53rd IEEE Conference on Decision and Control
December 15-17, 2014. Los Angeles, California, USA

Collaborative Extremum Seeking for Welfare Optimization

Anup Menon and John S. Baras

Abstract— **This paper addresses a distributed, model-free optimization problem in the context of multi-agent systems. The set-up comprises of a fixed number of agents, each of which can pick an action and receive/measure a private utility function that can depend on the collective actions taken by all agents. The exact functional form (or model) of the agent utility functions is unknown, and an agent can only measure the numeric value of its utility. The objective of the multi-agent system is to optimize the welfare function (i.e. sum of the individual utility functions). A model-free, distributed, on-line learning algorithm is developed that achieves this objective. The proposed solution requires information exchange between the agents over an undirected, connected communication graph, and is based on ideas from extremum seeking control. A result on local convergence of the proposed algorithm to an arbitrarily small neighborhood of a local minimizer of the welfare function is proved. Application of the solution to distributed control of wind turbines for maximizing wind farm-level power capture is explored via numerical simulations. Also included is a novel analysis of a dynamic average consensus algorithm that may be of independent interest.**

I. Introduction

Engineered multi-agent systems (MAS) comprise of multiple decision making entities (or agents) that have a collective task to accomplish. Each agent makes its individual decisions based on local information (such as its measurements and communicated information from other agents), and the challenge is to devise distributed decision making rules (or algorithms) that help realize the system-wide objective. While there has been extensive research in several formulations of these problems, emerging applications continue to provide new challenges.

We consider one such problem formulation in this paper with a salient feature of being model-free. In particular, consider a MAS comprising of n agents indexed by i; agent i takes action $u_i \in \mathbb{R}$, and receives/measures a private utility function $f_i(u)$ that can depend on the actions taken by all agents $u = (u_1, ..., u_n)$. The exact functional form of the utility functions $f_i(\cdot)$s, however, is unknown. An agent merely receives or measures the realized values of its respective utility function; for instance, if the collective action at time t is $u(t)$, agent i receives the corresponding realized numerical value $f_i(u(t))$. The MAS has the collaborative objective to optimize (without loss of generality, we consider minimization) the welfare function $W(u) = \sum_{i=1}^{n} f_i(u)$:

$$\min_{u\in\mathbb{R}^n} W(u). \tag{P}$$

Inter-agent communications can be realized in the context of engineered MAS, and are necessary to achieve such collaborative tasks. Hence, we shall allow for the possibility of such communications. Thus, distributed learning algorithms are sought that help each agent *learn* its (respective) action that corresponds to the optimizer of the welfare function.

Example application scenarios that can be modeled in this fashion include efficient coverage of source-fields by a group of robots [1], collaborative surveillance by a group of robots [2], etc. The specific application that has inspired our work is the problem of maximizing the total power production of wind farms. Due to aerodynamic interactions between wind turbines, each turbine maximizing its power production does not lead to farm-wide maximal power capture, and hence coordinated control of turbines can help improve power capture [3]–[7]. This problem is challenging because there are no accurate models for the said aerodynamic interactions, rendering model-based optimization inaccurate. Hence, model-free learning schemes, where turbines adapt their set-points on-line in response to measured value of their power production are needed.

While it is clear that (P) is distinct from the vast literature on model-based distributed optimization, the literature on Learning in Repeated Games is relevant to it. This area of research studies the emergent behavior resulting from players playing a game repeatedly while following specific (action-update) strategies, and has received a lot of interest in the context of distributed control recently (see [8] and references therein). While many of these algorithms have the desired feature of being model-free and payoff-based, being in the context of non-cooperative games, most focus on equilibration to Nash (and other related) equilibiria under appropriate assumptions on the payoff functions [9], [10]. However, such assumptions on payoff structure may be too restrictive and the equilibria may be inefficient in the present context. To the best of our knowledge, only [11] and the authors' earlier work [12], [13] address (P) using ideas from learning in games. However, these earlier works [11]–[13] assume the set of values an agent's action can take to be discrete; such discretization leads to loss of useful information (such as gradients) available in the present setting that can be exploited for improved performance (especially improved speed of convergence).

The main contribution of this paper is a solution to (P) using ideas from extremum seeking control, and a rigorous

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

justification that the proposed solution ensures local convergence of the agent actions $u(t)$ to an arbitrarily small neighborhood of a minimizer of $W(\cdot)$ as $t \to \infty$ (see Theorem 2). Extremum seeking control is a model-free adaptive control technique that performs a "simultaneous gradient estimation-descent". Loosely speaking, our solution is based on the agents running a dynamic average consensus algorithm with their payoffs as inputs so that the consensus output "tracks" $W(u)$. Next, this arrangement is interfaced suitably with an extremum seeking loop (see Figure 1) that leads agent i's action to evolve according to

$$\dot{u}_i \approx -\frac{\partial W(u)}{\partial u_i}.$$

In this sense, this work can be considered as an application of extremum seeking control and dynamic consensus for solving (P). Noteworthy in this context are recent works [14], [15] which apply extremum seeking control to learn Nash equilibria in non-cooperative $n-$player games.

