Supervisor Localization ⋆

Kai Cai [a]

[a] *Department of Electrical and Computer Engineering, Osaka City University*

8 June 2019

Abstract

In this article, we formulate a distributed control problem for multi-agent discrete-event systems, and introduce a *supervisor localization* approach to solving this problem. The approach defines a *control cover* on the state set of the supervisor, and constructs local controllers by merging the states residing in the same cells of the cover. The algorithm for supervisor localization is of polynomial complexity in the state size of the supervisor, and architectural and symbolic methods are explored to improve the algorithmic efficiency to tackle large-scale multi-agent systems. Further variations of the central concept of control cover are introduced, which expand the features of local controllers including time, partial observation, communication delay, and infinite behavior.

Key words: Supervisor localization, distributed control, discrete-event systems, multi-agent systems

Introduction

This article introduces a distributed control problem of multi-agent systems modeled as (finite-state) automata. Such systems are known as discrete-event systems (DES), whose dynamics is discrete (in time and usually in state space) and is asynchronous or event-driven (i.e. state transitions are driven by events or instantaneous happenings). We focus on the type of DES that consists of multiple component agents, interconnected and interacting with one another to achieve an overall goal. Examples of such multi-agent DES include machines in flexible manufacturing cells, mobile robots in warehouses, automated guided vehicles in loading docks, communication channels in cognitive radio networks, and parallel processors in multi-core supercomputers.

The objective of the distributed control problem is to synthesize a local controller for each component agent, and collectively the local controllers achieve a prescribed global goal. Equipped with these internal controllers, individual

⋆ This work was supported in part by JSPS KAKENHI Grant No. 16K18122.

agents make their own local control decisions, while communicate with peers (ideally nearest neighbors) for necessary information exchange. This is a purely distributed control architecture, in contrast with other architectures where external supervisory entities exist.

Problem Formulation

Consider a DES consisting of $N(>1)$ component agents:

$$\mathbf{G}_k = (Q_k, \Sigma_k, \delta_k, q_{0,k}, Q_{m,k}), \quad k = 1, \ldots, N. \tag{1}$$

$\mathbf{G}_k$ is the (finite-state) automaton model of agent k, where Q_k is the set of states, Σ_k the set of events, $\delta_k : Q_k \times \Sigma_k \to Q_k$ the (partial) state transition function (that describes how states transit on occurrences of events), $q_{0,k} \in Q_k$ the initial state, and $Q_{m,k} \subseteq Q_k$ the set of marker states (i.e. goal states of agent k). The state transition function δ_k can be extended to $\delta_k : Q_k \times \Sigma_k^* \to Q_k$, where Σ_k^* is the set of all finite-length strings of events in Σ_k, including the empty string ϵ. We write $\delta_k(q, s)!$ to mean that $\delta_k(q, s)$ is defined, namely string $s \in \Sigma_k^*$ can occur at state $q \in Q_k$.

The *closed behavior* of $\mathbf{G}_k$ is the language (i.e. set of strings)

$$L(\mathbf{G}_k) := \{s \in \Sigma_k^* \mid \delta_k(q_{0,k}, s)!\} \subseteq \Sigma_k^* \tag{2}$$

and the *marked behavior* is

$$L_m(\mathbf{G}_k) := \{s \in L(\mathbf{G}_k) \mid \delta_k(q_{0,k}, s) \in Q_{m,k}\} \subseteq L(\mathbf{G}_k). \tag{3}$$

Language $L(\mathbf{G}_k)$ represents the set of all possible strings that can occur in $\mathbf{G}_k$, while $L_m(\mathbf{G}_k)$ the subset of 'desired' strings that hit a marker state of $\mathbf{G}_k$. The *prefix closure* of $L_m(\mathbf{G}_k)$, written $\overline{L_m(\mathbf{G}_k)}$, is the set of all history strings of $L_m(\mathbf{G}_k)$, namely $\overline{L_m(\mathbf{G}_k)} = \{s \in \Sigma^* \mid (\exists t \in \Sigma^*) st \in L_m(\mathbf{G}_k)\}$. If $\overline{L_m(\mathbf{G}_k)} = L(\mathbf{G}_k)$, i.e. every string generated by $\mathbf{G}_k$ can be completed to reach a marker state of $\mathbf{G}_k$, then $\mathbf{G}_k$ is said to be *nonblocking*.

For control purposes, the event set Σ_k of $\mathbf{G}_k$ is partitioned into a subset $\Sigma_{c,k}$ of controllable events and a subset $\Sigma_{u,k}$ of uncontrollable events. Controllable events are those that a controller can prohibit from occurring; by contrast, uncontrollable events cannot be inhibited.

