

# On the Minimization of Communication in Networked Systems with a Central Station<sup>\*</sup>

Weilin Wang<sup>1</sup>, Stéphane Lafortune<sup>1</sup>, Feng Lin<sup>2</sup>

<sup>1</sup> Department of Electrical Engineering and Computer Science

University of Michigan

Ann Arbor, MI 48109

e-mail: {weilinw, stephane}@umich.edu

<sup>2</sup> Department of Electrical and Computer Engineering

Wayne State University

Detroit, MI 48202

e-mail: flin@ece.eng.wayne.edu

Revised in October 2007

**Abstract** The problem of minimizing communication in a distributed networked system is considered in a discrete-event formalism where the system is modeled as a finite-state automaton. The system consists of a central station and a set of  $N$  local agents, each observing a set of local events. The central station needs to know

---

\* The research of the first two authors is supported in part by ONR grant N0001-14-03-1-0232 and by NSF grants CCR-0325571 and ECS-0624821. The research of the third author is supported in part by NSF grant ECS-0624828.

exactly the state of the system, whereas local agents need to disambiguate certain pre-specified pairs of states for purposes of control or diagnosis. This requirement is achieved by communication, which occurs only between the central station and the local agents but not among the local agents. A communication policy is defined as a set of event occurrences to be communicated between the central station and the local agents. A communication policy is said to be minimal if any removal of communication of event occurrences will affect the correctness of the solution. Under an assumption on the absence of cycles (other than self-loops) in the system model, this paper presents an algorithm that computes a minimal communication policy in polynomial time in all parameters of the system. These results improve upon previous algorithms for solving minimum communication problems.

**Key words** distributed control, distributed diagnosis, minimum communication, discrete-event systems

## 1 Introduction

Distributed networked systems are becoming more and more prevalent due to advances in technologies. Examples can be found in wireless communications, transportation, telephony, and so forth. Nodes or agents in these systems need to communicate with each other and such communication is often “costly” due to one or more of the following considerations: power, bandwidth, security, or network topology. Because of such cost, there is a long-standing interest in minimizing communication among nodes in distributed networked systems. This problem can

take different forms for different modeling formalisms chosen to describe the system, different structural assumptions about the distribution of the sensors, the decentralization of information in the system, and the cost function to be minimized regarding communication. For example, the problem of minimizing the communication cost of a network management system consisting of a distributed hierarchy of cooperating intelligent agents is considered in [4]. In [17], minimum communication between distributed Bayesian networks is considered, where the problem of predicting the minimum expected communication cost is solved using Markov decision processes. More recently, [8] proposes an efficient communication strategy to improve response time.

Many communication problems in distributed systems can be investigated in the framework of discrete event systems. For example, for minimum communication in control, [24] considers a problem of decentralized control with a communication channel, where a necessary and sufficient condition is obtained for the existence of such a channel. In [11], states where supervisors do not know enough to solve the decentralized discrete event control problem are first identified. After the identification of such states, minimum communication is incorporated into the decentralized discrete event system so that the resulting augmented system results in supervisors with enough knowledge to make the correct control decisions. References [1–3] consider the problem of synthesizing several communicating supervisors, each with different information. Two fundamental difficulties associated with synthesis of optimal communication policies are discussed. Results are presented on the synthesis of communication policies for controllers that do not utilize

knowledge of the communication policy to affect their estimates. For minimum communication in diagnosis problems, a problem of failure diagnosis in discrete event systems with decentralized information is discussed in [7], where a coordinated decentralized architecture consisting of local sites communicating with a coordinator that is responsible for diagnosing the failures is proposed. A timed discrete event system model is used in [12] and decentralized failure diagnosis is proposed to treat failure diagnosis problems arising in communication networks. Reference [5] considers the trade-off between the speed of diagnosis and the cost of communication and computation. In the recent work [18, 19], a problem of turning on/off sensors dynamically for achieving diagnosability is considered for discrete event systems in the context of a general formulation based on information structures. A good survey on minimization of communication in the context of diagnosis or control of discrete-event systems where the information is decentralized can be found in [16]. It is generally believed that the problem of minimizing communication for diagnosis purposes is easier than minimizing communication for control purposes, due to the mutual coupling that occurs between the state estimation policy, the control policy, and the communication policy in decentralized control problems (see [3]).

Our approach considers minimum communication of *event occurrences* and it can be used in both control and diagnosis problems, along the line of work in [14] and [10]. This approach “separates” the communication problem from the diagnosis or control problem by assuming that the diagnosis and control policies are fixed and given *a priori*. Under this assumption, a minimum communication problem is

formulated and solved in [14], where a communication policy is defined as sets of event occurrences communicated among agents. The communication decisions of when, to whom, and which event occurrences to communicate are based on the agents' available information of the system trajectories. In other words, for any agent, the communication decisions for the occurrences of the same event after two different strings (trajectories) that look the same to that agent need to be consistent. This is captured in the notion of feasible communication policy in [14]. Also, the notion of implementability is introduced to restrict the communication policy to a subset of transitions of the automaton modeling the system. An algorithm that finds a minimal communication policy among two agents was proposed in [14]. The notion of minimality is a logical one: a communication policy is minimal if reducing one or more communications of event occurrences in the dynamic evolution of the system renders a correct solution incorrect (see [14] for precise definitions). However, the general strategy employed in the algorithm in [14] essentially proceeds by exhaustive enumeration of candidate solutions in a certain range.

A more general formulation of the minimum communication problem than that of [14] is proposed and investigated in [22] (see also Chapter II of [21]). It inherits some of the features of the problem treated in [14], including similar notions of feasibility and minimality of communication and an implementability condition to restrict solution space. However, the formulation in [21, 22] is more general than that in [14] in several respects: the number of agents, the communication structure, the goal of state disambiguation, and the algorithmic procedures. Based

on the assumptions made on the structure of the system regarding the absence of cycles other than self-loops, a solution procedure termed Algorithm MIN-COM-GEN is obtained to compute a minimal communication policy for each state of the system separately (one state per iteration). Algorithm MIN-COM-GEN of [22], together with the so-called “table technique” of [23] for testing potential solutions, computes a minimal communication policy in polynomial time complexity in the number of states of the system. However, Algorithm MIN-COM-GEN does not specify a specific strategy for the necessary computations at each iteration (or each state). The computational effort at each iteration may be substantial, as in the worst case it is exponential in the cardinality of a set whose size is the number of active events to the power of the number of agents in the system.

In this paper, we refine the problem formulation of [22]. We consider a distributed discrete event system with *one central station and  $N$  local agents*. Communications occur only between the central station and the local agents but not directly among the local agents. However, the central station may decide to relay the information received from local agent  $i$  to local agent  $j$ , which will be considered as two communications. The entire system is modeled by one automaton  $G$ . We assume that the events of  $G$  are observed by the central station and/or by at least one of the local agents. The central station must know the exact state of the system  $G$  at all times. Therefore, whenever an event that is observed by some local agents but not by the central station occurs and causes a change of the state in  $G$ , then one of these local agents must communicate that event occurrence to the central station. Our goal is to minimize the sets of communications from the

central station to each local agent  $i$  such that: (1) local agent  $i$  has no confusion in its communications to the central station; and (2) local agent  $i$  can distinguish certain pairs of states for its own control, diagnosis, or other purposes.

Examples given in [9, 22] show that agents observing fewer event occurrences may improve their knowledge of the system (as compared with when they observe more event occurrences) in the sense of that some previously indistinguishable states may become distinguishable; and this can happen even for feasible communication policies. Consequently, the fact that a set of event occurrences cannot be removed from a communication policy does not mean that a superset of it cannot be removed. This is referred to as the “lack of monotonicity” and it means that, counter-intuitively, more communications of event occurrences do not always mean better knowledge of the state. This brings up the following question: Does the refined problem formulation addressed in this paper suffer from the same problem of lack of monotonicity *within* each iteration of Algorithm MIN-COM-GEN that is used in [22] to solve the general problem? Unfortunately, the answer is yes. The problem of removing communications at each particular iteration (i.e., state) is still intricate. One of the main contributions of this paper is to present an algorithm that specializes Algorithm MIN-COM-GEN and achieves, at each iteration, polynomial complexity in the number of events in the active event set of the state under consideration. New techniques will be introduced to make this possible: a “dependency graph” and a special way of decomposing the post-language at each state. Moreover, we will show that the synthesis of the communication policies from the central station to the local agents can be decoupled and solved agent-by-agent.

Therefore, the end result is the first algorithm for solving a minimum communication problem that is polynomial in *all* the parameters: number of states, number of events, and number of agents.

This paper is organized as follows. Section 2 presents the details of the problem formulation. Then, in Section 3, an algorithm termed MIN-COM-SC is presented for the calculation of minimal communication policies for the  $N$  agents in the distributed system; the proof of the correctness (with respect to feasibility and local specifications) and minimality of Algorithm MIN-COM-SC is also presented in that section. In Section 4, a detailed procedure for removal of communications for a given state is presented based on a language decomposition analysis, together with the proof of its correctness. The overall complexity of Algorithm MIN-COM-SC is also derived in Section 4. Finally, an example is given in Section 5, followed by concluding remarks in Section 6. A preliminary version of some of the results in this paper appears in [20].

We assume basic knowledge of discrete-event systems and its common notations. The readers are directed to [6] for more complete introductory materials.

## 2 Description of Problem

### 2.1 System Model

We model a discrete event system as a deterministic finite-state automaton

$$G = (X, E, f, \Gamma, x_0) \quad (1)$$

where  $X$  is the finite set of states,  $E$  is the finite set of events,  $f : X \times E \rightarrow X$  is the transition function where  $f(x, e) = y$  means that there is a transition labeled by event  $e$  from state  $x$  to state  $y$ ,  $\Gamma$  is the active event function where  $\Gamma(x)$  is the set of all events  $e$  for which  $f(x, e)$  is defined (called the active event set of  $G$  at  $x$ ), and  $x_0$  is the initial state.

