseteq R^{-1}R(L_m(\mathbf{SUP})) \cap L_m(\mathbf{G}) \subseteq R^{-1}(L_m(\mathbf{RSUP})) \cap L_m(\mathbf{G}) = L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G})$. This leads to the following.

Corollary 1. *Consider the plant* $\mathbf{G}$ *as described in Section 2.2 and suppose that Assumptions (A1), (A2), (A3), (A4) hold. If* $L_m(\mathbf{M})$ *is controllable with respect to* $R(L(\mathbf{G}))$*, and* $R(L_m(\mathbf{SUP})) \subseteq L_m(\mathbf{RSUP})$*, then* **SSUP** *in (3) is the least restrictive scalable supervisor that solves SSCSP (i.e.* $L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G}) = L_m(\mathbf{SUP})$*).*

Although the least restrictive scalable solution in Corollary 1 is of theoretical interest, the additional condition $R(L_m(\mathbf{SUP})) \subseteq L_m(\mathbf{RSUP})$ may be too strong and its verification requires computing the monolithic supervisor which is infeasible for large multi-agent systems.

Alternatively, one may explore the threshold thd_i of k_i (the number of agents in group i that are allowed to work in parallel) such that if $k_i \geq thd_i$, then the scalable supervisor **SSUP** achieves the least restrictive controlled behavior (i.e. $L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G}) = L_m(\mathbf{SUP})$). For the small factory example in Fig. 1, the threshold for each of k_1 and k_2 is 2, the buffer size. More generally, for a small factory consisting of n_1 input machines, n_2 output machines, and a buffer of size b ($\leq n_1, n_2$), the threshold for each of k_1 and k_2 is b. A thorough study on the threshold of k_i that achieves the least restrictive controlled behavior will be pursued in our future work.

Remark 1. In Theorem 1, the condition that $L_m(\mathbf{M})$ is controllable with respect to $R(L(\mathbf{G}))$ rules out the case where agent models start with an uncontrollable event. To address this case, one approach is to replace the relabeled plant $\mathbf{M}$ in (P1) by $\mathbf{M} := R(\mathbf{G})$; the rest (P2)-(P4) remain the same. Suppose that the specification $E \subseteq L_m(\mathbf{G})$ is controllable with respect to $L(\mathbf{G})$. Then it is verified that $R(E)$ is controllable with respect to $R(L(\mathbf{G})) = L(\mathbf{M})$ (under Assumptions (A1), (A2)). Hence the resulting $L_m(\mathbf{SSUP}) = R^{-1}R(E)$. Therefore, since E is $(\mathbf{G}, R)$-normal (by Assumption (A3)), we derive $L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G}) = R^{-1}R(E) \cap L_m(\mathbf{G}) = E = L_m(\mathbf{SUP})$. The above reasoning leads to the following.

Corollary 2. *Consider the plant* $\mathbf{G}$ *as described in Section 2.2 and suppose that Assumptions (A1), (A2), (A3), (A4) hold. Also suppose that the relabeled plant* $\mathbf{M}$ *in (P1) is* $\mathbf{M} := R(\mathbf{G})$*. If the specification* $E \subseteq L_m(\mathbf{G})$ *is controllable with respect to* $L(\mathbf{G})$*, then* **SSUP** *in (3) solves SSCSP (with* $L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G}) = L_m(\mathbf{SUP})$*).*

Although Corollary 2 allows agents to start with an uncontrollable event, the assumption that $\mathbf{M} = R(\mathbf{G})$ requires computing the plant model $\mathbf{G}$ which is infeasible for large multi-agent systems. A special case where E is controllable with respect to $L(\mathbf{G})$ is when the specification $E = L_m(\mathbf{G})$. For the more general case where E is not controllable with respect to $L(\mathbf{G})$, we shall postpone the investigation to our future work.

4. Scalable distributed control

So far we have synthesized a scalable supervisor **SSUP** that effectively controls the entire multi-agent system, i.e. **SSUP** is a *centralized* controller. For the type of system considered in this paper which consists of many independent agents, however, it is also natural to design a *distributed* control architecture where each individual agent acquires its own local controller (thereby becoming autonomous).[3]

Generally speaking, a distributed control architecture is advantageous in reducing (global) communication load, since local controllers typically need to interact only with their (nearest) neighbors. A distributed architecture might also be more fault-tolerant, as partial failure of local controllers or the corresponding agents would unlikely to overhaul the whole system.