Another important contribution is the demonstration of the proposed solution on the wind farm power maximization problem via numerical simulations. While distributed data-driven approaches have been suggested for this problem (see [5], [7]), they either suffer from slow convergence or do not provide formal guarantees of convergence. In contrast we observe promising convergence speeds in our simulations. Yet another contribution is a novel analysis of a dynamic consensus algorithm which, while necessary for subsequent discussions, we believe is an elegant treatment of the topic and is of independent interest.

The remainder of the paper is organized as follows. Section II discusses the dynamic average consensus algorithm. The proposed solution to (P) and related results are discussed in Section III. The numerical simulations on the wind farm problem are presented in Section IV, followed by concluding remarks in Section V.

Notation

Given a matrix A,

1) its i^{th} row, j^{th} column element is denoted by $A[i,j]$.
2) the expression $A > (<) 0$ denotes that it is a symmetric positive(negative) definite matrix.
3) if it is symmetric, its largest and smallest eigenvalues are denoted by $\lambda_{max}(A)$ and $\lambda_{min}(A)$, respectively.
4) its rank is denoted by $rank(A)$.

The column vector $[1, ..., 1]^T$ is denoted by the bold-font $\mathbf{1}$. The i^{th} component of a vector v is denoted by v_i.

II. Dynamic Average Consensus Revisited

Consider the MAS discussed in Section I. Agent i has access to a reference signal r_i, and in this section we devise a dynamic average consensus algorithm that helps every agent track the average of the references $\frac{1}{n}\sum_{i=1}^{n} r_i$. This algorithm has appeared earlier in literature as 'Proportional-Integral Dynamic Consensus' in [16]; however, our analysis is quite different from this earlier work and is inspired by [17] where distributed optimization is treated along similar lines.

We begin by stating the communication requirements between the agents. It is assumed that the agents exchange information over a connected, undirected communication graph $\mathcal{G}_c = (V, E)$, where $V = \{1, ..., n\}$, and $(i,j) \in E (\Leftrightarrow (j,i) \in E)$ means that agents i and j can exchange information. An adjacency matrix A of $\mathcal{G}_c$ satisfies $(i,j) \notin E \Rightarrow A[i,j] = 0$; and the corresponding Laplacian matrix L is given by $L = diag(A\mathbf{1}) - A$[1]. It is clear that $L\mathbf{1} = 0$; however, when $rank(L) = (n-1)$, then $Lx = 0 \Leftrightarrow x = \alpha\mathbf{1}$ for some $\alpha \in \mathbb{R}$. A rank $(n-1)$ Laplacian can always be constructed for a connected graph by picking elements of the adjacency matrix such that $(i,j) \in E \Rightarrow A[i,j] > 0$ (this follows from the Perron-Frobenious Theorem [18]).

Now, consider the following optimization problem for the scalar variable $\hat{x}$:

$$\min_{\hat{x}} \frac{1}{2}\sum_{i=1}^{n}(\hat{x} - r_i)^2. \tag{P0}$$

Elementary calculations yield the optimizer to be $\frac{1}{n}\sum_{i=1}^{n} r_i$. Next, consider the following equivalent restatements of (P0):

$$\min \frac{1}{2}\sum_{i=1}^{n}(x_i - r_i)^2 \text{ subject to } x_i = x_j,\ \forall\, i, j;$$
$$\min \frac{1}{2}x^T x - r^T x + \frac{1}{2}r^T r \text{ subject to } L_I x = 0, \tag{P1}$$

where L_I is a Laplacian matrix of $\mathcal{G}_c$ with $rank(L_I) = (n-1)$ (as noted above, this essentially enforces $x_i = x_j,\ \forall\, i, j$). Finally, consider the following equivalent restatement of (P1):

$$\min \frac{1}{2}x^T x - r^T x + \frac{\rho}{2}x^T L_P x \text{ subject to } L_I x = 0, \tag{P2}$$

where L_P is a Laplacian matrix of $\mathcal{G}_c$. The last equivalence follows by noting that the constraint $L_I x = 0 \Rightarrow x = \alpha\mathbf{1} \Rightarrow x^T L_P x = 0$, so that (P2) is exactly the same as (P1). Addition of $x^T L_P x$ to the objective (with an appropriate choice of L_P) amounts to adding a penalty term for constraint violation that can help improve the speed of numerical algorithms for finding the solution.

