Distributed and Cooperative Task Processing: Cournot Oligopolies on a Graph

Theodore P. Pavlic, *Member, IEEE,* and Kevin M. Passino, *Fellow, IEEE*

***Abstract*—This paper introduces a novel framework for the design of distributed agents that must complete externally generated tasks but also can volunteer to process tasks encountered by other agents. To reduce the computational and communication burden of coordination between agents to perfectly balance load around the network, the agents adjust their volunteering propensity asynchronously within a fictitious trading economy. This economy provides incentives for nontrivial levels of volunteering for remote tasks, and thus load is shared. Moreover, the combined effects of diminishing marginal returns and network topology lead to competitive equilibria that have task reallocations that are qualitatively similar to what is expected in a load-balancing system with explicit coordination between nodes. In the paper, topological and algorithmic conditions are given that ensure the existence and uniqueness of a competitive equilibrium. Additionally, a decentralized distributed gradient-ascent algorithm is given that is guaranteed to converge to this equilibrium while not causing any node to over-volunteer beyond its maximum task-processing rate. The framework is applied to an autonomous-air-vehicle example, and connections are drawn to classic studies of the evolution of cooperation in nature.**

***Index Terms*—Agents and autonomous systems, distributed control, game theory, load balancing, network analysis and control, parallel algorithms.**

I. Introduction

We consider the problem of designing strategies to dynamically route externally generated tasks around a network to efficiently share the processing load of those tasks. In many cases, an optimal solution can be found either with centralized methods or methods that synchronize computation between networked nodes. To reduce the communication burden, we propose a totally asynchronous strategy that can achieve an adequate, if not optimal, task allocation that requires significantly less coordination between nodes. The load balancing that emerges in this case is a product of incentives built into a cooperation-trading economy as well as topological feedbacks. Thus, this paper shows how cooperation can be stabilized on networks of selfish agents that continuously climb local utility gradients.

Manuscript received July 13, 2012; revised January 18, 2013; accepted June 21, 2013. Date of publication July 16, 2013; date of current version May 13, 2014. This work was supported by the National Science Foundation under Grant EECS-0931669 and Grant CCF-1012029. Recommended by Associate Editor T. Vasilakos.

T. P. Pavlic is with the School of Life Sciences, Arizona State University, Tempe, AZ 85287 USA (e-mail: tpavlic@asu.edu).

K. M. Passino is with the Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail: passino.1@osu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2013.2271776

In particular, we consider a network of autonomous agents with some agents being responsible for processing tasks from one or more external sources. When a task arrives at one of these agents, the agent may advertise the task to other agents connected to it. If none of the connected agents volunteer to process the task, it must be processed by the advertising agent; otherwise,the task is processed by one of the volunteering agents. Agents that volunteer for tasks may themselves be connected to incoming task flows for which they can advertise task encounters. In general, an agent in the network may advertise task encounters to others, volunteer to process advertised tasks from others, or do both. Our challenge is to define a distributed asynchronous algorithm for automatically tuning how often agents volunteer to process advertised tasks so that the collection of volunteering tendencies across the network converges to a nontrivial Nash (i.e., competitive) equilibrium with desirable load-balancing features. In the rest of this section, we will review the existing cooperative processing works and discuss why those approaches are not adequate for solving the problem formulated here.

Grid computing [1] is one existing approach for achieving cooperative task processing across a group of networked task-processing agents. System designers work under the assumption of heterogeneous agents with conflicting priorities. They borrow from the economic theories of mechanism design [2, Chapter 23] and implementation theory [3, Chapter 10] to design mechanisms (e.g., brokering agents) and protocols that either encourage resource sharing [4]–[7] or discourage exploitation [8], [9] among groups of agents. The common element of these different methods of distributed algorithmic mechanism design (DAMD) [10] is that the designer has no direct control over individual agents; instead, she controls the structure of the interactions between given agents on a given network. Hence, DAMD is not appropriate for the design of the task-processing networks themselves.

Methods exist for the design of networks of interconnected task-processing agents that have desirable task flow characteristics. For example, a flexible manufacturing system (FMS) includes several machines that switch their current processing to one of several input task flows and then produce output task flows for other machines in the system. Reference [11] shows that distributed scheduling policies exist that guarantee such systems will have finite upper bounds on all buffers of tasks. Similarly, Cruz [12] shows how special network elements can be combined to form queueing systems with output traffic flows that are guaranteed to have finite burstiness constraints

so long as the input flows also satisfy similar constraints. These methods are not intended to describe how agents can dynamically adjust task flow to exploit unused processing ability on idle connected agents.

Because an optimal task flow configuration may be unknown, inaccessible, or changing over time, task-processing agents may need to use feedback to acquire and stabilize the optimal task-handling behavior. For example, a set of autonomous air vehicles (AAV) deployed for distributed search, surveillance, or task processing can coordinate their actions in order to converge on a holistically optimal behavior [13]–[15]. However, the coordination required between agents can be prohibitive. Additionally, the single optimality criterion being maximized ignores fatigue on individual agents. For example, in a smart power grid [16], it may be desirable for distributed power stations to share load; however, a single overloaded station should not result in a cascade of self-sacrificing failures. Here, noncooperative game theory is used to develop totally asynchronous distributed algorithms for task-processing agents that both respect local processing priorities while also sharing the processing burden of loaded neighbors.

Noncooperative game theory has been traditionally used to design optimal control strategies [17]; however, it can also be used to design simple selfish strategies that nonetheless assist neighbors. Several such techniques already exist for designing policies on nodes in *ad hoc* multihop communication networks [18]–[20]. In these cases, nodes can forward packets from other nodes in order to reduce network congestion or improve communication diversity, but nodes resist using all local resources for assisting other nodes. A salient feature of these forwarding networks is that packets can be duplicated or dropped at any time. Hence, these networks are ill-equipped to model task-processing scenarios where tasks that enter the network must be assigned and processed by exactly one agent. Instead, our approach passes volunteering requests around a network and uses an economics-inspired task-processing network game to determine how best to respond to these requests. The resulting volunteering policy is sensitive to both local processing requests and the presence of other agents on the network that can volunteer as well.

This paper is organized as follows. In Section II, the task-processing network (TPN) framework is defined and example TPNs are described. The optimization game is presented in Section III, and an asynchronous distributed computation method that ensures convergence to the game's Nash equilibrium is given in Section IV. In Section V, example results from an example task-processing network of autonomous air vehicles are presented. The resulting cooperation policy is analyzed for its ability to balance load around the vehicle network, and the price of anarchy (i.e., independent competition) is discussed. Conclusions and future areas of research are given in Section VI.

II. Task-Processing Network

In the following, we use real numbers $\mathbb{R}$, natural numbers $\mathbb{N} \triangleq \{1, 2, \dots\}$, whole numbers $\mathbb{W} \triangleq \{0, 1, 2, \dots\}$, and derived symbols like the nonnegative real numbers $\mathbb{R}_{\geq 0}$.

Take a finite set $\mathcal{A} \subset \mathbb{N}$ of task-processing agents. For each agent $i \in \mathcal{A}$, there exists a finite and possibly empty set $\mathcal{Y}_i \subset \mathbb{N}$ of task types that arrive at the agent, and for each $k \in \mathcal{Y}_i$, tasks of type k arrive at agent i from an external source at average rate $\lambda_i^k \in \mathbb{R}_{>0}$. Each external source of tasks is assumed to be independent of all other sources. Each agent $i \in \mathcal{A}$ has a maximum processing rate $R_i \in \mathbb{R}_{>0}$, and it is assumed that $R_i \geq \sum_{k \in \mathcal{Y}_i} \lambda_i^k$.

When a task arrives at an agent, the agent can convey the arrival to several connected agents that may volunteer to process the task. In particular, task arrivals are communicated along the directed edges in set $\mathcal{P} \subseteq \{(i, j) \in \mathcal{A}^2 : i \neq j\}$. For each agent $i \in \mathcal{A}$, set $\mathcal{V}_i \triangleq \{j \in \mathcal{A} : (j, i) \in \mathcal{P}\}$ is the collection of conveyors that advertise task arrivals to agent i, and set $\mathcal{C}_i \triangleq \{j \in \mathcal{A} : (i, j) \in \mathcal{P}\}$ is the collection of cooperators that agent i can advertise arrivals to. Furthermore, $\mathcal{V} \triangleq \{j \in \mathcal{A} : \mathcal{C}_j \neq \emptyset\} = \bigcup_{i \in \mathcal{A}} \mathcal{V}_i$ and $\mathcal{C} \triangleq \{i \in \mathcal{A} : \mathcal{V}_i \neq \emptyset\} = \bigcup_{j \in \mathcal{A}} \mathcal{C}_j$ are respectively the sets of all conveyors and cooperators in the network. Assume that:

1) for each conveyor $j \in \mathcal{V}$, there exists task type $k \in \mathcal{Y}_j$ with $\pi_j^k \neq 0$ where $\pi_j^k \in [0, 1]$ is the probability that conveyor j advertises an incoming k-type task to its connected cooperators $\mathcal{C}_j$. If $j \in \mathcal{V}$ does not advertise the task, it will be processed by agent j;
2) for each cooperator $i \in \mathcal{C}$, $\gamma_i \in [0, 1]$ is the probability that agent i will volunteer for an advertised task from one of its connected conveyors $\mathcal{V}_i$. Any task arriving at conveyor $j \in \mathcal{V}$ that is advertised to cooperators $\mathcal{C}_j$ will be processed with uniform probability by exactly one of the volunteers; if no cooperators volunteer for the task, then it is processed by conveyor j.