We use  $\mathcal{L}(G)$  to denote the language generated by  $G$ . We define the set of transitions  $TR(G)$  of  $G$  as

$$TR(G) := \{(x, e) \in X \times E : e \in \Gamma(x)\}. \quad (2)$$

The set of local agents is denoted by  $\mathcal{A} = \{1, 2, \dots, N\}$ . The set of events that can be observed by the central station is denoted by  $E_s$ . The set of events that can be observed by local agent  $i$  is denoted by  $E_i$ . We assume that every event is observed by the central station and/or by a local agent:

$$E = E_s \cup \left( \bigcup_{i \in \mathcal{A}} E_i \right). \quad (3)$$

For reasons that will become clear later on, we make the following assumption:

*Assumption (A1)* The graph of  $G$  has no cycles other than self-loops, i.e., for all  $x \in X$ , for all  $e \in E$  and  $t \in E^*$ ,

$$[f(x, et) \text{ is defined} \wedge f(x, e) \neq x] \Rightarrow f(x, et) \neq x. \quad (4)$$

From now on, we will assume that Assumption (A1) is satisfied.

## 2.2 Communication Model

Communications only occur from the central station to a local agent or from a local agent to the central station. The local agents do not communicate directly

among each other. Moreover, the central station does not broadcast or multicast to the local agents.

Communication from the central station to local agent  $i$  is described by the function

$$com_{si} : \mathcal{L}(G) \rightarrow 2^{E \setminus E_i}, \quad (5)$$

i.e., the central station can send out information from its own observations, as well as from the communications it receives from other local agents. Communication from local agent  $i$  to the central station is described by the function

$$com_{is} : \mathcal{L}(G) \rightarrow 2^{E_i \setminus E_s}, \quad (6)$$

i.e., local agent  $i$  can only send information from its own observations to the central station.

Given  $com_{is}, com_{si}, i \in \mathcal{A}$ , we use mutual induction to define the information mappings  $\theta_s : \mathcal{L}(G) \rightarrow E^*$  and  $\theta_i : \mathcal{L}(G) \rightarrow E^*, i \in \mathcal{A}$ , as follows. For the empty string  $\varepsilon$ ,

$$\theta_s(\varepsilon) = \varepsilon, \quad \theta_i(\varepsilon) = \varepsilon. \quad (7)$$

For all  $te \in \mathcal{L}(G)$  with  $e \in E$ ,

$$\theta_s(te) = \begin{cases} \theta_s(t)e & \text{if } e \in E_s \cup \bigcup_{i=1}^N com_{is}(t) \\ \theta_s(t) & \text{otherwise,} \end{cases} \quad (8)$$

$$\theta_i(te) = \begin{cases} \theta_i(t)e & \text{if } e \in E_i \cup com_{si}(t) \\ \theta_i(t) & \text{otherwise.} \end{cases} \quad (9)$$

The above definitions of  $com_{is}, com_{si}, i \in \mathcal{A}$ , are string-based. For implementation on automaton  $G$ , we need to define their state-based counterparts. For this

purpose, we introduce a so-called “implementability condition” that restricts the communication policies to the state structure of  $G$ . This condition is reminiscent of the implementability condition in [14].

*Implementability Condition on  $G$ :*

$$\begin{aligned} & (\forall i \in \mathcal{A}) (\forall s, s' \in \mathcal{L}(G)) f(x_0, s) = f(x_0, s') \\ & \Rightarrow [e \in \text{com}_{si}(s) \Leftrightarrow e \in \text{com}_{si}(s')] \end{aligned} \quad (10)$$

for the central station, and

$$\begin{aligned} & (\forall i \in \mathcal{A}) (\forall s, s' \in \mathcal{L}(G)) f(x_0, s) = f(x_0, s') \\ & \Rightarrow [e \in \text{com}_{is}(s) \Leftrightarrow e \in \text{com}_{is}(s')] \end{aligned} \quad (11)$$

for the local agents.

More specifically, under implementability, the transitions to be communicated from the central station to local agent  $i$  can be described by

$$\text{COM}_{si} \subseteq \{(x, e) \in \text{TR}(G) : e \in E \setminus E_i\}, \quad (12)$$

where  $(x, e) \in \text{COM}_{si}$  means that

$$((\forall s \in \mathcal{L}(G)) f(x_0, s) = x) e \in \text{com}_{si}(s). \quad (13)$$

Similarly, the transitions to be communicated from local agent  $i$  to the central station can be described by

$$\text{COM}_{is} \subseteq \{(x, e) \in \text{TR}(G) : e \in E_i \setminus E_s\}, \quad (14)$$

where  $(x, e) \in \text{COM}_{is}$  means that

$$((\forall s \in \mathcal{L}(G)) f(x_0, s) = x) e \in \text{com}_{is}(s). \quad (15)$$

We use the  $2 \times N$  matrix  $COM$  to record all communication policies among the central station and the local agents:

$$COM = \begin{bmatrix} COM_a \\ COM_s \end{bmatrix} = \begin{bmatrix} COM_{1s} & COM_{2s} & \cdots & COM_{Ns} \\ COM_{s1} & COM_{s2} & \cdots & COM_{sN} \end{bmatrix}.$$

The transitions observed by or communicated to the central station are described by the index function  $\mathcal{I}_s : TR(G) \rightarrow \{0, 1\}$  defined as

$$\mathcal{I}_s(x, e) := \begin{cases} 1 & \text{if } e \in E_s \text{ or } (\exists i \in \mathcal{A}) (x, e) \in COM_{is} \\ 0 & \text{otherwise} \end{cases}. \quad (16)$$

The transitions observed by or communicated to local agent  $i$  are described by the index function  $\mathcal{I}_i : TR(G) \rightarrow \{0, 1\}$  defined as

$$\mathcal{I}_i(x, e) := \begin{cases} 1 & \text{if } e \in E_i \text{ or } (x, e) \in COM_{si} \\ 0 & \text{otherwise} \end{cases}. \quad (17)$$

Moreover, it is required that any two sequences of events that are indistinguishable to a site (either the central station or a local agent) must be followed by the same communication decision for events whose occurrence is “known” to the site. Formally, from Equations (12) and (16), we define the notion of *feasibility* for  $COM_{si}$  as follows.

$$\begin{aligned} (\forall te, t'e \in \mathcal{L}(G)) \theta_s(te) = \theta_s(t'e) &= \theta_s(t)e = \theta_s(t')e \\ \Rightarrow [(f(x_0, t), e) \in COM_{si} \Leftrightarrow (f(x_0, t'), e) \in COM_{si}], \end{aligned} \quad (18)$$

and

$$(x, e) \in COM_{si} \Rightarrow e \in E_s \vee [(\exists j \in \mathcal{A}) e \in E_j \wedge (x, e) \in COM_{js}]. \quad (19)$$

From Equations (14) and (17), we define *feasibility* for  $COM_{is}$  as follows.

$$\begin{aligned} & (\forall te, t'e \in \mathcal{L}(G)) \theta_i(t) = \theta_i(t') \\ & \Rightarrow [(f(x_0, t), e) \in COM_{is} \Leftrightarrow (f(x_0, t'), e) \in COM_{is}]. \end{aligned} \quad (20)$$

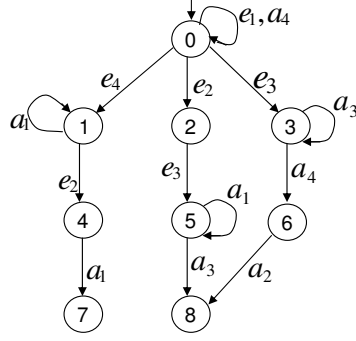
We note here that the feasibility of  $COM_{si}$  is different from that of  $COM_{is}$  because local agent  $i$  can only send out information about its own direct observations, as shown in Equation (14). We introduce the following terminology: (a)  $COM_a$  is feasible iff for all  $i \in \mathcal{A}$ ,  $COM_{is}$  is feasible, (b)  $COM_s$  is feasible iff for all  $i \in \mathcal{A}$ ,  $COM_{si}$  is feasible, and (c) matrix  $COM$  is feasible iff both  $COM_a$  and  $COM_s$  are feasible. Clearly, the notion of *feasibility* captured in Equations (18), (19), and (20) can be tested on  $G$ .

We use the following example to illustrate the meaning of feasibility.

*Example 1* Fig. 1 shows an automaton  $G$  that satisfies Assumption (A1). The states are  $X = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ . Suppose there are one central station and two local agents. Let  $E_s = \emptyset$ ,  $E_1 = \{a_1, a_2, a_3, a_4\}$ , and  $E_2 = \{e_1, e_2, e_3, e_4\}$ . Suppose agent 2 communicates everything to central station. Suppose the central station communicates everything except  $(0, e_2)$  to agent 1. Then, agent 1 cannot distinguish states 5 from 3 since  $\theta(e_2e_3) = e_3 = \theta(e_3)$ . By the feasibility requirement, this forces agent 1 to apply the same communication decisions to transitions  $(3, a_3)$  and  $(5, a_3)$ , that is, agent 1 has to communicate  $(3, a_3)$  if it wants to communicate  $(5, a_3)$ .