Denote the Lagrangian corresponding to (P2) by $\mathcal{L}(x,\lambda) = \frac{1}{2}x^T(I + \rho L_P)x - r^T x + \lambda^T L_I x$, where λ is the dual variable. It is well known fact from optimization theory that the solution x^* to (P2) corresponds to a saddle point (x^*, λ^*) of $\mathcal{L}(x,\lambda)$,

$$\mathcal{L}(x^*, \lambda) \le \mathcal{L}(x^*, \lambda^*) \le \mathcal{L}(x, \lambda^*).$$

Thus, consider the saddle-seeking system

$$\begin{aligned}\begin{bmatrix}\dot{x}\\ \dot{\lambda}\end{bmatrix} &= \begin{bmatrix}-\nabla_x \mathcal{L}(x,\lambda)\\ \nabla_\lambda \mathcal{L}(x,\lambda)\end{bmatrix}\\ &= \begin{bmatrix}-I - \rho L_P & -L_I^T\\ L_I & 0\end{bmatrix}\begin{bmatrix}x\\ \lambda\end{bmatrix} + \begin{bmatrix}I\\ 0\end{bmatrix} r. \end{aligned}\tag{1}$$

While r is a fixed vector in the context of the optimization problems (P0), (P1) and (P2), in what follows we will also

[1]For a vector v, $diag(v)$ denotes the diagonal matrix with $diag(v)[i,i] = v(i)$.

treat the system in (1) as an LTI system driven by the input r. Notice that the update rules for $\dot{x}_i$ and $\dot{\lambda}_i$ in (1), depend only on the values of x_j, λ_j and $L_I[j,i]$ such that $(i,j) \in E$ (apart from the i^{th} rows of L_I and L_P, and r_i which agent i already knows). Thus, (1) can be implemented in a distributed manner by means of exchange of appropriate variables and data over $\mathcal{G}_c$. So far the introduction of L_I and L_P only made sense due to their algebraic properties. The reason for requiring them to be Laplacians of $\mathcal{G}_c$ is to obtain such distributed implementation of (1).

By now the reader would have guessed that (1) is the proposed dynamic consensus algorithm, with the individual x_is expected to converge to $\frac{1}{n}\sum_{i=1}^{n} r_i$ (the solution of (P0)), for fixed values of r_is. The remainder of this section deals with formalizing this statement. Let μ be such that $\mu^T L_I = 0$, and let $U = [\tilde{U}\ \mu]$ be an orthonormal matrix. Replace λ in (1) with $\lambda = U \begin{bmatrix} \tilde{\lambda} \\ \lambda_\mu \end{bmatrix}$, where $\tilde{\lambda} \in \mathbb{R}^{n-1}$ and $\lambda_\mu \in \mathbb{R}$, to obtain

$$\begin{bmatrix} \dot{x} \\ \dot{\tilde{\lambda}} \\ \dot{\lambda}_\mu \end{bmatrix} = \left[\begin{array}{cc|c} -I-\rho L_P & -L_I^T \tilde{U} & 0 \\ \tilde{U}^T L_I & 0 & 0 \\ \hline 0 & 0 & 0 \end{array}\right] \begin{bmatrix} x \\ \tilde{\lambda} \\ \lambda_\mu \end{bmatrix} + \left[\begin{array}{c} I \\ 0 \\ \hline 0 \end{array}\right] r. \tag{2}$$

Since λ_μ does not interact with the rest of the state and inputs (and vise versa) in (2), one can eliminate it to obtain the reduced-order system

$$\begin{bmatrix} \dot{x} \\ \dot{\tilde{\lambda}} \end{bmatrix} = \begin{bmatrix} -I-\rho L_P & -L_I^T \tilde{U} \\ \tilde{U}^T L_I & 0 \end{bmatrix} \begin{bmatrix} x \\ \tilde{\lambda} \end{bmatrix} + \begin{bmatrix} I \\ 0 \end{bmatrix} r. \tag{3}$$

The following lemma is easily verified.

Lemma 2.1: Let $y = x$ be the output of (1), and $y_{ro} = x$ be that of (3). Let the same input $r : [0,T] \to \mathbb{R}^n$ be applied to both systems (1) and (3), with the same initial conditions for the variable x, and

- for a specified initial condition $\lambda(0) \in \mathbb{R}^n$, pick $\tilde{\lambda}(0) = \tilde{U}^T \lambda(0)$, or
- for a specified initial condition $\tilde{\lambda}(0) \in \mathbb{R}^{n-1}$, pick $\lambda(0) = \tilde{U}\tilde{\lambda}(0)$.

Then

$$y(t) = y_{ro}(t),\ \forall\, t \in [0,T].$$

The reason for introducing the reduced order system (3) is twofold. First, the properties of the equilibrium point of (3) are helpful in proving Theorem 1. Second, it turns out that the global asymptotic stability of the LTI system (3) (as opposed mere stability of (1)) will be helpful in the analysis of the next section. We will now prove the main convergence result for the dynamic average consensus algorithm (1). We will need the following technical lemma.

Lemma 2.2: Let $Q \in \mathbb{R}^{n\times n}$ be such that $Q^T + Q > 0$, and $R \in \mathbb{R}^{n\times m}$ $(m < n)$ be full rank. Then the matrix

$$M = \begin{bmatrix} -Q & -R \\ R^T & 0 \end{bmatrix}$$

is Hurwitz.