For these potential benefits, we aim in this section to design for the multi-agent system a distributed control architecture. In particular, we aim to design local controllers that have the same *scalability* as the centralized **SSUP**; namely their state sizes and computation are independent of the number of agents in the system. Thus when some agents break down and/or new agents are added in, there is no need of recomputing or reconfiguring these local controllers.

Let us now formulate the following Scalable Distributed Control Synthesis Problem (SDCSP):

Design a set of scalable local controllers $\mathbf{SLOC}_{ij}$ *(nonblocking generators), one for each agent* $\mathbf{G}_{ij}$ $(i \in \{1, \ldots, l\}, j \in \{1, \ldots, n_i\})$*such that*

(i) The number of states and computation of $\mathbf{SLOC}_{ij}$ *are independent of the number* n_i *of agents for all* $i \in \{1, \ldots, l\}$*;*

(ii) the set of $\mathbf{SLOC}_{ij}$ *is (collectively) control equivalent to the scalable supervisor* **SSUP** *with respect to plant* $\mathbf{G}$*:*

$$\bigcap_{\substack{i \in \{1,\ldots,l\} \\ j \in \{1,\ldots,n_i\}}} L_m(\mathbf{SLOC}_{ij}) \cap L_m(\mathbf{G}) = L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G}).$$

To solve SDCSP, we employ a known technique called *supervisor localization* (Cai & Wonham, 2010, 2016), which works to decompose an arbitrary supervisor into a set of local controllers whose collective behavior is equivalent to that of supervisor. Since we have synthesized **SSUP**, the scalable supervisor, a straightforward approach would be to apply supervisor localization to decompose the associated controlled behavior $L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G})$. This approach would require, however, the computation of $\mathbf{G}$ which is infeasible for large systems and causes the resulting local controllers non-scalable. Instead we propose the following procedure for designing scalable local controllers $\mathbf{SLOC}_{ij}$, for $i \in \{1, \ldots, l\}$ and $j \in \{1, \ldots, n_i\}$.

(Q1) Apply supervisor localization to decompose the relabeled supervisor **RSUP** into *relabeled local controllers* $\mathbf{RLOC}_i$, $i \in \{1, \ldots, l\}$, such that (by Cai & Wonham, 2010, 2016)

$$\bigcap_{i \in \{1,\ldots,l\}} L_m(\mathbf{RLOC}_i) \cap L_m(\mathbf{M}) = L_m(\mathbf{RSUP}).$$

(Q2) Compute $\text{trim}(\mathbf{RLOC}_i \parallel \mathbf{M}_i)$, where $\text{trim}(\cdot)$ operation removes blocking states (if any) of the argument generator.

(Q3) Inverse-relabel $\text{trim}(\mathbf{RLOC}_i \parallel \mathbf{M}_i)$ to obtain $\mathbf{SLOC}_{ij}$ ($j \in \{1, \ldots, n_i\}$), i.e.

$$\mathbf{SLOC}_{ij} := R^{-1}(\text{trim}((\mathbf{RLOC}_i \parallel \mathbf{M}_i))). \quad (5)$$

Notice that the computations involved in the above procedure are independent of the number n_i ($i \in \{1, \ldots, l\}$) of agents. In (Q1), computing $\mathbf{RLOC}_i$ by localization requires computing **RSUP** and $\mathbf{M}$ (in (P3) and (P1) respectively), both of which are independent of n_i (although dependent on k_i fixed in (P1)). In (Q2), for the synchronous product both $\mathbf{RLOC}_i$ and $\mathbf{M}_i$ are independent of n_i,

[3] In the centralized architecture, the communication from **SSUP** to the agents is typically done via event broadcasting. On the other hand, in a distributed architecture, the communication between local controllers of the agents is naturally pairwise.

