52nd IEEE Conference on Decision and Control
December 10-13, 2013. Florence, Italy

A Distributed Learning Algorithm with Bit-valued Communications for Multi-agent Welfare Optimization

Anup Menon and John S. Baras

***Abstract*— A multi-agent system comprising N agents, each picking actions from a finite set and receiving a payoff that depends on the action of the whole, is considered. The exact form of the payoffs are unknown and only their values can be measured by the respective agents. A decentralized algorithm was proposed by Marden et al. [1] and in the authors' earlier work [2] that, in this setting, leads to the agents picking welfare optimizing actions under some restrictive assumptions on the payoff structure. This algorithm is modified in this paper to incorporate exchange of certain bit-valued information between the agents over a directed communication graph. The notion of an interaction graph is then introduced to encode known interaction in the system. Restrictions on the payoff structure are eliminated and conditions that guarantee convergence to welfare minimizing actions w.p. 1 are derived under the assumption that the union of the interaction graph and communication graph is strongly connected.**

I. Introduction

An important direction of research in cooperative control of multi-agent systems is game theoretic control. This refers to the paradigm of: 1. designing individual utility functions for agents such that certain solution concepts (like Nash equilibria (NE)) correspond to desirable system-wide outcomes; and 2. prescribing learning rules that allows agents to learn such equilibria [3]. Also, the utilities and the learning rules must conform to the agents' information constraints. A popular choice is to design utilities such that the resulting game has a special structure so that the corresponding solution concepts are efficient w.r.t. system-wide objectives. NE of potential games, for instance, correspond to the extremal values of the potential function which can then be chosen so that its extrema correspond to desirable system-wide behavior. Examples of such utility design for specific applications range from distributed optimization [4] to coverage problems in sensor networks [5] and power control in wireless networks [6].

The other advantage of designing utilities with special structure is that players can be prescribed already available learning algorithms from the large body of work on *learning in repeated games* that helps them learn to play NE [7], [8], [9]. Some of these algorithms have the desirable feature of being payoff-based, i.e. an agent adjusts its play only on the basis of its past payoffs and actions, and does not require the agents to have any knowledge of the structure of the game. However, the success for most learning procedures is only guaranteed under an assumption on the game such as potential, weakly acyclic or congestion game.

Research partially supported by the US Air Force Office of Scientific Research MURI grant FA9550-09-1-0538, by the National Science Foundation (NSF) grant CNS-1035655, and by the National Institute of Standards and Technology (NIST) grant 70NANB11H148.

The authors are with the Institute for Systems Research and the Department of Electrical and Computer Engineering at the University of Maryland, College Park, MD 20742, USA `amenon@umd.edu`, `baras@umd.edu`

Thus, while an effective paradigm, game theoretic control has the following limitations:

- Since available algorithms are provably correct only for certain classes of games, there is a burden to design utilities that conform to such structure for each application.
- If the system requirements prohibit design of utilities with special structure, equilibrating to NE may be inefficient w.r.t. desirable outcome (also, may be unnecessary in non-strategic situations).

While within the paradigm of game theoretic control, our approach is complementary to that of utility design. Instead, we focus on algorithm design for welfare (i.e. the sum of individual utilities) optimization for arbitrary utilities. To motivate this approach we consider an application where the current paradigm is too restrictive: the problem of maximizing the total power production of a wind farm [10]. Aerodynamic interactions between different wind turbines are not well understood and there are no good models to predict the effects of one turbine's actions on another. However, it is clear that the amount of power a turbine extracts from the wind has a direct effect on the power production of turbines downstream (such a region of influence is called a wake, see Figure 1(a)). The information available to each turbine is its own power production and a decentralized algorithm that maximizes the total power production of the farm is sought. Since there are no good models for the interactions, there is little hope to design utilities with special structure that are functions of such individual power measurements. This points towards the need for algorithms that are applicable when there is little structural information about the utilities (for instance, a turbine can be assigned its individual power as its utility which, in turn, can depend on the actions taken by others in complex ways; see [10] for details).

A decentralized learning algorithm is presented in [1] with the intent to address this issue of unknown payoff structures. This algorithm allows agents to learn welfare maximizing actions and does not require any knowledge about the exact functional form of the utilities. In our earlier work [2], we provide conditions that guarantee convergence of this algorithm. However, convergence is guaranteed only under an assumption on the utilities called *interdependence* (which, for instance, need not hold for the wind farm problem).

The contribution of this paper is a distributed multi-agent learning algorithm that:

1) eliminates the need for any structural assumptions on the utilities by using inter-agent communication;
2) and, under appropriate conditions, ensures that agent actions converge to global extrema of the welfare function.

Regarding the use of inter-agent communication, we develop a framework for capturing known interaction in the system and prove results that help design "minimal" communication networks that guarantee convergence. The exchanged information in our algorithm is bit-valued which has implementation and robustness advantages. The framework developed also contrasts between implicit interaction between the agents via utilities and explicit interaction via communication. We also wish to point out that the problem formulated here can be thought of as a multi-agent formulation of a discretized extremum seeking problem [11] and the algorithm provides convergence to global optimal states with few restrictions on the functions involved.