We will return to this example throughout the paper. □

Before we proceed, let us introduce the following notations: For given matrices of sets  $B = [b_{ij}]$  and  $B' = [b'_{ij}]$  for  $i = 1, \dots, l$  and  $j = 1, \dots, m$ ,  $B$  is a subset



**Fig. 1** An automaton  $G$  that satisfies Assumption (A1).

(superset) of  $B'$ , denoted by  $B \subseteq (\supseteq) B'$ , iff

$$[(\forall i, j) b_{ij} \subseteq (\supseteq) b'_{ij}]. \quad (21)$$

$B$  is a proper subset (superset) of  $B'$ , denoted by  $B \subset (\supset) B'$ , iff

$$[(\forall i, j) b_{ij} \subseteq (\supseteq) b'_{ij}] \wedge [(\exists i, j) b_{ij} \subset (\supset) b'_{ij}]. \quad (22)$$

For given matrices of sets  $B = [b_{ij}]$  and  $B' = [b'_{ij}]$ , we define

$$B \setminus B' = [b_{ij} \setminus b'_{ij}]. \quad (23)$$

### 2.3 Objective for Central Station

For our system, it is required that the central station must know the exact state  $G$  is in, that is, for all  $s, s' \in \mathcal{L}(G)$ ,

$$\theta_s(s) = \theta_s(s') \Rightarrow f(x_0, s) = f(x_0, s'). \quad (24)$$

More specifically, let

$$T_{s,spec} = \{(x, y) \in X \times X : x \neq y\}. \quad (25)$$

Then, Equation (24) is equivalent to

$$\theta_s(s) = \theta_s(s') \Rightarrow (f(x_0, s), f(x_0, s')) \notin T_{s,spec}. \quad (26)$$

Any communication from local agents  $i$ ,  $i \in \mathcal{A}$  to the central station that achieves this requirement must satisfy

$$e \in E \setminus E_s \wedge f(x, e) \neq x \Rightarrow (\exists i \in \mathcal{A}) (x, e) \in COM_{is}. \quad (27)$$

In other words, if the central station needs to know the occurrence of an unobservable event, then at least one local agent  $i$  must communicate this occurrence. A communication  $COM_{is}^*$  satisfying Equation (27) is minimal if no other communication  $COM_{is}$  that is strictly less than  $COM_{is}^*$  (i.e.,  $COM_{is} \subset COM_{is}^*$ ) and satisfies Equation (27). Clearly a communication  $COM_{is}^*$  satisfying Equation (27) is minimal if

$$(\forall i, j \in \mathcal{A}, i \neq j) COM_{is}^* \cap COM_{js}^* = \emptyset. \quad (28)$$

Different  $COM_{is}^*$  satisfying Equations (27) and (28) are incomparable minimal solutions. We denote the fact that this communication is minimal by using the superscript \*. Let

$$COM_a^* = [COM_{1s}^*, COM_{2s}^*, \dots, COM_{Ns}^*].$$

Assume that some  $COM_a^*$  satisfying Equations (27) and (28) has been picked for all local agents. Since the central station knows all transitions that cause a change of state in  $G$ , it can relay this information to any local agent if needed.

**Lemma 1** *If  $COM'_a \supseteq COM_a^*$ , then*

$$\theta'_s(s) = \theta'_s(s') \Rightarrow f(x_0, s) = f(x_0, s'). \quad (29)$$

Furthermore, any  $COM_s$  that satisfies Equation (19) is feasible.

*Proof* By construction of  $COM_a^*$  and the fact that  $COM'_a \supseteq COM_a^*$ , it is clear that the central station knows all transitions that cause a change of state in  $G$  (only self loops are not communicated<sup>1</sup>). Therefore, under  $COM'_a$ , for all  $s, s' \in \mathcal{L}(G)$ ,

$$\theta'_s(s) = \theta'_s(s') \Rightarrow f(x_0, s) = f(x_0, s'). \quad (30)$$

We have

$$\begin{aligned} (\forall se, s'e \in \mathcal{L}(G)) \theta'_s(se) &= \theta'_s(s'e) = \theta'_s(s)e = \theta'_s(s')e \\ &\Rightarrow (f(x_0, s), e) = (f(x_0, s'), e). \end{aligned} \quad (31)$$

From this equation, it follows that for any  $COM_s$ , Equation (18) is always satisfied.

Therefore, if  $COM_s$  satisfies Equation (19), then  $COM_s$  is feasible.  $\square$

#### 2.4 Objective for Local Agents

We do not require that local agents know the exact state of  $G$ . Rather, it is only required that local agents can disambiguate certain pairs of states of  $G$  for their own control, diagnosis, or other purposes. Formally, for local agent  $i$ , we specify a relation  $T_{i,spec} \subseteq X \times X$  such that no state pair  $(x, y) \in T_{i,spec}$  should be indistinguishable from the viewpoint of local agent  $i$ , i.e., for all  $s, s' \in \mathcal{L}(G)$ ,

$$\theta_i(s) = \theta_i(s') \Rightarrow (f(x_0, s), f(x_0, s')) \notin T_{i,spec}. \quad (32)$$

The central station must communicate sufficient information to local agent  $i$  so that the above requirement is satisfied.

---

<sup>1</sup> It can be seen by building an observer  $G_{obs}$  that no state in  $G_{obs}$  contains more than one state of  $G$ .

Furthermore, since local agent  $i$  needs to communicate  $COM_{is}$  to the central station, the central station must communicate sufficient information to local agent  $i$  so that  $COM_{is}$  is feasible. To satisfy this feasibility requirement, we define a relation  $T_{i,feas} \subseteq X \times X$  describing the state pairs that cannot be confused for maintaining feasibility as follows. For each event  $e \in E_i \setminus E_s$  that may need to be communicated to the central station, let  $X_s(e)$  be the set of states where  $e$  needs to be communicated

$$X_s(e) = \{x \in X : (x, e) \in COM_{is}\} \quad (33)$$

and  $X_{ns}(e)$  be the set of states where  $e$  is not communicated

$$X_{ns}(e) = \{x \in X : (x, e) \in TR(G) \setminus COM_{is}\}. \quad (34)$$

In order for local agent  $i$  to communicate minimally to the central station (i.e., we do not want to add more communications to  $COM_{is}^*$ ), local agent  $i$  must not confuse states in  $X_s(e)$  with states in  $X_{ns}(e)$ . Therefore, we define

$$T_{i,feas} = \{(x, y) \in X \times X : (\exists e \in E_i \setminus E_s) x \in X_s(e) \wedge y \in X_{ns}(e) \vee (x \in X_{ns}(e) \wedge y \in X_s(e))\}. \quad (35)$$

**Lemma 2**  $COM_{is}^*$  is feasible if and only if for all  $s, s' \in \mathcal{L}(G)$ ,

$$\theta_i(s) = \theta_i(s') \Rightarrow (f(x_0, s), f(x_0, s')) \notin T_{i,feas}. \quad (36)$$

*Proof* The proof follows directly from the definition of  $T_{i,feas}$  and Equation (20).  $\square$

We are now ready to formally state the problem to be solved.

### 2.5 Problem Statement

Given a distributed system  $G$  with one central station and  $N$  local agents, and a set of local specifications,  $T_{i,spec}$ , one for each local agent. Assume that  $G$  satisfies (A1).

Given a vector of communications

$$COM_a^* = [COM_{1s}^*, \dots, COM_{Ns}^*] \quad (37)$$

that satisfies Equations (27) and (28).

**Goal:** Find a communication set

$$COM_{si}^* \subseteq \{(x, e) \in TR(G) : e \in E \setminus E_i\} \quad (38)$$

for each  $i \in \mathcal{A}$  such that

C1.  $COM_{is}^*$  is feasible: for all  $s, s' \in \mathcal{L}(G)$ ,

$$\theta_i^*(s) = \theta_i^*(s') \Rightarrow (f(x_0, s), f(x_0, s')) \notin T_{i,feas}$$

where  $\theta_i^*$  is the information mapping corresponding to  $COM_{si}^*$ .

C2. The local specification  $T_{i,spec}$  is satisfied: for all  $s, s' \in \mathcal{L}(G)$ ,

$$\theta_i^*(s) = \theta_i^*(s') \Rightarrow (f(x_0, s), f(x_0, s')) \notin T_{i,spec}.$$

C3. There is no other  $COM_{si}$  that is strictly less than  $COM_{si}^*$  ( $COM_{si} \subset COM_{si}^*$ ) and satisfies (C1) and (C2).

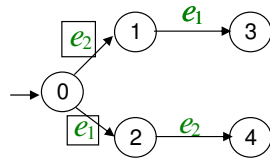
To solve the problem for local agent  $i$ , can we just remove transitions in  $\{(x, e) \in TR(G) : e \in E \setminus E_i\}$  one-by-one until we get a minimum set satisfying conditions

(C1) and (C2)? Unfortunately, the answer is “no”: the problem is so intricate that removing transitions one by one will not work as shown in following example.

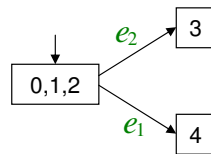
*Example 2* Consider a central station and a single local agent with system model given in Fig. 2. Let  $E_s = \{e_1, e_2\}$  and  $E_1 = \emptyset$ . Let

$$T_{1,spec} = \{(1,3), (2,4), (2,3), (1,4)\}.$$

It is obvious that an individual removal of communication  $(0, e_1)$  causes confusion of pair of states  $(1,4) \in T_{1,spec}$ , and an individual removal of communication  $(0, e_2)$  causes confusion of pair of states  $(2,3) \in T_{1,spec}$ . But, if we remove both  $(0, e_1)$  and  $(0, e_2)$  together, then  $T_{1,spec}$  is satisfied! This can be seen from the corresponding observer for local agent 1 in Fig. 3.



**Fig. 2** Automaton  $G$  for Example 2.



**Fig. 3** Observer for local agent 1 after central station removed communications of boxed transitions in Fig. 2.

The above example indicates that when removing transitions, we need to consider *sets* of transitions. A naive approach is to consider all possible subsets of transitions

$$W \subseteq \{(x, e) \in TR(G) : e \in E \setminus E_i\}. \quad (39)$$

For each  $W$ , check if the resulting  $\theta_i$  satisfies conditions (C1) and (C2), i.e., for all  $s, s' \in \mathcal{L}(G)$ ,

$$\theta_i(s) = \theta_i(s') \Rightarrow (f(x_0, s), f(x_0, s')) \notin T_{i,spec} \cup T_{i,feas}. \quad (40)$$

The number of iterations for such an approach is the same as the number of all possible communication policies. Since the system has  $N$  local agents, there are  $N$   $COM_{si}$  that need to be considered in  $COM$ . For each  $COM_{si}$ , the central station needs to make decisions for each transition in  $TR(G)$ , whose cardinality is in the order of  $\|X \times E\|$ . Considering all possible combinations, the set of all possible  $COM$ 's is in the order of  $2^{N\|X \times E\|}$ . To test each combination using the table technique in [23], the worst-case complexity is of order  $\mathcal{O}(\|E\|\|X\|^2)$ . Overall, the worst-case complexity of such an approach reaches  $\mathcal{O}(\|E\|\|X\|^2 2^{n\|E\|\|X\|})$ . This is clearly impractical and not scalable.