Given the agents' automata above, the multi-agent DES $\mathbf{G}$ is the *synchronous product* of $\mathbf{G}_1, \ldots, \mathbf{G}_N$, written

$$\mathbf{G} := \mathbf{G}_1 || \cdots || \mathbf{G}_N. \tag{4}$$

$\mathbf{G}$ is a DES of which the component automata $\mathbf{G}_k$ are free to execute their events independently except for synchronization on events that are shared. In general agents share events, i.e. $\Sigma_k \cap \Sigma_l \neq \emptyset$ for some $k, l \in \{1, \ldots, N\}$;

a shared event in $\Sigma_k \cap \Sigma_l$ must be either controllable for both agents $\mathbf{G}_k$, $\mathbf{G}_l$ or uncontrollable for both. The event set of $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$ is $\Sigma = \Sigma_1 \cup \cdots \cup \Sigma_N$, which is partitioned into the controllable $\Sigma_c = \Sigma_{c,1} \cup \cdots \cup \Sigma_{c,N}$ and the uncontrollable $\Sigma_u = \Sigma_{u,1} \cup \cdots \cup \Sigma_{u,N}$. The closed (resp. marked) behavior of $\mathbf{G}$ is the synchronous product of the individual agents' closed (resp. marked) behaviors:

$$L(\mathbf{G}) = L(\mathbf{G}_1)||\cdots||L(\mathbf{G}_N) \quad (\text{resp. } L_m(\mathbf{G}) = L_m(\mathbf{G}_1)||\cdots||L_m(\mathbf{G}_N)). \tag{5}$$

The control requirement on the multi-agent DES $\mathbf{G}$ is imposed as a sublanguage $K \subseteq L_m(\mathbf{G})$, namely part of the desired behavior of $\mathbf{G}$. The prefix closure of K is $\overline{K} = \{s \in \Sigma^* \mid (\exists t \in \Sigma^*)st \in K\}$. A sublanguage $K \subseteq L_m(\mathbf{G})$ is said to be *controllable* (with respect to $\mathbf{G}$) provided for every string $s \in \overline{K}$ and every uncontrollable event $\sigma \in \Sigma_u$, if $s\sigma \in L(\mathbf{G})$ then $s\sigma \in \overline{K}$. Succinctly

$$\overline{K}\Sigma_u \cap L(\mathbf{G}) \subseteq \overline{K}. \tag{6}$$

Whether or not $K(\subseteq L_m(\mathbf{G}))$ is controllable, bring in the family of sublanguages of K that are controllable (with respect to $\mathbf{G}$):

$$\mathcal{C}(K) := \{K' \subseteq K \mid \overline{K'}\Sigma_u \cap L(\mathbf{G}) \subseteq \overline{K'}\}. \tag{7}$$

This family $\mathcal{C}(K)$ is algebraically closed under arbitrary set unions; that is, the union of an arbitrary number of (controllable) sublanguages in $\mathcal{C}(K)$ is controllable and thus belongs again to $\mathcal{C}(K)$. As a result, $\mathcal{C}(K)$ contains a unique supremal element

$$\sup \mathcal{C}(K) := \bigcup \{K' \subseteq K \mid K' \in \mathcal{C}(K)\}. \tag{8}$$

This supremal controllable sublanguage of K can be represented by a nonblocking automaton, say $\mathbf{SUP} = (X, \Sigma, \xi, x_0, X_m)$, such that

$$L_m(\mathbf{SUP}) = \sup \mathcal{C}(K), \quad L(\mathbf{SUP}) = \overline{L_m(\mathbf{SUP})}.$$

This automaton **SUP** can serve as a centralized supervisor for the multi-agent DES $\mathbf{G}$, by enabling/disabling (controllable) events of $\mathbf{G}$ such that the imposed control requirement K is satisfied in a maximally permissive manner. Namely, **SUP** allows the generation by $\mathbf{G}$ of the largest possible set of marked strings that are legal (i.e. sublanguage of K) and controllable. In this sense, **SUP** is the optimal and nonblocking supervisor for $\mathbf{G}$.

The supervisor **SUP** provides a performance criterion for the distributed control problem to be formulated. In distributed control, one aims to design a local controller for each component agent $\mathbf{G}_k$ ($k \in \{1, \ldots, N\}$). A (nonblocking)

automaton

$$\mathbf{LOC}_k = (Y_k, \hat{\Sigma}_k, \eta_k, y_{0,k}, Y_{m,k}), \quad \hat{\Sigma}_k \subseteq \Sigma \tag{9}$$

is called a *local controller* for agent $\mathbf{G}_k = (Q_k, \Sigma_k, \delta_k, q_{0,k}, Q_{m,k})$ if $\mathbf{LOC}_k$ can disable controllable events in, and only in, $\Sigma_{c,k}$ (i.e. local control authority); and the disabling actions are consistent with the centralized supervisor **SUP**. The latter means that after a string generated by the multi-agent DES **G**, a controllable event $\sigma \in \Sigma_{c,k}$ is disabled by $\mathbf{LOC}_k$ if and only if σ is disabled by **SUP**. While the control authority of $\mathbf{LOC}_k$ is strictly local, the observation scope need not, and generally will not, be. In particular, the event set $\hat{\Sigma}_k$ of $\mathbf{LOC}_k$ does not bear any relation with the event set Σ_k of $\mathbf{G}_k$, and may generally include some events originated by other agents. Such events are indeed critical for collaboration among agents necessary to achieve a collective global objective. Think of nearest-neighbor observation, as for motorists maneuvering through a congested intersection.

