Cooperative Task-Processing Networks[*]

Theodore P. Pavlic[1] and Kevin M. Passino[2]

[1] Department of Computer Science and Engineering (`pavlic.3@osu.edu`)
[2] Department of Electrical and Computer Engineering (`passino@ece.osu.edu`)
The Ohio State University, Columbus, OH 43210 USA

Abstract. This paper introduces a novel framework for the analysis and design of distributed agents that must complete externally generated tasks but also can volunteer to process tasks encountered by other agents. A distributed asynchronous volunteering policy is presented that dynamically adjusts task flow around the network of agents. It is shown that even though agents independently adjust their tendency to volunteer to process tasks from other agents, the set of all volunteering tendencies converges to the unique Nash equilibrium of a cooperation game. An artificial cooperation trading economy ensures that at the equilibrium, non-zero cooperation tendencies are possible and vary across agents. In particular, an agent with relatively high task-encounter rate not only provides more incentive for connected neighbors to cooperate with it but also has less incentive to volunteer to cooperate with other agents. The framework is shown via simulation to be applicable to autonomous air vehicles, and the mathematical results of the paper are also shown to be consistent with classic studies of cooperation from science.

1 Introduction

We consider a network of autonomous agents for which some agents are 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 each agent volunteers to process advertised tasks so that the set of volunteering tendencies across the network converges to a Nash equilibrium. In the sequel, we review existing cooperative processing work and discuss why those approaches are not adequate for solving the problem we formulate here.

Grid computing [8] is an existing approach for achieving cooperative task processing across a group of networked agents. System designers work under the assumption of heterogeneous agents with conflicting priorities. They borrow from the economic theories of mechanism design [22, Ch. 23] and implementation theory [27, Ch. 10] to design mechanisms (e.g., brokering agents) and protocols that either encourage resource sharing [3, 15, 32] or discourage exploitation [26, 30] 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, they control the structure of the interactions between 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. Perkins and Kumar [29] show that distributed scheduling policies exist that guarantee such systems will have finite upper bounds on all buffers of tasks. Similarly, Cruz [9] 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 [11, 12, 14]. However, the coordination required between agents can be prohibitive. Additionally, the single optimality criteria being maximized ignores fatigue on individual agents. For example, in a smart power grid [17], 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, non-cooperative game theory is used to develop totally asynchronous and distributed algorithms for task-processing agents that both respect local processing priorities while also sharing the processing burden of highly loaded neighbors.

Non-cooperative game theory has been traditionally used to design optimal control strategies [5]; 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* multi-hop communication networks [1, 2, 7]. 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 Sect. 2, the task-processing network framework is defined and example task-processing networks are described. The optimization game is presented in Sect. 3, and an asynchronous distributed computation method that ensures convergence to the game's Nash equilibrium is given in Sect. 4. In Sect. 5, results from a simulated task-processing network of autonomous air vehicles are presented, and conclusions and areas of future research are discussed in Sect. 6.

2 Task-Processing Network

In the following, we use the real numbers $\mathbb{R}$, the natural numbers $\mathbb{N} \triangleq \{1, 2, \dots\}$, the whole numbers $\mathbb{W} \triangleq \{0, 1, 2, \dots\}$, and derived symbols like the non-negative real numbers $\mathbb{R}_{\geq 0}$. Take a finite but arbitrarily large set $\mathcal{A} \subset \mathbb{N}$ of *task-processing agents* and a set $\mathcal{P} \subseteq \{(i, j) \in \mathcal{A}^2 : i \neq j\}$ of directed arcs connecting distinct agents. For each agent $i \in \mathcal{A}$, $\mathcal{V}_i \triangleq \{j \in \mathcal{A} : (j, i) \in \mathcal{P}\}$ and $\mathcal{C}_i \triangleq \{j \in \mathcal{A} : (i, j) \in \mathcal{P}\}$ are respectively the sets of *conveyors* and *cooperators* connected to agent i. Hence, $\mathcal{V} \triangleq \{j \in \mathcal{A} : \mathcal{C}_j \neq \emptyset\} = \bigcup_{i \in \mathcal{A}} \mathcal{V}_i$ 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:

1. For all $i \in \mathcal{A}$, there exists a finite and possibly empty set $\mathcal{Y}_i \subset \mathbb{N}$ of *task types* such that for all $k \in \mathcal{Y}_i$, 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.
2. If $j \in \mathcal{V}$, then there exist $k \in \mathcal{Y}_j$ with $\pi_j^k \neq 0$ where $\pi_j^k \in [0, 1]$ represents 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 a task to its connected cooperators, the task will be processed by agent j.
3. If $i \in \mathcal{C}$, then there is some $\gamma_i \in [0, 1]$ that represents 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 cooperators that volunteer for it; 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 *task-processing network* (TPN).

The simple TPN shown in Fig. 1 represents a flexible manufacturing system (FMS) similar to the systems described by Perkins and Kumar [29]. 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.
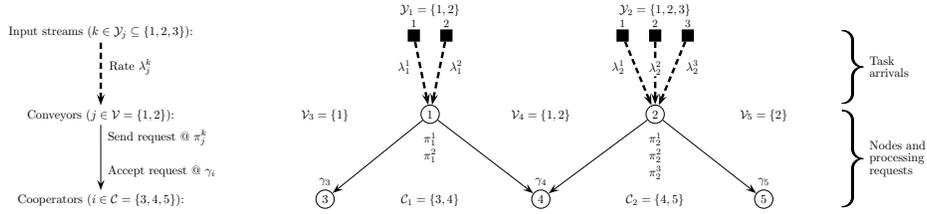
Fig. 1. Simple flexible manufacturing system example.

