Proceedings of the 18th World Congress
The International Federation of Automatic Control
Milano (Italy) August 28 - September 2, 2011

A Distributed Real-Time Algorithm for Preference-Based Agreement

Daniel Zelazo [*,1] **Mathias Bürger** [*,1] **Frank Allgöwer** [*,1]

[*] *Institute for Systems Theory and Automatic Control, University of Stuttgart, Pfaffenwaldring 9, 70550 Stuttgart, Germany (e-mail: {daniel.zelazo, buerger, allgower}@ ist.uni-stuttgart.de).*

Abstract: This paper studies a real-time distributed dual sub-gradient algorithm for multi-agent coordination. A finite-time optimal control problem with a terminal coupling consensus constraint is considered. The algorithm is shown to be equivalent to a linear time-varying (LTV) dynamical system with a distributed structure. The error between the multiplier values of the algorithm and the corresponding centralized solution is also given as an LTV system. This is used to derive an error bound for the terminal state of the algorithm to the optimal consensus value that is a function of the communication graph and the weights of each agents objective function.

Keywords: Distributed optimization; Multi-agent systems; Finite-time consensus; Sub-gradient methods

1. INTRODUCTION

Distributed algorithms for large-scale optimization problems are becoming increasingly important for a broad range of applications. These algorithms are motivated by scenarios where access to global information is either unavailable or unattainable due to the constraints of the system. These include limited computational resources, communication bandwidth, and power restrictions.

In multi-agent systems, it is often the goal of a team of agents to achieve through cooperation and coordination a global objective. Due to the same constraints listed earlier, this objective must be reached using distributed protocols for the decision making of each agent. One well-studied problem related to this is the consensus, or agreement protocol [Mesbahi and Egerstedt (2010)]. In agreement problems, each agent must agree upon a common value of interest. Within the control community, a primary focus is on the application of these distributed protocols to physical systems.

The elegance of the agreement protocol lies in its simplicity. It comes as no surprise that this algorithm also applies to other classes of problems beyond the control of physical systems. In fact, the origins of the agreement protocol can be traced to distributed computation and optimization problems [Bertsekas and Tsitsiklis (1989); Tsitsiklis (1984)]. More recently, sub-gradient algorithms have become a focal point for research in distributed optimization [Nedic and Ozdaglar (2009); Johansson et al. (2009); Zhu and Martínez (2011)]. Although the consensus problem and distributed optimization problems are strongly related, a major difference is that in the latter the agents are not physical entities but processing nodes.

There has been some recent work lying at the intersection of these two fields focusing simultaneously on the control of physical systems and distributed solutions to global optimization problems. Such a scenario has been considered in [Johansson et al. (2008)] where each agent negotiates a consensus value based on some cost function using a distributed optimization algorithm before controlling the physical system to that value. Dual decomposition is used in [Rantzer (2008)] for an optimal distributed controller design. In [Buerger et al. (2010)], a dual decomposition method is used to dynamically determine an optimal velocity for a team of self-interested agents.

This paper studies a distributed optimization problem which is coupled to a physical control system. A team of self-interested agents is considered that should achieve consensus at a specified time. In particular, we consider an ensemble of single integrator agents each equipped with a quadratic cost function penalizing its distance to a desired state, termed it's "preference," and its control energy. The agents are only coupled by the consensus constraint at the end of the time horizon. While this problem can be formulated as a centralized optimal control problem, we study a distributed solution that negotiates the consensus value in real-time based on a dual decomposition sub-gradient algorithm. Under the premise that communication and computation are not instantaneous, we assume that between communication rounds, the agents are already moving in the direction they consider to be optimal at that time instance. The dynamic element of this problem effectively changes the parameters of the optimization problem and we consider how this change deviates from the solution of the centralized static case.