Distributed Control Problem (DCP): Consider a multi-agent DES **G** (consisting of $\mathbf{G}_1, \ldots, \mathbf{G}_N$), and let **SUP** be an optimal and nonblocking supervisor for **G** (that achieves an imposed control requirement). Construct a set of local controllers $\{\mathbf{LOC}_k \mid k \in \{1, \ldots, N\}\}$, one for each agent $\mathbf{G}_k$, such that

$$||_{k\in\{1,\ldots,N\}}\ L(\mathbf{LOC}_k)||L(\mathbf{G}) = L(\mathbf{SUP})$$
$$||_{k\in\{1,\ldots,N\}}\ L_m(\mathbf{LOC}_k)||L_m(\mathbf{G}) = L_m(\mathbf{SUP}).$$

This problem requires that the collective behavior of local controllers be identical to the centralized controlled behavior, and therefore be globally optimal and nonblocking. Namely, no control performance is compromised when the architecture shifts from centralized to distributed.

For the sake of easy implementation and comprehensibility, it would be desired in practice that the state sizes of local controllers $\mathbf{LOC}_k$ be very much less than that of the centralized supervisor **SUP**. That is, for each $k \in \{1, \ldots, N\}$, the state size $|Y_k| \ll |X|$ where $|\cdot|$ denotes set cardinality. Inasmuch as this property is neither precise to state nor always achievable, it must needs be omitted from the formal problem statement; in applications, nevertheless, it should be kept in mind.

Key Concept: Control Cover

To solve DCP, we introduce a top-down approach that decomposes the centralized supervisor **SUP** into a local controller $\mathbf{LOC}_k$, for each agent $\mathbf{G}_k$ in turn. This approach is called *supervisor localization*. For the decomposition, a suitable cover (called *control cover*) is constructed on the state set of **SUP**; then **SUP** is reduced by merging the states in the same cells of the cover.

The cover is induced by a *binary relation* for pairs of states of **SUP**. We first introduce this binary relation. Recall the multi-agent DES $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$ and the centralized supervisor $\mathbf{SUP} = (X, \Sigma, \xi, x_0, X_m)$. Fixing an arbitrary $k \in \{1, \ldots, N\}$, we describe the control information concerning agent $\mathbf{G}_k$'s controllable events in $\Sigma_{c,k}$. First, for each state $x \in X$ of **SUP**, let $E_k(x)$ be the subset of $\mathbf{G}_k$'s controllable events that are enabled at x; namely

$$\begin{aligned} E_k(x) &= \{\sigma \in \Sigma_{c,k} \mid \sigma \text{ is defined at } x\} \\ &= \{\sigma \in \Sigma_{c,k} \mid \xi(x, \sigma)!\} \subseteq \Sigma_{c,k}. \end{aligned} \tag{10}$$

Next, for each state $x \in X$ of **SUP**, let $D_k(x)$ be the subset of $\mathbf{G}_k$'s controllable events that are disabled at x:

$$\begin{aligned} D_k(x) &= \{\sigma \in \Sigma_{c,k} \mid \sigma \text{ is not defined at } x \text{ but is defined at } \delta(q_0, s) \text{ for some } s \text{ that reaches } x\} \\ &= \{\sigma \in \Sigma_{c,k} \mid \neg\xi(x, \sigma)! \ \&\ (\exists s \in \Sigma^*)(\xi(x_0, s) = x \ \&\ \delta(q_0, s\sigma)!)\} \subseteq \Sigma_{c,k}. \end{aligned} \tag{11}$$

In addition to the control information, we describe the marking information. For each state $x \in X$ of **SUP**, let $M(x) = true$ if and only if x is a marker state in **SUP**, i.e. $x \in X_m$. Also let $T(x) = true$ if and only if $\delta(q_0, s)$ is a marker state in **G** for some s that reaches x, i.e. $(\exists s \in \Sigma^*)\xi(x_0, s) = x \ \&\ \delta(q_0, s) \in Q_m$.