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 system designer can choose different probabilities for each task type 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 the FMS example, 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 autonomous air vehicle (AAV) patrol scenario shown in Fig. 2(a) that is similar to others in resource allocation literature [11, 12, 14]. Each AAV $i \in \{1,2,3\}$ continuously
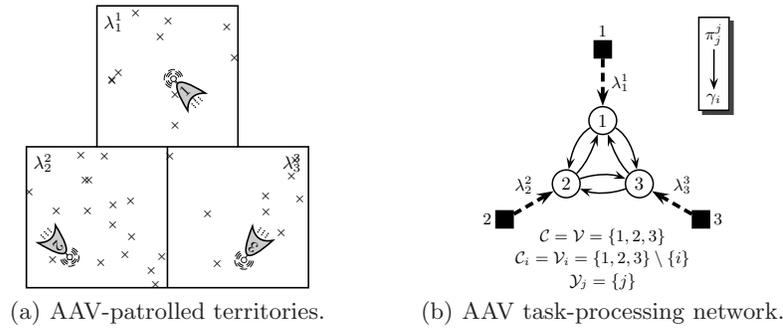

(a) AAV-patrolled territories.   (b) AAV task-processing network.

Fig. 2. A task-processing network formed by three autonomous air vehicles (AAV).

searches its territory for tasks (e.g., targets) to process, and these tasks are generated (i.e., found) at rate $\lambda_i > 0$. 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 generate a new task-processing request for that task; it must process it itself.

Task-processing networks describe a broad range of applications. The AAV example above can also serve as a model of a mobile software agent [19–21, 31, 33] that patrols for tasks to process or any general group of networked processors [13]. Additionally, by converting encounter rates to energetic rates (i.e., power demand), TPNs can model the behavior of smart power grids [17] made up of stations that request assistance from neighbors. That is, cooperator stations adjust additional supply provided in response to demand requests from remote conveyor stations.

3 Cooperation Game Among Selfish Agents

In a TPN, the probability (i.e., *cooperation propensity*) $\gamma_i \in [0, 1]$ that cooperator $i \in \mathcal{C}$ will volunteer for an advertised task from its connected conveyors must be chosen. It is assumed that this choice must be done in a distributed fashion and it is impractical for agents to coordinate in order to maximize some global utility. 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 [6, Sect. 3.5.1].

To inform each cooperator how to choose this policy, the network's designer assigns cost and rewards to agent operations in a common currency (e.g., proportional to dollars of net profit) that is called *points* here. In particular,

- Agent $i \in \mathcal{A}$ receives $(b_i^k - c_i^k)$ net points for processing a locally generated task of type $k \in \mathcal{Y}_i$.
- Conveyor $i \in \mathcal{V}$ receives r_i^k when a task of type $k \in \mathcal{Y}_i$ from i is processed by a $\mathcal{C}_i$ cooperator.
- If cooperator $j \in \mathcal{C}_i$ volunteers and is selected to process a task of type $k \in \mathcal{Y}_i$ from conveyor $i \in \mathcal{V}$, then j pays cost c_{ij}^k to process that task.

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:

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

So if any cooperator $i \in \mathcal{C}_j$ increases its cooperation propensity 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 propensity (i.e., $\gamma_i > 0$) and resource conservation (i.e., $\gamma_i < 1$).

To maximize net points earned over a long run time, each agent chooses a policy to maximize its own expected rate of point accumulation. So for a given vector $\boldsymbol{\gamma} = [\gamma_{c_1}, \gamma_{c_2}, \ldots, \gamma_{c_{|\mathcal{C}|}}]^T \in [0,1]^{|\mathcal{C}|}$ of cooperation policies (where unique $c_k \in \mathcal{C}$ for all $k \in \{1, 2, \ldots, |\mathcal{C}|\}$), the utility returned to cooperator $i \in \mathcal{C}$ is

$$
U_i(\boldsymbol{\gamma}) \triangleq b_i + \overbrace{\left(1 - \prod_{j \in \mathcal{C}_i}(1 - \gamma_j)\right)}^{\text{Conveyor part—constant with respect to } \gamma_i} r_i - Q_i p_i(Q_i)
$$

$$
\underbrace{\phantom{\left(1 - \prod_{j \in \mathcal{C}_i}(1 - \gamma_j)\right)}}_{\Pr(\text{Volunteer from } \mathcal{C}_i | \text{Advertisement from } i)}
$$

$$
+ \gamma_i \underbrace{\sum_{j \in \mathcal{V}_i}\left(p_{ij}(Q_j) - \overbrace{\mathrm{SOBP}_1(\mathcal{C}_j \setminus \{i\})}^{\Pr(i \text{ awarded task from } j | i \text{ volunteers})} c_{ij}\right)}_{\text{Cooperator part—} \gamma_i \text{ and } Q_j \text{ vary with } \gamma_i} \tag{1a}
$$

where

$$
b_i \triangleq \sum_{k \in \mathcal{Y}_i} \lambda_i^k \left(b_i^k - c_i^k\right), \qquad r_i \triangleq \sum_{k \in \mathcal{Y}_i} \lambda_i^k \pi_i^k \left(r_i^k - \left(b_i^k - c_i^k\right)\right),
$$

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

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

$$
c_{ij} \triangleq \sum_{k \in \mathcal{Y}_j} \lambda_j^k \pi_j^k c_{ij}^k, \qquad p_{ij}(Q_j) \triangleq \sum_{k \in \mathcal{Y}_j} \lambda_j^k \pi_j^k q_{ij}^k p_j^k(Q_j). \tag{1b}
$$

are the costs and benefits to $i \in \mathcal{C}$ for volunteering for tasks exported from $j \in \mathcal{V}_i$. The expression for SOBP is given in Definition 1.