Our approach to solve the minimum communication problem in a computationally tractable and scalable manner consists of two steps, presented sequentially in Sections 3 and 4. In Section 3, we apply the results in [21, 22] to decompose our problem and consider the transitions of each *state* of  $G$  *separately*. This gives an approach that achieves polynomial complexity with respect to the cardinality of state space of  $G$ . Then, in Section 4, we show how to reduce the computational

effort for the *transitions* at *each* state, resulting in polynomial complexity in the number of such transitions at the state.

### 3 Problem Solution - Part I

#### 3.1 Algorithm MIN-COM-CS

In this section, we present a new algorithm (termed Algorithm MIN-COM-CS) for finding a minimal communication  $COM_{si}^*$ , for all  $i \in \mathcal{A}$ , from the central station to local agent  $i$ , for a given minimal communication from the local agents to the central station  $COM_a^*$  as specified in the previous section. In the next section, we will show how to efficiently implement the key Step 3.2 of MIN-COM-CS.

#### ALGORITHM MIN-COM-CS

INPUT:  $COM_a^*$  and  $G$ .

**Step 1:** Initialization.

1.1 Define  $\mathcal{C} : X \rightarrow \{0, 1\}$  and set  $\mathcal{C}(x) = 0, \forall x \in X$ .

1.2 We start with both the central station and the local agents communicating all transitions, i.e.,  $COM$  is initially given by

$$COM_{si} = \{(x, e) \in TR(G)\},$$

$$COM_{is} = \{(x, e) \in TR(G) : e \in E_i \setminus E_s\},$$

for all  $i \in \mathcal{A}$ .

**Step 2:** Find a state  $x_l \in X$  that satisfies the equations:  $\mathcal{C}(x_l) = 0$  and

$$(\forall e \in \Gamma(x_l)) (f(x_l, e) = x_l) \vee (f(x_l, e) = x_k \Rightarrow \mathcal{C}(x_k) = 1).$$

**Step 3:** Remove communications of transitions at state  $x_l$  as follows.

**3.1** For each local agent  $i \in \mathcal{A}$ , set

$$RT_{is}^*(x_l) = \{(x_l, e) \in TR(G) : (x_l, e) \notin COM_{is}^*\} \text{ and}$$

$$COM_{is} \leftarrow COM_{is} \setminus RT_{is}^*(x_l).$$

(This removes all communications corresponding to self-loops at state  $x_l$  from  $COM_{is}$ .)

For each  $i \in \mathcal{A}$ , set

$$RT_{si}^{sl}(x_l) = \{(x_l, e) : f(x_l, e) = x_l\} \text{ and}$$

$$COM_{si} \leftarrow COM_{si} \setminus RT_{si}^{sl}(x_l).$$

(This removes all communications corresponding to self-loops at state  $x_l$  from  $COM_{si}$ .)

**3.2** For each  $i \in \mathcal{A}$ , consider the set of potentially removable transitions at  $x_l$ , i.e.,

$$PRT_s(x_l) = \{(x_l, e) \in TR(G) : (e \in \Gamma(x_l) \setminus E_i) f(x_l, e) \neq x_l\}. \quad (41)$$

Find  $RT_{si}^*(x_l) \subseteq PRT_s(x_l)$  for each  $i \in \mathcal{A}$ , such that under

$$COM_{si} \leftarrow COM_{si} \setminus RT_{si}^*(x_l) \quad (42)$$

the following conditions (I), (II) and (III) are satisfied:

I.  $COM_{is}$  is feasible.

II. The local specification  $T_{i,spec}$  is satisfied:

$$(\forall s, s' \in \mathcal{L}(G))[\theta_i(s) = \theta_i(s') \Rightarrow (f(x_0, s), f(x_0, s')) \notin T_{i,spec}]. \quad (43)$$

III. There is no other  $RT_{si}^{**}(x_l) \subseteq PRT_s(x_l)$  such that  $RT_{si}^{**}(x_l)$  satisfies (I) and (II)

with  $RT_{si}^*(x_l) \subset RT_{si}^{**}(x_l)$ .

**Step 4:** Set  $\mathcal{C}(x_l) = 1$ .

**Step 5:** Repeat Steps 2 to 4 until  $\mathcal{C}(x) = 1$  for all  $x \in X$ .

OUTPUT: The minimal communication policy  $COM_{si}^*$ ,  $i \in \mathcal{A}$ . ■

Algorithm MIN-COM-CS works as follows. After the necessary initialization in Step 1, each state of  $G$  is examined in an order essentially determined by a topological sort of the graph as specified by Step 2. Assumption (A1) makes this possible. The core of the algorithm is formed in Step 3 where a matrix of maximal removable communications at state  $x_l$  is calculated. In the context of the general problem formulation studied in [22], the method employed for finding such matrix of maximal removable communications is not specified. Certainly, exhaustive search will work. However, for the special problem considered in this paper, with a central station communicating with a set of local agents, there is an efficient way of implementing Step 3 (specifically, Step 3.2) as will discussed in the next section; it is called ALGORITHM CS-3.2. This is one of the main contributions of this paper.

### 3.2 Notation

For  $x \in X$  and  $V \subseteq TR(G)$ , we construct a subautomaton  $\hat{G}(x, V)$  of  $G$ , denoted by  $\hat{G}(x, V) \sqsubseteq G$ , by the following procedure. First, the initial state of  $\hat{G}(x, V)$  is state  $x$ . Then we define the transition function  $\hat{f}$  of  $\hat{G}(x, V)$  as

$$\hat{f}(x, e) := \begin{cases} f(x, e) & \text{if } (x, e) \in V; \\ \text{undefined} & \text{otherwise.} \end{cases} \quad (44)$$

$\hat{G}(x, V)$  is the accessible part of the resulting automaton. We use the notation  $\mathcal{L}(\hat{G}(x, V))$  for the language generated by  $\hat{G}(x, V)$  and the notation  $\hat{X}(x, V)$  for the set of states of  $\hat{G}(x, V)$ . Furthermore, denote  $\hat{G}(x) = \hat{G}(x, TR(G))$  and  $\hat{X}(x) = \hat{X}(x, TR(G))$ .

Let  $COM$  be a communication policy defined on  $G$ ; then  $COM$  on  $H \sqsubset G$  is the communication policy  $COM$  restricted to the set of transitions of  $H$ .

### 3.3 Properties of Algorithm MIN-COM-CS

We now show that Algorithm MIN-COM-CS is correct in the sense of it returns a solution to the problem formulated in Section 2.5.

Algorithm MIN-COM-CS is an adaptation and refinement of Algorithm MIN-COM-GEN of [21, 22] to the specific problem formulation considered herein. This enables us to solve our problem in polynomial complexity with respect to the cardinality of state space of the system. We justify Algorithm MIN-COM-CS as follows.

Lemma 3 shows that Step 3.1 of Algorithm MIN-COM-CS does not affect the existence of a solution for Step 3.2.

**Lemma 3** *After Step 3.1 of Algorithm MIN-COM-CS for removing communications of state  $x_i$ , the resulting COM satisfies*

1.  $COM_s$  and  $COM_a$  are feasible, and
2.  $T_{s,spec}$  and  $T_{i,spec}$ ,  $i = 1, \dots, N$ , are satisfied.

*Proof* Since Step 3.1 removes communications of self-loops at state  $x_i$  from both  $COM_s$  and  $COM_a$ , it can be verified that Equation (19) is satisfied after Step 3.1.

Furthermore, since  $COM_a \supseteq COM_a^*$ , by Lemma 1,  $COM_s$  is feasible and  $T_{s,spec}$  is satisfied.

If  $COM_{i_s}$  is not feasible or  $T_{i,spec}$  is not satisfied for some  $i \in \mathcal{A}$ , then after removing communication of self-loop of state  $x_i$  in Equation (41), we have, respectively,

$$(\exists s, t \in \mathcal{L}(\hat{G})) \theta_i(s) = \theta_i(t) \wedge (f(x_i, t), f(x_i, s)) \in T_{i,spec}, \quad (45)$$

or

$$(\exists s, t \in \mathcal{L}(\hat{G})) \theta_i(s) = \theta_i(t) \wedge (f(x_i, t), f(x_i, s)) \in T_{i,feas}. \quad (46)$$

In either case, we can find

$$e \in \Gamma(x_i) \text{ with } f(x_i, e) \neq x_i \quad (47)$$

and

$$s', t' \in \Gamma^*(x_i) \text{ with } f(x_i, s') = f(x_i, t') = x_i, \quad (48)$$

so that the traces  $s$  and  $t$  can be expressed as

$$s = s' e s'' \text{ and } t = t' e t''. \quad (49)$$

Since at this stage of the algorithm, for all  $(x_i, e)$  with

$$f(x_i, e) \neq x_i, \quad (50)$$

$\mathcal{I}_i(x_i, e) = 1$ , by the definition of information mapping, we have

$$\theta_i(s'') = \theta_i(t''). \quad (51)$$

Let  $f(x_l, e) = x_k$ . We have  $s'', t'' \in \mathcal{L}(\hat{G}(x_k))$ , with

$$(f(x_k, s''), f(x_k, t'')) \in T_{i,spec}, \quad (52)$$

or

$$(f(x_k, s''), f(x_k, t'')) \in T_{i,feas}. \quad (53)$$

But  $\theta_i(s'')$  and  $\theta_i(t'')$  have not changed since we removed the communications in  $RT^*(x_k)$ . It implies the removal of communications with respect to  $RT^*(x_k)$  violates (I) or (II), which is a contradiction.  $\square$

The following important proposition states that when Algorithm MIN-COM-CS checks, in Step 3.2, whether some communication can be further removed or not at state  $x_l$ , it is equivalent to checking the *subautomaton starting from*  $x_l$ .