Based on the above control and marking information, we introduce a binary relation $\mathcal{R}_k$ on the state set X of **SUP**. For arbitrary two states $x, x' \in X$, the pair (x, x') belongs to $\mathcal{R}_k$ if they are consistent in both control information and marking information. Consistency in control information means that any controllable events in $\Sigma_{c,k}$ enabled at x are not disabled at x', and vice versa; that is

$$E_k(x) \cap D_k(x') = \emptyset = E_k(x') \cap D_k(x). \tag{12}$$

Consistency in marking information means that if x and x' have corresponding states of the same marking status in **G**, then x and x' have the same marking status in **SUP**; namely

$$T(x) = T(x') \Rightarrow M(x) = M(x'). \tag{13}$$

Satisfying both (12) and (13), the binary relation $\mathcal{R}_k \subseteq X \times X$ is called a *consistency relation*. Note that $\mathcal{R}_k$ is not transitive in general: $(x, x') \in \mathcal{R}_k$ and $(x', x'') \in \mathcal{R}_k$ need not imply $(x, x'') \in \mathcal{R}_k$. For example, a controllable event $\sigma \in \Sigma_{c,k}$ is neither enabled nor disabled at x', is enabled at x, and is disabled at x''.

The binary relation $\mathcal{R}_k \subseteq X \times X$ induces a special cover $\mathcal{C}_k$ (generally nonunique) on the state set X of **SUP**. A cover on a set is a collection of nonempty subsets, called *cells*, whose union is the set itself. A cover generally allows its cells to share elements. In the special case where the cells are disjoint, a cover is called a *partition*. For a cover $\mathcal{C}_k = \{X_i \subseteq X \mid X_i \neq \emptyset \ \&\ i \in I_k\}$ (I_k some index set) induced by $\mathcal{R}_k$, it is first required that for every cell X_i, all

the states in X_i be pairwise consistent with respect to $\mathcal{R}_k$:

$$(\forall i \in I_k, \forall x, x' \in X_i)\ (x, x') \in \mathcal{R}_k. \tag{14}$$

Moreover, it is required that for every cell X_i and every event $\sigma \in \Sigma$, all states that can be reached from any state in X_i by a one-step transition σ belong to the same cell X_j; namely

$$(\forall i \in I_k, \forall \sigma \in \Sigma)\Big[\big((\exists x \in X_i)\xi(x,\sigma)!\big) \Rightarrow \big((\exists j \in I_k)(\forall x' \in X_i)\xi(x',\sigma)! \Rightarrow \xi(x',\sigma) \in X_j\big)\Big]. \tag{15}$$

Inductively, two states x, x' belong to a common cell of $\mathcal{C}_k$ if any two states that can be reached respectively from x and x' by some string in Σ^* are consistent with respect to $\mathcal{R}_k$. Satisfying both (14) and (15), the cover $\mathcal{C}_k$ on X is called a *control cover*. Based on this control cover $\mathcal{C}_k$, we decompose the centralized supervisor **SUP** into a local controller $\mathbf{LOC}_k$ for agent $\mathbf{G}_k$.

The problem of computing a control cover with the minimum number of cells is known to be NP-hard. Nevertheless, a polynomial algorithm called *supervisor localization algorithm* exists that computes a control cover (in fact a partition) with a small number of cells. The time complexity is $O(|X|^4)$. This algorithm is adapted from an existing *supervisor reduction algorithm*, and empirical evidence has shown the algorithm's effectiveness of achieving near-minimum number of cells.

Local Controllers

Having obtained a control cover $\mathcal{C}_k = \{X_i \mid i \in I_k\}$ on the state set X of **SUP**, we merge the states in the same cells to construct a local controller $\mathbf{LOC}_k = (Y_k, \hat{\Sigma}_k, \eta_k, y_{0,k}, Y_{m,k})$ for agent $\mathbf{G}_k$.

The state set Y_k is just the set of cell indices, i.e. $Y_k = I_k$. The initial state $y_{0,k} \in Y_k$ is the index of a cell where the initial state x_0 of **SUP** belongs. However, x_0 may be shared by multiple cells; in that case, pick an arbitrary cell that contains x_0, whose index is (say) $i_{0,k}$, and set $y_{0,k} = i_{0,k}$. The marker state subset $Y_{m,k} \subseteq Y_k$ is the subset of indices of cells that contain at least one marker state of **SUP**; namely $Y_{m,k} := \{i \in I_k \mid X_i \cap X_m \neq \emptyset\}$.

