Nash Equilibrium for Communication Protocols in Decentralized Discrete-Event Systems

W.H. Sadid, S.L. Ricker and S. Hashtrudi-Zad

Abstract—Finding optimal communication strategies for a controller in a decentralized control setting is challenging because the best strategy depends on the choices of other controllers, all of whom are also trying to optimize their own strategies. An optimal strategy in this context is one that minimizes the cost of the communication protocol for each controller. Communication policies that satisfy Nash equilibrium, an important solution concept in game theory, are of particular interest. A recent algorithm for efficiently calculating a Nash equilibrium point for multi-agent systems in a game-theoretic setting is adapted for the problem of incorporating communication into decentralized discrete-event systems.

I. INTRODUCTION

Equilibrium is a key idea in calculating optimal strategies for multi-agent systems. A Nash equilibrium is a collection of strategies, one for each agent in the system, such that if all other agents adhere to their strategies, any agent’s recommended strategy is strictly better than any other strategy it could execute. Intuitively, a Nash equilibrium represents each agent’s best response to the strategies of the other agents. If an agent knows the strategies of the other agents, then its best response is straightforward: the problem reduces to a centralized single-agent situation where the strategy that yields the (globally) optimal payoff is chosen. In contrast, a Nash equilibrium corresponds to a locally-optimal solution.

Optimal decentralized discrete-event control strategies have been examined in the context of Nash equilibrium [10]. Maximal decentralized controllers are synthesized with respect to the inclusion relation on the set of languages of the closed-loop system. There is no notion of cost associated with the synthesis process, and thus solutions are not necessarily comparable.

The equilibrium of asymmetric communication policies for decentralized diagnosis of discrete-event systems was examined in [2]. Nash equilibrium for communication between two diagnosers was computed assuming a uniform cost for communication. Additionally, a more restrictive constraint was imposed such that only one diagnoser had the ability to communicate. The decentralized control framework we consider here assumes that all supervisors are capable of communicating; however, communication is only initiated by a supervisor that observes and communicates information that will allow the supervisor receiving the information to make a correct control decision. In contrast, the format for communication in [1] allows any supervisor to initiate communication. Although there is no control or diagnosis objective in [15], the authors describe a locally optimal communication protocol for a two-agent system based on a set-theoretic definition of minimality. In this formulation, two different communication protocols are incomparable. This model was extended to decentralized control problems in [19], but the analysis is restricted to acyclic systems. Our interest lies in quantitative measures for synthesizing locally-optimal communication protocols.

Normal-form games (for two or more players) are those for which each player’s strategy can be represented by a finite set of strategies. In addition, strategies are associated with a payoff function. Algorithms to calculate a sample Nash equilibrium point for such two-player games include [6], [11], [17]. Only recently has the complexity of this problem been established as being in class PPAD-complete (polynomial parity argument, directed version) [3]. However, there is no known polynomial time complexity algorithm for this problem, and the precise computational complexity is also unknown [11].

We focus our attention on Algorithms 1 and 2 in [11] and adapt them to the problem of finding Nash equilibrium for, respectively, two communicating controllers and more than two communicating controllers in decentralized discrete-event systems. These algorithms rely on heuristic techniques to prune the solution space. In [11], the authors exploit game theory research that indicates the likelihood of finding equilibria increases when examining either strategies of small size, or those where the strategies of each player are of nearly the same size. The search for an equilibrium point is directed in this fashion and these algorithms, when tested on sets of randomly-generated games, outperform the standard algorithms such as Lemke-Howson [6], Simplicial Subdivision [18] and Govindan-Wilson [5].

II. BACKGROUND

The supervisory control framework of [12] describes the behavior of the system requiring control and the desired system behavior (design specification) as regular languages (over an alphabet \( \Sigma \) of discrete events). We denote these behaviors by languages \( L \) and \( K \), respectively. The decentralized supervisory control problem [4], [16] requires the synthesis of \( n \geq 2 \) controllers or supervisors that cooperatively keep the system in \( K \) by issuing control directives to prevent the system from performing behavior in \( L \setminus K \). Based on only a partial view of \( L \), each supervisor independently issues...
a control decision (e.g., enable, disable) where the control objective is reached when the correct pattern of control decisions is issued to keep the system in $K$. The ability to achieve a correct control policy relies on the existence of at least one supervisor that makes the correct control decision to keep the system within $K$. We denote the set of decentralized supervisors by $I$.