(The proof is based on constructing an appropriate Lyapunov function and is omitted due to space limitations.)

Theorem 1: Let $\mathcal{G}_c$ be a connected, undirected graph, L_I and L_P be Laplacians of $\mathcal{G}_c$ such that $rank(L_I) = (n-1)$, and ρ be such that $\frac{1}{2}\rho\lambda_{min}(L_P + L_P^T) < 1$. Then, for any $w \in \mathbb{R}^n$ and $r(t) = w\ \forall\, t \geq 0$,

1) the LTI system (3) has a unique globally exponentially stable equilibrium point $(x^{\mathrm{eq}}(w), \tilde{\lambda}^{\mathrm{eq}}(w))$, where $x_w = \frac{1}{n}(\mathbf{1}^T w)\mathbf{1}$, and
2) the state of the LTI system (1), with arbitrary initial conditions $x(0), \lambda(0) \in \mathbb{R}^n$, remains bounded and $x(t)$ converges exponentially to $\frac{1}{n}(\mathbf{1}^T w)\mathbf{1}$ as $t \to \infty$.

Proof: Notice that if $\frac{1}{2}\rho\lambda_{min}(L_P + L_P^T) < 1$, then $2I + \rho(L_P + L_P^T) > 0$. Since $rank(L_I) = (n-1)$ and U is orthonormal, $U^T L_I = \begin{bmatrix} \tilde{U}^T L_I \\ 0 \end{bmatrix} \Rightarrow L_I^T \tilde{U} \in \mathbb{R}^{n\times n-1}$ is full rank. Thus, by Lemma 2.2, the matrix

$$A_{ro} = \begin{bmatrix} -I-\rho L_P & -L_I^T \tilde{U} \\ \tilde{U}^T L_I & 0 \end{bmatrix}$$

is Hurwitz.

Next, notice that (P2) is a convex optimization problem (in particular, a quadratic program). Since $L_I x = 0 \Leftrightarrow \tilde{U}^T L x = 0$, the equality constraint in (P2) can be replaced with $\tilde{U}^T L x = 0$. The corresponding Lagrangian, with $\tilde{\lambda}$ as the Lagrange multiplier, is given by $\mathcal{L}(x,\tilde{\lambda}) = \frac{1}{2}x^T(I + \rho L_P)x - r^T x + \tilde{\lambda}^T \tilde{U}^T L_I x$. With $r = w$, the KKT conditions for this convex problem state that $x^{\mathrm{eq}}(w)$ is optimal *if and only if* $\exists\ \tilde{\lambda}^{\mathrm{eq}}(w)$ such that

$$\begin{bmatrix} -I-\rho L_P & -L_I^T \tilde{U} \\ \tilde{U}^T L_I & 0 \end{bmatrix} \begin{bmatrix} x^{\mathrm{eq}}(w) \\ \tilde{\lambda}^{\mathrm{eq}}(w) \end{bmatrix} + \begin{bmatrix} w \\ 0 \end{bmatrix} = 0. \tag{4}$$

Since A_{ro} is Hurwitz, there exists a unique solution $(x^{\mathrm{eq}}(w), \tilde{\lambda}^{\mathrm{eq}}(w))$ to (4). Further, subtracting (4) from (3) (with $r(t)$ set to w), we note the error variables $\begin{bmatrix} x - x^{\mathrm{eq}}(w) \\ \tilde{\lambda} - \lambda^{\mathrm{eq}}(w) \end{bmatrix}$ converge to zero exponentially since A_{ro} is Hurwitz. Finally, since (P2) is a restatement of (P1), which in turn is a restatement of (P0), we have $x^{\mathrm{eq}}(w) = \frac{1}{n}(\mathbf{1}^T w)\mathbf{1}$. This concludes the proof for the first claim.

The second claim regarding boundedness of the state of (1) follows by noting that by performing a similarity transformation of the state in (1), a mode with zero eigenvalue (λ_μ) has been isolated in (2) and shown to be unaffected by the input, while the rest of the state is Input-to-State-Stable as A_{ro} is Hurwitz. The exponential convergence $x(t) \to \frac{1}{n}(\mathbf{1}^T w)\mathbf{1}$ is due to Lemma 2.1 and the first claim of this theorem. ■

Hence, we have argued the correctness of the dynamic consensus algorithm (1) by associating it to an optimization problem and avoided the tedious calculations needed in the proof of [16, Theorem 5].