To determine the event set $\hat{\Sigma}_k \subseteq \Sigma$ and the (partial) transition function $\eta_k : Y_k \times \hat{\Sigma}_k \to Y_k$, first consider the transition structure among cells involving the whole event set Σ, i.e. $\iota_k : I_k \times \Sigma \to I_k$. For a cell labeled $i \in I_k$ and an event $\sigma \in \Sigma$, $\iota_k(i,\sigma)!$ and $\iota_k(i,\sigma) = j$ (for some cell labeled $j \in I_k$) if there exists a state x in cell X_i such that σ is defined at x and the transition of σ leads x to a state in cell X_j, and moreover all the states in X_i at which σ is defined transit on occurrence of σ to states in X_j; that is

$$\Big[(\exists x \in X_i)\xi(x,\sigma)!\ \&\ \xi(x,\sigma) \in X_j\Big]\ \ \&\ \ (\forall x' \in X_i)\Big[\xi(x',\sigma)! \Rightarrow \xi(x',\sigma) \in X_j\Big]. \tag{16}$$

If $i = j$, the transition $\iota_k(i, \sigma) = j$ is called a *selfloop*. In the case where an event $\sigma \in \Sigma$ is a selfloop at whichever cell it is defined, then σ does not play a role in the local controller $\mathbf{LOC}_k$ and hence can be excluded. For this reason, we determine the event set $\hat{\Sigma}_k$ to be the subset of events in Σ such that there exists at least one cell where these events are not selfloops:

$$\hat{\Sigma}_k := \{\sigma \in \Sigma \mid (\exists i, j \in I_k)\ i \neq j\ \&\ \iota_k(i, \sigma)!\ \&\ \iota_k(i, \sigma) = j\}. \tag{17}$$

Finally, the transition function η_k is just ι_k but only involves events in $\hat{\Sigma}_k$: for $y_{(i)} \in Y_k$ that corresponds to the cell labeled i and for $\sigma \in \hat{\Sigma}_k$,

$$\eta_k(y_{(i)}, \sigma) = y_{(j)} \quad \text{if and only if} \quad \iota_k(i, \sigma) = j. \tag{18}$$

The above shows the construction of one local controller $\mathbf{LOC}_k$ based on a control cover $\mathcal{C}_k$ on the state set of **SUP**. The multi-agent DES $\mathbf{G}$ consists of N agents $\mathbf{G}_1, \ldots, \mathbf{G}_N$; for each agent in turn, we construct a local controller based on a corresponding control cover. In total, construct N local controllers $\mathbf{LOC}_1, \ldots, \mathbf{LOC}_N$ from N control covers $\mathcal{C}_1, \ldots, \mathcal{C}_N$. The following theorem states that these local controllers provide a solution to DCP.

Theorem 1. The set of local controllers $\{\mathbf{LOC}_1, \ldots, \mathbf{LOC}_N\}$ constructed above is a solution to DCP.

Since the local controllers are derived from control covers on the stat set of **SUP**, control covers are sufficient for the solvability of DCP. In fact, control covers are also necessary for solving DCP, as asserted in the following theorem.

Theorem 2. If a set of local controllers $\{\mathbf{LOC}_1, \ldots, \mathbf{LOC}_N\}$ is a solution to DCP, then there exist control covers $\mathcal{C}_1, \ldots, \mathcal{C}_N$ on the state set of **SUP** from which the local controllers are constructed.

Example

The following example, Transfer Line, is an illustration of the supervisor localization approach. As displayed in Fig. 1, Transfer Line consists of two machines **M1**, **M2** followed by a test unit **TU**; these three component agents are linked by two buffers **B1**, **B2** with capacities 3 and 1, respectively. A workpiece entering the system is first processed by **M1** and stored in **B1**, then processed by **M2** and stored in **B2**. A processed workpiece tested by **TU** may be accepted or rejected; if accepted, it is released from the system; if rejected, it is returned to **B1** for reprocessing by **M2**. Thus the structure incorporates a 'material feedback loop'. The control requirements are to protect the two buffers **B1**, **B2** against overflow and underflow. Automaton models of the agents and the requirements are also displayed in Fig. 1. Controllable (resp. uncontrollable) events are odd (resp. even) numbered transitions.

Let $\mathbf{G}_1 := \mathbf{M1}$, $\mathbf{G}_2 := \mathbf{M2}$, and $\mathbf{G}_3 := \mathbf{TU}$. Then the multi-agent DES $\mathbf{G}$ is the synchronous product $\mathbf{G} = \mathbf{G}_1||\mathbf{G}_2||\mathbf{G}_3$. To describe the overall control requirement, form the synchronous product $\mathbf{K} = \mathbf{B1}||\mathbf{B2}||\mathbf{G}$; then the