For any strings $s, t \in \Sigma^*$, where $\Sigma^*$ is the Kleene closure of $\Sigma$, we say that $t$ is a prefix of $s$, denoted $t \preceq s$, if $\exists u \in \Sigma^*$ such that $s = tw$. If $L \subseteq \Sigma^*$, then the prefix-closure of $L$ is a language, denoted by $L^\preceq$, consisting of all prefixes of strings of $L$: $L^\preceq := \{ t \in \Sigma^* \mid t \preceq s \}$. A language is said to be prefix-closed if $L = L^\preceq$. We assume that the system language $L$ is prefix-closed.

The partial view that a decentralized supervisor $i$ has of $L$ is described by the set of observable events, denoted $\Sigma_{o,i} \subseteq \Sigma$, for $i \in I$. To describe a supervisor’s view of the system behavior, we use the canonical projection $\pi_i : \Sigma^* \rightarrow \Sigma_{o,i}^*$, for $i \in I$. This operator removes from every string in $\Sigma^*$ those events $\sigma$ that are not found in $\Sigma_{o,i}$: $\pi_i(\varepsilon) = \varepsilon$, where $\varepsilon$ denotes the empty string; $\pi_i(\sigma) = \sigma$ if $\sigma \in \Sigma \setminus \Sigma_{o,i}$; and $\pi_i(ta) = \pi_i(t)\pi_i(a)$, $t \in \Sigma^*, a \in \Sigma$.

The set of events $\Sigma$ is partitioned into the disjoint sets $\Sigma_e$ of controllable events, and $\Sigma_{uc}$ of uncontrollable events. Only controllable events can be prevented from occurring (i.e., may be disabled), as uncontrollable events are considered to be permanently enabled. Each supervisor $i \in I$ controls a subset of events: $\Sigma_{c,i} \subseteq \Sigma_e$, where $\Sigma_{c,i} := \cup_{i=1}^{\infty} \Sigma_{c,i}$. We denote the set of supervisors for which $\sigma \in \Sigma_{c,i}$ by $I_o$.

Decentralized supervisors (with no communications) can be synthesized if the specification $K$ is co-observable and controllable [16]. A language $K$ is co-observable with respect to $L$ and $\pi_i$, for $i \in I$, if for all $t \in K$ and for all $\sigma \in \Sigma$,

$$ (t\sigma \not\in K) \wedge (t\sigma \in L) \Rightarrow \exists i \in I_o \text{ s.t. } \pi_i^{-1}(\pi_i(t))\sigma \cap K = \emptyset. $$

It must be the case, based only on its partial view of a sequence, that at least one decentralized supervisor can make the correct control decision (i.e., determine that $t\sigma \in L \setminus K$).

The equivalent conditions in the case of a centralized supervisor are observability and controllability [7]. The observations of a centralized supervisor are captured by a projection operator $\pi : \Sigma^* \rightarrow \Sigma_{o}^*$. A language $K$ is said to be observable with respect to $L$, $\pi$, and $\Sigma_{o}$ if for all $t \in K$ and for all $\sigma \in \Sigma_{o}$,

$$ (t\sigma \not\in K) \wedge (t\sigma \in L) \Rightarrow \pi^{-1}(\pi(t))\sigma \cap K = \emptyset. $$

If specification $K$ is observable (and controllable) and not co-observable, the supervisory control problem can be solved using communicating decentralized supervisors.

A. Finite-state automaton

Every regular language $L$ can be recognized by a finite-state automaton. A deterministic finite automaton for $L$ is a tuple: $M_L = (Q, \Sigma, T, q_0)$, where $Q$ is a finite set of states; $\Sigma$ is a finite alphabet; $T \subseteq Q \times \Sigma \times Q$ is a (partial) transition relation; and $q_0 \in Q$ is the initial state. We can also partition transitions in $T$ according to the observable partition of $\Sigma$. In particular, $T_{o,i} = \{(q, \sigma, q') \in T \mid \sigma \in \Sigma_{o,i}\}$. An automaton for the language for $L = \{\varepsilon, a, b, ab, ab\sigma, ba\sigma\}$ is displayed in Fig. 1.