The distributed dual sub-gradient algorithm we develop relies on the notion of a *shrinking horizon* to account for the dynamic changes in the system as time progresses. We show that the evolution of the algorithm combined with the agents dynamics can be cast as a *linear time-varying dynamical system* (LTV). This provides insight into the influence of the step-size rule for the convergence of the algorithm. In particular, we show that the optimal Lagrange multipliers associated with the consensus constraint varies along the trajectory of the dynamic shrinking horizon problem. Based on this observation we analyze the performance of the dynamic algorithm, showing that the final distance to consensus between the agents is a function of the error between the estimated and the optimal multiplier at the final time. Additionally, we provide an LTV representation of the estimation error, which allows us to draw conclusions on suitable step-size rules for the optimization algorithm.

The organization of the paper is as follows. In the next subsection we introduce our notation. The general problem set-up, including the formulation of the centralized optimal control problem, is given in §2. In §3 the shrinking horizon preference agreement algorithm is presented, and a convergence analysis of the algorithm is given in §4. Finally, in §5 a simulation example is given, and §6 provides some concluding remarks.

[1] The authors would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart and within the Priority Program 1305 "Control Theory of Digitally Networked Dynamical Systems."

Notation The notation we employ is standard. The set of real numbers is denoted $\mathbb{R}$, and $\mathbb{R}_>$ ($\mathbb{R}_\geq$) is the set of positive (non-negative) numbers. For a vector $x \in \mathbb{R}^n$, we denote its transpose by x'. The ith component of the matrix-vector product is expressed as $[Ax]_i$ and the ijth element of the matrix A as $[A]_{ij}$. The all ones vector of length n is denoted $\mathbf{1}_n$ and I_n is the $n \times n$ identity matrix. The Euclidean norm of a vector x is denoted $\|x\|_2$. The unit vector of length n for the ith coordinate is denoted $e_{i,n}$.The communication structure between agents is captured by a graph $\mathcal{G}$ with node set $\mathcal{V} = \{v_1, \ldots, v_n\}$ and edge set $\mathcal{E}$. A *spanning tree* is a connected graph with $|\mathcal{V}| - 1$ edges and does not contain cycles. We define the set neighborhood of node i as $\mathcal{N}_i = \{\{i, j\} \in \mathcal{E}\}$. The *incidence matrix* of the graph $\mathcal{G}$, $E(\mathcal{G}) \in \mathbb{R}^{n \times |\mathcal{E}|}$, is a $\{0, \pm 1\}$-matrix with rows and columns indexed by the vertices and edges of $\mathcal{G}$ such that $[E(\mathcal{G})]_{ik}$ has the value '+1' if node i is the initial node of edge k, '-1' if it is the terminal node, and '0' otherwise [Godsil and Royle (2001)]. For brevity, we will omit the explicit dependence of the graph, and simply write E.

2. THE PREFERENCE-BASED AGREEMENT PROBLEM

We study the problem of self-interested dynamical agents that must agree upon a common state at the end of a given time horizon. The agents are modeled as a group of n single integrator systems,

$$x_i(t+1) = x_i(t) + u_i(t), \;\; x_i(0) = x_{i0}, \tag{1}$$

with $i = 1, \ldots, n$ and $x_i(t) \in \mathbb{R}$. The state and control vector for all n agents are denoted as $x(t) = [x_1(t), \ldots, x_n(t)]'$ and $u(t) = [u_1(t), \ldots, u_n(t)]'$. Agents can communicate with each other according to a fixed communication graph $\mathcal{G}$, assumed to be a spanning tree. Furthermore, we only consider *synchronous communication*, where all agents communicate at the same time instant.

The self-interest of each agent is modeled as a quadratic objective, attaining its minimum at a specific individual preference value ξ_i. Each agent aims to minimize the objective

$$J_i(t_0, T, x_i, u_i) = \frac{1}{2}\left(\sum_{t=t_0}^{T-1} q_i(x_i(t+1) - \xi_i)^2 + r_i u_i(t)^2\right), \tag{2}$$

where $q_i, r_i \in \mathbb{R}_>$ are the state and control weights. The individual agents are coupled by a requirement to achieve agreement at the end of the time horizon T; that is there is a terminal time constraint,

$$x_1(T) = x_2(T) = \cdots = x_n(T) \Leftrightarrow E'\mathbf{x}(T) = 0. \tag{3}$$