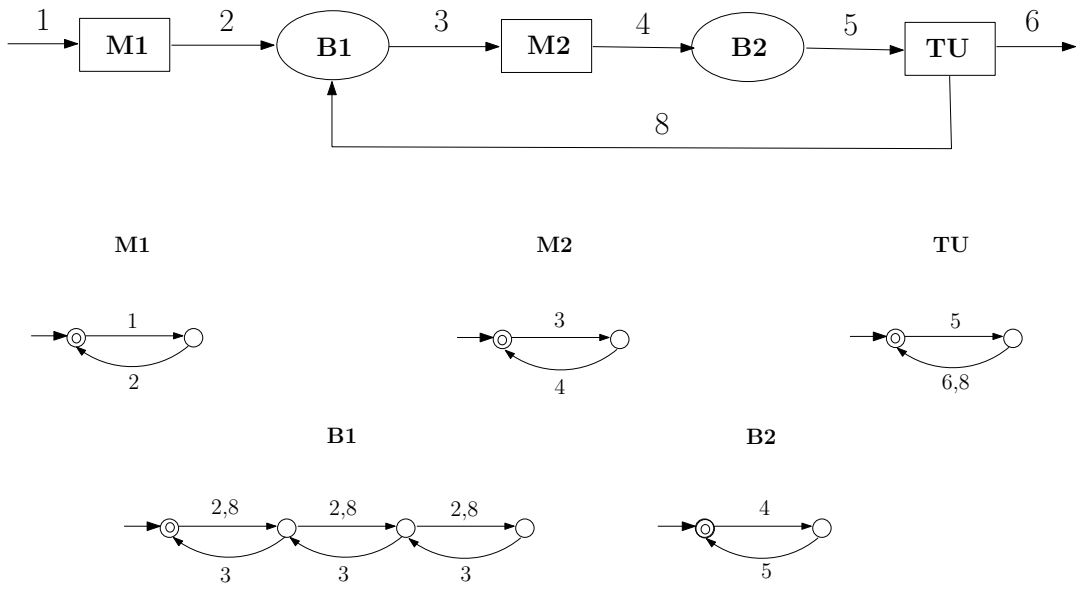

Fig. 1. Transfer Line

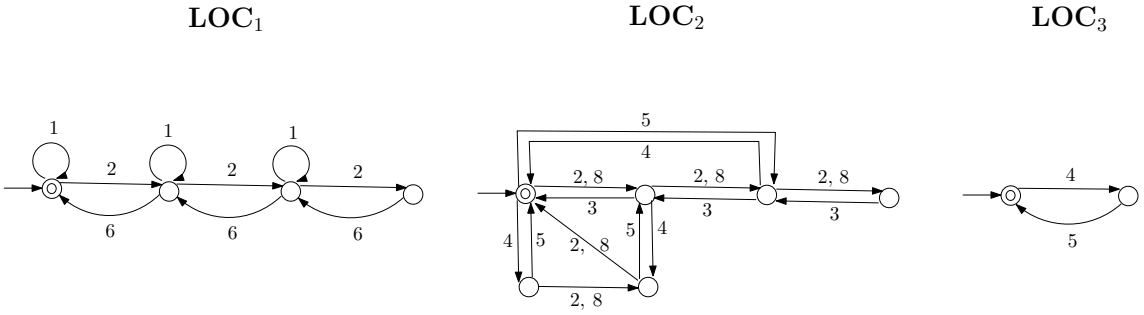

Fig. 2. Local Controllers

requirement language is $K = L_m(\mathbf{K}) \subseteq L_m(\mathbf{G})$. The supremal controllable sublanguage of K can be represented by a nonblocking automaton **SUP** having 28 states and 65 transitions. This **SUP** is the centralized supervisor that enforces the control requirement K for the multi-agent DES **G** in an optimal (i.e. maximally permissive) and nonblocking manner.

Now constructing suitable control covers on the state set of **SUP**, we decompose **SUP** into local controllers $\mathbf{LOC}_k$ for agents $\mathbf{G}_k$ $(k = 1, 2, 3)$. These controllers are displayed in Fig. 2, having 4, 6, and 2 states respectively. The ratios of state reduction are 7, 4.7, and 14. Moreover it is verified that the collective behavior of these local controllers is identical to the centralized controlled behavior:

$$||_{k\in\{1,2,3\}}\ L(\mathbf{LOC}_k)||L(\mathbf{G}) = L(\mathbf{SUP})$$
$$||_{k\in\{1,2,3\}}\ L_m(\mathbf{LOC}_k)||L_m(\mathbf{G}) = L_m(\mathbf{SUP}).$$

Equipped with these local controllers of reasonably small state sizes, the control logic of each individual agent is

transparent. Machine **M1** with $\mathbf{LOC}_1$, controlling event 1, ensures that no more than three workpieces can be processed simultaneously in the system; this is to prevent 'choking' the material feedback loop. Machine **M2** with $\mathbf{LOC}_2$, controlling event 3, simultaneously guarantees the safety of both buffers (against underflow and overflow). Finally, test unit **TU** with $\mathbf{LOC}_3$, controlling event 5, is responsible only for protecting buffer **B2** against underflow and overflow.

Large-Scale Multi-Agent Systems

To construct local controllers, we first need to compute a centralized supervisor **SUP** for the multi-agent DES **G**. When **G** consists of a large number of agents, the computation of **SUP** tends to become infeasible, inasmuch as the complexity is known to be exponential in the number of agents. Indeed, the computation of **G** itself as in (4) is the product of the component agents, and **G**'s state size tends to be exponential in the number of agents and can easily become astronomical.