The remainder of the paper is organized as follows. In section II we formulate the problem, develop the analysis framework, present the algorithm and state the main convergence result. Section III introduces *Perturbed Markov Chains* and states relevant results. In section IV, the results of Section III are used to prove the main result of section II. The paper concludes with some numerical illustrations and discussions about future work.

NOTATION

The paper deals exclusively with discrete-time, finite state space Markov chains. A time-homogeneous Markov chain with Q as its 1-step transition matrix means that the i^{th} row and j^{th} column entry $Q_{i,j} = \mathbb{P}(\mathbf{X}_{t+1} = j | \mathbf{X}_t = i)$, where $\mathbf{X}_t$ denotes the state of the chain at time t. If the row vector η_t denotes the probability distribution of the states at time t, then $\eta_{t+1} = \eta_t \ Q$. More generally, if $Q(t)$ denotes the 1-step transition probability matrix of a time-nonhomogeneous Markov chain at time t, then for all $m > n$, $\mathbb{P}(\mathbf{X}_m = j | \mathbf{X}_n = i) = Q^{(n,m)}_{i,j}$, where the matrix $Q^{(n,m)} = Q(n) \cdot Q(n+1) \cdots Q(m-1)$. The time indices of all Markov chains take consecutive values from the set of natural numbers $\mathbb{N}$. A Markov chain should be understood to be homogeneous unless stated otherwise. We denote the N-dimensional vector of all zeros and all ones by the bold font $\mathbf{0}$ and $\mathbf{1}$ respectively. For a multi-dimensional vector x, its i^{th} component is denoted by x_i; and that of x_t by $(x_t)_i$.

II. PROBLEM STATEMENT AND PROPOSED ALGORITHM

A. A Multi-agent Extremum Seeking Formulation

1) Agent Model: We consider N, possibly heterogeneous, agents indexed by i. The i^{th} agent can pick actions from a set $\mathcal{A}_i$, $1 < |\mathcal{A}_i| < \infty$; the joint action of the agents is an element of the set $\mathcal{A} = \prod_{i=1}^{N} \mathcal{A}_i$. The action of the i^{th} agent in the joint action $a \in \mathcal{A}$ is denoted by a_i. Further, given the i^{th} individual's present action is $b \in \mathcal{A}_i$, the choice of its very next action is restricted to be from $\mathcal{A}_i(b) \subset \mathcal{A}_i$.

Assumption 1: For any $b \in \mathcal{A}_i$, $b \in \mathcal{A}_i(b)$ and there exists an enumeration $\{b_1, ..., b_{|\mathcal{A}_i|}\}$ of $\mathcal{A}_i$ such that $b_{j+1} \in \mathcal{A}_i(b_j)$ for $j = 1, ..., (|\mathcal{A}_i| - 1)$ and $b_1 \in \mathcal{A}_i(b_{|\mathcal{A}_i|})$.

The first part of this assumption allows for the possibility of picking the same action in consecutive steps and the second ensures that any element of $\mathcal{A}_i$ is "reachable" from any other. Specific instances of such agent models in literature include the discretized position and viewing-angle sets for mobile sensors in [5], discretized position of a robot in a finite lattice in [12], [13] and the discretization of the axial induction factor of a wind turbine in [10].

An individual has a private utility that can be an arbitrary time-invariant function of the action taken by the whole but is measured or accessed only by the individual. Agent i's utility is denoted by $u_i : \mathcal{A} \to \mathbb{R}^+$. Examples include artificial potentials used to encode information about desired formation geometry for collaborative control of autonomous robots in [12], [13] and the measured power output of an individual wind turbine in [10]. At any time t, agent i only measures or receives $(u_t^{mes})_i = u_i(a_t)$ since neither the joint action a_t nor any information about $u_i(\cdot)$ is known to the agent.

The objective of the multi-agent system is to collaboratively minimize (or maximize) the welfare function $W^* = \min_{a \in \mathcal{A}} W(a)$, where $W(a) = \sum_{i=1}^{N} u_i(a)$. Achieving this objective results in a desirable behavior of the whole like a desired geometric configuration of robots in [12], [13], desired coverage vs. sensing energy trade-off in [5] and maximizing the power output of a wind farm in [10]. Thus we seek distributed algorithms for the agents to implement so that their collective actions converge in an appropriate sense to the set

$$\mathcal{A}^* = \{\arg\min_{a \in \mathcal{A}} W(a)\}.$$

2) Interaction Model: Interaction in a multi-agent setting can comprise of explicit communication between agents via communication or can be implicit with actions of an agent reflecting on the payoff of another. We present a general modeling framework that allows the designer to encode known inter-agent interactions in the system while explicit communication takes place over a simple signaling network with only a bit-valued variable exchanged.