The graph $\mathcal{G} \triangleq (\mathcal{A}, \mathcal{P})$, rates, and probabilities defined above characterize a TPN. In principle, each cooperator could have an independent volunteering probability for each type of task advertised to it. However, the convergence results later in this paper are greatly simplified in the case where each node has a single scalar decision variable. Thus, the more general case is left as a future direction.

The simple TPN shown in Fig. 1 represents an FMS similar to the systems described by [11]. Tasks of types 1, 2, and 3 arrive according to independent Poisson processes. Type-1 and type-2 tasks arrive at agent 1, and all three types of tasks arrive at agent 2. For tasks of type $k \in \mathcal{Y}_1 = \{1, 2\}$, agent 1 advertises task arrivals to agents 3 and 4 with probability π_1^k. Likewise, agent 2 advertises arrivals of tasks of type $k \in \mathcal{Y}_2 = \{1, 2, 3\}$ to agents 4 and 5 with probability π_2^k. The advertising probabilities for each task type can be chosen based on the specialized abilities of each agent. Each agent $i \in \{3, 4, 5\}$ volunteers for an advertised task with probability γ_i independent of task type. Hence, in this TPN, agents 1 and 2 are conveyors and agents 3, 4, and 5 are cooperators. In an FMS more like the ones described by , tasks enter at one node and are processed in an ordered sequence by a set of nodes. In these cases, the upstream nodes are not conveyors and the downstream nodes are not cooperators. Instead, the upstream nodes are the external task generators for the downstream conveyors. Thus, the TPN described in this paper represents

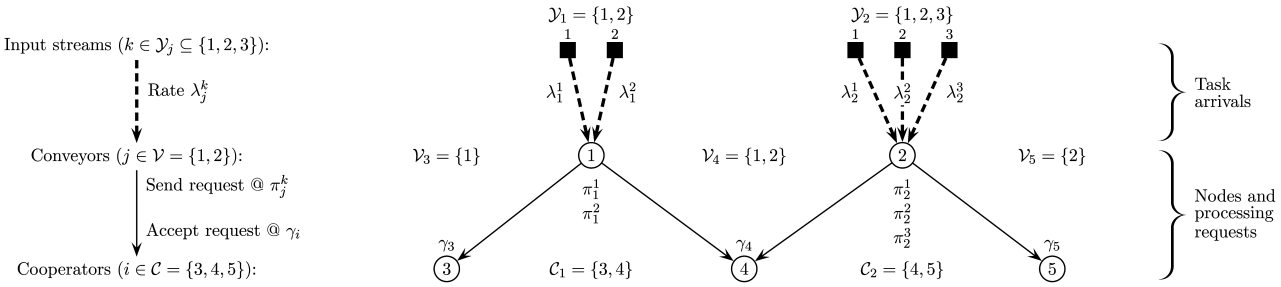

Fig. 1. Simple flexible manufacturing system example.

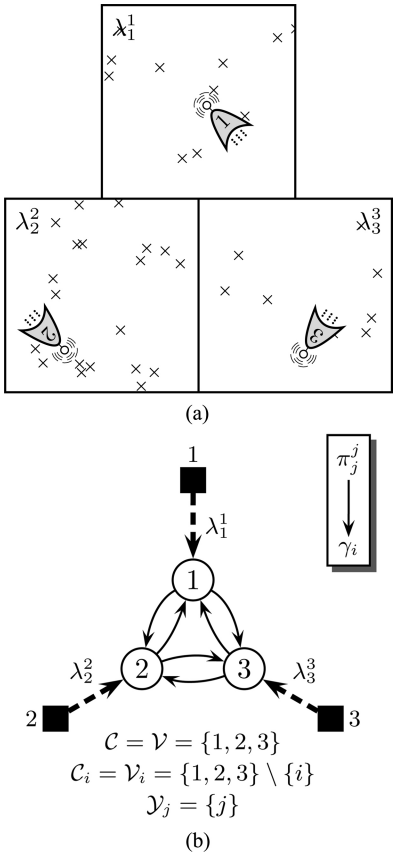

Fig. 2. Task-processing network formed by three autonomous air vehicles (AAVs). (a) AAVs-patrolled territories (b) Corresponding task-processing network.

how nodes of equal rank in the pipeline would share load generated by upstream nodes in the pipeline.

In the FMS example above, the set of conveyors and the set of cooperators are disjoint. In a general TPN, an agent can be both a cooperator and a conveyor. For example, the fully connected TPN shown in Fig. 2(b) models an AAV patrol scenario shown in Fig. 2(a) that is similar to others in resource-allocation literature [13]–[15]. Each AAV $i \in \{1, 2, 3\}$ continuously searches its territory for tasks (e.g., targets) to process, and these tasks are found at rate $\lambda_i^i > 0$. When a task is found, the AAV advertises the task to both of its neighbors. If neither neighbor volunteers for processing, the AAV processes the task itself. In this fully connected topology, all agents are both cooperators and conveyors. Although this network has several cycles, tasks do not move around the network—if a volunteering cooperator is given a task for processing, it cannot readvertise that task to its own neighboring cooperators; it must process the task itself.

TPNs describe a broad range of applications. The AAV example also models groups of networked processors [21] or mobile software agents [22]–[26] that patrol for tasks to process. Additionally, by converting encounter rates to energetic rates (i.e., power demand), TPNs can model the behavior of smart power grids [16] made up of stations that request assistance from neighbors. Thus, cooperator stations adjust additional supply provided in response to demand requests from remote conveyor stations.

III. Cooperation Game Among Selfish Agents

In a TPN, the cooperation willingness is the probability $\gamma_i \in [0, 1]$ that cooperator $i \in \mathcal{C}$ will volunteer for an advertised task from its connected conveyors. Here, to simplify the analysis later, this decision variable is independent of task type. Assuming that it is impractical for agents to coordinate to maximize global utility, each cooperator adjusts its cooperation willingness in a distributed fashion. So each agent independently chooses a cooperation policy that maximizes its individual utility (i.e., agents are selfish). Hence, optimality is given in terms of the Nash equilibrium [27, Section 3.5.1].

To inform each cooperator how to choose this policy, cost and rewards are assigned to agent operations in a common currency (e.g., proportional to dollars of net profit) that is called points here. In particular:

1) conveyor $i \in \mathcal{V}$ pays no cost for a locally generated task of type $k \in \mathcal{Y}_i$ if it is processed by a $\mathcal{C}_i$ cooperator; otherwise, the task is processed locally and i pays cost c_i^k;
2) if cooperator $j \in \mathcal{C}_i$ volunteers and is selected to process a task of type $k \in \mathcal{Y}_i$ from conveyor $i \in \mathcal{V}_j$, then cooperator j pays cost c_{ij}^k to process that task.

The cost described in 2 is only paid if the cooperator volunteers and is selected to process a task. Given that cooperator $i \in \mathcal{C}$ volunteers to process a task from conveyor $j \in \mathcal{V}_i$, the probability of being selected to process that task is

$$P_s(j|i) \triangleq \sum_{s=0}^{|\mathcal{C}_j|-1} \frac{1}{1+s} \sum_{\substack{\mathcal{B} \subseteq \mathcal{C}_j \setminus \{i\} \\ |\mathcal{B}|=s}} \left(\left(\prod_{k \in \mathcal{B}} \gamma_k \right) \left(\prod_{\substack{k \in \mathcal{C}_j \setminus \mathcal{B} \\ k \neq i}} (1-\gamma_k) \right) \right),$$

and so cooperation willingness from other cooperators in $\mathcal{C}_j$ reduces the chances that cooperator $i \in \mathcal{C}_j$ will be selected and have to pay the processing cost.

However, these costs and benefits alone do not provide cooperators with any incentive to volunteer to process conveyor tasks, and so a payment mechanism is required. Consider conveyor $j \in \mathcal{V}$ and task type $k \in \mathcal{Y}_j$. If one or more cooperators in $\mathcal{C}_j$ volunteer frequently to process requests from agent j, the other cooperators in the set should conserve resources by volunteering infrequently. To ensure this qualitative behavior, each cooperator $i \in \mathcal{C}_j$ receives volunteering payment $q_{ij}^k p_j^k(Q_j)$ from conveyor $j \in \mathcal{V}_i$ where

1) $Q_j \triangleq \sum_{k \in \mathcal{C}_j} \gamma_k$ is the total quantity of cooperation willingness available to conveyor j.
2) $p_j^k(Q_j)$ is a decreasing payment function that represents the price that conveyor j pays to its connected cooperators each time they volunteer for a task of type $k \in \mathcal{Y}_j$.
3) $q_{ij}^k \in \mathbb{R}_{>0}$ is a value factor that scales payment $p_j^k(Q_j)$ from conveyor j into the currency of cooperator $i \in \mathcal{C}_j$ (i.e., i perceives $q_{ij}^k p_j^k(Q_j)$ value from the contribution $p_j^k(Q_j)$ from j).