**Proposition 1** *In Step 3.2 of the Algorithm MIN-COM-CS, suppose we are given  $RT_{si}(x_l) \supseteq RT_{si}^{sl}(x_l)$ . Let*

$$COM_{si} \leftarrow COM_{si} \setminus RT_{si}(x_l) \quad (54)$$

*be the corresponding communication. Then conditions*

- I.  $COM_{is}$  is feasible, and*
- II. no  $(x, y) \in T_{i,spec}$  is indistinguishable*

*are satisfied for  $G$  if and only if they are satisfied for  $\hat{G}(x_l)$ .*

*Proof* The initialization of the communication policy of Algorithm MIN-COM-CS given by Step 1 is feasible and, under this policy, each agent will have a perfect observation of the system. Furthermore, Lemma 3 ensures that we can find

a solution for Step 3 of Algorithm MIN-COM-CS. Therefore, Algorithm MIN-COM-CS satisfies all the conditions in the proof of Proposition 1 for Algorithm MIN-COM-GEN in [22]. Hence, the proof of this proposition resembles the proof of Proposition 1 in [22], which can be briefly summarized as follows.

In view of Assumption (A1), Algorithm MIN-COM-CS can proceed state by state in  $G$  according to an order of topological sort. From Algorithm MIN-COM-CS, when a state  $x_l$  is selected for removal of communications, the central station and agents communicate all transitions for all states which are “upstream” of  $x_l$ . Therefore, each agent as well as the central station still has perfect observation for all strings that arrive at  $x_l$  for the first time and end at that state. Thus, two strings that become indistinguishable because of the removal of communications at state  $x_l$  at that time have to have the same prefix which reaches state  $x_l$ . Because both feasibility and  $T_{i,spec}$  are defined based on indistinguishable strings, their violation, if any, must occur in  $\hat{G}(x_l)$ . Therefore, conditions I and II are satisfied for  $G$  if and only if they are satisfied for  $\hat{G}(x_l)$ .  $\square$

We will use Proposition 1 to further save computational efforts for Step 3.2 in the next section.

Theorem 1 states the correctness of Algorithm MIN-COM-CS.

**Theorem 1** *Algorithm MIN-COM-CS gives a solution to the minimum-communication problem stated in Section 2.5.*

*Proof* The proof resembles the proof of Theorem 1 in [22] and can be briefly summarized as follows. The topological sort ensures that Algorithm MIN-COM-CS checks and removes, if possible, communications for each state of  $G$  exactly

once. Lemma 3 ensures that we can find a solution for Step 3 of Algorithm MIN-COM-CS. If there is a solution which is strictly better than the solution of Algorithm MIN-COM-CS, then we can find a set of states for which the corresponding communications are strictly less than those obtained by Algorithm MIN-COM-CS. Because of Assumption (A1), for such set of states, we can find at least one state  $x$  whose communications are strictly less than those obtained by Algorithm MIN-COM-CS, but not strictly less for any of its reachable states. State  $x$  and its reachable part of  $G$  form the subautomaton  $\hat{G}(x)$ . By Proposition 1, there is a contradiction to the minimality of Step 3.2 of Algorithm MIN-COM-CS for this state  $x$ .  $\square$

Theorem 1 also implies that in order to solve the problem in Section 2.5, we only need to iterate every state in  $G$  once. At each iteration (from Step 2 to Step 4), we find a maximal set of removable communications at that state. Furthermore, the problem can be solved *agent by agent*, irrespective of (i) the number of local agents, (ii) the events directly observed by each agent, and (iii) the local specification  $T_{i,spec}$  for each agent. This decomposition of the problem leads to significant computational savings.

*Example 3* Consider the automaton  $G$  of Example 1, shown in Fig. 1; each event can be directly observed by at least one of the local agents or by the central station. We can iterate of Step 2 to Step 4 of Algorithm MIN-COM-CS for all states of  $G$  in an order of topological sort, such as 8, 7, 6, 5, 4, 3, 2, 1, 0 or, alternatively, 7, 4, 8, 1, 5, 6, 2, 3, 0. The problem of Section 2.5 can be solved by removing communications for one state per iteration and this can be done agent by agent  $\square$

We will revisit this example in Section 5 after we present our solution procedure for Step 3.2 of Algorithm MIN-COM-CS in the next section.

Unlike Algorithm MIN-COM-GEN of [22], where all local agents must be considered together in Step 3, in Algorithm MIN-COM-CS, each local agent can be considered separately in Step 3. Hence, with the results presented so far, we can solve the problem of minimization of communication for acyclic systems with a central station in a computation effort of  $\mathcal{O}(n\|E\|\|X\|^22^{\|E\|})$ . The only computational hurdle remaining is that our current solution is still exponential in the cardinality of the event set  $E$ . We resolve this issue in the next section.

## 4 Problem Solution - Part II

### 4.1 Language Decomposition

A “trivial” implementation of Step 3.2 of Algorithm MIN-COM-CS is to test all subsets of  $PRT_s(x_l)$ . But this requires an exponential number of tests with respect to the cardinality of  $PRT_s(x_l)$ . We want to develop a procedure with polynomial number of iterations with respect to the cardinality of both  $X$  and  $PRT_s(x_l)$ .

From Section 3.3, we know that given state  $x_l$ , finding  $RT_{si}^*(x_l) \subseteq PRT(x_l)$  for  $G$  is equivalent to finding  $RT_{si}^*(x_l)$  for  $\hat{G} = \hat{G}(x_l)$ . Furthermore, besides  $T_{i,spec}$ , the choice of  $RT_{si}^*$  in Step 3.2 of Algorithm MIN-COM-CS only depends on the given  $RT_{is}^*(x_l)$ . Therefore, we can solve  $RT_{si}^*(x_l)$  for each local agent  $i \in \mathcal{A}$  separately in  $\hat{G}$ . This decomposition of the problem is important and very useful. The subsequent discussion considers a fixed agent labeled by  $i$ .

Because of the problem of lack of monotonicity, before we can efficiently remove the communications at state  $x_l$ , i.e., calculate  $RT_{si}^*(x_l)$ , we analyze the relations among transitions defined at state  $x_l$  for an arbitrary fixed  $RT_{si}(x_l)$ . To do so, we first partition event set  $E$  according to its observability properties and to the communication policy of corresponding transitions at state  $x_l$  as follows:

- Observable or communicated non self-loop:

$$V_1 = \{e \in \Gamma(x_l) : f(x_l, e) \neq x_l \wedge \mathcal{I}_i(x_l, e) = 1\}, \quad (55)$$

- Unobservable and non communicated non self-loop:

$$V_2 = \{e \in \Gamma(x_l) : f(x_l, e) \neq x_l \wedge \mathcal{I}_i(x_l, e) = 0\}, \quad (56)$$

- Non self-loop:

$$V_{ns} = \{e \in \Gamma(x_l) : f(x_l, e) \neq x_l\} \quad (57)$$

- Observable or communicated self-loop<sup>2</sup>:

$$V_3 = \{e \in \Gamma(x_l) : f(x_l, e) = x_l \wedge e \in E_i\} \quad (58)$$

- Unobservable and non communicated events:

$$V_4 = E \setminus (V_1 \cup V_3). \quad (59)$$

Based on the above partition of event set  $E$ , we wish to decompose the language  $\mathcal{L}(\hat{G})$  in order to obtain an efficient solution procedure for Step 3.2. This decomposition has four steps.

---

<sup>2</sup> Because the communications for self-loops have already been removed by Step 3.1 of Algorithm MIN-COM-CS, we only need to consider the set of events defined as self-loops with  $e \in E_i$  here.

1. Define the set of strings that self-loop at state  $x_l$ :

$$L_1 = \{s \in \mathcal{L}(\hat{G}) : f(x_l, s) = x_l\}. \quad (60)$$

2. Define

$$TR_1 = \{(x, e') \in TR(\hat{G}) : \mathcal{I}_i(x, e') = 0 \vee e' \in V_3\}. \quad (61)$$

It corresponds to unobservable and non-communicated transitions or to transitions with events in observable self-loops. Define  $L_2(e)$  the set of strings which leave  $x_l$  with an event  $e \in V_2$  and only use transitions in  $TR_1$  afterwards,

$$L_2(e) = \{s \in \mathcal{L}(\hat{G}) : s \in L_1 e \mathcal{L}(\hat{G}(f(x_l, e), TR_1))\}. \quad (62)$$

Define  $L_2 = \bigcup_{e \in V_2} L_2(e)$ . Hence, any  $s \in L_2$  will cause confusion between states  $x_l$  and  $f(x_l, s)$ .

3. For  $e \in V_1$ , define

$$L_3(e) = \{s \in \mathcal{L}(\hat{G}) : s \in L_1 e \mathcal{L}(\hat{G}(f(x_l, e)))\}. \quad (63)$$

$$L_3 = \bigcup_{e \in V_1} L_3(e). \quad (64)$$

Hence,  $L_3$  is the set of strings that leave state  $x_l$  by observable or communicated events.

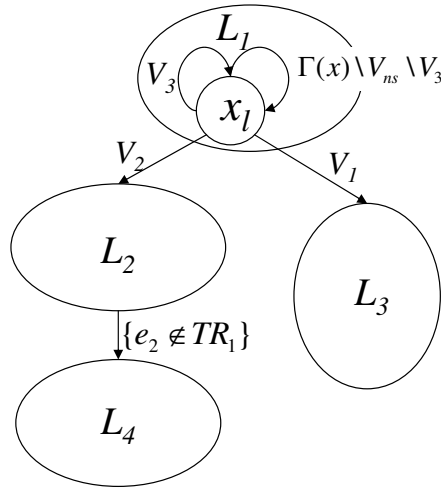
4. For  $e_1 \in V_2$ ,  $e_2 \in V_1 \cup V_4$ ,

$$L_4(e_1, e_2) = \{s \in \mathcal{L}(\hat{G}) : \exists t e_2 \in \bar{s}, t \in L_2(e_1), \mathcal{I}(f(x_l, t), e_2) = 1\}, \quad (65)$$

where  $\bar{s}$  is the prefix closure of string  $s$ .

$$L_4 = \bigcup_{e_1 \in V_2} \bigcup_{e_2 \in V_1 \cup V_4} L_4(e_1, e_2). \quad (66)$$

In words,  $L_4$  is the set of strings which start with a string in  $\cup_{e_1 \in V_2} L_2(e_1)$  followed by observable transition  $(f(x_l, t), e_2)$ , then followed by any suffix. Note that the label  $e_2$  of the transition  $(f(x_l, t), e_2)$  which leaves language  $\cup_{e_1 \in V_2} L_2(e_1)$  is not the label of a transition defined at  $x_l$  nor is it the label of an observable self-loop at  $x_l$ . Hence, this label  $e_2$  is in the set  $V_1 \cup V_4$  and after  $(f(x_l, t), e_2)$  happens, we know that the trace has left language  $\cup_{e_1 \in V_2} L_2(e_1)$ .