1) **Interaction Graph**
Consider a directed graph $\mathcal{G}_I(a)$ for every $a \in \mathcal{A}$ with a vertex assigned to each agent. Its edge set contains the directed edge (j, i) if and only if $\exists \ b \in \mathcal{A}_j$ such that $u_i(a) \neq u_i(b, a_{-j})$.[1] Thus, for every action profile a, $\mathcal{G}_I(a)$ encodes the set of agents whose actions can (and must) affect the payoffs of other specific agents. We call this graph the *interaction graph*.
In the case of a wind farm, power production of a turbine downstream of another is affected by the

[1]We borrow notation from the game theory literature: a_J denotes the actions taken by the agents in subset J from the collective action a and the actions of the rest is denoted by a_{-J}.

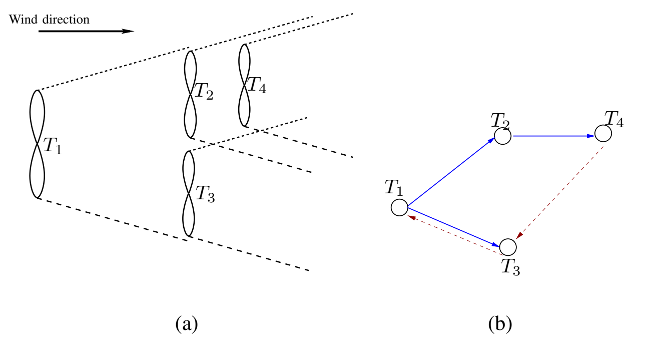

Fig. 1. (a) Schematic diagram of a wind farm; a loop represents a wind turbine and the dotted lines its corresponding wake. (b) Solid arrows represent edges in $\mathcal{G}_I$ and the dotted arrows edges in $\mathcal{G}_c$.

actions of the latter (see Figure 1 (b)). For the collaborative robotics problem, all robots that contribute to the artificial potential of a given robot constitute the latter's in-neighbors in the interaction graph. Essentially, the interaction graph is a way of encoding certain "coarse" information about the structure of the payoff functions even in the absence of explicit knowledge of their functional forms.

2) **Communication Graph**
The agents are assumed to have a mechanism to transmit a bit-valued message to other agents within a certain range. The mode of communication is broadcast and an agent need not know which other agents are receiving its message. For each $a \in \mathcal{A}$, we model this explicit information exchange by a directed graph over the set of agents $\mathcal{G}_c(a)$ called the *communication graph*. A directed edge (j, i) in $\mathcal{G}_c(a)$ represents that agent j can send a message to agent i when the joint action being played is a. Let $\mathcal{N}_i(a)$ denote the in-neighbors of agent i in the communication graph. Each transmission is assumed to last the duration of the algorithm iteration.

We stress this framework is for modeling and analysis at the level of the system designer; the agents neither know the joint action nor the corresponding neighbors in $\mathcal{G}_I(\cdot)$ or $\mathcal{G}_c(\cdot)$ and simply go about measuring their utilities, broadcasting messages and receiving such broadcast messages from other agents when permitted by $\mathcal{G}_c(a)$.

B. The Decentralized Algorithm

Endow agent i with a state $x_i = [a_i, m_i]$; $a_i \in \mathcal{A}_i$ corresponds to the action picked and m_i is the $\{0,1\}$-valued 'mood' of agent i. When the mood variable equals 1 we call the agent "content", else "discontent". The collective state of all agents is denoted by $x = (a, m)$. Other than recording its state, each agent maintains a variable $\overline{u}_i$, which records the payoff it received in the previous iterate. At $t = 0$, the agent i initializes $(m_0)_i = 0$, picks an arbitrary $(a_0)_i \in \mathcal{A}_i$ and records the received payoff $\overline{u}_i = (u_0^{mes})_i$. With slight abuse of notation, we let $\mathcal{G}_c(a_t) = \mathcal{G}_c(t)$ and $\mathcal{A}_i((a_t)_i) = \mathcal{A}_i(t)$.

For a certain pre-specified monotone decreasing sequence $\{\epsilon_t\}_{t\in\mathbb{N}}$, with $\epsilon_t \to 0$ as $t \to \infty$, and constants $c > W^*$, $\beta_1, \beta_2 > 0$, agent i performs the following sequentially at every ensuing time instant $t > 0$.

Start

Step 1: Receive $(\mathbf{m}_{t-1})_j$ from all $j \in \mathcal{N}_i(t-1)$, i.e. the in-neighbors of i in $\mathcal{G}_c(t-1)$. Compute temporary variable $\tilde{m}_i$ as follows.

1) If $(\mathbf{m}_{t-1})_i = 0$, set $\tilde{m}_i = 0$;
2) else, if $(\mathbf{m}_{t-1})_i = 1$ and $\prod_{j\in\mathcal{N}_i(t-1)} (\mathbf{m}_{t-1})_j = 1$, set $\tilde{m}_i = 1$;
3) and if $(\mathbf{m}_{t-1})_i = 1$ and $\prod_{j\in\mathcal{N}_i(t-1)} (\mathbf{m}_{t-1})_j = 0$, set $\tilde{m}_i = \{0,1\}$ w.p. $\{1-\epsilon_t^{\beta_1}, \epsilon_t^{\beta_1}\}$.