From a centralized perspective, the preference-based agreement problem can be stated as the optimal control problem with terminal constraint

$$OCP(t_0, T, x_0): \min_{x,u} \sum_{i=1}^{n} J_i(t_0, T, x_i, u_i) \quad \text{s.t. (1) and (3).} \tag{4}$$

We collect the entire state and control trajectories of each agent into the row vectors $\mathbf{x}_i = [\, x_i(t_0+1) \cdots x_i(T) \,]$ and $\mathbf{u}_i = [\, u_i(t_0) \cdots u_i(T-1) \,]$. As we are considering a team of n agents, we introduce further notation to streamline the presentation. The bold-face vectors $\mathbf{x} = [\, (\mathbf{x}_1)' \cdots (\mathbf{x}_n)' \,]' \in \mathbb{R}^{n \times T}$ and $\mathbf{u} = [\, (\mathbf{u}_1)' \cdots (\mathbf{u}_n)' \,]' \in \mathbb{R}^{n \times T}$ denote the complete trajectories for the state and control of the entire ensemble of agents, and $(\overline{\mathbf{x}}, \overline{\mathbf{u}})$ denotes the *optimal* trajectory generated by the solution of $OCP(t_0, T, x_0)$. At times, we will be interested in the state or control trajectory value for all agents at a particular time τ; we will denote this by $\mathbf{x}(\tau) \in \mathbb{R}^{n \times 1}$ and $\mathbf{u}(\tau) \in \mathbb{R}^{n \times 1}$; similarly, $\mathbf{x}_i(\tau) \in \mathbb{R}$ refers to the value of agent i at time τ.

Note that problem $OCP(t_0, T, x_0)$ can be reformulated as a static quadratic program. Using the new notation, the objective for each agent can be stated as $J_i(t_0, T, \mathbf{x}_i, \mathbf{u}_i) = \frac{1}{2}(q_i\|\mathbf{x}_i - \mathbf{1}_T'\xi_i\|_2^2 + r_i\|\mathbf{u}_i\|_2^2)$, and the dynamic constraint as the linear equation

$$\mathbf{x}_i = \mathbf{1}_T' x_{i0} + \mathbf{u}_i B_T'. \tag{5}$$

Here, $B_T \in \mathbb{R}^{T \times T}$ is defined such that $[B_T]_{kl} = 1$ for $k \geq l$ and zero otherwise.

Throughout this paper, we will not only rely on the primal problem formulation (4), but we will often consider the dual problem. The dual problem is obtained by relaxing the coupling constraint with a multiplier μ into the objective to obtain the Lagrangian,

$$\mathcal{L}(\mathbf{x}, \mathbf{u}, \mu) = \sum_{i=1}^{n} J_i(t_0, T, \mathbf{x}_i, \mathbf{u}_i) + \mu' E'\mathbf{x}(T). \tag{6}$$

The dual function is obtained by minimizing (6) subject to the dynamic constraint (5), $q(\mu) = \min_{\mathbf{x},\mathbf{u}} \mathcal{L}(\mathbf{x}, \mathbf{u}, \mu)$. We denote the optimal solution of the primal and dual problems as $(\overline{\mathbf{x}}, \overline{\mathbf{u}}, \overline{\mu})$. As $OCP(t_0, T, x_0)$ is a strictly convex problem (a quadratic program with linear constraints), we have strong duality which implies that $q(\overline{\mu}) = J(t_0, T, \overline{\mathbf{x}}, \overline{\mathbf{u}})$ [Ruszczynski (2006)].

Note that the multiplier μ is associated with each edge in $\mathcal{G}$. Equivalently, we can consider a variable associated with each agent by defining

$$\gamma = E\mu; \tag{7}$$

the Lagrangian can now be written as a function of γ with the associated dual function $q(\gamma)$.