III. Collaborative Extremum Seeking

In this section we discuss in detail the proposed extremum seeking based solution to (P). We assume the agents can exchange information over a communication graph $\mathcal{G}_c$ as described in Section II. Consider the schematic representation shown in Figure 1. Agent i applies its input u_i and receives the corresponding payoff or utility value $f_i(u)$. The agent uses this observed/measured value as the reference command r_i in the dynamic consensus algorithm. Finally, the 'consensus output' x_i is used as the feedback to the extremum seeking block which, in turn, generates u_i, thus closing the loop. Observe that variables in agent i's 'loop' (highlighted in Figure 1) comprise of only variables measured, controlled or communicated to it over $\mathcal{G}_c$, leading to the required distributed implementation. Within the extremum seeking blocks, agents use their respective sinusoidal 'dither' signal $\nu_i(t) = \sin(\omega_i t + \phi)$ and certain parameters which satisfy

$$\begin{aligned} \omega_i &= \omega_c \overline{\omega}_i, \\ a_i &= a_c \overline{a}_i, \end{aligned} \tag{5}$$

where, for all i, ω_c, a_c and $a_i \in \mathbb{R}$ are positive, and ω_i is a positive rational number. The constant $k_i = \omega_c K_i$, for some $K_i > 0$.

The system depicted in Figure (1) is given by[2]

$$\begin{aligned} \begin{bmatrix} \dot{x} \\ \dot{\lambda} \end{bmatrix} &= \begin{bmatrix} -I - \rho L_P & -L_I^T \\ L_I & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} + \begin{bmatrix} f(u) \\ 0 \end{bmatrix}, \\ \dot{\hat{u}}_i &= -\epsilon k_i \nu_i x_i, \; \forall\, i, \\ u &= \hat{u} + a\nu(t); \end{aligned} \tag{6}$$

where $f(\cdot) = [f_1(\cdot), ..., f_n(\cdot)]^T$ and, by slight abuse of notation, $a\nu(t) = [a_1\nu_1(t), ..., a_n\nu_n(t)]$.

Assumption 1: The function $f_i(\cdot)$ is smooth for each i, and there exists $u^* \in \mathbb{R}^n$ such that $\frac{\partial W(u^*)}{\partial u} = 0$ and $\frac{\partial^2 W(u^*)}{\partial u^2} > 0$.

This assumption ensures existence of a strict local minimizer for W. We are now ready to state the main result of this section which provides conditions that guarantee local convergence of the variable u to a neighborhood of u^*.

Theorem 2: Let the hypothesis of Theorem 1 and Assumption 1 hold. Let $\omega_i \neq \omega_j$, $2\omega_i \neq \omega_j$ and $\omega_i \neq \omega_j + \omega_k$, for all distinct $i, j, k \in \{1, ..., n\}$[3]. Then there exists $\omega_c^* > 0$ such that for any $\omega_c \in (0, \omega_c^*)$, there exist $\epsilon^* > 0$ and $a_c^* > 0$, such that for the chosen ω_c, and any $\epsilon \in (0, \epsilon^*)$ and $a_c \in (0, a_c^*)$, the solution to (6) with initial conditions $\hat{u}(0)$ sufficiently close to u^* (and $x(0), \lambda(0)$ arbitrary) is such that $u(t)$ converges exponentially to an $O(\omega_c + \epsilon + a_c)$ neighborhood of u^*.

The proof uses averaging and singular perturbation arguments that are standard in proving local convergence of extremum seeking schemes [19], and is hence omitted in view of space limitations. It is worth mentioning that the 'exponential stability' of the reduced order system (3) (proved in Section II) plays a key role in the analysis of (6) by enabling the use of relevant singular perturbation results (which require such exponential stability).

[2]We ignore the low-pass filters (LPF) and the high-pass filters (HPF) shown in Figure 1 since these are not needed for the convergence result of this section and including them makes the discussion cumbersome due to additional states. We will use these in the simulations of Section IV.

[3]For $n = 2$, drop conditions that refer to ω_k.

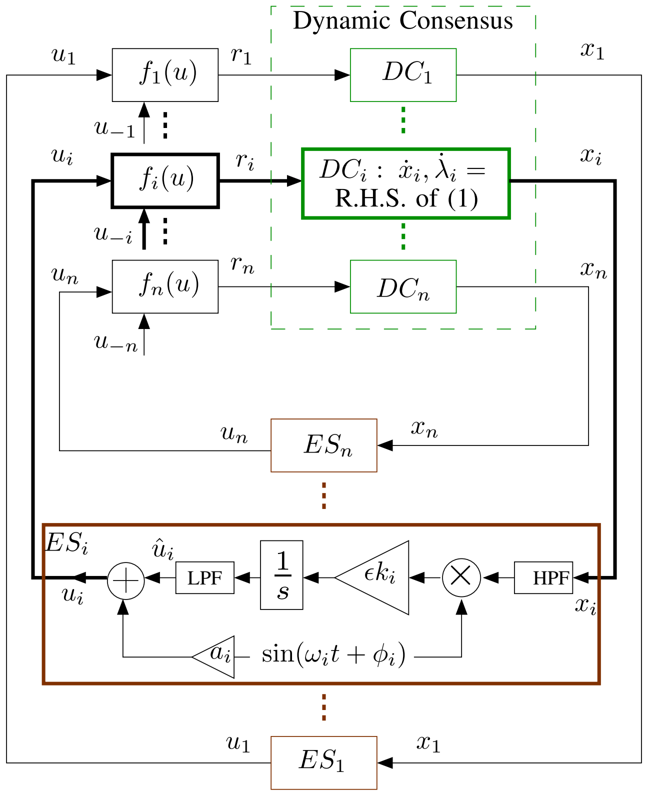

Fig. 1. A schematic representation of the proposed solution. DC_i refers to part of the dynamic consensus algorithm (1) implemented by agent i, ES_i refers to the extremum seeking law implemented by agent i, and u_{-i} refers to the elements of the vector u other than u_i.