![Fig. 1. A finite-state automaton with the initial state indicated by a small entry arrow.](image)

B. Decentralized communication protocols

Incorporating instantaneous communication into decentralized discrete-event control problems has been explored with a variety of models (e.g., [1], [8], [13], [14], [20]). Communication is introduced into the system when, for given languages $K$ and $L$, $K$ does not satisfy (1). The structure on which we will reason about communication is isomorphic to the plant automaton in the case of cyclic systems and isomorphic to a deterministic version of the $U$ structure defined in [13] in the case of cyclic systems.

We denote the set of messages introduced into the system by $\Delta = \cup_{i,j \in I} \Delta_{i,j}$, where $\Delta_{i,j} \subseteq \Sigma_{o,i}$ is the set of messages sent from supervisor $i$ to supervisor $j$, for $i, j \in I$. The goal is to allow the supervisors to make the correct control decisions not just based on their partial observation of the system, but also taking into account the information received from other communicating supervisors (senders). There are many options for choosing when communication should occur: each supervisor/sender can communicate everything it observes followed by subsequent refinement based on information it has received from the others; or specific events/transitions can be identified by the sender as providing useful information to a receiver.

**Definition 1:** [13] A communication protocol from supervisor $i$ to supervisor $j$ is a mapping from the system behavior to a message or silence $\phi_{i,j} : \Sigma \rightarrow \Delta_{i,j}$, where $\Delta_{i,j} = \Delta_{i,j} \cup \{\varepsilon\}$. Such a protocol instructs supervisor $i$ to send a message to supervisor $j$ after an observation of a sequence in $L$.

That is, when $\phi_{i,j}(t)$ is a symbol in $\Delta_{i,j}$, supervisor $i$ initiates communication with supervisor $j$, otherwise supervisor $i$ is silent. Note that supervisor $i$ can only communicate after it sees an event in $\Sigma_{o,i}$ and, thus, for $t \in L$ and $\forall \sigma \in (\Sigma \setminus \Sigma_{o,i})$, $\phi_{i,j}(t\sigma) = \varepsilon$.

The last information that supervisor $i$ obtained about a sequence $t = \sigma_1 \ldots \sigma_m \in L$, is given by $\psi_i : L \rightarrow \Delta_{i,j} \cup (\cup_{t \in L} \Delta_{i,j})$, and is defined as follows, where $\psi_i(\varepsilon) = \varepsilon$. 3385
\[ \psi_i(t) = \begin{cases} 
\sigma_m, & \text{if } \sigma_m \in \Sigma_{o,i}; \\
\sigma_m, & \text{if } \sigma_m \notin \Sigma_{o,i} \text{ and } \exists j \in I \text{ s.t. } \phi_{j,i}(t) \neq \varepsilon; \\
\varepsilon, & \text{otherwise.} 
\end{cases} \]

The extended canonical projection \( \pi^{\Delta}_i \) is a mapping that includes the messages of \( \Delta \). Formally, \( \pi^{\Delta}_i : \Sigma^* \rightarrow (\Sigma_{o,i} \cup (\cup_{j \in I} \Delta_{j,i}))^* \), where \( \pi^{\Delta}_i(\varepsilon) = \varepsilon \), and \( \pi^{\Delta}_i(t) = \psi_i(\sigma_1) \psi_i(\sigma_2) \ldots \psi_i(\sigma_1 \ldots \sigma_m) \), for \( t = \sigma_1 \ldots \sigma_m \).

**Definition 2:** The supervisor-to-supervisor communication protocols \( \phi_{i,j} \) are coherent\(^1\) if for all sequences \( t, t' \in L \), and every \( i, j \in I, i \neq j \),

\[ \pi^\Delta_i(t) = \pi^\Delta_i(t') \Rightarrow \phi_{i,j}(t') = \phi_{i,j}(t'). \]

When communication is identified as necessary to resolve a violation of co-observability, it must also be the case that the same message is sent after the occurrence of all sequences that have the same extended canonical projection as the original communication.

The decentralized communication problem can be formally described as follows.