Step 2: Pick $(\mathbf{a}_t)_i$ as follows.

1) If $\tilde{m}_i = 1$, pick $(\mathbf{a}_t)_i$ from $\mathcal{A}_i(t-1)$ according to the p.m.f.

$$p(b) = \begin{cases} 1-\epsilon_t^c & \text{if } b = (\mathbf{a}_{t-1})_i \\ \frac{\epsilon_t^c}{|\mathcal{A}_i(t-1)|-1} & \text{otherwise.} \end{cases} \tag{1}$$

2) Else, if $\tilde{m}_i = 0$, pick $(\mathbf{a}_t)_i$ according to the uniform distribution

$$p(b) = \frac{1}{|\mathcal{A}_i(t-1)|} \quad \text{for all } b \in \mathcal{A}_i(t-1). \tag{2}$$

Step 3: Measure or receive payoff $(\mathbf{u}_t^{mes})_i (= u_i(\mathbf{a}_t))$.

Step 4: Update $(\mathbf{m}_t)_i$ as follows.

1) If $\tilde{m}_i = 1$ and $((\mathbf{a}_t)_i, (\mathbf{u}_t^{mes})_i) = ((\mathbf{a}_{t-1})_i, \overline{u}_i)$, then set $(\mathbf{m}_t)_i = 1$;
2) else, if $\tilde{m}_i = 1$ and $((\mathbf{a}_t)_i, (\mathbf{u}_t^{mes})_i) \neq ((\mathbf{a}_{t-1})_i, \overline{u}_i)$, set $(\mathbf{m}_t)_i = \{0,1\}$ w.p. $\{1-\epsilon_t^{\beta_2}, \epsilon_t^{\beta_2}\}$;
3) and if $\tilde{m}_i = 0$, set

$$(\mathbf{m}_t)_i = \begin{cases} 0 & \text{w.p.} \quad 1-\epsilon_t^{(\mathbf{u}_t^{mes})_i} \\ 1 & \text{w.p.} \quad \epsilon_t^{(\mathbf{u}_t^{mes})_i}. \end{cases} \tag{3}$$

Update $\overline{u}_i (\leftarrow \mathbf{u}_t^{mes})_i$.

Step 5: Broadcast $(\mathbf{m}_t)_i$ to all out-neighbors in $\mathcal{G}_c(t)$.

Stop

It is easy to see that the algorithm defines a nonhomogeneous Markov chain on the state space $S = \mathcal{A} \times \{0,1\}^N$.

C. Convergence Guarantees

The following is the main convergence result for the decentralized algorithm described above.

Theorem 1: Let

1) $\sum_{t=1}^{\infty} \epsilon_t^c = \infty$ and
2) For every $a \in \mathcal{A}$, $\mathcal{G}_c(a) \cup \mathcal{G}_I(a)$ be strongly connected.

Then, if $\mathbf{X}_t = [\mathbf{a}_t, \mathbf{m}_t]$ denotes the collective state of the agents at time t,

$$\lim_{t\to\infty} \mathbb{P}[\mathbf{a}_t \in \mathcal{A}^*] = 1.$$

From a practical view-point, the result provides a system-designer with guidelines on how to guarantee convergence of the above algorithm. The first assumption translates to a constraint on the sequence $\{\epsilon_t\}_{t\in\mathbb{N}}$ (or an "annealing schedule") on how fast it may approach zero.

The second provides flexibility to design a 'minimal' communication network by utilizing information about the payoff structure (such a choice is made in Figure 1 (b)). For instance, if the designer has no information about the structure of the payoff function ($\mathcal{G}_I(\cdot) = \emptyset$), a communication network such that $\mathcal{G}_c(a)$ is strongly connected for all $a \in \mathcal{A}$ can be installed to guarantee convergence. The other extreme case is for all $a \in \mathcal{A}$, $\mathcal{G}_I(a)$ is strongly connected; then the algorithm converges even in the absence of any explicit communication.

The proof for Theorem 1 relies on the theory of *perturbed Markov chains*. The general theory is reviewed in Section III and two important results are stated - Theorem 2 from [14] and Theorem 3 from [2]. Theorem 1 is then proved in Section IV with this theory applied to the special case of the algorithm.

III. Perturbed Markov Chains

The theory of perturbed Markov chains, developed by Young [14], is reviewed in this section. Consider a homogeneous Markov chain with possibly several stationary distributions. Perturbed Markov chains are essentially a way of "choosing" amongst these by perturbing elements of the Markov chains. This section describes this theory in detail and also presents results on how to reduce ϵ over time while evolving according to the perturbed chain (rendering the chain nonhomogeneous) while retaining ergodicity.

A. Perturbed Markov Chains

Let $P(0)$ be the 1-step transition probability matrix of a Markov chain on a finite state space S. We refer to this chain as the *unperturbed chain*.