So if any cooperator $i \in \mathcal{C}_j$ increases its cooperation willingness γ_i, it increases how often it receives payment $p_j^k(Q_j)$ while also decreasing the payment itself. For each cooperator $i \in \mathcal{C}_j$, these two pressures encourage cooperation (i.e., equilibrium $\gamma_i^* > 0$) and resource conservation (i.e., $\gamma_i^* < 1$).

To maximize net points earned over a long run time, each agent chooses a policy that maximizes its own expected rate of point accumulation. So for a given vector $\underline{\gamma} = [\gamma_{c_1}, \gamma_{c_2}, \ldots, \gamma_{c_{|\mathcal{C}|}}] \in [0,1]^{|\mathcal{C}|}$ of cooperation policies (where unique $c_k \in \mathcal{C}$ for all $k \in \{1, \ldots, |\mathcal{C}|\}$), the utility (i.e., long-term rate of point gain) returned to cooperator $i \in \mathcal{C}$ is

$$U_i(\underline{\gamma}) \triangleq \underbrace{-c_i \quad (1-\gamma_j)}_{\text{Conveyor costs}} + \underbrace{\gamma_i \sum_{j \in \mathcal{V}_i} \left(p_{ij}(Q_j) - P_s(j|i) c_{ij} \right)}_{\text{Cooperator part} - \gamma_i \text{ and } Q_j \text{ vary with } \gamma_i} \tag{1a}$$

where

$$c_i \triangleq \sum_{k \in \mathcal{Y}_i} \lambda_i^k \pi_i^k c_i^k, \tag{1b}$$

$$p_i(Q_i) \triangleq \sum_{k \in \mathcal{Y}_i} \lambda_i^k \pi_i^k p_i^k(Q_i) \tag{1c}$$

are the costs and benefits of local processing on $i \in \mathcal{V}$, and

$$c_{ij} \triangleq \sum_{k \in \mathcal{Y}_j} \lambda_j^k \pi_j^k c_{ij}^k, \tag{1d}$$

$$p_{ij}(Q_j) \triangleq \sum_{k \in \mathcal{Y}_j} \lambda_j^k \pi_j^k q_{ij}^k p_j^k(Q_j) \tag{1e}$$

are the costs and benefits to $i \in \mathcal{C}$ for volunteering for tasks exported from $j \in \mathcal{V}_i$.

Consider a cooperator $i \in \mathcal{C}$ that volunteers to process a task advertised from conveyor $j \in \mathcal{V}_i$. In the cooperator's utility function U_i, the conditional selection probability $P_s(j|i)$ scales the cost of processing the advertised task, and so for $j \in \mathcal{V}_i$, the impact of cost rate c_{ij} decreases as other cooperators from $\mathcal{C}_j$ increase their own cooperation willingness. So for a conveyor $j \in \mathcal{V}$, its connected cooperators $\mathcal{C}_j$ form a Cournot oligopoly [28] (i.e., a set of independent agents that provide a service for a demand-driven price) with a positive externality [29] (i.e., the cost of processing decreases as more cooperators enter the market). The underbraced cooperator part of (1a) shows that cooperator i must set its cooperation willingness γ_i (i.e., its quantity of supplied cooperation) based on the summed returns from several such markets.

IV. Distributed Computation of the Nash Equilibrium

Let $n \triangleq |\mathcal{C}|$ be the number of cooperators that individually adjust cooperation willingness to maximize local utility. To ensure that each cooperator $i \in \mathcal{C}$ is operating below its maximum processing rate R_i, the cooperation level $\gamma_i \in [0, \gamma_i^{\max}]$ where

$$\gamma_i^{\max} \triangleq \max\left\{0, \min\left\{1, \frac{R_i - \sum_{k \in \mathcal{Y}_i} \lambda_i^k}{\sum_{j \in \mathcal{V}_i} \sum_{k \in \mathcal{Y}_j} \lambda_j^k}\right\}\right\}$$

is the cooperation level that would bring the processing rate on i to $R_{\max}$ if no other cooperators volunteered for tasks. So, because there is no coordination between players, the n-dimensional play space is the Cartesian product $\mathcal{X} \triangleq \prod_{i \in \mathcal{C}} [0, \gamma_i^{\max}]$ where $\mathcal{X} \subseteq [0,1]^n$, and the collection of cooperation policies across all cooperators is the vector $\underline{\gamma} \triangleq [\gamma_{c_1}, \gamma_{c_2}, \ldots, \gamma_{c_n}] \in \mathcal{X}$ (where unique $c_k \in \mathcal{C}$ for all $k \in \{1, 2, \ldots, n\}$). For each $i \in \mathcal{C}$, it is assumed that the utility function $U_i : \mathcal{X} \to \mathbb{R}$ is twice continuously differentiable, and so, by Weirstrass' theorem, U_i is bounded above and below and achieves its extrema. Following [27], Proposition 5.7 from Chapter 3, the Nash equilibria of the cooperation game can be found by solving n separate 1-D variational inequality problems. In particular, $\underline{\gamma}^* \in \mathcal{X}$ is a Nash equilibria of the cooperation game if and only if, for all $i \in \mathcal{C}$

$$(\gamma_i - \gamma_i^*) \nabla_i U_i(\underline{\gamma}^*) \leq 0 \quad \text{for all } \gamma_i \in \mathcal{X} \tag{2}$$

where the block gradient (i.e., the ith row of the gradient)

$$\nabla_i U_i(\underline{\gamma}) = \sum_{j \in \mathcal{V}_i} \left(\overbrace{p_{ij}(Q_j) + \gamma_i \frac{\partial}{\partial \gamma_i}(\gamma_i p_{ij}(Q_j))}^{p_{ij}(Q_j) + \gamma_i p_{ij}(Q_j)} - P_s(j|i) c_{ij} \right).$$

So in a local neighborhood of the Nash equilibrium $\gamma^* \in \mathcal{X}$, any unilateral perturbation of a coordinate of γ^* will result in equal or reduced utility.

The solution to the n simultaneous nonlinear equations in (2) is not guaranteed to exist in general and may be difficult to find analytically even when it does exist. However, variational inequalities over product spaces are well suited for parallel and asynchronous computation [27]. Under special conditions on each utility function, a unique Nash equilibrium is guaranteed to exist, and each of its coordinates in (2) can be computed independently in the distributed and asynchronous fashion described by Definition 1.

Definition 1 (Totally asynchronous distributed iteration): Take $\{c_1, c_2, \ldots, c_n\}$ to be the set $\mathcal{C}$ of the n

distinct cooperators. Let $\mathcal{T} \triangleq \mathbb{W}$ to be the indices of a sequence of physical times, and let $(\underline{\gamma}(t))_{t\in\mathcal{T}} \triangleq ([\gamma_{c_1}(t), \gamma_{c_2}(t), \ldots, \gamma_{c_n}(t)]\ )_{t\in\mathcal{T}}$ be a sequence of iterated calculations in the $\mathcal{X}$ play space. For each $i \in \mathcal{C}$, subset $\mathcal{T}^i \subseteq \mathcal{T}$ corresponds to the times when coordinate $\gamma_i(t)$ is computed. Additionally, for each $i, j \in \mathcal{C}$ and each $t \in \mathcal{T}$, there is an index $\tau_j^i(t) \in \mathcal{T}$ of the least-outdated version of coordinate γ_j available for the computation of coordinate γ_i with transition mapping $T_i : \mathcal{X} \to [0, \gamma_i^{\max}]$ at time t such that $0 \le \tau_j^i(t) \le t$. That is, an outdated state estimate

$$\begin{aligned}\underline{\gamma}^i(t) &\triangleq [\gamma_{c_1}^i(t), \gamma_{c_2}^i(t), \ldots, \gamma_{c_n}^i(t)] \\ &\triangleq [\gamma_{c_1}(\tau_{c_1}^i(t)), \gamma_{c_2}(\tau_{c_2}^i(t)), \ldots, \gamma_{c_n}(\tau_{c_n}^i(t))]\end{aligned}$$

is available for the computation $\gamma_i(t+1) = T_i(\underline{\gamma}^i(t))$ for each $t \in \mathcal{T}$ and $i \in \mathcal{C}$. It is assumed that:

1) set $\mathcal{T}^i$ is countably infinite (i.e., $|\mathcal{T}^i| = |\mathcal{T}| = |\mathbb{N}|$) for all $i \in \mathcal{C}$;
2) if subsequence (t_k) of $\mathcal{T}^i$ is such that $\lim_{k\to\infty} t_k = \infty$, then $\lim_{k\to\infty} \tau_j^i(k) = \infty$ for all $i, j \in \{1, 2, \ldots, n\}$. That is, $\liminf_{t\to\infty} \tau_j^i(t) = \infty$ for all $i, j \in \{1, 2, \ldots, m\}$.