**Problem 1:** Given two regular languages \( K, L \) defined over a common alphabet \( \Sigma \), where \( K \subseteq L \subseteq \Sigma^* \) is observable and controllable, but is not co-observable with respect to \( L \), \( \pi_i, \Sigma_{c,i} \), controllable events \( \Sigma_{c,1}, \ldots, \Sigma_{c,n} \subseteq \Sigma \), observable events \( \Sigma_{o,1}, \ldots, \Sigma_{o,n} \subseteq \Sigma \), and a finite set of communication messages \( \Delta = \cup_{i,j \in I} \Delta_{i,j} \). Construct communication protocols \( \phi_{i,j} \) for every \( i, j \in I \), such that \( (\sigma \notin K) \land (\tau \in L) \Rightarrow \exists i \in I, s.t. \pi^{\Delta} \pi^{\Delta}_i(t) \cap \mathcal{K} = \emptyset \).

Thus we can synthesize communicating supervisors when \( K \) is co-observable with respect to \( L \), \( \pi^\Delta_i \), and \( \Sigma_{c,i} \).

### III. NASH EQUILIBRIUM FOR COMMUNICATION PROTOCOLS

The results in this section are predicated on the fact that we can express a decentralized control and communication problem as a normal-form game.

**Definition 3:** [11] A (finite, \( n \)-person) normal-form game is a tuple \( (N, A, u) \), where:

- \( N \) is a finite set of \( n \) players, indexed by \( i \);
- \( A = A_1 \times \ldots \times A_n \), where \( A_i \) is a finite set of actions available to player \( i \), each vector \( a = \langle a_1, \ldots, a_n \rangle \in A \) is called an action profile;
- \( u = \langle u_1, \ldots, u_n \rangle \) where \( u_i : A \rightarrow \mathcal{R} \) is a real-valued utility function for player \( i \).

We consider a decentralized discrete-event control and communication problem where:

- \( I \) is a finite set of \( n \) supervisors, indexed by \( i \);
- \( \Phi = \Phi_1 \times \ldots \times \Phi_i \times \ldots \times \Phi_n \) is a finite set of communication protocols, where \( \Phi_i = \{ \phi_i | \phi_i = \langle \phi_{i,1}, \ldots, \phi_{i,n} \rangle, \phi_{i,j} : L \rightarrow \Delta_{i,j} \} \); and
- \( u = \langle u_1, \ldots, u_n \rangle \) where \( u_i : \Phi \rightarrow \mathcal{R} \) is a real-valued utility function for each supervisor \( i \).

\(^1\)In the discrete-event system literature, this property is referred to as feasibility, but we will use this word in the sequel as it has a particular meaning in the context of finding Nash equilibrium points.

Wlog we assume a uniform cost for communication (i.e., same cost for every message(event)). For acyclic systems, this corresponds to finding protocols that have an overall minimal number of communications; however, it is straightforward to extend \( u_i \) to different cost functions. Let \( \omega \) denote the set of all communication protocols \( \{ \phi_k | k \in I \land k \neq i \} \). Similarly, let \( \Phi_{\omega} \) denote the set \( \{ \Phi_k | k \in I \land k \neq i \} \).

We focus on two ways in which a supervisor can choose its communication protocol: (i) select a single action and execute it; (ii) randomize over the set of available actions according to some probability distribution. The former case is called a pure strategy, while the latter case is called a mixed strategy.

A mixed strategy for a supervisor specifies the probability distribution used to select the protocol that a supervisor will use to solve the control problem. The probability distribution for a supervisor \( i \) is denoted by \( p_i : \Phi_i \rightarrow [0,1] \), such that \( \sum_{\phi_i \in \Phi_i} p_i(\phi_i) = 1 \).

**Definition 4:** (Adapted from [11].) The subset of communication protocols \( \phi_i \) that are assigned positive probability by the mixed strategy \( p_i \), is called the support of \( p_i \).

A pure strategy is a special type of mixed strategy when the support is a single action.

The expected utility function for mixed strategies can be extended from \( u_i \), defined previously:

\[ u_i(\Phi) = \sum_{\phi \in \Phi} u_i(\phi) \prod_{j \in I} p_j(\phi_j) \]

We can now define Nash equilibrium in the context of Problem 1.