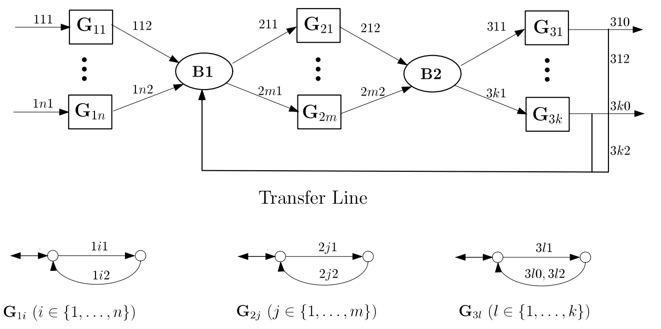

Transfer Line
B1
B2
G_{1i} $(i \in \{1,\ldots,n\})$
G_{2j} $(j \in \{1,\ldots,m\})$
G_{3l} $(l \in \{1,\ldots,k\})$

Fig. 2. Transfer line: system configuration and component agents. Event $1i1$ ($i \in \{1,\ldots,n\}$) means that $\mathbf{G}_{1i}$ starts to work by taking in a workpiece, and $1i2$ means that $\mathbf{G}_{1i}$ finishes work and deposits a workpiece to buffer **B1**; event $2j1$ ($j \in \{1,\ldots,m\}$) means that $\mathbf{G}_{2j}$ starts to work by taking in a workpiece, and $2j2$ means that $\mathbf{G}_{2j}$ finishes work and deposits a workpiece to buffer **B2**; event $3l1$ ($l \in \{1,\ldots,k\}$) means that $\mathbf{G}_{3l}$ starts to work by testing a workpiece, $3l0$ means that $\mathbf{G}_{3l}$ detects a fault and sends the faulty workpiece back to buffer **B1**, and $3l2$ means that $\mathbf{G}_{3l}$ detects no fault and output the successfully processed workpiece.

while trim may only reduce some states. Finally in (Q3), inverse-relabeling does not change the number of states. Therefore the state number of the resulting scalable local controller $\mathbf{SLOC}_{ij}$ and its computation are independent of the number n_i of agents.

The synchronous product in (Q2) may produce blocking states; such an example is provided in Section 5. Thus the trim operation is needed to ensure that the resulting $\mathbf{SLOC}_{ij}$ is a nonblocking generator. In addition, note that $\mathbf{SLOC}_{ij}$ are the same for all $j \in \{1,\ldots,n_i\}$. This means that every agent $\mathbf{G}_{ij}$ in the same group $\mathcal{G}_i$ obtains the same local controller, although each local controller will be dedicated to enabling/disabling only the controllable events originated from its associated agent.

The main result of this section is the following.

Theorem 2. *The set of* $\mathbf{SLOC}_{ij}$ *(*$i \in \{1,\ldots,l\}, j \in \{1,\ldots,n_i\}$*) as in (5) is a set of scalable local controllers that solves SDCSP.*

5. Illustrating example

We present an example of a transfer line system, adapted from (Wonham & Cai, 2019). As displayed in Fig. 2, transfer line consists of machines ($\mathbf{G}_{11},\ldots,\mathbf{G}_{1n}$; $\mathbf{G}_{21},\ldots,\mathbf{G}_{2m}$) and test units ($\mathbf{G}_{31},\ldots,\mathbf{G}_{3k}$), linked by two buffers **B1** and **B2** both with capacities 1. The generators of the agents are shown in Fig. 2. Based on their different roles, the machines are divided into 3 groups: $\mathcal{G}_1 = \{\mathbf{G}_{11},\ldots,\mathbf{G}_{1n}\}$, $\mathcal{G}_2 = \{\mathbf{G}_{21},\ldots,\mathbf{G}_{2m}\}$, and $\mathcal{G}_3 = \{\mathbf{G}_{31},\ldots,\mathbf{G}_{3k}\}$. Let the relabeling map R be given by $R(1i1) = 11$, $R(1i2) = 12$, $i \in \{1,\ldots,n\}$, $R(2j1) = 21$, $R(2j2) = 22$, $j \in \{1,\ldots,m\}$, and $R(3l0) = 30$, $R(3l1) = 31$, $R(3l2) = 32$, $l \in \{1,\ldots,k\}$, where odd-number events are controllable and even-number events are uncontrollable. It is easily observed that Assumptions (A1), (A2) hold.