For all $t \in \mathcal{T}$, sequence $(\underline{\gamma}(t))$ is generated by the totally asynchronous distributed iteration (TADI)

$$\gamma_i(t+1) \triangleq \begin{cases} T_i(\underline{\gamma}^i(t)), & \text{if } t \in \mathcal{T}^i, \\ \gamma_i(t), & \text{if } t \notin \mathcal{T}^i \end{cases} \tag{3}$$

where $\underline{\gamma}(t) \triangleq [\gamma_{c_1}(t), \gamma_{c_2}(t), \ldots, \gamma_{c_n}(t)]$. For each $i \in \mathcal{C}$, the transition mapping $T_i : \mathcal{X} \to [0, \gamma_i^{\max}]$ in (3) is defined by

$$T_i(\underline{\gamma}) \triangleq \min\{\gamma_i^{\max}, \max\{0, \gamma_i + \sigma_i \nabla_i U_i(\underline{\gamma})\}\}$$

where $\sigma_i \in \mathbb{R}_{>0}$ is a step-size parameter that scales movement along the utility gradient $\nabla_i U_i$.

A. Conditions for Distributed Convergence

The TADI-generated $(\underline{\gamma}(t))$ sequence represents the collective motion of n self-interested agents that each climb their respective utility gradient in order to maximize their expected rate of point return. That is, (3) may be viewed as a dynamical system model of coupled agents that each take independent actions. In particular, the conditional probability $P_s(j|i)$ is bounded away from zero, and so assuming that $c_{ij} > 0$ and payment $p_{ij} \equiv 0$ for all $i, j \in \mathcal{A}$, the response of the system reaches $\underline{\gamma}(T) = \underline{0}$ in some finite time $T \in \mathbb{W}$. That is, the intrinsic agent behavior is not to cooperate. For each $i \in \mathcal{C}$, it is desirable to find a control law, which is implemented through the choice of payment function, to destabilize the no-cooperation equilibrium and provide feedback to stabilize the Nash equilibrium. It will be shown that Definition 2 gives sufficient characteristics for such stabilizing payment functions. Moreover, Proposition 1 provides a simpler criterion to test for such functions.

Definition 2 (Stabilizing payment function): For $k \in \mathbb{N}$, a stabilizing payment function (SPF) $p : [0, k] \to \mathbb{R}$ is a twice-continuously differentiable function such that:

1) it is strictly decreasing. In particular, $p\,(Q) \triangleq \partial p(Q)/\partial Q < 0$ for all $Q \in [0, k]$;
2) it is convex. In particular, $p\ (Q) \triangleq \partial^2 p(Q)/\partial Q^2 \ge 0$ for all $Q \in [0, k]$;
3) its convexity is eventually dominated by its slope. In particular,

$$\gamma p\ (Q) \le -p\,(Q), \quad \text{for all } Q \in [\gamma, k - (1-\gamma)] \text{ with } \gamma \in [0, \gamma_i^{\max}].$$

Proposition 1 (Sufficient conditions for stabilization): Take $k \in \mathbb{N}$ and function $p : [0, k] \to \mathbb{R}$. If $0 \le p\ (Q) < -p\,(Q)$ for all $Q \in [0, k]$, then p is a stabilizing payment function.

The set of SPFs is closed under conical combinations. So for $i \in \mathcal{C}$, if p_{ij} is an SPF for all $j \in \mathcal{V}_i$, then the sum $\sum_{j\in\mathcal{V}_i} p_{ij}(Q_j)$ is itself an SPF. Additionally, by the definition of $p_{ij}(Q_j)$ in (1e), if $p_j^k(Q_j)$ is an SPF for all $j \in \mathcal{V}$ and $k \in \mathcal{Y}_j$, then $p_{ij}(Q_j)$ will also be an SPF for all $i \in \mathcal{C}$ and $j \in \mathcal{V}_i$.

Four example SPFs are shown in Fig. 3. Each payment function meets the conditions of Proposition 1; however, by the weaker condition 3 of Definition 2, the p_h function in (d) can also have $\varepsilon \ge \kappa$ and still be an SPF.

Convergence to the Nash equilibrium depends not only on the structure of the payment functions but also on the structure of the TPN graph itself. Sufficient convergence conditions for a TPN network and its payment functions are given in Theorem 1, which uses Definition 3 to describe the topological constraints on the TPN graph.

Definition 3 (k-conveyor): Conveyor $i \in \mathcal{V}$ is a *k-conveyor* if it has exactly $k \in \mathbb{N}$ outgoing connections to cooperators (i.e., if $k = |\mathcal{C}_i|$);

Theorem 1 (Convergence of cooperation): Assume that:

1) for all $i \in \mathcal{C}$ and $j \in \mathcal{V}_i$, p_{ij} is a stabilizing payment function;
2) for all $j \in \mathcal{V}$, $|\mathcal{C}_j| \le 3$ (i.e., no conveyor can have more than 3 outgoing links to cooperators);
3) for $i \in \mathcal{C}$ and $j \in \mathcal{V}_i$, if j is a 3-conveyor, then there must be some $k \in \mathcal{V}_i$ that is a 2-conveyor.

Also define iteration mapping $T : \mathcal{X} \to \mathcal{X}$ by $T(\underline{\gamma}) \triangleq [T_1(\underline{\gamma}), T_2(\underline{\gamma}), \ldots, T_n(\underline{\gamma})]$ where, for each $i \in \mathcal{C}$

$$T_i(\underline{\gamma}) \triangleq \min\{1, \max\{0, \gamma_i + \sigma_i \nabla_i U_i(\underline{\gamma})\}\}, \tag{4a}$$

where

$$\frac{1}{\sigma_i} \ge 2|\mathcal{V}_i| \max_{k\in\mathcal{V}_i} |p_{ik}(0)| \tag{4b}$$

for all $\underline{\gamma} \in \mathcal{X}$. If

$$\min_{j\in\mathcal{V}_i} |p_{ij}\left(|\mathcal{C}_j|\right)| > \left(|\mathcal{V}_i| - \frac{1}{2}\right) \max_{j\in\mathcal{V}_i} |c_{ij}|, \quad \text{for all } i \in \mathcal{C}, \tag{5}$$

then the TADI sequence $(\underline{\gamma}(t))$ generated with mapping T and the outdated estimate sequence $(\underline{\gamma}^i(t))$ for all $i \in \mathcal{C}$ each converge to the unique Nash equilibrium of the cooperation game.

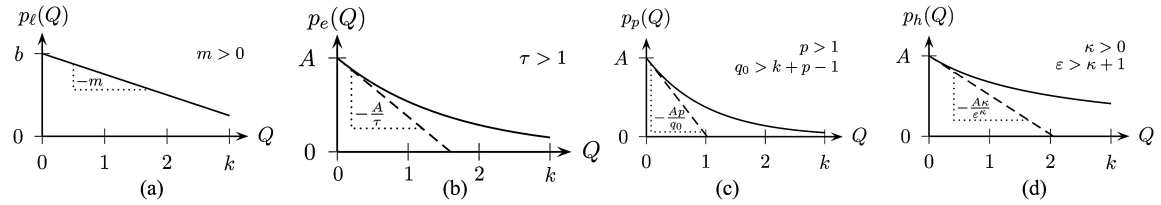

Fig. 3. Sample stabilizing payment (i.e., inverse-demand) functions. (a) $p_\ell(Q) \triangleq b - mQ$. (b) $p_e(Q) \triangleq A\exp(-Q/\tau)$. (c) $p_p(Q) \triangleq A(1 - Q/q_0)^p$. (d) $p_h(Q) \triangleq A\varepsilon^\kappa/(\varepsilon + Q)^\kappa$.

Proof of Theorem 1: By assumption 1 (i.e., all payment functions are stabilizing), for any $\underline{\gamma} \in \mathcal{X}$ and $i \in \mathcal{C}$

$$\begin{aligned}\nabla^2_{ii}U_i(\underline{\gamma}) &= \sum_{j\in\mathcal{V}_i}\left(2p_{ij}(Q_j) + \gamma_i p_{ij}(Q_j)\right)\\ &= \sum_{j\in\mathcal{V}_i}\overbrace{p_{ij}(Q_j)}^{<0} + \sum_{j\in\mathcal{V}_i}\left(\overbrace{p_{ij}(Q_j) + \gamma_i p_{ij}(Q_j)}^{\leq 0}\right) < 0\end{aligned}$$

and

$$\begin{aligned}\nabla^2_{ii}U_i(\underline{\gamma}) &= \sum_{j\in\mathcal{V}_i}\left(\overbrace{2p_{ij}(Q_j)}^{<0} + \overbrace{\gamma_i p_{ij}(Q_j)}^{\geq 0}\right) \geq -2\sum_{j\in\mathcal{V}_i}|p_{ij}(Q_j)|\\ &\geq -2\sum_{j\in\mathcal{V}_i}\max_{k\in\mathcal{V}_i}|p_{ik}(0)| = -2|\mathcal{V}_i|\max_{k\in\mathcal{V}_i}|p_{ik}(0)|\\ &\geq -2|\mathcal{V}_i|\max_{k\in\mathcal{V}_i}|p_{ik}(0)|. \qquad (6)\end{aligned}$$

So by the assumed limits on step size σ_i given in (4b), $0 > \nabla^2_{ii}U_i(\underline{\gamma}) \geq -1/\sigma_i$ for all $i \in \mathcal{C}$.