**Definition 5:** A communication protocol \( \phi^* = (\phi_1^*, \ldots, \phi_n^*) \) is a Nash equilibrium if

- for all \( i \in I \), \( u_i(\phi_i^*, \phi^{\omega}_n) \leq u_i(\phi_i, \phi^*_{\omega}) \) for every protocol \( \phi_i \in \Phi_i \);
- \( \phi^* \) and \( (\phi_i, \phi^*_{\omega}) \) are coherent; and
- \( \phi^* \) and \( (\phi_i, \phi^*_{\omega}) \) solve Problem 1.

Note that this definition seeks a least-costly best response communication protocol for each supervisor.

**Theorem 1:** [9] Every normal-form game has at least one Nash equilibrium.

### A. Nash equilibrium for two supervisors

In [11], a novel approach for finding a sample Nash equilibrium for normal-form games is presented. Algorithm 1 in [11], now referred to in the literature as SEM (Support-Enumeration Method), introduces heuristics based on the space of supports and a notion of dominated actions that are pruned from the search space. It may still be the case that an exponential number of iterations are required to find a sample Nash equilibrium, but as noted in [11], when tested on large sets of random games, SEM outperformed the standard algorithms because of the heuristics used to refine the search space.

In the acyclic case, the brute force approach would examine \( 2^{T_{n-1}} \) possibilities. In the cyclic case, we use the size of the power set of the finite transition set of \( \mathcal{U} \) as an upper bound.
bound. We will use this set of communication protocols as input to the Nash equilibrium algorithm and establish that we have (i) made the protocols coherent; (ii) selected a set of protocols such that the control problem can be solved; and (iii) submitted the now-coherent set of protocols to ensure that we have a feasible solution wrt criteria for finding a Nash equilibrium.

The search for a sample Nash equilibrium requires a lexicographic ordering on the sizes of the supports of prospective solutions. For instance, if we identify sets of transitions that range in size from 1 (a single transition) to \( k_1 \) that supervisor 1 could communicate to supervisor 2 that would allow the latter supervisor to make all of its correct control decisions, then the support size profile for supervisor 1 will be \( 1, 2, 3, \ldots, k_1 \). We also include the possibility that no communication occurs. Assume that we have a similar range of protocols such that the control problem can be solved; and allow the latter supervisor to make all of its correct control steps: since we initially consider coherent (where coherent versions of communication protocols) and thus, Algorithm 1 has exponential running time.

Defin ition 6: A coherent communication protocol \( \phi_i \in \Phi_i \) is conditionally dominated given the sets of available (coherent) protocols \( \Phi_	ext{remaining} \subseteq \Phi_i \) for the remaining supervisors, if \( \exists \phi'_i \in \Phi_i \) such that \( \forall \phi_w \in \Phi_	ext{remaining}, u_i(\phi_i, \phi_w) > u_i(\phi'_i, \phi_w) > 0 \).

We assume that when determining conditional domination, all communication protocols are coherent, or are made coherent for the purpose of testing conditional domination.

In adapting SEM for the decentralized communication and control problem, Algorithm 1 contains two additional steps: since we initially consider \( \phi \) that are not coherent, we must make the prospective communication protocols coherent (where coherent versions of \( \phi \) are denoted by \( \varphi \) (Line 7) and a check to ensure that the protocols being considered for Nash equilibrium actually solve the control problem (Line 8). In the worst case, there are an exponential number of supports (in the number of possible communication protocols) and thus, Algorithm 1 has exponential running time.

We also need a feasibility program to determine whether or not a potential solution is a true Nash equilibrium. A standard feasibility program from [11], adapted for our notation, is described by Program 1. The input is a set of coherent communication protocols that solve Problem 1 and the output is a protocol \( \phi \) that satisfies Nash equilibrium. The first two constraints ensure that the supervisor has no preference for one protocol over another within the input set and it must not prefer a protocol that is not part of the input set. The third and fourth constraints check that the protocols in the input set are chosen with a non-zero probability, whereas any protocols outside of the input set are chosen with zero probability. The last constraint simply determines that there is a valid probability distribution over the communication protocols.