The important feature of this re-formulation is that the dual function $q(\gamma)$ is completely separable across each agent. This then motivates the dual sub-gradient algorithm, which can be stated as follows. At each iteration step k of the algorithm, the dual function is computed for a fixed value of $\hat{\gamma}^{[k]}$. That is, each agent solves the following quadratic program, $QP_i(k)$,

$$(\hat{\mathbf{x}}_i^{[k+1]}, \hat{\mathbf{u}}_i^{[k+1]}) = \arg\min_{\hat{\mathbf{x}}_i^{[k]}, \hat{\mathbf{u}}_i^{[k]}} J_i(t_0, T, \hat{\mathbf{x}}_i^{[k]}, \hat{\mathbf{u}}_i^{[k]}) + \hat{\gamma}_i^{[k]}\hat{\mathbf{x}}_i^{[k]}(T) \tag{8}$$
$$\text{s.t. } \hat{\mathbf{x}}_i^{[k]} = \mathbf{1}_{\tilde{T}}' x_{i0} + \hat{\mathbf{u}}_i^{[k]} B_{\tilde{T}}',$$

Here we have temporarily abused our notation to facilitate this discussion. The superscript, as in $\gamma^{[k]}$, denotes the iteration count for the sub-gradient algorithm, and the notation $(\hat{\mathbf{x}}_i^{[k]}, \hat{\mathbf{u}}_i^{[k]})$ denotes the optimization variables for $QP_i(k)$. While ensuring that the initial values of the dual variables satisfy $\gamma^{[0]} = E(\mathcal{G})\mu^{[0]}$, the next step is then to update the multiplier using the sub-gradient as

$$\hat{\gamma}^{[k+1]} = \hat{\gamma}^{[k]} + \alpha^{[k]} EE'\hat{\mathbf{x}}^{[k+1]}(T). \tag{9}$$

The sub-gradient for the edge multiplier μ is precisely $E'\hat{\mathbf{x}}^{[k]}(T)$, and using (7) leads to (9). The matrix EE' is the *graph Laplacian* of $\mathcal{G}$ [Godsil and Royle (2001)]. The choice of the step-size $\alpha^{[k]}$ is critical for the convergence properties of this algorithm. While there are many step-size rules that can guarantee convergence of this algorithm, tuning the step-size to achieve desirable convergence rates can be non-trivial. With a suitable choice for the step-size, the sub-gradient algorithm will converge to the optimal solution of $OCP(t_0, T, x_0)$,

$$\lim_{k \to \infty} (\hat{\mathbf{x}}^{[k]}, \hat{\mathbf{u}}^{[k]}, \hat{\gamma}^{[k]}) = (\overline{\mathbf{x}}^{(t_0, x_0)}, \overline{\mathbf{u}}^{(t_0, x_0)}, E\,\overline{\mu}^{(t_0, x_0)}).$$

For a more detailed discussion of appropriate step-size rules and sub-gradient methods the reader is referred to [Ruszczynski (2006)]. The notation $\overline{\mathbf{x}}^{(t_0,x_0)}$ is used to emphasize the dependence of the solution on the initial time and initial condition. When these are unambiguously understood, we will use the shorthand notation, i.e., $\overline{\mathbf{x}}^t$, instead.

The appeal of this method is that the update rule (9) is inherently distributed. That is, each agent can compute the value $\gamma_i^{[k+1]}$ to use in the next iteration step solely through communication with its neighbors, as defined by the communication graph $\mathcal{G}$. In particular, agent i must only send the value $\hat{\mathbf{x}}_i^{[k]}(T)$ to all neighboring agents.

While the sub-gradient algorithm is attractive due to its distributed and relatively simple architecture, we note that this algorithm must be performed *before* each agent can begin moving along its optimal trajectory. Indeed, for good convergence of the algorithm, it may be required to run for a time significantly longer than the desired horizon time T. This then motivates the question if it is possible to derive an algorithm that can be implemented *on-line*. That is, we would like to develop an algorithm where each iteration step corresponds to the actual physical time, while additionally propagating the agents along a calculated trajectory. Such an algorithm should also negotiate the final consensus value in real-time and, if possible, satisfy the terminal time constraint at the real time T while simultaneously minimizing the local performance index for each agent.