Next, we bound the cross terms $\nabla^2_{i\ell}U_i$ of the utility Hessian. Take $\underline{\gamma} \in \mathcal{X}$ and cooperator $i \in \mathcal{C}$ connected to conveyor $j \in \mathcal{V}_i$. For another cooperator $\in \mathcal{C} \setminus \{i\}$, if $\ell \notin \mathcal{C}_j$ (i.e., is not an outgoing cooperator for j), then $\partial Q_j/\partial\gamma = 0$ and $\partial P_s(j|i)/\partial\gamma = 0$. So the only other cooperators that contribute to the cross terms of the Hessian are those that share a conveyor with cooerpator i. Hence

$$\begin{aligned}0 &\leq \sum_{\substack{\in\mathcal{C}\\ \neq i}}|\nabla^2_{i\ell}U_i(\underline{\gamma})|\\ &= \sum_{\substack{\in\mathcal{C}\\ \neq i}}\left|\sum_{j\in\mathcal{V}_i}[\ \in\mathcal{C}_j]\left(p_{ij}(Q_j) + \gamma_i p_{ij}(Q_j) - c_{ij}\frac{\partial P_s(j|i)}{\partial\gamma}\right)\right|\end{aligned}$$

where $[\cdot]$ is the Iverson bracket (i.e., $[S] = 1$ or $[S] = 0$ when statement S is true or false). However, the series of sums of mixed binomial products in P_s can be converted into a simpler alternating series of sums of monomial products which is more amenable to differentiation [30, Proposition C.11]. Furthermore

$$\frac{\partial P_s(j|i)}{\partial\gamma} = -\sum_{s=0}^{|\mathcal{C}_j|-2}(-1)^s\frac{1}{2+s}\sum_{\substack{\mathcal{B}\subseteq\mathcal{C}_j\setminus\{i,\ \}\\ |\mathcal{B}|=s}}\left(\prod_{k\in\mathcal{B}}\gamma_k\right), \qquad (7)$$

which, with the use of the binomial theorem [31], can be shown to be negative and bounded below by $-1/2$ [30]. Hence,

$$\begin{aligned}&\sum_{\substack{\in\mathcal{C}\\ \neq i}}\left|\nabla^2_{i\ell}U_i(\underline{\gamma})\right|\\ &\quad\leq \sum_{\substack{\in\mathcal{C}\\ \neq i}}\sum_{j\in\mathcal{V}_i}[\ \in\mathcal{C}_j]\left(\underbrace{|p_{ij}(Q_j) + \gamma_i p_{ij}(Q_j)|}_{\leq 0} + \frac{1}{2}|c_{ij}|\right)\\ &\quad= \sum_{j\in\mathcal{V}_i}\left(\left|p_{ij}(Q_j) + \gamma_i p_{ij}(Q_j)\right| + \frac{1}{2}|c_{ij}|\right)\sum_{\substack{\in\mathcal{C}\\ \neq i}}[\ \in\mathcal{C}_j]\\ &\quad= \sum_{j\in\mathcal{V}_i}\left(\left|p_{ij}(Q_j) + \gamma_i p_{ij}(Q_j)\right| + \frac{1}{2}|c_{ij}|\right)\left(\left|\mathcal{C}_j\right| - 1\right).\end{aligned}$$

However, by assumption 2, each conveyor $j \in \mathcal{V}$ has no more than three outgoing connections to cooperators (i.e., $|\mathcal{C}_j| \leq 3$). Additionally, by assumption 3, if $j \in \mathcal{V}_i$ is a 3-conveyor (i.e., it has 3 outgoing cooperator connections), then there must be some other conveyor $m \in \mathcal{V}_i \setminus \{j\}$ that is a 2-conveyor. So letting $m \in \mathcal{V}_i$ be the 2-conveyor that is guaranteed to exist

$$\begin{aligned}\sum_{\substack{\in\mathcal{C}\\ \neq i}}\left|\nabla^2_{i\ell}U_i(\underline{\gamma})\right| \leq\ &\overbrace{2\sum_{j\in\mathcal{V}_i\setminus\{m\}}\left(\underbrace{|p_{ij}(Q_j) + \gamma_i p_{ij}(Q_j)|}_{\leq 0} + \frac{1}{2}|c_{ij}|\right)}^{\text{Doubled contribution to sum from other cooperators connected to assumed 3-conveyors in } \mathcal{V}_i\setminus\{m\}}\\ &\underbrace{+\underbrace{|p_{im}(Q_m) + \gamma_i p_{im}(Q_m)|}_{\leq 0} + \frac{1}{2}|c_{im}|.}_{\text{Contribution to sum from other cooperator of 2-conveyor } m\in\mathcal{V}_i}\end{aligned}$$

By Definition 2 of an SPF

$$\begin{aligned}\sum_{\substack{\in\mathcal{C}\\ \neq i}}\left|\nabla^2_{i\ell}U_i(\underline{\gamma})\right| \leq &-\sum_{j\in\mathcal{V}_i\setminus\{m\}}\left(2p_{ij}(Q_j) + \gamma_i p_{ij}(Q_j)\right)\\ &-\left(p_{im}(Q_m) + \gamma_i p_{im}(Q_m)\right)\\ &+\left(\overbrace{|\mathcal{V}_i\setminus\{m\}|}^{m\in\mathcal{V}_i} + \frac{1}{2}\right)\max_{j\in\mathcal{V}_i}|c_{ij}|\\ =&-\sum_{j\in\mathcal{V}_i\setminus\{m\}}\left(2p_{ij}(Q_j) + \gamma_i p_{ij}(Q_j)\right)\\ &-\left(p_{im}(Q_m) + \gamma_i p_{im}(Q_m)\right)\\ &+\left(|\mathcal{V}_i| - \frac{1}{2}\right)\max_{j\in\mathcal{V}_i}|c_{ij}|.\end{aligned}$$

or, equivalently

$$\begin{aligned}\sum_{\substack{\in\mathcal{C}\\ \neq i}} \left|\nabla^2_{i\ell}U_i(\underline{\gamma})\right| \quad &\leq -\textstyle\sum_{j\in\mathcal{V}_i}\left(2p_{ij}(Q_j)+\gamma_i p_{ij}(Q_j)\right)\\ &\quad +\left(2p_{im}(Q_m)+\gamma_i p_{im}(Q_m)\right)\\ &\quad -\left(p_{im}(Q_m)+\gamma_i p_{im}(Q_m)\right)\\ &\quad +\left(|\mathcal{V}_i|-\tfrac{1}{2}\right)\max_{j\in\mathcal{V}_i}|c_{ij}|.\end{aligned}$$

Thus

$$\sum_{\substack{\in\mathcal{C}\\ \neq i}} \left|\nabla^2_{i\ell}U_i(\underline{\gamma})\right| \leq -\overbrace{\sum_{j\in\mathcal{V}_i}\left(2p_{ij}(Q_j)+\gamma_i p_{ij}(Q_j)\right)}^{\nabla^2_{ii}U_i(\underline{\gamma})} + \overbrace{p_{im}(Q_m)}^{<0} + \left(|\mathcal{V}_i|-\frac{1}{2}\right)\max_{j\in\mathcal{V}_i}|c_{ij}|.$$

So

$$\begin{aligned}\sum_{\substack{\in\mathcal{C}\\ \neq i}} &\left|\nabla^2_{i\ell}U_i(\underline{\gamma})\right|\\ &\leq -\nabla^2_{ii}U_i(\underline{\gamma})\\ &\quad -\underbrace{\left(\min_{j\in\mathcal{V}_i}|p_{ij}(Q_j)| - \left(|\mathcal{V}_i|-\frac{1}{2}\right)\max_{j\in\mathcal{V}_i}|c_{ij}|\right)}_{>0 \text{ by (5)}}\end{aligned}$$

where, by the assumption in (5), the underbraced expression is strictly positive. The desired result follows from this inequality and (6). In particular, as shown by [27], Chapter 3 for the special case of iterative mappings with scalar block components, T is a maximum-norm contraction mapping [27, Proposition 1.11]. Consequently, it has a unique fixed point $\underline{\gamma}^*$ [27, Proposition 1.1] that is the Nash equilibrium [27, Proposition 5.1] of the cooperation game. Futhermore, the TADI sequence $(\underline{\gamma}(t))_{t\in\mathcal{T}}$ generated by T converges to $\underline{\gamma}^*$ [27, Proposition 2.1 from Chapter 6]. ■

B. Interpretations

The proof of Theorem 1 makes use of a kind of diagonal dominance of $\nabla^2_{ii}U_i$ for cooperator $i \in \mathcal{C}$ over the combined contribution of off-diagonal terms $\nabla_{i\ell}U_i$ for each $\in \mathcal{C}\setminus\{i\}$. For each conveyor $j \in \mathcal{V}_i$ connected to i, there are two p_{ij} terms contributed to the diagonal. However, if a conveyor is a 3-conveyor, it also contributes two p_{ij} terms to the off-diagonals. Consequently, if all conveyors connected to a cooperator are 3-conveyors, diagonal dominance is not possible because the terms exactly cancel. In essence, a unique Nash equilibrium is not guaranteed to exist in this case because cooperator choices become decoupled when the probability of actually having to pay a cost for volunteering is sufficiently low (i.e., such cooperators always cooperate regardless of the decision of other cooperators). When one of the 3-conveyors is replaced by a 2-conveyor, the two cooperators connected to the 2-conveyor are sufficiently coupled to restore a single Nash equilibrium. In other words, diagonal dominance is restored because the 2-conveyor only contributes a single p_{ij} term to the off-diagonals, and so the two diagonal p_{ij} terms can dominate it. So 2-conveyors are themselves stabilizers that allow a cooperator $i \in \mathcal{C}$ to focus its decision making on the conveyors in $\mathcal{V}_i$ for which there is only one other cooperator competing for payment. For example, in the complex TPN in Fig. 4, the 3-conveyors in the network (e.g., 2, 4, 7, and 10) could destabilize the gradient ascent if the 2-conveyors (e.g., 1, 5, 6, 8, 9, and 11) were not also present.