Definition 1 (Sum of binomial products). *Let $\mathcal{I}$ be a finite index set and $\Omega \triangleq \{\gamma_i\}_{i \in \mathcal{I}}$ be an indexed family where $\gamma_i \in [0,1]$ for each $i \in \mathcal{I}$. For $\Gamma \subseteq \mathcal{I}$ and $g \in \mathbb{N}$, the* sum of binomial products

$$
\mathrm{SOBP}_g(\Gamma) \triangleq \sum_{\ell=0}^{|\Gamma|} \frac{1}{g+\ell} \sum_{\substack{\mathcal{C} \subseteq \Gamma \\ |\mathcal{C}|=\ell}} \left(\left(\prod_{i \in \mathcal{C}} \gamma_i\right)\left(\prod_{k \in \mathcal{C}}(1 - \gamma_k)\right)\right). \tag{2}
$$

For a cooperator $i \in \mathcal{C}$, $\mathrm{SOBP}_1(\mathcal{C}_j \setminus \{i\})$ is the probability that cooperator i is chosen to process an advertised task from conveyor $j \in \mathcal{V}_i$ when it is given that it volunteers for the task. Hence, 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 propensity because the

probability that agent i will be selected decreases. So for a conveyor $j \in \mathcal{V}$, its connected cooperators $\mathcal{C}_j$ form a Cournot oligopoly [23] (i.e., independent agents that provide a service for a demand-driven price) with a positive externality [4] (i.e., processing cost decreases as more cooperators enter the market). The underbraced part of (1a) shows that cooperator i sets its cooperation propensity γ_i based on the summed returns from several such markets.

4 Distributed Computation of the Nash Equilibrium

Let $n \triangleq |\mathcal{C}|$. Because there is no coordination between players, the n-dimensional play space is the Cartesian product $\prod_{i \in \mathcal{C}}[0,1] = [0,1]^n$, and the collection of cooperation policies across all cooperators is the vector $\boldsymbol{\gamma} \triangleq [\gamma_{c_1}, \gamma_{c_2}, \ldots, \gamma_{c_n}]^T \in [0,1]^n$ (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 $U_i : [0,1]^n \mapsto \mathbb{R}$ is twice-continuously differentiable, and so, by Weirstrass' theorem, U_i is bounded above and below and achieves its extrema. Following Bertsekas and Tsitsiklis [6, Proposition 5.7 from Ch. 3], the Nash equilibria of the cooperation game can be found by solving n separate one-dimensional variational inequality problems. In particular, $\boldsymbol{\gamma}^* \in [0,1]^n$ is a Nash equilibria of the cooperation game if and only if, for all $i \in \mathcal{C}$,

$$(\gamma_i - \gamma_i^*)\nabla_i U_i(\boldsymbol{\gamma}^*) \le 0 \quad \text{for all } \gamma_i \in [0,1] \tag{3}$$

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

$$\nabla_i U_i(\boldsymbol{\gamma}) = \sum_{j \in \mathcal{V}_i} \Big(\overbrace{p_{ij}(Q_j) + \gamma_i p'_{ij}(Q_j)}^{\frac{\partial}{\partial \gamma_i}(\gamma_i p_{ij}(Q_j))} \Big) - \text{SOBP}_1(\mathcal{C}_j \setminus \{i\}) c_{ij}).$$

So in a local neighborhood of the Nash equilibrium $\gamma^* \in [0,1]^n$, any unilateral perturbation of a coordinate of γ^* will result in equal or reduced utility.

The existence of a solution to the n simultaneous nonlinear equations in (3) is not guaranteed in general and may be difficult to find analytically. However, variational inequalities over product spaces are well suited for parallel and asynchronous computation [6]. Under special conditions on each utility function, a unique Nash equilibrium is guaranteed to exist, and each of its coordinates in (3) can be computed independently in the distributed and asynchronous fashion described by Assumption 1.

Assumption 1 (Totally asynchronous distributed iteration). *Take* $(c_1, c_2, \ldots, c_n) \triangleq \mathcal{C}$ *to represent the n distinct cooperators of $\mathcal{C}$. Let $\mathcal{T} \triangleq \mathbb{W}$ to be the indices of a sequence of physical times, and let $\{\boldsymbol{\gamma}(t)\}_{t \in \mathcal{T}} \triangleq \{(\gamma_{c_1}(t), \gamma_{c_2}(t), \ldots, \gamma_{c_n}(t))\}$ be a sequence of iterated calculations in the $[0,1]^n$ 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 $T_i : [0,1]^n \mapsto [0,1]$ at time t such that $0 \le \tau_j^i(t) \le t$. That is, an outdated state estimate

$$\boldsymbol{\gamma}^i(t) \triangleq (\gamma_{c_1}^i(t), \gamma_{c_2}^i(t), \ldots, \gamma_{c_n}^i(t)) \triangleq (\gamma_{c_1}(\tau_{c_1}^i(t)), \gamma_{c_2}(\tau_{c_2}^i(t)), \ldots, \gamma_{c_n}(\tau_{c_n}^i(t)))$$

is available to compute $\gamma_i(t+1) = T_i(\boldsymbol{\gamma}^i(t))$ for $t \in \mathcal{T}$ and $i \in \mathcal{C}$. It is assumed:

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 $\{\boldsymbol{\gamma}(t)\}$ is generated by the totally asynchronous distributed iteration (TADI)

$$\gamma_i(t+1) \triangleq \begin{cases} T_i(\boldsymbol{\gamma}^i(t)), & \text{if } t \in \mathcal{T}^i, \\ \gamma_i(t), & \text{if } t \notin \mathcal{T}^i \end{cases} \tag{4}$$

where $\boldsymbol{\gamma}(t) \triangleq (\gamma_{c_1}(t), \gamma_{c_2}(t), \ldots, \gamma_{c_n}(t))$. For each $i \in \mathcal{C}$, the transition mapping $T_i : [0,1]^n \mapsto [0,1]$ in (4) is defined by $T_i(\boldsymbol{\gamma}) \triangleq \min\{1, \max\{0, \gamma_i + \sigma_i \nabla_i U_i(\boldsymbol{\gamma})\}\}$ where $\sigma_i \in \mathbb{R}_{>0}$ is a step-size parameter that scales motion along gradient $\nabla_i U_i$.

4.1 Conditions for Distributed Convergence

The TADI-generated $\{\boldsymbol{\gamma}(t)\}$ 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, (4) may be viewed as a dynamical system model of coupled agents that each take independent actions. In particular, it can be shown that there exists a constant $\underline{\text{SOBP}} > 0$ such that $\text{SOBP}_1(\Gamma) \ge \underline{\text{SOBP}}$ for all $\Gamma \subseteq \mathcal{C}$. 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 $\boldsymbol{\gamma}(T) = \mathbf{0}$ 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 functions satisfying Definition 2 the necessary characteristics.

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

1. *It is strictly decreasing. That is, $p'(Q) \triangleq dp(Q)/dQ < 0$ for all $Q \in [0,k]$.*
2. *It is convex. That is, $p''(Q) \triangleq d^2p(Q)/d^2Q \ge 0$ for all $Q \in [0,k]$.*
3. *Its convexity is eventually dominated by its slope. That is,*

$$\gamma p''(Q) \le -p'(Q) \quad \text{for all } Q \in [\gamma, k - (1-\gamma)] \text{ with } \gamma \in [0,1]. \tag{5}$$

The set of SPFs is closed under conical combinations (i.e., it is a filled cone). 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 (1b), 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 $i \in \mathcal{C}$. Four example SPFs are shown in Fig. 3. Each payment function meets the conditions of Proposition 1; however, the weaker condition 3 of Definition 2 is met for $\varepsilon \geq \kappa$ in Fig. 3(d). Additionally, Fig. 3(c) generalizes Fig. 3(a).

Proposition 1 (Sufficient conditions for SPF). *Take $k \in \mathbb{N}$ and function $p : [0, k] \mapsto \mathbb{R}$. If $0 \leq p''(Q) < -p'(Q)$ for all $Q \in [0, k]$, then p is an SPF.*

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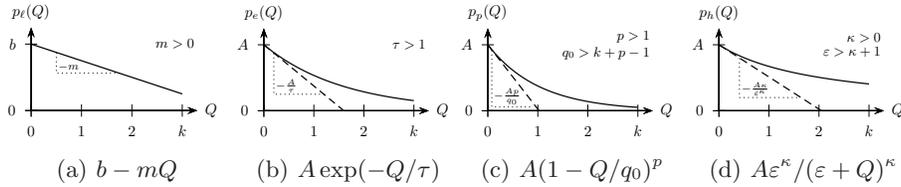
Fig. 3. Sample stabilizing payment (i.e., inverse-demand) functions.

Convergence to the Nash equilibrium depends both on the structure of the payment functions and 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). *For $k \in \mathbb{N}$, $i \in \mathcal{V}$ is a k-conveyor if $|\mathcal{C}_i| = k$.*

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 $j \in \mathcal{V}$, $|\mathcal{C}_j| \leq 3$ (i.e., less than 4 outgoing cooperator links per conveyor).*
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.*

Define $T : [0, 1]^n \mapsto [0, 1]^n$ by $T(\boldsymbol{\gamma}) \triangleq (T_1(\boldsymbol{\gamma}), \ldots, T_n(\boldsymbol{\gamma}))$ so that for each $i \in \mathcal{C}$,

$$T_i(\boldsymbol{\gamma}) \triangleq \min\{1, \max\{0, \gamma_i + \sigma_i \nabla_i U_i(\boldsymbol{\gamma})\}\} \quad where \quad \frac{1}{\sigma_i} \geq 2|\mathcal{V}_i| \max_{k \in \mathcal{V}_i} |p'_{ik}(0)| \quad (6)$$

for all $\boldsymbol{\gamma} \in [0, 1]^n$. If

$$\min_{j \in \mathcal{V}_i} |p'_{ij}(|\mathcal{C}_j|)| > \left(|\mathcal{V}_i| - \frac{1}{2}\right) \max_{j \in \mathcal{V}_i} |c_{ij}| \quad for \ all \ i \in \mathcal{C}, \quad (7)$$