Definition 3.1: A regular perturbation of $P(0)$ consists of a stochastic matrix valued function $P(\epsilon)$ on some non-degenerate interval $(0, a]$ that satisfies, for all $x, y \in S$,

1) $P(\epsilon)$ is irreducible and aperiodic for each $\epsilon \in (0, a]$,
2) $\lim_{\epsilon\to 0} P_{x,y}(\epsilon) = P_{x,y}(0)$ and
3) if $P_{x,y}(\epsilon) > 0$ for some ϵ, then $\exists\ r(x,y) \geq 0$ such that $0 < \lim_{\epsilon\to 0} \epsilon^{-r(x,y)} P_{x,y}(\epsilon) < \infty$.

An immediate consequence of the first requirement is that there exists a unique stationary distribution $\mu(\epsilon)$ satisfying $\mu(\epsilon)P(\epsilon) = \mu(\epsilon)$ for each $\epsilon \in (0, a]$. The other two requirements dictate the way the perturbed chain converges to the unperturbed one as $\epsilon \to 0$. Let $\mathfrak{L} = \{f \in \mathcal{C}^\infty |\ f(\epsilon) \geq 0, f(\epsilon) = \sum_{i=1}^{L} a_i \epsilon^{b_i}$ for some $a_i \in \mathbb{R}, b_i \geq 0\}$ for some large enough but fixed $L \in \mathbb{N}$, where $\mathcal{C}^\infty$ is the space of smooth functions. The following assumption will be invoked later.

Assumption 2: For all $x, y \in S$, $P_{x,y}(\epsilon) \in \mathfrak{L}$.

We develop some notation that will help state the main result regarding perturbed Markov chains. The parameter $r(x, y)$ in the definition of regular perturbation is called the *1-step transition resistance* from state x to y. Notice that $r(x, y) = 0$ only for the one step transitions $x \to y$ allowed under $P(0)$. A *path* $h(a \to b)$ from a state $a \in S$ to $b \in S$ is an ordered set $\{a = x_1, x_2, \ldots, x_n = b\} \subseteq S$ such that every transition $x_k \to x_{k+1}$ in the sequence has positive 1-step probability according to $P(\epsilon)$. The resistance of such a path is given by $r(h) = \sum_{k=1}^{n-1} r(x_k, x_{k+1})$.

Definition 3.2: For any two states x and y, the *resistance* from x to y is defined by $\rho(x, y) = \min\{r(h) |\ h(x \to y)$ is a path$\}$.

Definition 3.3: Given a subset $A \subset S$, its *co-radius* is given by $CR(A) = \max_{x\in S\setminus A} \min_{y\in A} \rho(x, y)$.

Thus, $\rho(x, y)$ is the minimum resistance over all possible paths starting at state x and ending at state y and the co-radius of a set specifies the maximum resistance that must be overcome to enter it from outside. We will extend the definition of resistance to include resistance between two subsets $S_1, S_2 \subset S$:

$$\rho(S_1, S_2) = \min_{x\in S_1, y\in S_2} \rho(x, y).$$

Since $P(\epsilon)$ is irreducible for $\epsilon > 0$, $\rho(S_1, S_2) < \infty$ for all $S_1, S_2 \subset S$.

Definition 3.4: A *recurrence or communication class* of a Markov chain is a non-empty subset of states $E \subseteq S$ such that for any $x, y \in E$, $\exists\ h(x \to y)$ and for any $x \in E$ and $y \in S \setminus E$, $\nexists\ h(x \to y)$.

Let us denote the recurrence classes of the unperturbed chain $P(0)$ as $E_1, ..., E_M$. Consider a directed graph $\mathcal{G}_{RC}$ on the vertex set $\{1, ..., M\}$ with each vertex corresponding to a recurrence class. Let a *j-tree* be a spanning subtree in $\mathcal{G}_{RC}$ that contains a unique directed path from each vertex in $\{1, ..., M\} \setminus \{j\}$ to j and denote the set of all j-trees in $\mathcal{G}_{RC}$ by $\mathcal{T}^j_{RC}$.

Definition 3.5: The *stochastic potential* of a recurrence class E_i is

$$\gamma(E_i) = \min_{T\in\mathcal{T}^i_{RC}} \sum_{(j,k)\in T} \rho(E_j, E_k).$$

Let $\gamma^* = \min_{E_i} \gamma(E_i)$.

We are now ready to state the main result regarding perturbed Markov chains.

Theorem 2 ([14], Theorem 4): Let $E_1, ..., E_M$ denote the recurrence classes of the Markov chain $P(0)$ on a finite state space S. Let $P(\epsilon)$ be a regular perturbation of $P(0)$ and let $\mu(\epsilon)$ denote its unique stationary distribution. Then,

1) As $\epsilon \to 0$, $\mu(\epsilon) \to \mu(0)$, where $\mu(0)$ is a stationary distribution of $P(0)$ and
2) A state is stochastically stable i.e. $\mu_x(0) > 0 \Leftrightarrow x \in E_i$ such that $\gamma(E_i) = \gamma^*$.