**Fig. 4** Decomposition of  $\mathcal{L}(\hat{G})$  as  $L_1$ ,  $L_2$ ,  $L_3$ , and  $L_4$ .

Using 1.–4. above, as shown in Fig. 4, language  $\mathcal{L}(\hat{G})$  can be decomposed as follows:

$$\mathcal{L}(\hat{G}) = L_1 \cup (\cup_{e \in V_2} L_2(e)) \cup (\cup_{e \in V_1} L_3(e)) \cup (\cup_{e_1 \in V_2} \cup_{e_2 \in V_1 \cup V_4} L_4(e_1, e_2)). \quad (67)$$

We illustrate language decomposition by the following example.

*Example 4* Consider the language generated by  $G$  in Fig. 1 of Example 1. We decompose  $\mathcal{L}(G)$  from the point of view of agent 1. Suppose that only  $(0, e_3), (2, e_3)$

are unobservable to agent 1, i.e., agent 1 does not know the occurrence of  $e_3$  after states 0 and 2. Let  $\bar{L}$  be the prefix-closure of language  $L$ , let  $x_l$  be state 0. Then,  $V_1 = \{e_2, e_4\}$ ,  $V_2 = \{e_3\}$ ,  $V_3 = \{a_4\}$ , and  $V_4 = \{a_1, a_2, a_3, e_1, e_3\}$ . In the notation of regular expressions,

$$\begin{aligned}
L_1 &= (e_1 + a_4)^*, \\
L_2 &= L_2(e_3) = (e_1 + a_4)^* e_3 \bar{a}_4, \\
L_3 &= L_3(e_2) + L_3(e_4) = (e_1 + a_4)^* (e_2 e_3 \bar{a}_1 a_3 + e_4 \bar{a}_1 e_2 a_1), \text{ and} \\
L_4 &= (e_1 + a_4)^* e_3 (a_3 \bar{a}_3^* a_4 a_2 + a_4 a_2).
\end{aligned} \tag{68}$$

It can be verified that  $\mathcal{L}(G) = L_1 \cup L_2 \cup L_3 \cup L_4$ .  $\square$

To examine (I) and (II) for a given  $RT(x_l)$ , it is sufficient to examine all pairs of traces  $(s, t) \in \mathcal{L}(\hat{G}) \times \mathcal{L}(\hat{G})$ . Building on the above decomposition of  $\mathcal{L}(\hat{G})$ , we partition  $\mathcal{L}(\hat{G}) \times \mathcal{L}(\hat{G})$  into five sets of pairs of traces as  $P_i, i = 1, \dots, 5$ :

$$\begin{aligned}
P_1 &= [L_1 \cup (\cup_{e \in V_2} L_2(e)) \cup (\cup_{e_1 \in V_2} \cup_{e_2 \in V_1 \cup V_4} L_4(e_1, e_2))] \\
&\quad \times [L_1 \cup (\cup_{e \in V_2} L_2(e)) \cup (\cup_{e_1 \in V_2} \cup_{e_2 \in V_1 \cup V_4} L_4(e_1, e_2))],
\end{aligned} \tag{69}$$

$$P_2 = [\cup_{e \in V_1} L_3(e)] \times [\cup_{e \in V_1} L_3(e)], \tag{70}$$

$$\begin{aligned}
P_3 &= [L_1 \cup (\cup_{e \in V_2} L_2(e))] \times [\cup_{e \in V_1} L_3(e)] \\
&\quad \cup [\cup_{e \in V_1} L_3(e)] \times [L_1 \cup (\cup_{e \in V_2} L_2(e))],
\end{aligned} \tag{71}$$

$$\begin{aligned}
P_4 &= [\cup_{e_1 \in V_2} \cup_{e_2 \in V_1 \cup V_4} \cup_{e_3 \in V_1, e_3 \neq e_2} L_4(e_1, e_2) \times L_3(e_3)] \\
&\quad \cup [\cup_{e_1 \in V_2} \cup_{e_2 \in V_1 \cup V_4} \cup_{e_3 \in V_1, e_3 \neq e_2} L_3(e_3) \times L_4(e_1, e_2)],
\end{aligned} \tag{72}$$

$$\begin{aligned}
P_5 &= [\cup_{e_1 \in V_2} \cup_{e_2 \in V_1} L_4(e_1, e_2) \times L_3(e_2)] \\
&\quad \cup [\cup_{e_1 \in V_2} \cup_{e_2 \in V_1} L_3(e_2) \times L_4(e_1, e_2)].
\end{aligned} \tag{73}$$

Note that  $P_4$  and  $P_5$  correspond to the complementary case where  $e_2 = e_3$ . By Equation (67), we can write

$$\mathcal{L}(\hat{G}) \times \mathcal{L}(\hat{G}) = \cup_{i=1}^5 P_i. \quad (74)$$

From now on, the mention of “ $P_i$  test” means testing all pairs of traces  $(s, t) \in P_i$  for  $(f(x_I, s), f(x_I, t)) \in T_{i,spec} \cup T_{i,feas}$ , that is, testing (I) and (II).

We now describe how to test (I) and (II) over  $\mathcal{L}(\hat{G}) \times \mathcal{L}(\hat{G})$  by testing these conditions over the sets  $P_i$ ,  $i = 1, \dots, 5$ . It turns out that the cases  $P_2, P_3$ , and  $P_4$  always satisfy (I) and (II):

- For  $(s, t) \in P_2$ , we construct  $G'$ , a subautomaton of  $\hat{G}$ , as

$$G' = \hat{G}(x_I, TR(\hat{G}) \setminus \{(x_I, e) : e \in V_2\})$$

with state space  $X'$ . We have  $G' \sqsubseteq \hat{G}$ . Because for all  $(x, e) \in TR(G')$ ,  $\mathcal{J}_i(x, e)$  does not change before and after removing communications in  $RT_{st}^*(x_I)$  for all  $i \in \mathcal{A}$  in Step 3.2, the checking of these pairs of traces will always pass.

- For  $(s, t) \in P_3$ , all traces in  $L_3$  have at least one observed transition corresponding to an event from  $V_1$ . But for all  $t \in L_1 \cup (\cup_{e \in V_2} L_3(e))$ , no such transition is available. We can conclude that  $\theta_i(s) \neq \theta_i(t)$ .
- For  $(s, t) \in P_4$ , the first observed events  $e_2, e_3 \in V_1 \cup V_4$  contained in  $s$  and  $t$  are different. Hence, we also have  $\theta_i(s) \neq \theta_i(t)$ .

Consequently, for testing (I) and (II) in Step 3.2 of Algorithm MIN-COM-CS for given  $RT'(x_I) \subseteq PRT(x_I)$ , we only need to test  $P_1$  and  $P_5$ . We use this fact to design the following algorithm for Step 3.2 of Algorithm MIN-COM-CS.

#### 4.2 Algorithm for Step 3.2 of MIN-COM-CS

Here we are concerned with communications from the central station to local agent  $i$ . The goal is to find  $RT_{si}^*(x_l)$  for Step 3.2 of Algorithm MIN-COM-CS. We present the following algorithm for this purpose.

ALGORITHM CS-3.2:

INPUT: Intermediate communication policy of Algorithm MIN-COM-CS  $COM_{si}$ ,  $i \in \mathcal{A}$  of previous iteration, observable event sets  $E_i$  for agent  $i$ ,  $i \in \mathcal{A}$ .

**3.2.1.** For each  $(e_1, e_2)$  with  $e_1 \in V_{ns} \setminus E_i$ ,  $e_2 \in V_{ns}$ ,  $e_1 \neq e_2$ , test (I) and (II) for  $L_4(e_1, e_2) \times L_3(e_2)$ . Construct directed graph  $D$  (Dependency Graph) whose nodes are  $e \in V_{ns}$  and arcs are  $(e_i, e_j)$  if the test of  $L_4(e_i, e_j) \times L_3(e_j)$  failed.

**3.2.2.** Set  $V_2 = \emptyset$ ,  $V_1 = E_i \cap V_{ns}$ , and  $V^T \leftarrow V_{ns} \setminus V_1$ .

**3.2.3.** Find a minimal set  $V_2' \supset V_2$ , and  $V_2' \setminus V_2 \subseteq V^T$ , which satisfies the following conditions: for all  $e \in V_2'$ , there are no directed paths starting with  $e$  and ending with  $e' \notin V_2'$ .

**3.2.4.** Test (I) and (II) for current  $P_1$  with respect to  $V_2'$ .

- If it satisfies the test, set  $V_2 \leftarrow V_2'$ .
- If it does not, set  $V_1 \leftarrow V_1 \cup (V_2' \setminus V_2)$ , and then, set

$$V_1 \leftarrow V_1 \cup \{e \in V^T : \exists \text{ a directed path from } e \text{ to } e' \in V_2\}.$$

**3.2.5.** Set  $V^T \leftarrow V^T \setminus (V_1 \cup V_2)$ .

If  $V^T \neq \emptyset$  go back to Step 3.2.3. Otherwise, set  $V_2^* \leftarrow V_2$ ,  $RT_{si}^*(x_l) = \{(x_l, e) : e \in V_2^*\}$  and

$$COM_{si} \leftarrow COM_{si} \setminus RT_{si}^*(x_l).$$

Go to Step 4 of Algorithm MIN-COM-CS.

OUTPUT: Updated intermediate communication policy  $COM_{si}, i \in \mathcal{A}$ .

*Remark:* We perform corresponding tests for every  $(e_1, e_2)$  with  $e_1 \in V_{ns} \setminus E_i, e_2 \in V_{ns}, e_1 \neq e_2$  such that the Dependency Graph is constructed for  $P_5$  tests of all possible solutions instead of a particular  $RT_{si}(x_l)$  as in Section 4.1.

#### 4.3 Properties of Algorithm CS-3.2

We now show the correctness of Algorithm CS-3.2, when it is used to implement Step 3.2 of Algorithm MIN-COM-CS.