To solve DCP for large-scale multi-agent systems, we need to combine the construction of local controllers with an efficient supervisory synthesis approach. There are two basic such approaches: architectural and symbolic.

The architectural approach explores decentralized and hierarchical modularization, or in a heterarchical combination. Consider a multi-agent DES **G** consisting of N agents $\mathbf{G}_1, \ldots, \mathbf{G}_N$, and suppose that the total control requirement on **G** is comprised of M sublanguages $K_1, \ldots, K_M \subseteq L_m(\mathbf{G})$ (as subrequirement). Each subrequirement is typically imposed on a subset of agents; thus we synthesize M *decentralized* supervisors $\mathbf{SUP}_1, \ldots, \mathbf{SUP}_M$ each enforcing a subrequirement for the involved subset of agents. If these supervisors are *nonconflicting* (in the sense that a string generated by a supervisor can also be generated by other supervisors), then the synchronous product of these supervisors is the same as the centralized supervisor **SUP**. In case the decentralized supervisors do conflict, a *coordinator* **CO** can be designed (by suitable abstraction techniques) to resolve the conflict, which yields

$$\mathbf{SUP}_1 || \cdots || \mathbf{SUP}_M || \mathbf{CO} = \mathbf{SUP}. \tag{19}$$

That is, the family of M decentralized supervisors and the coordinator is collectively identical to the centralized supervisor. Computing each member in this family is typically more efficient than computing the centralized supervisor. After synthesizing the M decentralized supervisors and the coordinator, we decompose each of them into local controllers by the same construction introduced above. These local controllers again solve DCP, but can be computed more efficiently.

The second, symbolic approach explores efficient representation of DES. *State tree structure* is such an approach that represents the states and transitions of DES as a tree organization, and stores the organization intensionally by *binary decision diagram* encoding. Computation is performed using algorithmic recipes defined specifically for

this organization, which avoids explicit enumeration of the plain states and transitions. The central concept for localization, i.e. control cover, can be introduced for state tree structures, and thereby the computation of local controllers is made efficient by taking advantage of efficient DES representation. The resulting symbolic local controllers again solve DCP, in the sense that they collectively achieve the same controlled behavior as the centralized symbolic supervisor.

Extensions

Supervisor localization has been extended in a number of directions, including timed DES, partial observation, communication delay, and infinite-behavior DES. Invariably in all these extensions, the central concept is control cover of different variations.

For timed DES, a global clock is introduced and each event has a (countdown) timer. Thus rather than, or at least in addition to, permanent disablement of a (controllable) event as in the untimed case, the centralized supervisor needs to *force* certain events in a timely fashion to meet a temporal requirement. For this reason, a subset of *forcible* events is introduced at the supervisor's disposal, which can be used to preempt the tick of the global clock. As a result, the controlled behavior of a timed supervisor includes tick-preempting actions (in addition to disabling actions). The corresponding new feature of supervisor localization is to decompose the tick-preempting actions into *local preemptors* for the individual agents; the central concept for this decomposition is a *preemption cover*, defined similarly to control cover but accounting for the tick-preempting actions instead.

Partial observation is a well-studied phenomenon for both untimed and timed DES, which refers to the realistic constraint that not all events can be observed by the supervisor. The necessary and sufficient condition for the existence of a partial-observation supervisor is *observability*, in addition to controllability. Unlike controllability, however, observability is not algebraically closed under set unions, and consequently there generally does not exist a unique supremal observable sublanguage of a given (control requirement) language. A remedy to observability, but more conservative, is *normality*, which more recently is relaxed to *relative observability*; the latter two are closed under unions and permit the existence of supremal elements, respectively. In any case, once a partial-observation supervisor is computed, it can be decomposed into local controllers (and local preemptors in the timed case) for individual agents. The central concept is *partial-observation control cover* (and *partial-observation preemption cover* in the timed case) which requires those states indistinguishable under partial observation to be consistent (in the same sense as for the full-observation case).

Equipped with local controllers, agents need to communicate certain events with peers to achieve critical synchronization. Indeed, agent $\mathbf{G}_k$ needs to receive events in $\hat{\Sigma}_k - \Sigma_k$ ($\hat{\Sigma}_k$ as in (17)) from other agents in order to make state transitions in its own local controller $\mathbf{LOC}_k$. If event communication was perfect, when an agent $\mathbf{G}_l$ sends an