IV. WIND FARM POWER MAXIMIZATION

In this section we explore the applicability of the proposed solution in Section III to the problem of maximizing the total power production of a wind farm via numerical simulations. The approach is to build a simulation model of a wind farm based on a model for computing wake affected wind speeds and a model for power production of turbines. Next, we run simulations of our model-free solution (6) on this simulation model and evaluate its performance (as though the model were the "ground truth"). Due to space limitations, we will only briefly describe the models used and will provide appropriate references for further details.

A. Wind Farm Model [7]

The action variable u_i of a turbine is its *axial induction factor* which takes values in $[0, 1/2]$. The power produced is given by the model

$$f_i(u) = \frac{1}{2}\rho_{\text{air}} A_i C_p(u_i) V_i(u)^3,$$

where $C_p(u_i)$ is the power efficiency coefficient and is modeled as $C_p(u_i) = u_i(1-u_i)^2$, $V_i(u)$ is the wind speed at turbine i in m/s, ρ_{air} is the density of air and is fixed at the constant value $1.225 kg/m^3$, and A_i is the area swept by the blades of turbine i in m^2. We have assumed all turbines in the farm to be identical with diameters equal to $77m$. It can be seen from the power capture expression that the influence of other turbine's actions on turbine i's power production (or utility) is due to the $V_i(u)$ term. This term captures the effect of aerodynamic interaction between turbines and we will use analytical models to describe it.

Consider a wind farm with a given layout and a given wind direction. Let us suitably enumerate the turbines so that the coordinates of the turbines $\{(x_1, y_1), ..., (x_n, y_n)\}$ are such that $x_j < x_i$ implies turbine i is downstream from turbine j. Then the wind speed at turbine i is given by

$$V_i(u) = V_\infty \left(1 - \sqrt{\sum_{j \in \{1,\ldots,n\}: x_j < x_i} (u_j C[j,i])^2} \right)$$

where V_∞ is the free stream wind speed in m/s, and the matrix $C \in \mathbb{R}^{n \times n}$ depends on the farm layout. The model used for computation of the C matrix is the popular low-fidelity wake model called the *Park Model* that can be found in references [3], [5]–[7]. In the simulation results reported in this section, we set $V_\infty = 10m/s$, $n = 3$, consider the three turbines located at coordinates $\{(0,0),(0,5D),(0,10D)\}$ (where D is the turbine diameter), and consider the wind blowing along the positive horizontal axis. Specifically, with regard to reference [7], the roughness coefficient 'k' used in computation of the C matrix is set to 0.04.

B. Simulation Results

A Simulink model is developed for implementing the proposed solution in Figure 1 with the functions $f_i(\cdot)$ modeled as described above. The parameters for the extremum seeking loop are chosen as $\omega = [1, 1.3\bar{3}, 1.90476]^T$, $\omega_c = 1$, $a_c = 5 \cdot 10^{-2}$, $a = [1,1,1]^T$, $\epsilon = 5 \cdot 10^{-1}$, and ϕ is chosen at random. It is typical for each turbine to be set to maximize its power production in present-day wind farm operations, and this corresponds to $u_i = \frac{1}{3}$. We shall think of this setting as the 'baseline' and compare the performance of our solution against it. Also, unless stated otherwise, reported simulations are run with initial conditions $\hat{u}(0) = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]^T$, and with $x(0), \lambda(0)$ set to zero. The value of gains K_i are set to the inverse of the baseline power of turbine i, $(f_i(\frac{1}{3}))^{-1}$. The filters depicted in Figure 1 are implemented using simple first order filters

$$\text{LPF: } \frac{1}{s + \omega_l}, \quad \text{HPF: } \frac{s}{s + \omega_h},$$

where $\omega_l = \min\{\omega\}$ and $\omega_h = 1.5 \cdot \max\{\omega\}$. Finally, we choose $L_p = 0$ and

$$L_I = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}.$$

Figure 2 shows time traces of relevant variables when simulations are run with the aforementioned parameters. It can be seen that the action variables $u(t)$ and total farm power $W(u(t))$ continue to oscillate around constant values. Such oscillations are to be expected in any extremum seeking scheme since the dither $a_i \nu(t)$ is injected additively in $u_i(t)$.