3. PREFERENCE-BASED AGREEMENT PROTOCOL

The need for a real-time distributed algorithm for solving problem $OCP(t_0, T, x_0)$ is based on the assumption that the time required to compute a sufficiently good solution using an off-line algorithm corresponds to a period where agents must remain idle. If we consider the horizon time T as an absolute deadline, then an optimal strategy would require each agent to move towards their preference state in order to minimize their individual objectives before maneuvering to the consensus state. [2]

In this direction, we propose a real-time preference-based agreement algorithm inspired by the sub-gradient algorithm. The algorithm also relies heavily on the existence of analytic solutions to quadratic programs. The general strategy of this algorithm is to physically propagate the states forward at each iteration. The corresponding sub-problem to be solved at the next time step is the quadratic program $QP_i(t)$ described in (8), but with the horizon window reduced; instead of minimizing from $t = 0$ to the horizon T, we minimize from $t = 1$. It can be considered as a "shrinking-horizon" sub-gradient algorithm. We note that a similar strategy was proposed in [Skaf et al. (2010)] in the context of finite-horizon stochastic control problems.

Here we recall that the state signal $x_i(t)$ corresponds to the true physical state of agent i at time t, where as the vectors $\hat{\mathbf{x}}_i^t$, $\hat{\mathbf{u}}_i^t \in \mathbb{R}^{\tilde{T}}$ correspond to the optimization variables associated with problem $QP_i(t)$. Finally, the notation $\hat{\mathbf{u}}_i^t(k)$ refers the element of the vector $\hat{\mathbf{u}}_i^t$ at time k. [3] The algorithm is presented in Algorithm 1.

At the discrete time instant $t < T$, each agent i solves an optimal control problem with the finite horizon $\tilde{T} = T - t$, using the given $\mu(t)$ for the estimated terminal constraint multiplier value. The optimal solution of $QP_i(t)$ is then used to propagate the actual *physical system state*, $x_i(t)$, forward. The new state is then used as the initial condition for the subsequent iteration. The key point here is that at each step of the algorithm, the agents are physically moving along the optimal trajectory calculated for a given multiplier value.

Algorithm 1 Shrinking Horizon Preference Agreement (SHPA)

Data: Initial conditions $x_i(0) = x_{i0}$ and $\mu(0) = \mu_0$; $t = 0$.
for $t := 0$ **to** $T - 1$ **do**
$\gamma^t = E\,\mu(t)$, $\tilde{T} = T - t$
Each agent solves the sub-problem $QP_i(t)$:
$\min_{\hat{\mathbf{x}}_i^t, \hat{\mathbf{u}}_i^t} J_i(t, T, \hat{\mathbf{x}}_i^t, \hat{\mathbf{u}}_i^t) + \gamma_i^t \hat{\mathbf{x}}_i^t(\tilde{T})$ s.t. $\hat{\mathbf{x}}_i^t = \mathbf{1}_{\tilde{T}} x_i(t) + B_{\tilde{T}} \hat{\mathbf{u}}_i^t$
Propagate state and multipliers using solution of $QP_i(t)$:

$$x_i(t+1) = x_i(t) + \hat{\mathbf{u}}_i^t(t),\ i = 1, \ldots, n \tag{10}$$

$$\mu(t+1) = \mu(t) + \alpha(t) E' \hat{\mathbf{x}}^t(T) \tag{11}$$

where $\alpha(t)$ satisfies some step-size rule.
end for

We now analyze the dynamic behavior of Algorithm 1. An immediate result that can be derived from this algorithm is a reformulation to a linear time-varying (LTV) system. This linear system results from the analytic solution that the problem $QP_i(t)$ admits [Boyd and Vandenberghe (2004)]. Before we present the LTV system, we first state the following result on the analytic solution for $\hat{\mathbf{u}}_i^t(t)$ and $\hat{\mathbf{x}}_i^t(T)$. The proof is omitted due to space constraints.