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

Proof. By assumption 1 (i.e., payments from SPF), for any $\boldsymbol{\gamma} \in [0,1]^n$ and $i \in \mathcal{C}$,

$$\nabla_{ii}^2 U_i(\boldsymbol{\gamma}) = \sum_{j \in \mathcal{V}_i} \left(2p_{ij}'(Q_j) + \gamma_i p_{ij}''(Q_j)\right) = \sum_{j \in \mathcal{V}_i} \overbrace{p_{ij}'(Q_j)}^{<0} + \sum_{j \in \mathcal{V}_i} \overbrace{\left(p_{ij}'(Q_j) + \gamma_i p_{ij}''(Q_j)\right)}^{\leq 0},$$

which shows $\nabla_{ii}^2 U_i(\boldsymbol{\gamma}) < 0$. Moreover, for any $\boldsymbol{\gamma} \in [0,1]^n$ and $i \in \mathcal{C}$,

$$\nabla_{ii}^2 U_i(\boldsymbol{\gamma}) = \sum_{j \in \mathcal{V}_i} \left(\overbrace{2p_{ij}'(Q_j)}^{<0} + \overbrace{\gamma_i p_{ij}''(Q_j)}^{\geq 0}\right) \geq -2 \sum_{j \in \mathcal{V}_i} |p_{ij}'(Q_j)| \geq -2 \sum_{j \in \mathcal{V}_i} \max_{k \in \mathcal{V}_i} |p_{ik}'(0)|$$

$$= -2|\mathcal{V}_i| \max_{k \in \mathcal{V}_i} |p_{ik}'(0)| \geq -2|\mathcal{V}_i| \max_{k \in \mathcal{V}_i} |p_{ik}'(0)|. \tag{8}$$

So by (6), $0 > \nabla_{ii}^2 U_i(\boldsymbol{\gamma}) \geq -1/\sigma_i$ for all $i \in \mathcal{C}$.

Next in this proof, we bound the cross terms $\nabla_{i\ell}^2 U_i$ of the utility Hessian. These bounds require the introduction of SOMS in Definition 4, which represents the slope of the SOBP. Moreover, Lemmas 1, 2, and 3 put bounds on SOMS and precisely relate it to the SOBP; they follow from the binomial theorem. [28].

Definition 4 (Sum of monomial sums). *Let $\mathcal{I}$ be a finite index set and $\Omega \triangleq \{\gamma_i\}_{i \in \mathcal{I}}$ be an indexed family where $\gamma_i \in [0,1]$ for each $i \in \mathcal{I}$. For $\Gamma \subseteq \mathcal{I}$ and $h \in \mathbb{N}$, the* sum of monomial sums

$$\mathrm{SOMS}_h(\Gamma) \triangleq \sum_{\ell=0}^{|\Gamma|} (-1)^\ell \frac{1}{h+\ell} \sum_{\substack{\mathcal{C} \subseteq \Gamma \\ |\mathcal{C}|=\ell}} \left(\prod_{i \in \mathcal{C}} \gamma_i\right). \tag{9}$$

Lemma 1. *If $\Gamma \subseteq \mathcal{I}$ and $k \in \Gamma$, then $\partial \mathrm{SOBP}_1(\Gamma)/\partial \gamma_k = -\mathrm{SOMS}_2(\Gamma \setminus \{k\})$.*

Lemma 2. *If $\Gamma \subseteq \mathcal{I}$ and $h \in \mathbb{N}$, then $\mathrm{SOMS}_h(\Gamma) \geq (1/h) \prod_{k=1}^{|\Gamma|} k/(h+k)$.*

Lemma 3. *If $\Gamma \subseteq \mathcal{I}$ and $h \in \mathbb{N}$, then $\mathrm{SOMS}_h(\Gamma) \leq 1/h$.*

Take $\boldsymbol{\gamma} \in [0,1]^n$ and cooperator $i \in \mathcal{C}$. For another cooperator $\ell \in \mathcal{C} \setminus \{i\}$, if $\ell \notin \mathcal{C}_j$ (i.e., ℓ is not an outgoing cooperator for j), then $\partial Q_j/\partial \gamma_\ell = 0$ and $\partial \mathrm{SOBP}_1(\mathcal{C}_j - \{i\})/\partial \gamma_\ell = 0$ where $Q_j \triangleq \sum_{k \in \mathcal{C}_j} \gamma_k$ and SOBP is from Definition 1. So by introducing SOMS from Lemma 1,

$$0 \leq \sum_{\substack{\ell \in \mathcal{C} \\ \ell \neq i}} |\nabla_{i\ell}^2 U_i(\boldsymbol{\gamma})| = \sum_{\substack{\ell \in \mathcal{C} \\ \ell \neq i}} \left| \sum_{j \in \mathcal{V}_i} [\ell \in \mathcal{C}_j] \left(\begin{array}{c} p_{ij}'(Q_j) + \gamma_i p_{ij}''(Q_j) \\ + \underbrace{\mathrm{SOMS}_2(\mathcal{C}_j \setminus \{i,\ell\})}_{\partial/\partial \gamma_\ell \, \mathrm{SOBP}_1(\mathcal{C}_j - \{i\})} c_{ij} \end{array} \right) \right|$$

where $[\cdot]$ is the Iverson bracket (i.e., $[S] = 1$ or $[S] = 0$ when statement S is true or false). By Lemmas 2 and 3, $0 < \mathrm{SOMS}_2(\Gamma) \leq 1/2$ for all $\Gamma \subseteq \mathcal{C}$. Hence,

$$\sum_{\substack{\ell \in \mathcal{C} \\ \ell \neq i}} |\nabla_{i\ell}^2 U_i(\boldsymbol{\gamma})| \leq \sum_{\substack{\ell \in \mathcal{C} \\ \ell \neq i}} \sum_{j \in \mathcal{V}_i} [\ell \in \mathcal{C}_j] \left(|\underbrace{p_{ij}'(Q_j) + \gamma_i p_{ij}''(Q_j)}_{\leq 0}| + \frac{1}{2}|c_{ij}| \right),$$

which decouples the summations. So

$$\sum_{\substack{\ell \in \mathcal{C} \\ \ell \neq i}} |\nabla^2_{i\ell} U_i(\boldsymbol{\gamma})| = \sum_{j \in \mathcal{V}_i} \left(\left| p'_{ij}(Q_j) + \gamma_i p''_{ij}(Q_j) \right| + \frac{1}{2} |c_{ij}| \right) \sum_{\substack{\ell \in \mathcal{C} \\ \ell \neq i}} [\ell \in \mathcal{C}_j]$$

$$= \sum_{j \in \mathcal{V}_i} \left(\left| p'_{ij}(Q_j) + \gamma_i p''_{ij}(Q_j) \right| + \frac{1}{2} |c_{ij}| \right) (|\mathcal{C}_j| - 1).$$