event $\sigma(\in \hat{\Sigma}_k - \Sigma_k)$ to agent $\mathbf{G}_k$, $\mathbf{G}_k$ receives σ instantaneously. In practice, however, communication goes through physical channels, which are typically subject to delay. Consequently, $\mathbf{G}_k$ can receive the event σ only after some delay. This communication delay is reflected by a channel model that treats σ as sending the event by $\mathbf{G}_l$, while a delayed version σ' as receiving the event by $\mathbf{G}_k$, such that σ' occurs after σ with unbounded delay (and bounded delay counted by the global clock in the timed DES case). With the channel model, one may test if the local controllers are *robust* to communication delay, in the sense that the channeled behavior is *sound* and *complete* with respect to the original, zero delay controlled behavior. Moreover, treating the channels as part of the plant (namely synchronous product of DES $\mathbf{G}$ and the channels), we may synthesize a centralized supervisor tolerant of delay, and decompose the supervisor into local controllers. Since the local controllers are collectively equivalent to the centralized supervisor, they are ensured to tolerate the same delay as the supervisor does. Central to this decomposition is implicitly a *delayed control cover* (untimed or timed).

DES with *infinite behavior* (often modeled by a Büchi automaton of which strings may have infinite length) describes systems' behavior in the long run and asymptotic sense. For such DES, one can impose control requirement that some desired property becomes true *eventually*, or some behavior holds *infinitely often*; these are referred to as *liveness specifications*, which may be stated using temporal logic formulas. Accordingly, a centralized supervisor for infinite-behavior DES needs to enforce liveness specifications, although its control decisions must be made at certain finite times (so that these decisions may be realistically implemented). The centralized supervisor may be decomposed into local controllers, through defining a *liveness control cover* on the state set of the supervisor. The resulting local controllers collectively achieve the same global liveness behavior as the supervisor does.

Conclusions

In this article, we have formulated one distributed control problem for multi-agent DES, and introduced a supervisor localization approach to solving this problem. The approach defines a control cover on the state set of the supervisor, and constructs local controllers by merging the states residing in the same cells of the cover. The algorithm for supervisor localization is of polynomial complexity in the state size of the supervisor, and architectural and symbolic methods are explored to improve the algorithmic efficiency to tackle large-scale multi-agent systems. Further variations of the central concept of control cover are introduced, which expand the features of local controllers including time, partial observation, communication delay, and infinite behavior.

At its core, supervisor localization is a structural reduction procedure and the resulting local controllers are the quotient structures. The takeaway message of supervisor localization is, everything that can be done globally can be done locally.

Notes and References

Supervisor localization was initially proposed in [1], and further developed in [2]. For large-scale systems, the architectural approach was introduced in [3] while the symbolic approach in [4]. Extensions to timed DES, partial observation, communication delay, and infinite behavior were presented in [5], [6], [7-8], and [9] respectively. A comprehensive treatment of supervisor localization can be found in the monograph [10].

[1] K. Cai and W.M. Wonham, "Supervisor localization: a top-down approach to distributed control of discrete-event systems", IEEE Transactions on Automatic Control, vol. 55, no. 3, pp. 605-618, Mar. 2010.

[2] K. Cai and W.M. Wonham, "New results on supervisor localization, with case studies", Discrete Event Dynamic Systems, vol. 25, no. 1-2, pp. 203-226, Jun. 2015.

[3] K. Cai and W.M. Wonham, "Supervisor localization for large discrete-event systems – case study Production Cell", International Journal of Advanced Manufacturing Technology, vol. 50, no. 9-12, pp. 1189-1202, Oct. 2010.

[4] K. Cai and W.M. Wonham, "Supervisor localization of discrete-event systems based on state tree structures", IEEE Transactions on Automatic Control, vol. 59, no. 5, pp. 1329-1335, May 2014.

[5] R. Zhang, K. Cai, Y. Gan, Z. Wang, and W.M. Wonham, "Supervision localization of timed discrete-event systems", Automatica, vol. 49, no. 9, pp. 2786-2794, Sep. 2013.

[6] R. Zhang, K. Cai, and W.M. Wonham, "Supervisor localization of discrete-event systems under partial observation", Automatica, vol. 81, no. 7, pp. 142-147, Jul. 2017.

[7] R. Zhang, K. Cai, Y. Gan, Z. Wang, and W.M. Wonham, "Distributed supervisory control of discrete-event systems with communication delay", Discrete Event Dynamic Systems, vol. 26, no. 2, pp. 263-293, Jun. 2016.

[8] R. Zhang, K. Cai, Y. Gan, and W.M. Wonham, "Delay-robustness in distributed control of timed discrete-event systems based on supervisor localization", International Journal of Control, vol. 89, no. 10, pp. 2055-2072, Oct. 2016.

[9] R. Zhang and K. Cai, "Supervisor localization of discrete-event systems with infinite behavior", in Proceedings of Workshop on Discrete Event Systems, pp. 372-377, Sorrento, Italy, May 2018.

[10] K. Cai and W.M. Wonham, "Supervisor Localization: A Top-Down Approach to Distributed Control of Discrete-Event Systems", Lecture Notes in Control and Information Sciences, vol. 459, Springer, 2016.