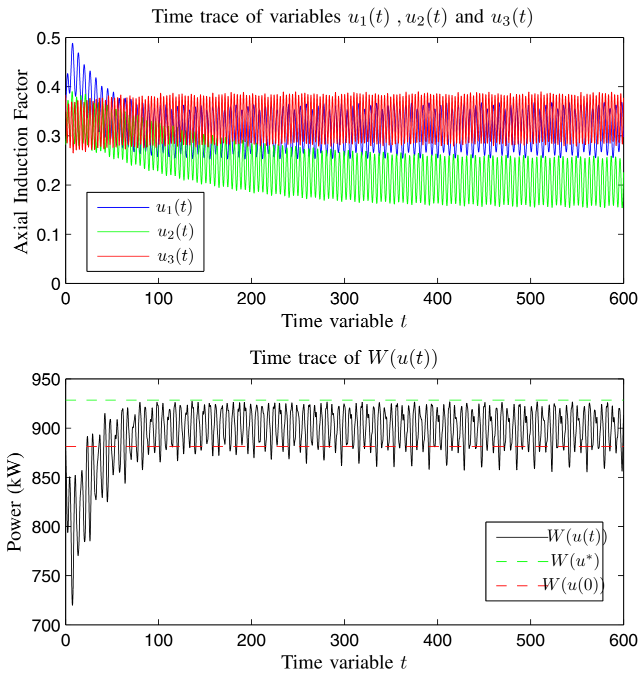

Fig. 2. A typical run of the proposed solution

The performance of the solution is better understood by plotting the "learning variable" $\hat{u}$. In Figure 3 we plot the evolution of $(\hat{u}_1, \hat{u}_2)$ starting with initial condition $\hat{u}(0) = [0,0,0]^T$ over contours of $W(\cdot)$ (the variable $\hat{u}_3$, excluded from the plot, converges to $\frac{1}{3}$; this is optimal (= u_3^*) for the most downstream turbine which only has to maximize its individual power regardless of what the other turbines do). The proposed solution appears to perform very well, and $(\hat{u}_1, \hat{u}_2)$ converge to a small neighborhood of the optimal values (u_1^*, u_2^*) (computed by using standard optimization software on the model).

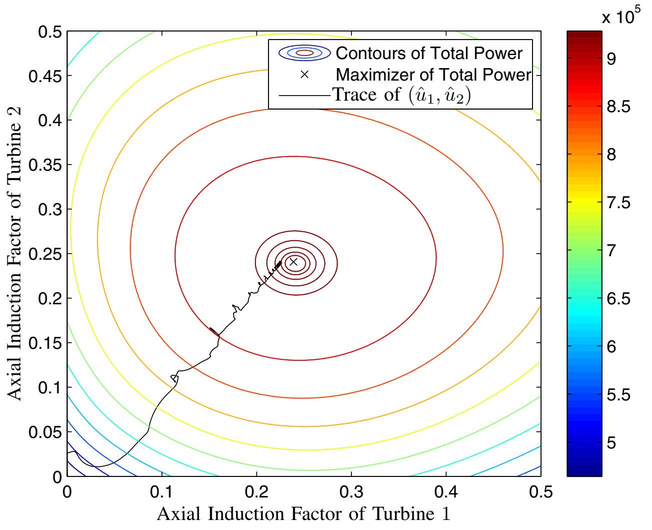

Fig. 3. Trace of $(\hat{u}_1, \hat{u}_2)$ plotted over contours of total power $W(\cdot)$ (Watts)

C. Consensus Time Scale vs. Learning Time Scale

The choice of the parameters made so far are somewhat arbitrary. Indeed, an accurate choice of the parameters in the dynamic consensus stage require additional hardware specific information. To put things in perspective, we repeat the simulation in Section IV-B for different values of time-scale separation between the learning dynamics and the consensus dynamics. This is done by multiplying the R.H.S. of the consensus part of dynamics in (6) by a positive constant α_{TS}. This effectively translates to the consensus dynamics (1) running at a time scale $\alpha_{\text{TS}} \cdot t$ when compared to the learning dynamics of $\hat{u}$ that run at time scale t. In Figure 4 we plot the performance of the algorithm with varying values of α_{TS} (all other parameters are kept fixed during these runs). Results show that so long as the consensus algorithm runs at least an order of magnitude faster than the learning dynamics, learning is successful. The degraded performance observed for small values of α_{TS} is expected since in the absence of information exchange, each turbine tries to maximize its power production leading to convergence to a neighborhood of the 'Nash equilibrium' $[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$; reminiscent of the Nash-seeking result in [14].