Algorithm 1 SEM for DES (n=2)

1: for all \( x = (x_1, x_2) \) sorted in increasing order \(|x_1 - x_2|\), and in the event of a tie, sorted in increasing order by \( x_1 + x_2 \) do
2: \( \Phi_{x_1} = \{(\phi_{1,1}, \phi_{1,2}) \in \Phi_1 | |\phi_{1,2}| = x_1\} \)
3: \( \Phi'_2 \leftarrow \{\phi_2 \in \Phi_2 \text{ not conditionally dominated, given } \Phi_{x_1}\} \)
4: if \( \forall \phi_1 \in \Phi_{x_1}, \phi_1 \text{ is not conditionally dominated, given } \Phi'_2 \) then
5: \( \Phi_{x_2} = \{(\phi_{2,1}, \phi_{2,2}) \in \Phi_2 | |\phi_{2,1}| = x_2\} \)
6: if \( \forall \phi_1 \in \Phi_{x_1}, \phi_1 \text{ is not conditionally dominated, given } \Phi_{x_2} \) then
7: \( \Phi \leftarrow \{\{(\phi_{1,1}, \phi_{2,1}) | (\phi_{1,2}, \phi_{2,2}) \text{ coherent}\} \)
8: if \( \Phi \not\subseteq \Phi \) \)
9: if Program 1 is satisfiable for \( \Phi \) then
10: return found NE \( \phi^* \)

Program 1 Feasibility Program TGS (Test Given Supports)

Input: \( \Phi = \Phi_1 \times \cdots \times \Phi_n \)
Output: \( \phi \) is a Nash equilibrium if there exist both \( \phi = (\phi_1, \ldots, \phi_n) \) and \( v = (v_1, \ldots, v_n) \) such that:
1: \( \forall i \in I, \phi_i \in \Phi_i : \sum_{\phi_j \in \Phi_j} p(\phi_j)u_i(\phi_i, \phi_j) = v_i \)
2: \( \forall i \in I, \phi_i \notin \Phi_i : \sum_{\phi_j \in \Phi_j} p(\phi_j)u_i(\phi_i, \phi_j) \geq v_i \)
3: \( \forall i \in I, \phi_i \in \Phi_i : p_i(\phi_i) \geq 0 \)
4: \( \forall i \in I, \phi_i \notin \Phi_i : p_i(\phi_i) = 0 \)
5: \( \forall i \in I : \sum_{\phi_i \in \Phi_i} p_i(\phi_i) = 1 \)

Example 1: We illustrate Algorithm 1 using the automaton in Fig. 1. Suppose that \( L \) is the language generated by the collection of all transitions, whereas \( K \) is the language generated by transitions with solid lines. Let \( \Sigma_{o,1} = \{a\}, \Sigma_{o,2} = \{b\} \) and \( \Sigma_{c,1} = \{a, \sigma\} \), \( \Sigma_{c,2} = \{b, \sigma\} \). Note that \( K \) is not co-observable as there is no supervisor that controls \( \sigma \) that can distinguish between \( ab\sigma \) and \( b\sigma \).

The input to Algorithm 1 is \( \Phi_1 = \{(0,0), (0,\{(1, a, 2\})\), (0,\{(5, a, 6\})\) and \( \Phi_2 = \{(0,0), \{(1, b, 5\}), (0,\{(2, b, 3\})\), (0,\{(1, b, 5\}), (2, b, 3\})\}\). The smallest support size for \( \Phi_1 \) is 0, corresponding to supervisor 1 sending no information at all to supervisor 2, whereas the largest support size is 2, when supervisor...
1 communicates all of its observations to supervisor 2. Similarly, the smallest and largest support sizes for \( \Phi_2 \) are 0 and 2. Thus, we begin by searching profiles where \( x = (0,0) \), followed by \( x = (1,1), x = (2,2), x = (0,1), x = (1,0), x = (1,2), x = (2,1), x = (0,2) \) and \( x = (2,0) \). We will ignore the case when \( x = (0,0) \) as this is the situation when no communication occurs. By assumption, the communication protocol corresponding to this situation, \( \Phi = (\emptyset, \emptyset, \emptyset) \), does not solve the control problem.