**Lemma 4** *Consider testing the removal of communications for the transitions going out of state  $x_l$  and labeled by events in  $V_2' \subseteq V_{ns}$ . Then, the test for  $P_5$  in Equation (73) is successful iff, for all  $e \in V_2'$ , all directed paths that start with event  $e$  in Dependency Graph  $D$  constructed in Step 3.2.1 contain only events in set  $V_2'$ .*

*Proof* The proof follows directly from the construction of the Dependency Graph  $D$ .  $\square$

**Theorem 2** *Algorithm CS-3.2 gives an  $RT^*(x_l)$  that satisfies (I), (II) and (III) of Step 3.2 of Algorithm MIN-COM-CS.*

*Proof* By Step 3.2.1, Step 3.2.4, and language decomposition analysis, we can conclude that the resulting  $RT^*(x_l)$  satisfies (I) and (II). It remains to prove that (III) holds.

Consider the iterations of Steps 3.2.3 to 3.2.5. We have tested in previous iterations that the communications of transitions for all events within  $V_2$  are removable. Then

we find a minimal set  $V_2' \supset V_2$  with  $e \in V_2' \setminus V_2$ , where  $V_2'$  satisfies the requirement of Step 3.2.3. By Lemma 4, we only need to test  $P_1$  for  $V_2'$ . Suppose that in Step 3.2.4 the test of  $V_2'$  fails. Suppose there exists  $V_2'' \supset V_2$  with  $e \in V_2''$ , and suppose the communications of events in  $V_2''$  are removable. Then, the removal of  $V_2''$  must satisfy the test of  $P_5$ . By Lemma 4, there are no directed paths coming out of  $V_2''$  in the Dependency Graph  $D$ . By minimality of  $V_2'$ , we have  $V_2'' \supseteq V_2'$ .

By Lemma 4 and Step 3.2.3, we only need to test  $P_1$  for  $V_2''$ . Let test  $P_1'$  and  $P_1''$  denote the  $P_1$  tests corresponding to  $V_2'$  and  $V_2''$ , respectively. The failure of test  $P_1'$  indicates

$$(\exists s, t \in \mathcal{L}(\hat{G})) \theta_i(s) = \theta_i(t) \wedge (f(x_l, s), f(x_l, t)) \in T_{i,spec} \cup T_{i,feas}. \quad (75)$$

But, for both tests  $P_1'$  and  $P_1''$ , the information mappings are the same for  $s$  and  $t$ . Consequently,  $s$  and  $t$  should also cause a failure for test  $P_1''$ . This is a contradiction. We can conclude that, in each iteration of Steps 3.2.3 to 3.2.5, there are no supersets of  $V_2$  containing events in  $V_1'$  with corresponding communications removable. Since  $V_2^* \cup V_1' = V_{ns}$ , we have that  $V_2^*$  contains a maximal set of events in  $V_{ns}$  whose corresponding communications are removable. This result shows that  $RT^*(x_l)$  satisfies (III).  $\square$

#### 4.4 Tests in Step 3.2

In this subsection, we first suggest an algorithm for calculating confusable states for an automaton with given information mapping. Then, we consider the test  $L_4(e_i, e_j) \times L_3(e_j)$  in Step 3.2.1 of Algorithm CS-3.2 and the test  $P_1$  in Step 3.2.4.

*4.4.1 Calculating Confusable States* We can use the table technique in [23] or the nondeterministic product automaton in Chapter II of [21] (related to the M-Machine in [15] and the verifier in [25]) to calculate, in polynomial time, the set of confusable state pairs. Then we use the set of confusable state pairs to verify feasibility and local specifications for a given communication policy with corresponding index function  $\mathcal{S}_k$  for each agent  $k$  (defined in Equation (17)).

Let  $cp_i$  be the set of all confused pairs for agent  $i$  obtained by any of the methods mentioned above. Clearly, Equations (18) and (20) for feasibility holds iff

$$\begin{aligned} & (\nexists (x, y) \in cp_i) [(x, e), (y, e) \in TR(G)] \\ & \wedge [(x, e) \in COM_{is} \wedge (y, e) \notin COM_{is}] \end{aligned} \quad (76)$$

for all local agents  $i \in \mathcal{A}$ .

Equation (32) for local specification conditions holds iff

$$(\forall i \in \mathcal{A}) cp_i \cap T_{i,spec} = \emptyset \quad (77)$$

for all local agents  $i \in \mathcal{A}$ .

*4.4.2 Tests in Step 3.2.1* The algorithm for testing  $L_4(e_i, e_j) \times L_3(e_j)$  is as follows.

**Step 1.** Construct a deterministic automaton  $\bar{G}$  as follows:

Construct an automaton  $G'_m(e_i, e_j)$  to represent marked language  $L_4(e_i, e_j)$  as a subautomaton of  $\hat{G}$ . Let  $X'_m$  represent the corresponding marked states of the resulting automaton. Here, the marking of states can be done as follows. For  $x \in X'_m$ , to mark state  $x$ , we change the label of state  $x$  to  $x'$ .

Construct an automaton  $G''_m(e_j)$  to represent marked language  $L_3(e_j)$  as a subautomaton of  $\hat{G}$ . Let  $X''_m$  represent the corresponding marked states of the resulting automaton. Here, the marking of states can be done as follows. For  $x \in X''_m$ , to mark state  $x$ , we change the label of state  $x$  to  $x''$ .

Construct  $\bar{G}$  by merging the initial states of  $G'_m(e_i, e_j)$  and  $G''_m(e_j)$ , both labeled by  $x_i$ , as initial state but keeping all other transitions and states in both of  $G'_m(e_i, e_j)$  and  $G''_m(e_j)$ . Note that self-loops at initial state  $x_i$  of both automata match before these two states are merged.

**Step 2.** Calculate pairs of states in  $\bar{G}$  which are confusable using any of the methods mentioned above. If there exist  $x' \in X'_m$  and  $y'' \in X''_m$  with  $(x', y'')$  confusable with each other and  $(x, y) \in T_{i,feas} \cup T_{i,spec}$ , we can conclude that for state  $x_i$  the removal of communication of event  $e_i$  depends on removal of communication of event  $e_j$ .

4.4.3 Tests in Step 3.2.4 The algorithm for testing Step 3.2.4 is as follows.

**Step 1.** Construct an automaton  $\tilde{G}$  as a subautomaton of  $\hat{G}$  to represent language

$$L_1 \cup (\cup_{e \in V_2} L_2(e)) \cup (\cup_{e_1 \in V_2 \cup e_2 \in E \setminus V_3} L_4(e_1, e_2)). \quad (78)$$

**Step 2.** Calculate pairs of states in  $\tilde{G}$  which are confusable using any of the methods mentioned above. If there exists  $(x, y)$  confusable with each other and  $(x, y) \in T_{i,feas} \cup T_{i,spec}$ , we can conclude that the test of  $P_i$  for  $V'_2$  fails. Otherwise, the test passes.

In order to construct the Dependency Graph, we need to consider each pair of transitions for a particular state. Therefore, in the worst case we need to con-

sider  $\|E\|^2$  pairs. For each of those event pairs, we need to build an automaton that represents  $L_4(e_i, e_j) \times L_3(e_j)$ , which needs a computational effort of the order  $\|X \times E\|$ . A test for each  $L_4(e_i, e_j) \times L_3(e_j)$  has complexity  $\|E\|\|X\|^2$  if we use the table technique in [23]. Then, the number of iterations of Steps 3.2.3 to 3.2.5 is upper bounded by number of events  $\|E\|$ . And for each iteration, the computation for testing I and II can be done by using the table technique, which has a computational efforts  $\|E\|\|X\|^2$ . In all, Algorithm CS-3.2 is of the same order as building the Dependency Graph, which is  $\|E\|^3\|X\|^2$ .

#### 4.5 Computational Complexity

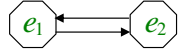
As a consequence of the iterative procedure in Algorithm MIN-COM-CS, the procedure described in Section 4 to resolve Step 3.2 of Algorithm MIN-COM-CS, together with the strategies described in Section 4.4 for the tests of Step 3.2, the following result on computational complexity can be obtained.

**Theorem 3** *The problem formulated in Section 2.5 can be solved with polynomial complexity with respect to the cardinality of the state space of  $G$ , the cardinality of the event set of  $G$ , and the number of local agents.*

*Proof* The number of iterations for Algorithm MIN-COM-CS is  $\|X\|$ . The amount of computation for each iteration is determined by the computation complexity of Algorithm CS-3.2, which is in the order of  $\|E\|^3\|X\|^2$ . Since there are  $N$  agents, overall, in the worst case, the complexity of solving the problem in Section 2.5 is  $\mathcal{O}(N\|E\|^3\|X\|^3)$ . □

## 5 Illustrative Examples

*Example 5* This example continues Example 2 presented earlier. By iterating Algorithm MIN-COM-CS for Example 2, we examine states 4, 3, 2, 1 in this order. No communications are removed for these states. Next, by Step 2, we examine



**Fig. 5** Dependency Graph for transitions at state 0 of Example 2.

state 0. Go to Step 3. Step 3.1 is trivial here. Go to Step 3.2. By Step 3.2.1, we find that the removal of communications of transitions  $(0, e_1)$  and  $(0, e_2)$  are mutual dependent. The Dependency Graph is shown in Fig. 5. Go to Step 3.2.2. Iterate Step 3.2.3 to Step 3.2.5. By looking at the observer in Fig. 3 or by using the techniques discussed in Section 4.4.1, we find that we can remove  $(0, e_1)$  and  $(0, e_2)$  together. Finally

$$COM_{s1}^* = \{(1, e_2), (2, e_1)\}, COM_{1s}^* = \emptyset. \quad (79)$$

*Example 6* We continue Examples 1 and 3 presented earlier. Consider the system model given in Fig. 1. Let  $E_s = \emptyset$ ,  $E_1 = \{a_1, a_2, a_3, a_4\}$  and  $E_2 = \{e_1, e_2, e_3, e_4\}$ . Suppose the only requirement for  $COM_{s1}$  is to satisfy the feasibility of  $COM_{1s}^*$ , that is,  $T_{1,spec} = \emptyset$ . However, agent 2 needs to distinguish state pair  $\{(3, 8)\}$ , that is,  $T_{2,spec} = \{(3, 8)\}$ . From Equations (27) and (28) in Section 2.3, we have

$$COM_{1s}^* = \{(3, a_4), (4, a_1), (5, a_3), (6, a_2)\}, \text{ and}$$

$$COM_{2s}^* = \{(0, e_4), (0, e_2), (0, e_3), (1, e_2), (2, e_3)\}.$$

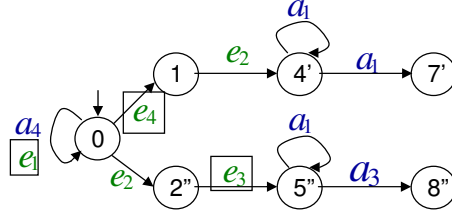
Thus, we have  $T_{1,feas} = \{(0,3), (3,5), (4,5), (1,4)\}$ , and  $T_{2,feas} = \emptyset$ .