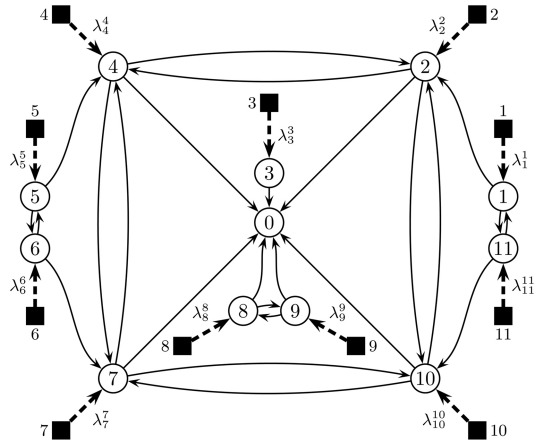

Fig. 4. Many-agent task-processing network with stable topology.

It may be possible to weaken Theorem 1 to allow for conveyors with $n > 3$ outgoing connections to cooperators so long as the slopes of the n-conveyor payment functions can be dominated by those of other 1-conveyors. That is, although n-conveyors with $n \geq 2$ contribute to both diagonal $\nabla^2_{ii}U_i$ and off-diagonal $\nabla^2_{i\ell}U_i$ terms, 1-conveyors contribute only to the diagonal. Consequently, if the p_{ij} from the 1-conveyor is large enough, it can dominate uncanceled terms from the off-diagonals. In other words, a cooperator connected to an n-conveyor with $n > 3$ would normally be decoupled from other cooperator decisions because the chance of actually having to process an advertised task would be negligible. However, if such cooperator also faced a 1-conveyor whose payment function was sufficiently steep, then the loss of payment from the 1-conveyor for being overly cooperative would restore a nontrivial cooperation willingness.

The restriction in (5) is similar to the network generalization of Hamilton's rule [32] discussed by Ohtsuki *et al.* [33] and [34]. In their case, the graph consists of individuals at graph nodes that have relationships modeled by graph links. Each individual is either a cooperator, which pays a cost to deliver a benefit to a neighbor, or a defector, which pays no cost and delivers no benefit. Behavioral strategies spread by way of birth–death processes, and they show that a sufficient condition for cooperation to spread is that the benefit-to-cost ratio is greater than the average degree (i.e., number of neighbors connected to each node) of the network. Here, the task-processing network is not a substrate for birth–death processes; however, the rule in (5) plays a similar role relating payment, cost, and cooperator degree. Moreover, it is a sufficient condition for individual gradient ascent to converge upon a stable cooperation policy. Hence, just as Hamilton's rule allows scientists to reason about cooperation in natural networks, it also ensures that automata will find stable cooperation strategies in artificial networks.

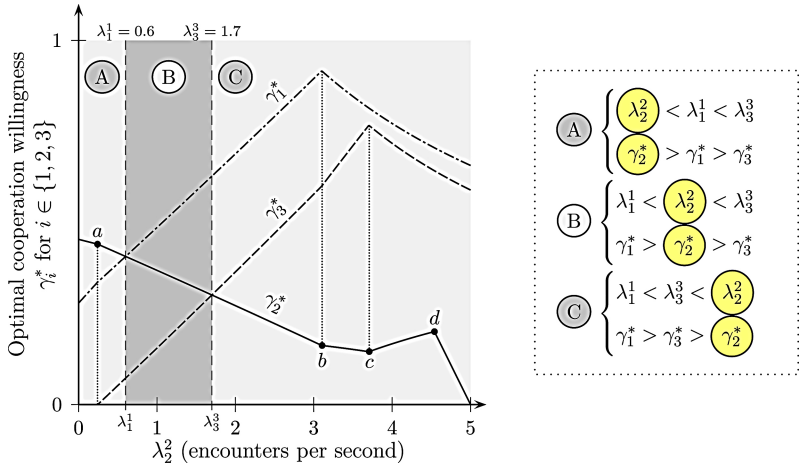

Fig. 5. AAV optimal cooperation willingness as encounter rates vary. The solid, dashed, and dot-dashed lines represent the Nash-equilibrium cooperation levels for three fully connected AAVs that differ in their encounter rates. The encounter rate for AAV 2 is swept from zero to its maximum capacity while the encounter rates for the other two AAVs are held constant. Each AAV has a linear payment function whose slope is inversely proportional to the local encounter rate; otherwise, the AAVs are identical. AAV 3 begins to cooperate at point a but truncates the rise in cooperation due to capacity constraints at point c. Likewise, AAV 1 increases its cooperation until truncating at point b due to capacity constraints. At point d, the encounter rate at AAV 2 becomes too high for it to sustain any further cooperation, and its cooperation decreases rapidly to zero when its encounter rate reaches its maximum capacity.

V. Example Cooperative AAV Scenario

Consider the AAV scenario shown in Fig. 2. Assume that $\pi_i^k = 1$, $c_{ij} = 0.1$, $q_{ij} = 1$, and $R_i = R_{\max} = 5$ for all $i \in \mathcal{A} = \{1, 2, 3\}$, $j \in \mathcal{A} \setminus \{i\}$, $k \in \mathcal{Y}_i$, and $\in \mathcal{Y}_j$. Also assume that $\lambda_1^1 = 0.6$, $\lambda_3^3 = 1.7$, $0 < \lambda_2^2 \leq R_{\max}$, and the linear payment function $p_i^i(Q_i) \triangleq 1 - Q_i/\lambda_i^i$ for all $i \in \mathcal{A}$. Hence, the three otherwise equivalent agents face different task-encounter rates, and their payment functions have slopes that are inversely proportional to each encounter rate. So agents associated with higher encounter rates have a higher demand for cooperation and thus have inelastic payment functions (i.e., cooperation retains its high value even when a high quantity is available).

A conservative choice of step size $\sigma \triangleq 1/(4 \max_{i \in \mathcal{A}, j \in \mathcal{V}_i} p_{ij}([0, 0, 0]\ ))$ for all $\in \mathcal{A}$ yields a convergent TADI for this scenario, and so the equilibrium can be found by parallel numerical computation on autonomous vehicles in an asynchronous and decentralized fashion (i.e., the equilibrium need not be calculated in a central location and broadcasted continuously to the agents). simulation results of this parallel computation of the Nash equilibrium $\underline{\gamma}^*$ are summarized in Fig. 5 for the different combinations of encounter rates so as to show how the equilibrium depends on relative encounter rates.

The Nash equilibrium has the desirable feature that $\lambda_i^i > \lambda_j^j$ implies that $\gamma_i^* < \gamma_j^*$ for all $i, j \in \mathcal{A}$. So agents that are locally busy are less willing to cooperate, and agents that are relatively idle are more willing to cooperate. In Fig. 5, as λ_2^2 increases from a relatively low encounter rate to a relatively high encounter rate, payment function p_2 to agents 1 and 3 becomes shallower and causes the optimal γ_1^* and γ_3^* to increase. However, as γ_1^* (or γ_3^*) increases, payment $p_3(Q_3)$ [or $p_1(Q_1)$] to agent 2 is depressed and γ_2^* decreases. So even though the encounter rate λ_2^2 does not appear in the gradient of U_2, agent 2 still arrives at a Nash equilibrium where its own increased encounter rate corresponds to a decreased willingness to cooperate. Thus, this desirable effect is emergent and due to implicit coupling within the network. Even though each agent's own encounter rate has no direct relationship to the gradient it climbs, the cyclic relationships on the network weigh the local encounter rate against the neighboring encounter rates and adjust cooperation accordingly.

The cooperation game is meant to provide incentives to independent agents to cooperate without forcing the use of a rigid coordination protocol and the communication costs that come with it. Consequently, the balancing action of the cooperation game is destined to be worse than a true load-balancing strategy with explicit coordination or centralized control. For the case of particular AAV example described above, Fig. 6(a) shows the processing rates that correspond to the competitive equilibria for each encounter rate combination. The dotted line labeled R_{opt} represents the perfectly balanced solution. Clearly, the three processing rates, $R_1(\gamma_1^*)$, $R_2(\gamma_2^*)$, and $R_3(\gamma_3^*)$, are not perfectly balanced. However, the task allocation is preferable over the no-cooperation case shown in Fig. 6(b). In particular, in the no-cooperation case, the rates are nowhere balanced and AAV 2 reaches its maximum processing rate. However, in the competitive case, the rates are initially very close to optimal and all agents stay below their maximum processing rate. Furthermore, the competitive volunteering equilibrium has redirected the most tasks to AAV 1, which is the AAV with the smallest local encounter rate. So without explicit coordination, the cooperation game has achieved a weak balance of the load in the system. Additionally, as in virtual load balancing [35], the payment, costs, and currency scalars in (1a) can be adjusted to bias the system toward preferential task-allocation shapes.