B. Ergodicity of Nonhomogeneous Perturbed Markov Chains

Consider the nonhomogeneous Markov chain resulting from picking the ϵ along the evolution of $P(\epsilon)$ at time instant t as the corresponding element ϵ_t of the sequence $\{\epsilon_t\}_{t\in\mathbb{N}}$. We henceforth refer to this sequence as the annealing schedule and the resulting Markov chain as the nonhomogeneous perturbed chain. Theorem 3 provides conditions on the annealing schedule that guarantee ergodicity of the nonhomogeneous perturbed chain with $\mu(0)$ (as in Theorem

2) being the limiting distribution. We denote the time-varying transition matrix of the nonhomogeneous perturbed chain by the bold-font $\mathbf{P}$, i.e. $\mathbf{P}(t) = P(\epsilon_t)$.

Define

$$\kappa = \min_{E \in \{E_i\}} CR(E). \tag{4}$$

Theorem 3 ([2], Theorem 3 or [15], Theorem 5): Let the recurrence classes of the unperturbed chain $P(0)$ be aperiodic and the parameter ϵ in the perturbed chain be scheduled according to the monotone decreasing sequence $\{\epsilon_t\}_{t\in\mathbb{N}}$, with $\epsilon_t \to 0$ as $t \to \infty$, as described above. Then, a sufficient condition for weak ergodicity of the resulting nonhomogeneous Markov chain $\mathbf{P}(t)$ is

$$\sum_{t\in\mathbb{N}} \epsilon_t^{\kappa} = \infty.$$

Furthermore, if the chain is weakly ergodic and Assumption 2 holds, then it is strongly ergodic with the limiting distribution being $\mu(0)$ as described in Theorem 2.

IV. Analysis of the Algorithm

The objective of this section is to prove Theorem 1. The proof relies on some Lemmas that are stated in the following. The proofs for the Lemmas are excluded for want of space and may be found in [15]. We will first consider the algorithm of section II-B with the parameter ϵ_t held constant at $\epsilon > 0$. The algorithm then describes a Markov chain on the finite state space $S = \mathcal{A} \times \{0,1\}^N$ and we denote its 1-step transition matrix as $P(\epsilon)$. The reason for choosing the same notation here as for the general perturbed Markov chain discussed in section III is that we wish to view the Markov chain induced by the algorithm as a perturbed chain and analyze it using results from section III. Similarly, $\mathbf{P}(t)$ denotes the 1-step transition probability matrix for the duration $(t, t+1)$ of the nonhomogeneous Markov chain induced by the algorithm as described in section III, i.e. with time varying ϵ_t. Henceforth, the components of any $x \in S$ will be identified with a superscript i.e. $x = [a^x, m^x]$.

Lemma 4.1 ([15], Lemma 4.1): The Markov chain $P(\epsilon)$ is irreducible and aperiodic.

Lemma 4.1 implies that $P(\epsilon)$ has a unique stationary distribution which we denote, as in the previous section, by $\mu(\epsilon)$. It is also clear that $P(\epsilon)$ is a regular perturbation of $P(0)$ (the latter obtained by setting $\epsilon_t \equiv 0$ in the algorithm). Thus, by Theorem 2, $\mu(\epsilon) \to \mu(0)$ as $\epsilon \to 0$ where $\mu(0)$ is a stationary distribution of $P(0)$.

A. Stochastically Stable States: Support of $\mu(0)$

Definition 4.1: Let

$$C^0 = \{x \in S | m^x = \mathbf{1}\} \text{ and}$$
$$D^0 = \{x \in S | m^x = \mathbf{0}\}.$$

Lemma 4.2 ([15], Lemma 4.2): If for every $a \in \mathcal{A}$, $\mathcal{G}_c(a) \cup \mathcal{G}_I(a)$ is strongly connected, the recurrence classes of the unperturbed chain $P(0)$ are D^0 and the singletons $z \in C^0$.

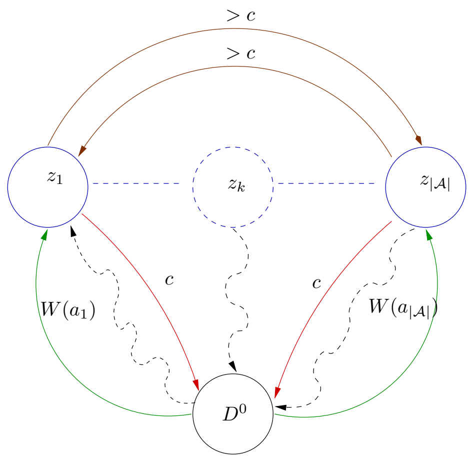

Fig. 2. The circles represent recurrence classes of $P(0)$ and weights on the arrows the corresponding $\rho(\cdot,\cdot)$s. If $W(a^{z_1}) = W^*$, the zig-zag lines represent edges in the minimum resistance tree rooted at z_1.

Guided by Theorem 2, we now proceed to calculate the stochastic potential of the recurrence classes of $P(0)$. But first some calculations are organized in the following Lemma.