**Iteration 1:** \( x = (1,1) \). Line 2: \( \Phi_{x_1} = \{ (\emptyset,\{1,2\}), (\emptyset,\{5,6\}) \} \). Line 3: We can then determine the set \( \Phi_1 \) by calculating those elements of \( \Phi_2 \) that are not conditionally dominated by the elements of \( \Phi_{x_1} \). No elements of \( \Phi_2 \) are conditionally dominated by the elements of \( \Phi_{x_1} \), thus \( \Phi_2 = \Phi_2 \). Note that conditional domination is tested based on coherent communication policies. We temporarily transform elements of \( \Phi_2 \) and \( \Phi_{x_1} \) so that they satisfy coherence. For example, when \( \phi_1 = (\emptyset,\{5,6\}) \) and \( \phi_2 = (\{2,3\},\emptyset) \), first make \( \phi_2 \) coherent wrt \( \phi_1 \), so that \( \phi_2 \) becomes \( (\{1,5,2,3\},\emptyset) \) and now \( \phi_1 \) is already coherent wrt the coherent \( \phi_2 \). Line 4: No elements of \( \Phi_{x_1} \) are conditionally dominated given \( \Phi_2 \). Line 5: \( \Phi_{x_2} = (\{\{1,5,2,3\},\emptyset\}, \{\{2,3\},\emptyset\}) \). Line 6: None of the elements of \( \Phi_{x_1} \) are conditionally dominated by the elements of \( \Phi_{x_2} \). Line 7: \( \Phi \) contains the following coherent communication protocols:

- \( (\emptyset,\emptyset,\emptyset) \)
- \( (\emptyset,\{1,2\},\emptyset) \)
- \( (\emptyset,\emptyset,\emptyset) \)

**Line 8:** Each element of \( \Phi \) solves the control problem. 
**Line 9:** Since each supervisor has three choices that are equally likely, let \( p_i(\phi_i) = \frac{1}{3} \) for each \( \phi_i \in \Phi_i \). Program TGS returns the Nash equilibrium communication protocol \( \phi^* = (\emptyset,\{1,2\},\emptyset) \).

**B. Nash equilibrium for more than two supervisors**

Algorithm 2 is a modification of Algorithm 1 to accommodate the case of more than two supervisors. One subtle difference in Algorithm 2 is the change in the ordering of the support sizes: sorted first by size and then by balance. The justification for this decision comes from [11]: when there are more than two players (supervisors), balance is not as important a criterion when finding a sample Nash equilibrium. Additionally, the algorithm relies on recursive backtracking (Procedure 1) to explore the search space.

For the case of more than two supervisors, it is slightly more complicated to remove dominated communication protocols. The input to Procedure 2 (line 11 in Procedure 1) is now the set of domains for support of each supervisor. When the support for a supervisor is instantiated, the domain contains only the instantiated support. For all other supervisors, the domain contains the supports of size \( x_i \) that were not previously eliminated by earlier calls to this procedure.

**Example 2:** We illustrate Algorithm 2 using the automaton in Fig. 2. As before, let \( L \) be the language generated by the collection of all transitions, and \( K \) be the language of only the solid-line transitions. Suppose that \( \Sigma_{o,1} = \{a\}, \Sigma_{o,2} = \{b\}, \) and \( \Sigma_{o,3} = \{c\} \), while \( \Sigma_{c,1} = \{a,\sigma\}, \Sigma_{c,2} = \{b,\sigma\} \), and \( \Sigma_{c,3} = \{c,\sigma\} \). \( K \) is not co-observable since none of the supervisors can make the correct control decisions regarding \( \sigma \).

The support sizes for each supervisor ranges from 0 to 3, thus when checking the possible support sizes there are 64 possibilities, beginning with \( x = (0,0,0) \), which we have previously indicated we would ignore, and ending with \( x = (3,3,3) \). The first three supports (all with a cumulative sum of 1 and a max difference of 1) to examine are of size \( (1,0,0) \), \( (0,1,0) \) and \( (0,0,1) \). We will begin with \( (0,1,0) \).

**Algorithm 2, iteration 1:** \( x = (0,1,0) \). Line 2: \( \Phi_1 = \emptyset \). Line 3: \( D_{x_1} = \{({\emptyset},{\emptyset},{\emptyset})\} \) . \( D_{x_2} = \{(1, b, 6)\} \) \( \{3, b, 4\} \) \( \{0, 0\} \) \( \{10, b, 11\} \). \( D_{x_3} = \{0, 0\} \) \( \{3, b, 4\} \) \( \{0, 0\} \) \( \{10, b, 11\} \). Line 4: Call to Procedure 1 with \( \Phi'_1 \), \( D_{x_1} \), and \( 1 \).