Initially, by Step 1 of Algorithm MIN-COM-CS,

$$COM_{s1} = COM_{2s} = \{(0, e_1), (0, e_2), (0, e_3), (0, e_4), (1, e_2), (2, e_3)\},$$

$$COM_{s2} = COM_{1s} = \{(0, a_4), (1, a_1), (3, a_3), (3, a_4), (4, a_1), (5, a_1), (5, a_3), (6, a_2)\}.$$

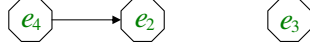
By iterating Algorithm MIN-COM-CS, we examine states 8, 7, 6, 5, 4, 3, 2, 1 in this order. Remove the communications of transitions  $(5, a_1), (3, a_3), (1, a_1)$  from  $COM_{1s}$ . Remove the communication of transition  $(2, e_3)$  from  $COM_{s1}$ . Remove the communication of transitions  $(6, a_2), (5, a_1), (5, a_3), (4, a_1), (3, a_3), (1, a_1)$  from  $COM_{s2}$ . Notice that the removal of communication  $(3, a_4)$  from  $COM_{s2}$  will cause agent 2 to confuse  $(3, 8)$ , which violates  $T_{2,spec}$ . So far there is nothing to be removed for  $COM_{2s}$ . Now, by Step 2, we examine state 0. By Step 3.1, we further remove  $(0, e_1)$  from  $COM_{s1}$  and  $COM_{2s}$ , and remove  $(0, a_4)$  from  $COM_{1s}$  and  $COM_{s2}$ . Go to Step 3.2.1. The automaton used for testing  $L_4(e_4, e_2) \times L_3(e_2)$  is shown in Fig. 6, where all boxed transitions are “unknown” to local agent 1 in the test. State



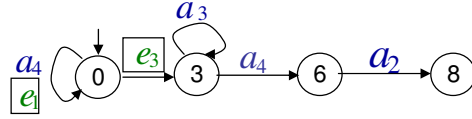
**Fig. 6** Automaton for testing  $L_4(e_4, e_2) \times L_3(e_2)$  in  $\hat{G}(0)$ .

$4'$  is confused with state  $5''$  by agent 2 because agent 2 has no information about  $(0, e_4)$  or  $(2'', e_3)$ . Since  $(4, 5) \in T_{1,feas}$ , this indicates that the removal of  $e_4$  depends on the removal of  $e_2$ . It can be verified that no further dependencies exist.

The Dependency Graph for this case is shown in Fig. 7. Go to Step 3.2.2. Iterate Step 3.2.3 to Step 3.2.5. We try to remove communication  $(0, e_3)$ . The automaton used for testing  $P_1$  is shown in Fig. 8. The test fails due to the confusion of pair



**Fig. 7** Dependency Graph for transitions at state 0 of  $\hat{G}(0)$ .



**Fig. 8** Automaton for Testing  $P_1$  for removal of  $(0, e_3)$ .

of states  $(0, 3) \in T_{1,feas}$ . Then, we further remove communications  $(0, e_2)$ ,  $(0, e_4)$  in this order from  $COM_{s1}$  because the corresponding  $P_1$  tests passed. Finally, we have

$$\begin{aligned}
 COM_{s1}^* &= \{(0, e_3), (1, e_2)\}, \\
 COM_{1s}^* &= \{(3, a_4), (4, a_1), (5, a_3), (6, a_2)\} \\
 COM_{s2}^* &= \{(3, a_4)\}, \\
 COM_{2s}^* &= \{(0, e_2), (0, e_3), (0, e_4), (1, e_2), (2, e_3)\}.
 \end{aligned}$$

## 6 Conclusion

Building on the novel approach proposed in [21, 22] for solving general minimum-communication problems, we have considered the important special case of one

central station communicating with a set of local agents. We have proposed a specialized version of Algorithm MIN-COM-GEN of [21,22], termed MIN-COM-CS, which exploits the special features of the problem to improve the computational efficiency of MIN-COM-GEN, namely, MIN-COM-CS achieves polynomial complexity in all key parameters (the cardinality of the state space, the cardinality of the event set, and the number of local agents). The techniques used to achieve this result, such as dependency graph and language decomposition, may prove applicable to other classes of minimum-communication problems. This remains to be investigated.

## References

1. Barrett, G., Lafortune, S.: On the synthesis of communicating controllers with decentralized information structures for discrete-event systems. In: Proc. 37th IEEE Conf. on Decision and Control, pp. 3281–3286 (1998)
2. Barrett, G., Lafortune, S.: Some issues concerning decentralized control problems with communication. In: Proc. 38th IEEE Conf. on Decision and Control (1999)
3. Barrett, G., Lafortune, S.: Decentralized supervisory control with communicating controllers. *IEEE Trans. Automatic Control* **45**(9), 1620–1638 (2000)
4. Bhutani, K., Khan, B.: Minimizing communication costs in hierarchical multi-agent systems. In: Proc. 6th Joint Conference on Information Sciences, pp. 1435–1442 (2002)
5. Boel, R.K., vanSchuppen, J.H.: Decentralized failure diagnosis for discrete-event systems with costly communication between diagnosers. In: Proc. 6th International Workshop on Discrete Event Systems (WODES'02), pp. 175–181 (2002)
6. Cassandras, C.G., Lafortune, S.: *Introduction to Discrete Event Systems*. Kluwer Academic Publishers (1999)

7. Debouk, R., Lafortune, S., Teneketzis, D.: Coordinated decentralized protocols for failure diagnosis of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications* **10**(1/2), 33–86 (2000)
8. Han, G., Wang, J.: An efficient communication strategy for mobile agent based distributed spatial data mining application. In: *Proc. SPIE - The International Society for Optical Engineering*, pp. 538–541 (2005)
9. Lafortune, S.: On decentralized and distributed control of partially-observed discrete event systems. In: *Advances in Control Theory and Applications - Lecture Notes in Control and Information Sciences*, Edited by C. Bonivento, A. Isidori, L. Marconi and C. Rossi. Springer (2007)
10. Lin, F., Rudie, K., Lafortune, S.: Minimal communication for essential transitions in a distributed discrete-event system. *IEEE Trans. Automatic Control* **52**(8), 1495–1502 (2007)
11. Ricker, S.L., Rudie, K.: Incorporating communication and knowledge into decentralized discrete-event systems. In: *Proc. . 38th IEEE Conf. on Decision and Control*, pp. 1326–1332 (1999)
12. Ricker, S.L., van Schuppen, J.H.: Decentralized failure diagnosis with asynchronous communication between supervisors. In: *Proc. European Control Conf.*, pp. 1002–1006 (2001)
13. Rudie, K., Lafortune, S., Lin, F.: Minimal communication in a distributed discrete-event control system. In: *Proc. 1999 American Control Conf.*, pp. 1965–1970 (1999)
14. Rudie, K., Lafortune, S., Lin, F.: Minimal communication in a distributed discrete-event system. *IEEE Trans. Automatic Control* **48**(6), 957–975 (2003)
15. Rudie, K., Willems, J.C.: The computational complexity of decentralized discrete-event control problems. *IEEE Trans. Automatic Control* **40**(7), 1313–1318 (1995)

16. van Schuppen, J.H.: Decentralized control with communication between controllers. In: V.D. Blondel, A. Megretski (eds.) *Unsolved Problems in Mathematical Systems and Control Theory*, pp. 144–150. Princeton University Press, Princeton (2004)
17. Shen, J., Lesser, V., Carver, N.: Minimizing communication cost in a distributed bayesian network using a decentralized MDP. In: *Proc. 2nd International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 678–685 (2003)
18. Thorsley, D., Teneketzis, D.: Active acquisition of information for diagnosis of discrete event systems. In: *Proc. Allerton Conference on Control, Communication, and Computing* (2004)
19. Thorsley, D., Teneketzis, D.: Diagnosis of cyclic discrete-event systems using active acquisition of information. In: *Proc. 8th International Workshop on Discrete Event Systems (WODES'06)*. University of Michigan-Ann Arbor (2006)
20. Wang, W., Lafortune, S., Lin, F.: A polynomial algorithm for minimizing communication in a distributed discrete event system with a central station. In: *Proc. 45th IEEE Conf. on Decision and Control*, 6034-6040 (2006)
21. Wang, W.: Optimization of communication and coverage in classes of distributed systems. Ph.D. thesis, University of Michigan, Department of Electrical Engineering and Computer Science (2007)
22. Wang, W., Lafortune, S., Lin, F.: Minimization of communication in distributed discrete event systems. In: *Proc. of 2007 European Control Conf.* (2007)
23. Wang, W., Lafortune, S., Lin, F.: An algorithm for calculating indistinguishable states and clusters in finite-state automata with partially observable transitions. *Systems & Control Letters* **56**(9), 656–661 (2007)
24. Wong, K.C., van Schuppen, J.H.: Decentralized supervisory control of discrete-event systems with communication. In: *Proc. 3rd International Workshop on Discrete Event Systems (WODES'96)*, pp. 284–289 (1996)

25. Yoo, T.S., Lafortune, S.: Polynomial-time verification of diagnosability of partially-observed discrete event systems. *IEEE Trans. Automatic Control* **47**(9), 1491–1495 (2002)