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,

$$\sum_{\substack{\ell \in \mathcal{C} \\ \ell \neq i}} |\nabla^2_{i\ell} U_i(\boldsymbol{\gamma})| \leq \overbrace{2 \sum_{j \in \mathcal{V}_i \setminus \{m\}} \left(\underbrace{\left| p'_{ij}(Q_j) + \gamma_i p''_{ij}(Q_j) \right|}_{\leq 0} + \frac{1}{2} |c_{ij}| \right)}^{\substack{\text{Doubled contribution to sum from other cooperators} \\ \text{connected to assumed 3-conveyors in } \mathcal{V}_i \setminus \{m\}}}$$

$$\underbrace{+ \left| \underbrace{p'_{im}(Q_m) + \gamma_i p''_{im}(Q_m)}_{\leq 0} \right| + \frac{1}{2} |c_{im}|.}_{\substack{\text{Contribution to sum from} \\ \text{other cooperator of 2-conveyor } m \in \mathcal{V}_i}}$$

By Definition 2 of an SPF,

$$\sum_{\substack{\ell \in \mathcal{C} \\ \ell \neq i}} |\nabla^2_{i\ell} U_i(\boldsymbol{\gamma})| \leq - \sum_{j \in \mathcal{V}_i \setminus \{m\}} \left(2p'_{ij}(Q_j) + \gamma_i p''_{ij}(Q_j) \right) - \left(p'_{im}(Q_m) + \gamma_i p''_{im}(Q_m) \right)$$

$$+ \left(\overbrace{|\mathcal{V}_i \setminus \{m\}|}^{m \in \mathcal{V}_i} + \frac{1}{2} \right) \max_{j \in \mathcal{V}_i} |c_{ij}|$$

$$= - \sum_{j \in \mathcal{V}_i \setminus \{m\}} \left(2p'_{ij}(Q_j) + \gamma_i p''_{ij}(Q_j) \right) - \left(p'_{im}(Q_m) + \gamma_i p''_{im}(Q_m) \right)$$

$$+ \left(|\mathcal{V}_i| - \frac{1}{2} \right) \max_{j \in \mathcal{V}_i} |c_{ij}|$$

$$\leq - \sum_{j \in \mathcal{V}_i} \left(2p'_{ij}(Q_j) + \gamma_i p''_{ij}(Q_j) \right) + \left(2p'_{im}(Q_m) + \gamma_i p''_{im}(Q_m) \right)$$

$$- \left(p'_{im}(Q_m) + \gamma_i p''_{im}(Q_m) \right) + \left(|\mathcal{V}_i| - \frac{1}{2} \right) \max_{j \in \mathcal{V}_i} |c_{ij}|$$

$$\leq - \overbrace{\sum_{j \in \mathcal{V}_i} \left(2p'_{ij}(Q_j) + \gamma_i p''_{ij}(Q_j) \right)}^{\nabla^2_{ii} U_i(\boldsymbol{\gamma})} + \overbrace{p'_{im}(Q_m)}^{<0}$$

$$+ \left(|\mathcal{V}_i| - \frac{1}{2} \right) \max_{j \in \mathcal{V}_i} |c_{ij}|.$$

So

$$\sum_{\substack{\ell \in \mathcal{C} \\ \ell \neq i}} |\nabla^2_{i\ell} U_i(\boldsymbol{\gamma})| \leq -\nabla^2_{ii} U_i(\boldsymbol{\gamma}) - \underbrace{\left(\min_{j \in \mathcal{V}_i} |p'_{ij}(Q_j)| - \left(|\mathcal{V}_i| - \frac{1}{2} \right) \max_{j \in \mathcal{V}_i} |c_{ij}| \right)}_{> \, 0 \text{ by } (7)}$$

where, by the assumption in (7), the underbraced expression is strictly positive. The desired result follows from this inequality and (8). In particular, as shown by Bertsekas and Tsitsiklis [6, Propositions 1.1, 1.11, and 5.1 from Ch. 3], T is a maximum-norm contraction mapping with unique fixed point $\boldsymbol{\gamma}^*$ that is the Nash equilibrium of the cooperation game. Futhermore, the TADI sequence $\{\boldsymbol{\gamma}(t)\}_{t \in \mathcal{T}}$ generated by T converges to $\boldsymbol{\gamma}^*$ [6, Proposition 2.1 from Ch. 6]. $\qquad\square$

4.2 Interpretations

As shown in the proof of Theorem 1, every 3-conveyor contributes two payment slope p'_{ij} terms to $\nabla^2_{i\ell} U_i$ that are canceled by the two slope terms in $\nabla^2_{ii} U_i$. Hence, when 3-conveyors are connected to a cooperator, the cooperator loses control of its utility gradient along its cooperation coordinate unless there exists a 2-conveyor that it can dominate. 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. It may be possible to weaken Theorem 1 to
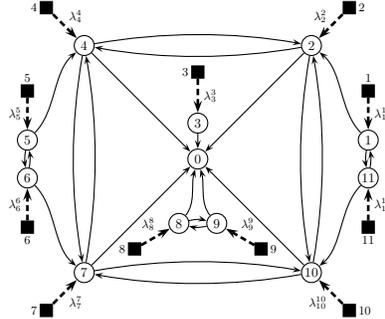

Fig. 4. Many-agent task-processing network with stable topology.