For comparison, a summary of a Pareto-optimal policy for this 3-AAV case is shown in Fig. 7. That is, (a) shows the cooperation levels $\underline{\gamma}^*$ that maximize the simple sum $U_1(\underline{\gamma}) + U_2(\underline{\gamma}) + U_3(\underline{\gamma})$ over the capacity-constrained

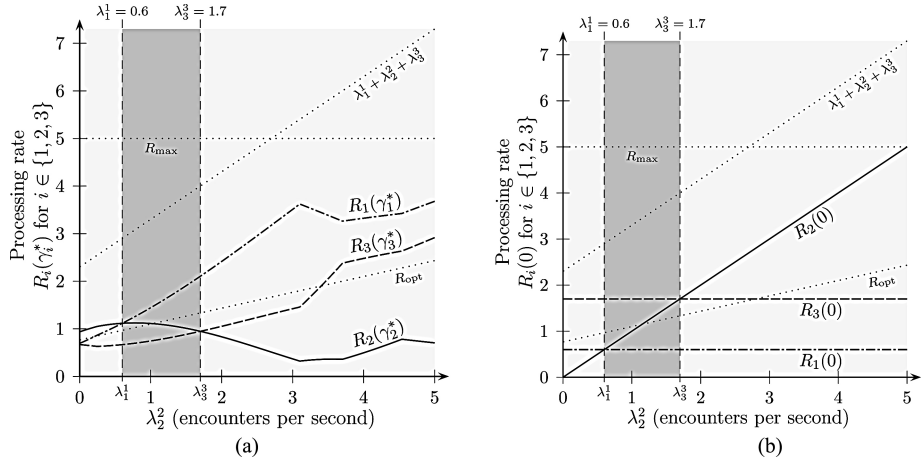

Fig. 6. AAV processing rates as encounter rates vary. The solid, dashed, and dot-dashed lines represent the processing rate for three fully connected AAVs that differ in their encounter rates. The encounter rate for AAV 2 is swept from zero to its maximum capacity while the encounter rates for the other two AAVs are held constant. In (a), the processing rates corresponding to the Nash-equilibrium cooperation levels are shown. In (b), the processing rates corresponding to zero cooperation are shown. In either case, the processing rates sum to the oblique line labeled $\lambda_1^1 + \lambda_2^2 + \lambda_3^3$ that represents the total encounter rate for the 3-AAV system. Likewise, the oblique line labeled R_{opt} represents the processing rate that each AAV would attain if the system load was perfectly balanced. The horizontal dotted line labeled $R_{\max}$ represents the maximum processing rate for each AAV. Although the Nash-equilibrium policy does not track R_{opt}, it does spread the load across the three AAVs such that the AAV with the smallest local encounter rate processes the most remote load.

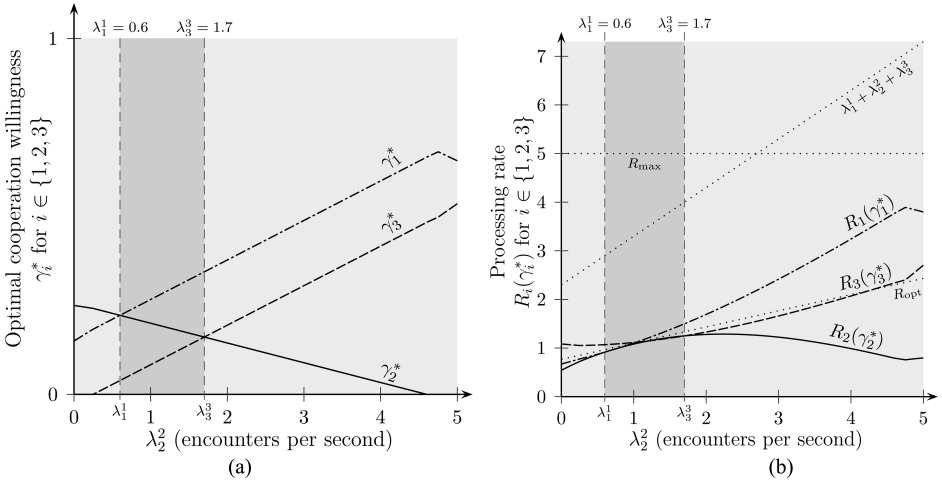

Fig. 7. Pareto-optimal AAV cooperation levels and processing rates. The solid, dashed, and dot-dashed lines represent the optimal cooperation levels [in (a)] and corresponding processing rates [in (b)] for three fully connected AAVs that differ in their encounter rates. The encounter rate for AAV 2 is swept from zero to its maximum capacity while the encounter rates for the other two AAVs are held constant. The cooperation levels are chosen to maximize the simple sum of the three individual agent utility functions. In (b), the processing rates sum to the oblique line labeled $\lambda_1^1 + \lambda_2^2 + \lambda_3^3$ that represents the total encounter rate for the 3-AAV system. Likewise, the oblique line labeled R_{opt} represents the processing rate that each AAV would attain if the system load was perfectly balanced. The horizontal dotted line labeled $R_{\max}$ represents the maximum processing rate for each AAV. Like the Nash-equilibrium policy, the Pareto-optimal policy does not track R_{opt} but it does spread the load across the three AAVs such that the AAV with the smallest local encounter rate processes the most remote load. Thus, the Pareto-optimal policy may be viewed as an ideal case of the Nash-optimal policy. However, if global control was possible, a true load-balancing solution would likely be used instead.

volunteering-policy space $\mathcal{X}$, and (b) shows the corresponding processing rates. Because the interests of the AAVs are aligned in this case, it may make sense to remove the payment functions that were added in the competitive case as an incentive for cooperation among the selfish agents. However, the resulting no-incentive optimization objective has a trivial no-cooperation optimizer. That is, because the cost of processing a remote task is no worse on the system than the cost of processing it locally, there is no incentive for increased cooperation. Consequently, payment is still needed to induce nontrivial cooperation; however, like a cartel, the team coordinates its actions to maximize the total payment to the group. The resulting shapes of the cooperation and processing-rate curves are similar to the Nash-optimal curves in Figs. 5 and 6(a), and the apparent price of anarchy is low for this case. However, the agent utility functions used in this paper are only introduced to induce behavior with cooperative features from inherently competitive agents. If agents are synchronized or centrally controlled, a policy focused on true load balancing would likely be used.

VI. Conclusion and Future Work

A framework for cooperative task processing on a network was presented. Using this framework, a totally asynchronous cooperative control policy was shown to stabilize

the Nash equilibrium of a cooperation game. By introducing a cooperation-trading economy into the formulation, the agents individually climbed their own local utility functions yet still achieved an equilibrium where task processing was shared among different agents; this balancing of incoming load was achieved without central control or synchronized coordination between nodes. Thus, the central contribution of this paper was a dynamic task-processing framework that was able to share network resources without explicit coordination between members of the network.

In order to simplify the analysis, a result of [27], Propositions 1.11 from Chapter 3 was used that required that the decision variable on each node be a scalar. Future work should address the case where each agent associates a different cooperation willingness to different types of tasks and to different conveyors. Likewise, forwarding probabilities can also be made into decision variables that could be adjusted on each conveyor across the different task types being advertised to the different cooperators connected to it. In particular, an additional incentive scheme could be introduced to induce nontrivial forwarding behavior on the conveyors. Furthermore, if both forwarding and volunteering can be adjusted simultaneously, protocols based on fairness and reciprocity can be developed that may not require explicit incentives introduced with a fictitious trading economy. In general, future work should allow for multiple decision variables per agent and a multidimensional decision space which need not be a simple Cartesian product.

This paper assumed that each agent had a maximum processing rate, and the decision variables were conservatively constrained to ensure that maximum rate was not violated. Consequently, these capacity constraints have room to be relaxed in future work. For example, if an agent has independent cooperation levels for two different task types, the constraint on each cooperation level can be made to vary with the other cooperation level. More importantly, the role of time can be made more explicit. Processing and switching durations are central motivations for the work of [11]. Effects like these can be added to this model by making the average processing time of each task explicit. In particular, the present work optimizes the long-term rate of gain of each agent based on rewards issued at the instant each task arrives. If each task type has an average processing time associated with it, then an agent will sometimes be able to increase its long-term rate of gain by processing fewer items that have long processing times. That is, the time spent processing a task is an implicit opportunity cost due to the lost time available for encountering other tasks that return higher profit. Because task arrivals are independent, a utility function that captures the processing time of tasks may be derived as an average reward rate of this Markov renewal–reward process [36]. If analytically tractable, optimal cooperation willingness corresponding to such a time-aware utility function can be derived using similar methods as in this paper.

REFERENCES

[1] R. Buyya, "Economic-based distributed resource management and scheduling for grid computing," Ph.D. dissertation, Monash Univ., Melbourne, Australia, Apr. 2002.

[2] A. Mas-Colell, M. D. Whinston, and J. R. Green, *Microeconomic Theory*. New York, NY, USA: Oxford Univ. Press, 1995.

[3] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*. Cambridge, MA, USA: MIT Press, 1994.

[4] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta, "Spawn: A distributed computational economy," *IEEE Trans. Softw. Eng.*, vol. 18, no. 2, pp. 103–117, Feb. 1992.