The specification is to avoid underflow and overflow of buffers **B1** and **B2**, which is enforced by the two generators **E1** and **E2** in Fig. 3. Thus the overall specification E is $E = L_m(\mathbf{E1}) \cap L_m(\mathbf{E2})$, which can be verified to satisfy Assumption (A3). It is also verified that Assumption (A4) holds. In addition, it is checked that $L_m(\mathbf{H}_i) := L_m(R(\mathbf{G}_{i1}))$ ($i = 1,2,3$) is controllable with respect to $R(L(\mathbf{G}_{i1} \parallel \mathbf{G}_{i2}))$. By Proposition 1, we have that $L_m(\mathbf{M})$ is controllable with respect to $R(L(\mathbf{G}))$. Therefore the sufficient condition of Theorem 1 is satisfied.

By the procedure (P1)-(P4) with $k_1 = 2$, $k_2 = 3$, $k_3 = 1$, we design a scalable supervisor **SSUP**, displayed in Fig. 3. The state size of **SSUP** and its computation are independent of the agent numbers n, m, k. Moreover, the controlled behavior of **SSUP**

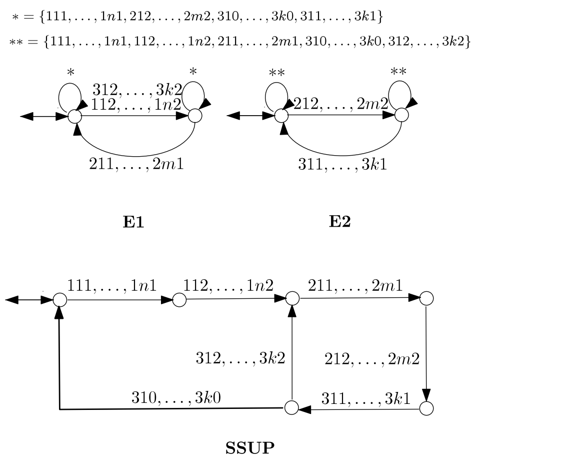

$* = \{111,\ldots,1n1,212,\ldots,2m2,310,\ldots,3k0,311,\ldots,3k1\}$
$** = \{111,\ldots,1n1,112,\ldots,1n2,211,\ldots,2m1,310,\ldots,3k0,312,\ldots,3k2\}$
E1
E2
SSUP

Fig. 3. Transfer line: specification generators **E1**, **E2**, and scalable supervisor **SSUP**.

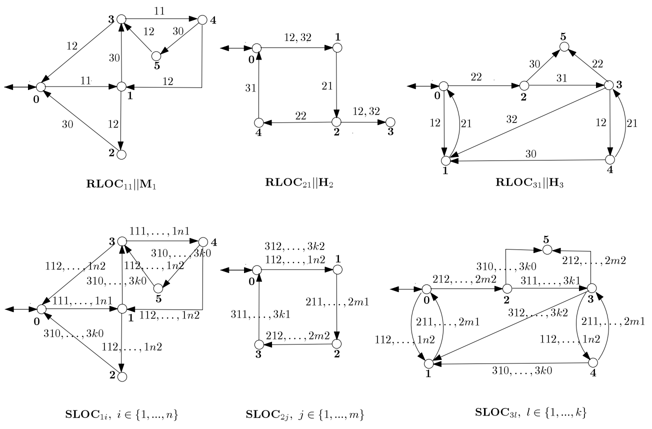

$\mathbf{RLOC}_{11}||\mathbf{M}_1$
$\mathbf{RLOC}_{21}||\mathbf{H}_2$
$\mathbf{RLOC}_{31}||\mathbf{H}_3$
$\mathbf{SLOC}_{1i},\ i \in \{1,\ldots,n\}$
$\mathbf{SLOC}_{2j},\ j \in \{1,\ldots,m\}$
$\mathbf{SLOC}_{3l},\ l \in \{1,\ldots,k\}$

Fig. 4. Transfer line: scalable local controllers ($\mathbf{SLOC}_{1i}$ for machine $\mathbf{G}_{1i}$, $i \in \{1,\ldots,n\}$; $\mathbf{SLOC}_{2i}$ for machine $\mathbf{G}_{2j}$, $j \in \{1,\ldots,m\}$; $\mathbf{SLOC}_{3i}$ for test unit $\mathbf{G}_{3l}$, $l \in \{1,\ldots,k\}$).

is in fact equivalent to that of the monolithic supervisor **SUP**, i.e. $L_m(\mathbf{SSUP}) \cap L_m(\mathbf{G}) = L_m(\mathbf{SUP})$, for arbitrary fixed values of n, m, k. This is owing to that both buffers have only one slot, and thus the restriction due to relabeling is already enforced by the monolithic supervisor in order to satisfy the specification.