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.

The restriction in (7) is similar to the network generalization of Hamilton's rule [16] discussed by Ohtsuki et al. [25] and Nowak [24]. 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 it is shown that a sufficient condition for cooperation to spread is that the benefit-to-cost ratio is greater than the average network degree (i.e., number of neighbors connected to each node). Here, the TPN is not a substrate for birth–death processes; however, the rule in (7) 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.

5   Simulation of Cooperative AAV Scenario

Consider the AAV scenario shown in Fig. 2. Assume that $\pi_i^k = 1$, $c_{ij}^\ell = 0.1$, and $q_{ij}^\ell = 1$ for all $i \in \mathcal{A}$, $j \in \mathcal{A} \setminus \{i\}$, $k \in \mathcal{Y}_i$, and $\ell \in \mathcal{Y}_j$. Also assume that $\lambda_1^1 = 0.6$, $\lambda_3^3 = 1.7$, $0 < \lambda_2 \le 5$, 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 $\sigma_\ell \triangleq 1/(4 \max_{i \in \mathcal{A}, j \in \mathcal{V}_i} p_{ij}'([0,0,0]^T))$ for all $\ell \in \mathcal{A}$ yields a convergent TADI for this scenario. MATLAB simulation results summarized in Fig. 5 show how the resulting Nash equilibrium $\boldsymbol{\gamma}^*$ depends upon the AAV encounter rates. In particular, the Nash equilibrium has the desirable feature that $\lambda_i > \lambda_j$ 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 increases, payment func-
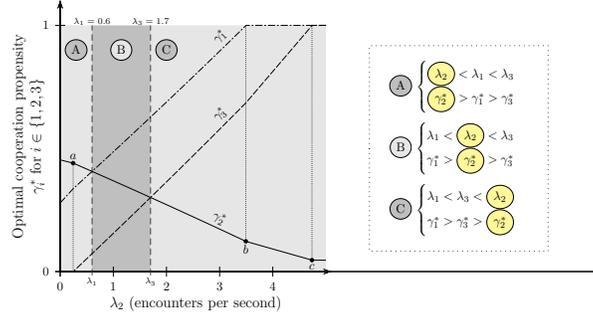

Fig. 5. AAV optimal cooperation propensity as encounter rates vary.

tion 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. Moreover, at point b when the ascent of γ_1^* truncates, the rate that γ_2^* decreases shallows. At point c when γ_3^* also truncates, the γ_2^* graph flattens entirely. Hence, to reduce the load on saturated cooperators, agent 2 reciprocates for their cooperation by not reducing its own cooperation level to zero. So even though each agent's own encounter rate has no direct relationship to its TADI-directed movement, a desirable coupling between encounter rates and optimal cooperation levels emerges from cyclic relationships on the network.

6 Conclusion

A framework for cooperative task processing on a network has been presented. Using this framework, a particular 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 climb their own local utility functions yet still achieve an equilibrium where task processing is shared among different agents. The present work adjusts each agent's overall cooperation propensity in order to maximize economic returns over a lifetime of task encounters and processing requests. Future work should address the case where each agent associates a different cooperation propensity with each of its connected conveyors. Likewise, forwarding probabilities could be considered to be decision variables that should be adjusted across each agent's connected cooperators. The present work associates only one decision variable with each distributed agent, and so it makes the simplifying assumption that those variables come from a Cartesian product space. However, if future frameworks place multiple decision variables on a single distributed agent, that assumption may be relaxed.

A weakness of the present work is that it implicitly assumes that agents either have infinite processing capacity or that all tasks have negligible processing time. Processing and switching durations are central motivations for the work of Perkins and Kumar [29] just as finite capacity motivates the work of Cruz [9]. These effects can incorporated by explicitly modeling the average processing time of each task. 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, and this rate will be depressed by the processing time of each task. Moreover, the time spent processing a task represents a opportunity cost due to the lost time available for encountering other tasks that return higher profit. Because each arrival is independent, the average reward results of Johns and Miller [18] for Markov renewal–reward processes can be used to model the long-term rate of gain in this case. Hence, the utility functions discussed in this work can be easily modified to include these effects. If analytically tractable, optimal results can be found that account for appreciable processing times.

Bibliography

[1] Altman, E., Kherani, A.A., Michiardi, P., Molva, R.: Non-cooperative forwarding in ad-hoc networks. In: Proc. Networking. Lecture Notes in Computer Science, vol. 3462, pp. 486–498 (2005)

[2] Altman, E., Kumar, A., Kumar, D., Venkatesh, R.: Cooperative and non-cooperative control in IEEE 802.11 WLANs. In: Proc. 19th Int. Teletraffic Congr. Beijing (August 29 – September 2, 2005)

[3] Amir, Y., Awerbuch, B., Barak, A., Borgstrom, R.S., Keren, A.: An opportunity cost approach for job assignment in a scalable computing cluster. IEEE Trans. Parallel Distrib. Syst. 11, 760–768 (July 2000)

[4] Bacharach, M.: Economics and the Theory of Games. Macmillan, London (1976)

[5] Başar, T., Olsder, G.J.: Dynamic Noncooperative Game Theory. No. 23 in Classics in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, PA, second edn. (1999)