Lemma 4.3 ([15], Lemma 4.3): Under the same assumption as Lemma 4.2, for any $y \in D^0$ and $z \in C^0$,

$$\rho(x,y) = c, \ \forall\, x \in C^0, \tag{5}$$
$$\rho(y,z) = W(a^z), \tag{6}$$
$$\rho(x,z) \le W(a^z), \ \forall\, x \in S \setminus C^0, \tag{7}$$
$$\text{and } \rho(z',z) > c, \ \forall\, z' \in C^0, \ z' \ne z. \tag{8}$$

From Lemma 4.2, there are exactly $|\mathcal{A}|+1$ recurrence classes of $P(0)$ - $|\mathcal{A}|$ corresponding to each $a \in \mathcal{A}$ (i.e. each element of C^0) and one for the set D^0. Let $\{z_1, ..., z_{|\mathcal{A}|}\}$ be an enumeration for C^0; together with the calculations in Lemma 4.3, Figure 2 emerges as a picture for $\mathcal{G}_{RC}$ of the algorithm. The following Lemma can be derived on the basis of Figure 2.

Lemma 4.4 ([15], Lemma 4.4): Under the same assumption as Lemma 4.2, the stochastically stable set is $\{z_i \in C^0 | W(a^{z_i}) = W^*\}$.

B. Proof of Theorem 1

We return to the analysis of the nonhomogeneous Markov chain, $\mathbf{P}$, induced by the algorithm with the annealing schedule $\{\epsilon_t\}_{t\in\mathbb{N}}$. The proof relies on noting that if the annealing schedule satisfies $\sum_{t=1}^{\infty} {\epsilon_t}^c = \infty$, $\mathbf{P}$ is strongly ergodic with the limiting distribution having support over states with efficient actions as described by Lemma 4.4.

Lemma 4.5 ([15], Lemma 4.5): Under the same assumption as Lemma 4.2, for the nonhomogeneous Markov chain defined on S by the algorithm, κ as defined in (4) equals c.

Proof: [Proof of Theorem 1] The Assumption of Lemma 4.2 is included in the statement of the Theorem. All transition probabilities in the algorithm of section II-B belong to $\mathfrak{L}$; thus Assumption 2 holds. For any $y \in D^0$ and $z \in C^0$, $P_{y,y}(0) > 0$ and $P_{z,z}(0) > 0$. Hence the recurrence classes of the unperturbed Markov chain are aperiodic and, from Theorem 3 and Lemma 4.5, the chain is strongly ergodic if

$$\sum_{t=1}^{\infty} {\epsilon_t}^{c} \;=\; \infty. \tag{9}$$

Next, for any initial distribution η_0 on S and any subset $\tilde{S} \subset S$, $\mathbb{P}(\mathbf{X}_t \in \tilde{S}) = \sum_{j\in\tilde{S}} (\eta_0 \mathbf{P}^{(1,t)})_j$. Since (9) implies strong ergodicity with limiting distribution $\mu(0)$ as in Theorem 2, $\lim_{t\to\infty} \mathbb{P}(\mathbf{X}_t \in \tilde{S}) = \sum_{j\in\tilde{S}} \mu_j(0)$. Let $\tilde{S} = \{x \in S | W(a^x) = W^*, m^x = \mathbf{1}\}$. Then in view of Lemma 4.4,

$$\lim_{t\to\infty} \mathbb{P}[\mathbf{a}_t \in \mathcal{A}^*] = 1.$$

$\blacksquare$

V. Numerical Simulations and Conclusions

An interesting question is how the performance of the algorithm depends on $\mathcal{G}_c$. We present results of some simple numerical experiments to motivate such questions. Consider N identical agents with $\mathcal{A}_i = \{0.1, 1\}$ and $u_i(a) = a_{i-1}$ for $i = 2, ..., N$ and $u_1(a) = a_N$. Thus $\mathcal{G}_I(a)$ is a directed ring for all $a \in \mathcal{A}$ (see Figure 3 (a)) and the welfare function $W(a) = \sum_{i=1}^{N} a_i$, has a unique minimum at $(0.1, ..., 0.1)$. Let directed edges $(i, i-q)$ (where subtraction is mod N) for all i constitute $\mathcal{G}_c^q(a)$ for all $a \in \mathcal{A}$ and $\mathcal{G}(q) = \mathcal{G}_c^q \cup \mathcal{G}_I$. The algorithm is implemented in MATLAB for the case $N = 10$, $c = 1.1$, $\beta_1 = \beta_2 = 0.5$, $\epsilon_t = \frac{1}{\sqrt[c]{t}}$ and is allowed to run while $\epsilon_t > 10^{-4}$. The experiment is carried out for different values of q and performance is measured as the percentage of times the welfare minimal action is picked averaged over 100 runs for each value of q. The performance measure, the length of the longest shortest-path (SP) in $\mathcal{G}(q)$ and length of a cycle in $\mathcal{G}(q)$ are plotted for values of q in $\{0, .., (N-1)\}$ in Figure 3 (b). The results suggest a heuristic: To improve performance, pick $\mathcal{G}_c(a)$ to comprise of edges exactly opposite of $\mathcal{G}_I(a)$ and thereby reducing the cycle lengths.