Scalable distributed control. Following the procedure (Q1)-(Q3) in Section 4, we compute the scalable local controllers for the individual agents. In (Q2), certain synchronous products turn out to be blocking, as displayed in Fig. 4 (upper part). Hence the trim operation in (Q2) is important to ensure that the resulting local controllers are nonblocking. In Fig. 4 (lower part), $\mathbf{SLOC}_{1i}$ (6 states) is for the machine $\mathbf{G}_{1i}$, $i \in \{1,\ldots,n\}$; $\mathbf{SLOC}_{2j}$ (4 states) for the machine $\mathbf{G}_{2j}$, $j \in \{1,\ldots,m\}$; and $\mathbf{SLOC}_{3i}$ (6 states) for the test unit $\mathbf{G}_{3l}$, $l \in \{1,\ldots,k\}$. It is verified that the desired control equivalence between the set of local controllers and the supervisor **SSUP** in Fig. 3 is satisfied, i.e. the condition (ii) of SDCSP holds.

The control logic of the scalable local controllers is as follows. First for $\mathbf{SLOC}_{1i}$ ($i \in \{1,\ldots,n\}$), which controls only the event $1i1$ of machine $\mathbf{G}_{1i}$, observe that event $1i1$ is disabled at states 1, 2, and 4 to protect buffer **B1** against overflow, while it is disabled at state 5 due to the restriction of relabeling. As mentioned above, relabeling allows parallel operations of two machines in group one. Next for $\mathbf{SLOC}_{2j}$ ($j \in \{1,\ldots,m\}$), which is responsible only for event $2j1$ of machine $\mathbf{G}_{2j}$, observe that event $2j1$ is disabled at states 0, 2 and 3. This is to protect buffer **B1** against underflow and buffer **B2** against overflow. Finally for $\mathbf{SLOC}_{3l}$ ($l \in \{1,\ldots,k\}$), which is responsible only for event $3l1$ of test unit $\mathbf{G}_{3l}$, observe that event $3l1$ is disabled at states 0, 1, 3, 4 and 5. This is to protect buffer **B2** against underflow and buffer **B1** against overflow.

6. Conclusions

We have studied multi-agent discrete-event systems that can be divided into several groups of independent and similar agents. We have employed a relabeling map to generate template structures, based on which scalable supervisors are designed whose state sizes and computational process are independent of the number of agents. We have presented a sufficient condition for the validity of the designed scalable supervisors, and shown that this condition may be verified with low computational effort. Moreover, based on the scalable supervisor we have designed scalable local controllers, one for each component agent. An example has been provided to illustrate our proposed synthesis methods.

References

Cai, K., & Wonham, W. M. (2010). Supervisor localization: a top-down approach to distributed control of discrete-event systems. *IEEE Transactions on Automatic Control*, *55*(3), 605–618.

Cai, K., & Wonham, W. M. (2016). *Supervisor localization: a top-down approach to distributed control of discrete-event systems*. Springer.

ElMaraghy, H. A. (2005). Flexible and reconfigurable manufacturing systems paradigms. *International Journal of Flexible Manufacturing Systems*, *17*(4), 261–276.

Gohari, P., & Wonham, W. M. (2000). On the complexity of supervisory control design in the RW framework. *IEEE Transactions on Systems, Man, and Cybernetics: Cybernetics*, *30*(5), 643–652.

Jiao, T., Gan, Y., Xiao, G., & Wonham, W. M. (2017). Exploiting symmetry of state tree structures for discrete-event systems with parallel components. *International Journal of Control*, *90*(8), 1639–1651.

Jiao, T., Gan, Y., Xiao, G., & Wonham, W. M. (2018). Exploiting symmetry of discrete-event systems by relabeling and reconfiguration. *IEEE Transctions on Systems, Man, and Cybernetics: Systems*, http://dx.doi.org/10.1109/TSMC.2018.2795011.