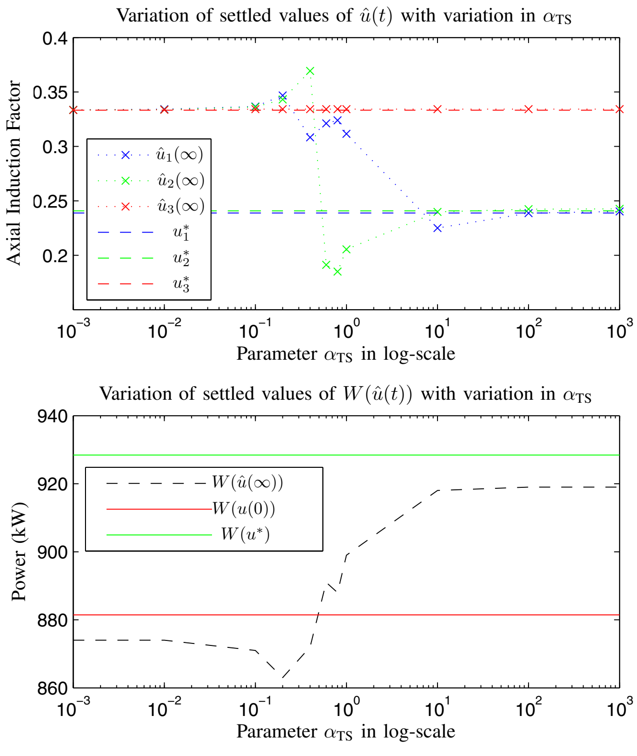

Fig. 4. Performance across varying learning and consensus time scales

V. Conclusion

An important data-driven distributed optimization problem is formulated in this paper. Besides the several applications that can be modeled in this framework, an underlying motivation for this work is to show that in engineered multi-agent systems (MAS), inter-agent communications and collaboration can be leveraged to achieve behaviors beyond Nash equilibria. The desired emergent property of the MAS is indeed application specific, and the present work adds the option of a welfare optimal outcome to the literature.

Extensions of the presented results to the case with non-linear dynamical agent models, and on semi-global practical stability of the proposed solution are forthcoming. An important next-step is to obtain a discrete-time counterpart to the proposed solution. Also, the delay between a change in a turbine's action and the measurement of its consequences by a downstream turbine are ignored in the reported simulation study (see [5], [6]). The proposed algorithm will be tested on higher fidelity wind farm models in future work.

References

[1] 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*, pp. 5505–5510, 2010.

[2] 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.

[3] K. E. Johnson and N. Thomas, "Wind farm control: addressing the aerodynamic interaction among wind turbines," in *Proc. of the American Control Conference, 2009*, pp. 2104–2109, 2009.

[4] E. Bitar and P. Seiler, "Coordinated control of a wind turbine array for power maximization," in *Proc. of the American Control Conference, 2013*, pp. 2898–2904, 2013.

[5] P. M. Gebraad, F. C. van Dam, and J.-W. van Wingerden, "A model-free distributed approach for wind plant control," in *Proc. of the American Control Conference, 2013*, pp. 628–633, 2013.

[6] K. E. Johnson and G. Fritsch, "Assessment of extremum seeking control for wind farm energy production," *Wind Engineering*, vol. 36, no. 6, pp. 701–716, 2012.

[7] 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.

[8] J. Marden and J. S. Shamma, "Game theory and distributed control," in *Handbook of Game Theory* (H. P. Young and S. Zamir, eds.), vol. 4, Elsevier. To appear; preprint available at http://www.prism.gatech.edu/ jshamma3/downloads/2099gta.pdf.

[9] H. P. Young, *Strategic learning and its limits*. Oxford University Press, 2004.

[10] D. Fudenberg, *The theory of learning in games*. MIT press, 1998.

[11] J. R. Marden, H. P. Young, and L. Y. Pao, "Achieving Pareto optimality through distributed learning," in *Proc. of 51st Annual IEEE Conference on Decision and Control*, pp. 7419–7424, 2012.

[12] 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.

[13] A. Menon and J. S. Baras, "A distributed learning algorithm with bit-valued communications for multi-agent welfare optimization," in *Proc. of the 52nd Annual IEEE Conference on Decision and Control*, pp. 2406–2411, 2013.

[14] P. Frihauf, M. Krstic, and T. Basar, "Nash equilibrium seeking in noncooperative games," *IEEE Transactions on Automatic Control*, vol. 57, no. 5, pp. 1192–1207, 2012.

[15] M. S. Stankovic, K. H. Johansson, and D. M. Stipanovic, "Distributed seeking of Nash equilibria with applications to mobile sensor networks," *IEEE Transactions on Automatic Control*, vol. 57, no. 4, pp. 904–919, 2012.

[16] R. Freeman, P. Yang, and K. Lynch, "Stability and convergence properties of dynamic average consensus estimators," in *45th IEEE Conference on Decision and Control, 2006*, pp. 338–343, Dec 2006.

[17] I. Matei and J. Baras, "A non-heuristic distributed algorithm for non-convex constrained optimization," tech. rep., Institute for Systems Research, 2013.

[18] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[19] M. Krstić and H.-H. Wang, "Stability of extremum seeking feedback for general nonlinear dynamic systems," *Automatica*, vol. 36, no. 4, pp. 595–601, 2000.