[6] Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and Distributed Computation: Numerical Methods. Athena Scientific, Belmont, Massachusetts (1997)

[7] Buttyán, L., Hubaux, J.P.: Stimulating cooperation in self-organizing mobile ad hoc networks. Mob. Networks Appl. 8, 579–592 (2003)

[8] Buyya, R.: Economic-based Distributed Resource Management and Scheduling for Grid Computing. Ph.D. thesis, Monash University, Melbourne, Australia (April 2002)

[9] Cruz, R.L.: A calculus for network delay, part II: network analysis. IEEE Trans. Inf. Theory 37, 132–141 (January 1991)

[10] Feigenbaum, J., Shenker, S.: Distributed algorithmic mechanism design: recent results and future directions. In: Proc. 6th Int. Workshop Discrete Algorithms and Methods Mobile Computing and Communication. pp. 1–13. Atlanta, Georgia, USA (September 28, 2002)

[11] Finke, J., Passino, K.M.: Stable cooperative vehicle distributions for surveillance. J. Dyn. Syst., Meas., Control. 129(5), 597–608 (2007)

[12] Finke, J., Passino, K.M., Sparks, A.G.: Stable task load balancing for cooperative control of networked autonomous air vehicles. IEEE Trans. Control. Syst. Technol. 14, 789–803 (September 2006)

[13] Gil, A.E., Passino, K.M.: Stability analysis of network-based cooperative resource allocation strategies. Automatica 42(2), 245–250 (2005)

[14] Gil, A.E., Passino, K.M., Ganapathy, S., Sparks, A.: Cooperative task scheduling for networked uninhabited air vehicles. IEEE Trans. Aerosp. Electron. Syst. 44, 561–581 (April 2008)

[15] Grosu, D., Chronopoulos, A.T.: Algorithmic mechanism design for load balancing in distributed systems. IEEE Trans. Syst., Man, Cybern. 34, 77–84 (February 2004)

[16] Hamilton, W.D.: The genetical evolution of social behavior. I. J. Theor. Biol. 7(1), 1–16 (1964)

[17] Ipakchi, A., Albuyeh, F.: Grid of the future. IEEE Power Energy Mag. 7(2), 52–62 (March/April 2009)

[18] Johns, M.V., Miller, Jr., R.G.: Average renewal loss rates. Ann. Math. Stat. 34(2), 396–401 (June 1963)

[19] Lange, D.B.: Mobile objects and mobile agents: the future of distributed computing? In: Jul, E. (ed.) ECOOP'98 – Object-Oriented Programming: 12th European Conf., Brussels, Belgium, July 20–24, 1998, Proc. Lecture Notes in Computer Science, vol. 1445, pp. 1–12. Springer, Berlin (July 20–24, 1998)

[20] Lange, D.B., Oshima, M.: Seven good reasons for mobile agents. Commun. ACM 42(3), 88–89 (March 1999)

[21] Maheswaran, R.T., Imer, O.Ç., Başar, T.: Agent mobility under price incentives. In: Proc. 38th IEEE Conf. Decision and Control. vol. 4, pp. 4020–4025. Phoenix, Arizona, USA (December 7–10, 1999)

[22] Mas-Colell, A., Whinston, M.D., Green, J.R.: Microeconomic Theory. Oxford University Press, New York (1995)

[23] Nicholson, W.: Microeconomic Theory: Basic Principles and Extensions. Dryden Press, Fort Worth, TX, fifth edn. (1992)

[24] Nowak, M.A.: Five rules for the evolution of cooperation. Science 314(5805), 1560–1563 (December 8, 2006)

[25] Ohtsuki, H., Hauert, C., Lieberman, E., Nowak, M.A.: A simple rule for the evolution of cooperation on graphs. Nature 441(7092), 502–505 (2006)

[26] Oram, A. (ed.): Peer-to-Peer: Harnessing the Power of Disruptive Technologies. O'Reilly & Associates, Sebastopol, CA (2001)

[27] Osborne, M.J., Rubinstein, A.: A Course in Game Theory. MIT Press, Cambridge, MA (1994)

[28] Pavlic, T.P., Passino, K.M.: Cooperative task-processing networks: parallel computation of non-trivial volunteering equilibria. Tech. Rep. OSU-CISRC-3/11-TR05, The Ohio State University (2011)

[29] Perkins, J.R., Kumar, P.R.: Stable, distributed, real-time scheduling of flexible manufacturing/assembly/disassembly systems. IEEE Trans. Autom. Control. 34, 139–148 (February 1989)

[30] Qiu, D., Srikant, R.: Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In: Yavatkar, R., Zegura, E.W., Rexford, J. (eds.) Proc. ACM SIGCOMM 2004 Conf. Applications, Technologies, Architectures, and Protocols Computer Communication. pp. 367–378. Portland, Oregon, USA (August 30 – September 3, 2004)

[31] Reed, D., Pratt, I., Menage, P., Early, S., Stratford, N.: Xenoservers: accountable execution of untrusted programs. In: Proc. 7th Workshop Hot Topics in Operating Systems. pp. 136–141. Rio Rico, Arizona, USA (March 28–30, 1999)

[32] Waldspurger, C.A., Hogg, T., Huberman, B.A., Kephart, J.O., Stornetta, W.S.: Spawn: a distributed computational economy. IEEE Trans. Softw. Eng. 18, 103–117 (February 1992)

[33] White, J.E.: Telescript technology: mobile agents. In: Milojičić, D., Douglis, F., Wheeler, R. (eds.) Mobility: Processes, Computers, and Agents, pp. 460–493. ACM Press, New York (1999)