**Procedure 1, call for \( i = 1 \):** Line 9: \( \Phi'_1 = \{({\emptyset},{\emptyset},{\emptyset})\} \). Line 10: \( D_{x_1} = \emptyset \) . Line 11: Call to Procedure 2 with \( \Phi'_1 \), \( D_{x_2} \) and \( D_{x_3} \).

**Procedure 2, call for \( D_{x_1} = \{({\emptyset},{\emptyset},{\emptyset})\} \) :**
Procedure 2 Iterated Removal of Dominated Communication Protocols (IRDCP)

Input: $D_x = (D_{x1}, \ldots, D_{xn})$
Output: Updated domains or failure

1. repeat
   2. $dominated \leftarrow false$
   3. for all $i \in I$ do
   4. for all $\phi_i \in D_{xi}$ do
   5. for all $\phi'_i \in \Phi_i$ do
   6. if $\phi_i$ is conditionally dominated by $\phi'_i$ given $D_{x'}$, then
   7. $D_{x} \leftarrow D_{x} \setminus \{\phi_i\}$
   8. $dominated \leftarrow true$
   9. if $D_{x} = \emptyset$ then
      10. return failure
   11. until $dominated = false$
   12. return $D_{x}$

Fig. 2. An example for three communicating supervisors.

\[
\{(\{1, b, 6\}, \emptyset, \emptyset), (\{3, b, 4\}, \emptyset, \emptyset), (\{10, b, 11\}, \emptyset, \emptyset), (\emptyset, \emptyset, (1, b, 6)), (\emptyset, \emptyset, (3, b, 4)), (\emptyset, \emptyset, (10, b, 11))\} \times \{(\emptyset, \emptyset, \emptyset), (\emptyset, \emptyset, \emptyset)\}.
\]

When $i = 2$ and 3, no communication protocols are removed by these procedure calls. Then make the recursive call to Procedure 1 with $\Phi', D_x$, and 2.

Procedure 1, return to line 11. Line 12: Recursive call to Procedure 1 with $\Phi', D_x$ and 4.

Procedure 1, call for $i = 4$. Line 2: At this point, there are six different possible communication protocols: $\{(\emptyset, \emptyset, \emptyset)\} \times \{(\{1, b, 6\}, \emptyset, \emptyset), (\{3, b, 4\}, \emptyset, \emptyset), (\{10, b, 11\}, \emptyset, \emptyset)\} \times \{(\emptyset, \emptyset, \emptyset)\}$. When we make these combinations coherent, $\Phi'$ contains two unique protocols:

- $((\emptyset, \emptyset, \emptyset), (\{1, b, 6\}, (3, b, 4), (10, b, 11)), (\emptyset, \emptyset), (\emptyset, \emptyset, \emptyset))$,
- $((\emptyset, \emptyset, \emptyset), (\emptyset, \emptyset, (1, b, 6), (3, b, 4), (10, b, 11)), (\emptyset, \emptyset, \emptyset))$.

Line 3: Only one of these communication protocols solves the control problem, so now $\Phi' = \{(\emptyset, \emptyset, \emptyset), (\emptyset, \emptyset, (1, b, 6), (3, b, 4), (10, b, 11))\}$. Line 4: Call to Program 1 with $\Phi'$ returns NE $\phi^* = \{(\emptyset, \emptyset, \emptyset), (\emptyset, \emptyset, (1, b, 6), (3, b, 4), (10, b, 11))\}$. This is a pure strategy.

IV. CONCLUSION

We have presented two algorithms, adapted from the original game theory context [11], to find a sample Nash equilibrium (e.g., locally-optimal solution) for communication protocols that solve Problem 1. The algorithms terminate when a first Nash equilibrium point is found. Based on experimental evidence in [11], the heuristic for finding equilibrium points quickly is captured in the ordering of supports. The SEM algorithm is prejudiced towards finding Nash equilibrium points in balanced support sizes, i.e., when each agent communicates a similar number of messages. The algorithms were demonstrated with simple acyclic examples that assumed a simple uniform cost associated with the utility function. To extend the results to cyclic examples, the algorithms need not change. Instead, the utility function must be updated to accommodate more appropriate cost models.

REFERENCES