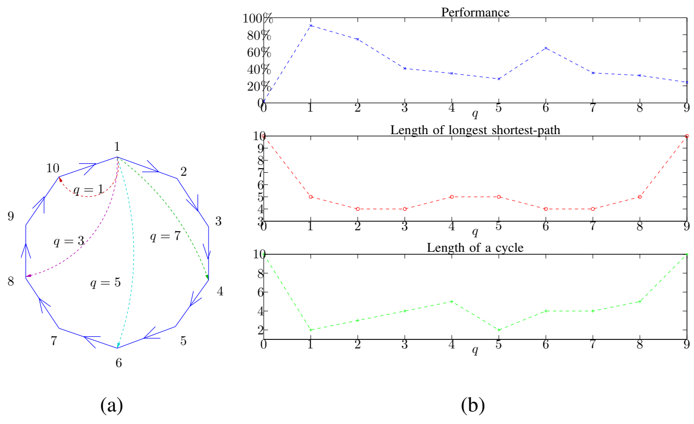

Fig. 3. Effects of varying $\mathcal{G}_c$ for $N = 10$. (a) The blue solid arrows represent $\mathcal{G}_I$ and the dotted arrows denote the edge $(1, 1-q)$ in the corresponding $\mathcal{G}_c^q$. (b) Plot of performance, longest SP and cycle length w.r.t. different values of q.

An important open question is determining the rate of convergence of the algorithm. One way to answer this question is to calculate the rate of convergence of $||\eta_t - \mu(0)||$ as $t \to \infty$, where η_t is the density of $\mathbf{X}_t = [\mathbf{a}_t, \mathbf{m}_t]$. This is difficult since the Markov chain is nonhomogeneous and the best results we know in such situations are for the simulated annealing algorithm [16]. We will address this issue along such lines in future work. We expect that such an investigation will also shed light on the issue of how the communication and interaction graphs play a role in determining the speed of convergence.

References

[1] J. R. Marden, H. P. Young, and L. Y. Pao, "Achieving Pareto optimality through distributed learning," in *Proc. of 51st Annual Conference on Decision and Control (CDC)*, pp. 7419–7424, IEEE, 2012. Available online at http://ecee.colorado.edu/marden/files/Pareto%20Optimality.pdf.

[2] A. Menon and J. S. Baras, "Convergence guarantees for a decentralized algorithm achieving Pareto optimality," in *Proc. of the 2013 American Control Conference (ACC)*, pp. 1932–1937, 2013.

[3] R. Gopalakrishnan, J. R. Marden, and A. Wierman, "An architectural view of game theoretic control," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 3, pp. 31–36, 2011.

[4] N. Li and J. R. Marden, "Designing games for distributed optimization," in *Proc. of 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC), 2011*, pp. 2434–2440, IEEE, 2011.

[5] M. Zhu and S. Martnez, "Distributed coverage games for energy-aware mobile sensor networks," *SIAM Journal on Control and Optimization*, vol. 51, no. 1, pp. 1–27, 2013.

[6] E. Altman and Z. Altman, "S-modular games and power control in wireless networks," *IEEE Transactions on Automatic Control*, vol. 48, no. 5, pp. 839–842, 2003.

[7] J. R. Marden and J. S. Shamma, "Revisiting log-linear learning: Asynchrony, completeness and payoff-based implementation," *Games and Economic Behavior*, vol. 75, no. 2, pp. 788–808, 2012.

[8] J. R. Marden, H. P. Young, G. Arslan, and J. S. Shamma, "Payoff based dynamics for multi-player weakly acyclic games," *SIAM Journal on Control and Optimization*, vol. 48, pp. 373–396, Feb 2009.

[9] H. P. Young, "Learning by trial and error," *Games and Economic Behavior*, vol. 65, no. 2, pp. 626–643, 2009.

[10] J. Marden, S. Ruben, and L. Pao, "A model-free approach to wind farm control using game theoretic methods," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1207–1214, 2013.

[11] N. Ghods, P. Frihauf, and M. Krstic, "Multi-agent deployment in the plane using stochastic extremum seeking," in *Proc. of 49th IEEE Conference on Decision and Control (CDC), 2010*, pp. 5505–5510, IEEE, 2010.

[12] X. Tan, W. Xi, and J. S. Baras, "Decentralized coordination of autonomous swarms using parallel Gibbs sampling," *Automatica*, vol. 46, no. 12, pp. 2068–2076, 2010.

[13] W. Xi, X. Tan, and J. S. Baras, "Gibbs sampler-based coordination of autonomous swarms," *Automatica*, vol. 42, no. 7, pp. 1107–1119, 2006.

[14] H. P. Young, "The evolution of conventions," *Econometrica: Journal of the Econometric Society*, pp. 57–84, 1993.

[15] A. Menon and J. S. Baras, "A distributed learning algorithm with bit-valued communications for multi-agent welfare optimization." Institute of Systems Research Technical Report. Available online at http://drum.lib.umd.edu/advanced-search, 2013.

[16] D. Mitra, F. Romeo, and A. Sangiovanni-Vincentelli, "Convergence and finite-time behavior of simulated annealing," *Advances in Applied Probability*, pp. 747–771, 1986.