Liu, Y., Cai, K., & Li, Z. (2018a). *On scalable supervisory control of multi-agent discrete-event systems*: *Technical Report*, Available at: https://arxiv.org/pdf/1704.08858.pdf (Last accessed 31 December 2018).

Liu, Y., Cai, K., & Li, Z. (2018b). On scalable supervisory control of multi-agent discrete-event systems. In *Proceeding of 14th workshop on discrete-event system* (pp. 25–30). Italy.

Nooruldeen, A., & Schmidt, K. W. (2014). State attraction under language specification for the reconfiguration of discrete event systems. *IEEE Transactions on Automatic Control*, *60*(6), 1630–1634.

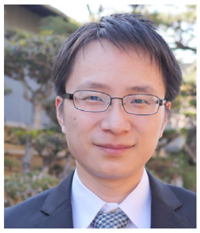
Wonham, W. M., & Cai, K. (2019). *Supervisory control of discrete-event systems*. Springer.

Wonham, W. M., & Ramadge, P. J. (1987). On the supremal controllable sublanguage of a given language. *SIAM Journal on Control and Optimization*, *25*(3), 637–659.

Wu, N., & Zhou, M. (2007). Deadlock resolution in automated manufacturing systems with robots. *IEEE Transactions on Automation Science and Engineering*, *4*(3), 474–480.

Yingying Liu received her B.S. degree in Automation from Taiyuan University of Technology, Taiyuan, China, in 2013. She is currently pursuing the Ph.D degree in Mechanical and Electronic Engineering at Xidian University, Xi'an, China. She received Best Student Paper Award from Workshop on Discrete Event Systems 2018.

Her research interests include supervisory control of multi-agent discrete-event systems (DES) with applications to large-scale manufacturing systems, supervisory control of multi-agent DES with partial observations, and distributed control of DES.



Kai Cai received the B.Eng. degree in Electrical Engineering from Zhejiang University, Hangzhou, China, in 2006; the M.A.Sc. degree in Electrical and Computer Engineering from the University of Toronto, Toronto, ON, Canada, in 2008; and the Ph.D. degree in Systems Science from the Tokyo Institute of Technology, Tokyo, Japan, in 2011. He is currently an Associate Professor at Osaka City University. Previously, he was an Assistant Professor at the University of Tokyo (2013–2014), and a Postdoctoral Fellow at the University of Toronto (2011–2013).

Dr. Cai's research interests include distributed control of discrete-event systems and cooperative control of networked multi-agent systems. He is the co-author (with W.M. Wonham) of Supervisory Control of Discrete-Event Systems (Springer 2019) and Supervisor Localization (Springer 2016). He is serving as the Chair for the IEEE CSS Technical Committee on Discrete Event Systems and an Associate Editor for the IEEE Transactions on Automatic Control.

Zhiwu Li received the B.S., M.S., and Ph.D. degrees in mechanical engineering, automatic control, and manufacturing engineering, respectively, all from Xidian University, Xi'an, China, in 1989, 1992, and 1995, respectively. He joined Xidian University in 1992 and now he is also with the Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau. Over the past decade, he was a Visiting Professor at the University of Toronto, Technion (Israel Institute of Technology), Martin-Luther University of Halle--Wittenburg, Conservatoire National des Arts et Métiers (CNAM), Meliksah Universitesi, and King Saud University.

Dr. Li's research interests include Petri net theory and application, supervisory control of discrete event systems, workflow modeling and analysis, system reconfiguration, game theory, and data and process mining. He is a member of Discrete Event Systems Technical Committee of the IEEE Systems, Man, and Cybernetics Society, and a member of IFAC Technical Committee on Discrete Event and Hybrid Systems from 2011 to 2014. He serves as a frequent reviewer for 90+ international journals including Automatica and a number of the IEEE Transactions as well as many international conferences. He is listed in Marquis Who's Who in the world, 27th Edition, 2010. Dr. Li is a recipient of an Alexander von Humboldt Research Grant, Alexander von Humboldt Foundation, Germany, and Research in Paris, France. He is the founding chair of Xi'an Chapter of IEEE Systems, Man, and Cybernetics Society.