[5] Y. Amir, B. Awerbuch, A. Barak, R. S. Borgstrom, and A. Keren, "An opportunity cost approach for job assignment in a scalable computing cluster," *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 7, pp. 760–768, Jul. 2000.

[6] D. Grosu and A. T. Chronopoulos, "Algorithmic mechanism design for load balancing in distributed systems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 1, pp. 77–84, Feb. 2004.

[7] R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Trans. Comput.*, vol. 29, no. 12, pp. 1104–1113, Dec. 1980.

[8] A. Oram, Ed., *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. Sebastopol, CA, USA: O'Reilly, 2001.

[9] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Architect., Protocols Comput. Commun.*, Portland, OR, USA, Aug. 30–Sep. 3, 2004, pp. 367–378.

[10] J. Feigenbaum and S. Shenker, "Distributed algorithmic mechanism design: Recent results and future directions," in *Proc. 6th Int. Workshop Discrete Algorithms Methods Mobile Comput. Commun.*, Atlanta, GA, USA, Sep. 28, 2002, pp. 1–13.

[11] J. R. Perkins and P. R. Kumar, "Stable, distributed, real-time scheduling of flexible manufacturing/assembly/disassembly systems," *IEEE Trans. Autom. Control.*, vol. 34, no. 2, pp. 139–148, Feb. 1989.

[12] R. L. Cruz, "A calculus for network delay, part II: Network analysis," *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 132–141, Jan. 1991.

[13] J. Finke, K. M. Passino, and A. G. Sparks, "Stable task load balancing strategies for cooperative control of networked autonomous air vehicles," *IEEE Trans. Control. Syst. Technol.*, vol. 14, no. 5, pp. 789–803, Sep. 2006.

[14] J. Finke and K. M. Passino, "Stable cooperative vehicle distributions for surveillance," *J. Dyn. Syst., Meas., Control.*, vol. 129, no. 5, pp. 597–608, 2007.

[15] A. E. Gil, K. M. Passino, S. Ganapathy, and A. Sparks, "Cooperative task scheduling for networked uninhabited air vehicles," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 44, no. 2, pp. 561–581, Apr. 2008.

[16] A. Ipakchi and F. Albuyeh, "Grid of the future," *IEEE Power Energy*, vol. 7, no. 2, pp. 52–62, Mar./Apr. 2009.

[17] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory*, 2nd ed., Classics in Applied Mathematics Series. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1999, no. 23.

[18] E. Altman, A. Kumar, D. Kumar, and R. Venkatesh, "Cooperative and non-cooperative control in IEEE 802.11 WLANs," in *Proc. 19th Int. Teletraffic Congr.*, Beijing, Aug. 29–Sep. 2, 2005, pp. 1663–1672.

[19] L. Buttyán and J.-P. Hubaux, "Stimulating cooperation in self-organizing mobile ad hoc networks," *Mob. Netw. Appl.*, vol. 8, no. 5, pp. 579–592, Oct. 2003.

[20] E. Altman, A. A. Kherani, P. Michiardi, and R. Molva, "Non-cooperative forwarding in ad-hoc networks," in *Proc. Networking*, (LNCS), vol. 3462, 2005, pp. 486–498.

[21] A. E. Gil and K. M. Passino, "Stability analysis of network-based cooperative resource allocation strategies," *Automatica*, vol. 42, no. 2, pp. 245–250, 2005.

[22] D. B. Lange and M. Oshima, "Seven good reasons for mobile agents," *Commun. ACM*, vol. 42, no. 3, pp. 88–89, Mar. 1999.

[23] D. B. Lange, "Mobile objects and mobile agents: The future of distributed computing?" in *Proc. ECOOP,* Brussels, Belgium, Jul. 20–24, 1998, pp. 1–12.

[24] R. T. Maheswaran, O. Ç. Imer, and T. Başar, "Agent mobility under price incentives," in *Proc. 38th IEEE Conf. Decision Control*, vol. 4, Phoenix, AZ, USA, Dec. 7–10, 1999, pp. 4020–4025.

[25] J. E. White, "Telescript technology: Mobile agents," in *Mobility: Processes, Computers, and Agents*, D. Milojičić, F. Douglis, and R. Wheeler, Eds. New York, NY, USA: ACM Press, 1999, pp. 460–493.

[26] D. Reed, I. Pratt, P. Menage, S. Early, and N. Stratford, "Xenoservers: Accountable execution of untrusted programs," in *Proc. 7th Workshop Hot Topics Oper. Syst.*, Rio Rico, AZ, USA, Mar. 28–30, 1999, pp. 136–141.

[27] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Belmont, MA, USA: Athena Scientific, 1997.
[28] W. Nicholson, *Microeconomic Theory: Basic Principles and Extensions*, 5th ed. Fort Worth, TX, USA: Dryden Press, 1992.
[29] M. Bacharach, *Economics and the Theory of Games*. London, U.K.: Macmillan, 1976.
[30] T. P. Pavlic and K. M. Passino, "Cooperative task-processing networks: Parallel computation of non-trivial volunteering equilibria," The Ohio State Univ., Tech. Rep. OSU-CISRC-3/11-TR05, 2011 [Online]. Available: ftp://ftp.cse.ohio-state.edu/pub/tech-report/2011/TR05.pdf
[31] R. D. Gustafson, P. D. Frisk, and J. Hughes, *College Algebra*, 10th ed. Belmont, CA, USA: Brooks/Cole Publishing, Dec. 2008.
[32] W. D. Hamilton, "The genetical evolution of social behavior. I." *J. Theor. Biol.*, vol. 7, no. 1, pp. 1–16, 1964.
[33] H. Ohtsuki, C. Hauert, E. Lieberman, and M. A. Nowak, "A simple rule for the evolution of cooperation on graphs," *Nature*, vol. 441, no. 7092, pp. 502–505, 2006.
[34] M. A. Nowak, "Five rules for the evolution of cooperation," *Science*, vol. 314, no. 5805, pp. 1560–1563, Dec. 2006.
[35] K. L. Burgess and K. M. Passino, "Stability analysis of load balancing systems," *Int. J. Control.*, vol. 61, no. 2, pp. 357–393, 1995.
[36] M. V. Johns and R. G. Miller, Jr., "Average renewal loss rates," *Ann. Math. Stat.*, vol. 34, no. 2, pp. 396–401, Jun. 1963.

Theodore P. Pavlic (S'02–M'10) received the Ph.D. degree in electrical and computer engineering from The Ohio State University (OSU), Columbus, USA, in 2010. From 2010 to 2012, he was a Post-Doctoral Researcher in computer science and engineering at OSU, studying the use of software-verification formal methods for autonomous urban vehicle development. He is currently a Post-Doctoral Scholar in the School of Life Sciences at Arizona State University (ASU), Tempe, AZ, USA.

He is currently associated with a social-insect laboratory where he studies ants, like Temnothorax rugatulus and Aphaenogaster cockerelli, that serve as experimental model systems for distributed decentralized decision making. Target applications of interest include distributed power generation, building lighting and HVAC control, and coordination problems in multirobot systems. He has also held engineering positions in software and hardware research and development at IBM Network Storage, Research Triangle Park, NC, USA, and National Instruments, Austin, TX, USA, as well as a number of small telecommunications companies. In 2000, his load-balancing work with the then-nascent Linux Virtual Server project was referenced in a *Wired* magazine article on global collaboration in open-source-software development. His current research interests include multiagent systems, bio-inspiration and bio-mimicry, distributed problem solving, and ecological rationality. In general, he is interested in complex adaptive systems with applications in control systems engineering and behavioral ecology.

Kevin M. Passino (S'79–M'90–SM'96–F'04) received the Ph.D. degree in electrical engineering from the University of Notre Dame, Notre Dame, IN, USA, in 1989.

He is currently a Professor of electrical and computer engineering at The Ohio State University (OSU), Columbus, USA, and was the Director of the OSU Collaborative Center of Control Science that was funded by AFOSR and AFRL/VA. He is the co-editor (with P. J. Antsaklis) of the book *An Introduction to Intelligent and Autonomous Control* (Kluwer Academic, 1993), co-author (with S. Yurkovich) of the book *Fuzzy Control* (Addison Wesley Longman, 1998), co-author (with K. L. Burgess) of the book *Stability Analysis of Discrete Event Systems* (Wiley, 1998), co-author (with V. Gazi, M. L. Moore, W. Shackleford, F. Proctor, and J. S. Albus) of the book *The RCS Handbook: Tools for Real Time Control Systems Software Development* (Wiley, 2001), co-author (with J. T. Spooner, M. Maggiore, and R. Ordonez) of the book *Stable Adaptive Control and Estimation for Nonlinear Systems: Neural and Fuzzy Approximator Techniques* (Wiley, 2002), author of the book *Biomimicry for Optimization, Control, and Automation* (Springer-Verlag, 2005), and co-author (with V. Gazi) of the book *Swarm Stability and Optimization* (Springer-Verlag, 2011).

Dr. Passino has served as the Vice President of Technical Activities of the IEEE Control Systems Society (CSS), and was an Elected Member of the IEEE CSS Board of Governors. He was also the Program Chair of the 2001 IEEE Conference on Decision and Control, and is currently a Distinguished Lecturer for the IEEE CSS.