Lemma 1. For a given $\mu(t)$, the optimal control $\hat{\mathbf{u}}_i^t(t)$ and terminal state $\hat{\mathbf{x}}_i^t(T)$ used in the update rule (10) and (11) can be calculated analytically as

$$\hat{\mathbf{u}}_i^t(t) = -r_i^{-1}\kappa_i^1(\tilde{T})(x_i(t) - \xi_i) - r_i^{-1} q_i^{-1} \kappa_i^2(\tilde{T}) \gamma_i(t) \tag{12}$$

$$\hat{\mathbf{x}}_i^t(T) = q_i^{-1}\kappa_i^2(\tilde{T})(x_i(t) - \xi_i) - q_i^{-1}\kappa_i^3(\tilde{T})\gamma_i(t) + \xi_i, \tag{13}$$

where $\tilde{T} := T - t$ is the time horizon at time t, $\gamma(t) = E\mu(t)$, and $\kappa_i^j(\tilde{T})$ are defined in Table 1.

The optimal solution $(\hat{\mathbf{u}}_i^t(t), \hat{\mathbf{x}}_i^t(T))$ depends on the optimization parameters q_i, r_i, ξ_i, the actual systems state $x_i(t)$, the available multiplier $\gamma_i(t) = [E\mu(t)]_i$, and on time-varying functions $\boldsymbol{\kappa}^j(\tilde{T})$ which are defined on the remaining time-horizon $\tilde{T} = T - t$. Explicit expressions for $\boldsymbol{\kappa}^j(\tilde{T})$ are given in Table 1, and a more detailed discussion of these quantities are given in Appendix A. The appendix provides a recursive relationship for computing these functions in addition to some useful properties describing their behavior. We briefly note that these functions are independent of the actual states of the system, and can be computed off-line and *a-priori* given the values for q_i, r_i. This is in analogy to finite-time LQ controllers.

Having established the analytic solution of $QP_i(t)$ in Lemma 1, we can state the following result equating Algorithm 1 with an LTV dynamical system.

Lemma 2. Algorithm 1 is equivalent to the linear time-varying dynamical system

$$\begin{bmatrix} x(t+1) \\ \mu(t+1) \end{bmatrix} = \begin{bmatrix} I - \boldsymbol{\kappa}^1(\tilde{T})R^{-1} & -R^{-1}\boldsymbol{\kappa}^2(\tilde{T})Q^{-1}E \\ \alpha(t)E'\boldsymbol{\kappa}^2(\tilde{T})Q^{-1} & I - \alpha(t)E'\boldsymbol{\kappa}^3(\tilde{T})Q^{-1}E \end{bmatrix} \begin{bmatrix} x(t) \\ \mu(t) \end{bmatrix} + \begin{bmatrix} \boldsymbol{\kappa}^1(\tilde{T})R^{-1} \\ -\alpha(t)E'(\boldsymbol{\kappa}^2(\tilde{T})Q^{-1} - I) \end{bmatrix} \xi, \tag{14}$$

where $R = \mathbf{diag}\{r_1, \ldots, r_n\}$ and $Q = \mathbf{diag}\{q_1, \ldots, q_n\}$.

Proof: Insert the expressions for $(\hat{\mathbf{x}}^t(T), \hat{\mathbf{u}}_i^t(t))$ into the recursions (10) and (11). □

[2] This reasoning assumes that T is sufficiently large. For a shorter horizon each agent might not have enough time to reach its preference.
[3] For example, for time $\tau = t + k$, $\hat{\mathbf{u}}_i^t(\tau)$ refers to the $(k+1)$-th element of the vector.

Table 1. Auxiliary functions (See Appendix A)

$$\begin{array}{rcl@{\qquad}rcl} \Gamma_i(\tilde{T}) & := & q_i^{-1} I_{\tilde{T}} + r_i^{-1} B_{\tilde{T}} B_{\tilde{T}}' & \kappa_i^2(\tilde{T}) & := & \mathbf{1}_{\tilde{T}}' \Gamma_i^{-1}(\tilde{T}) e_{\tilde{T},\tilde{T}} \\ \kappa_i^1(\tilde{T}) & := & \mathbf{1}_{\tilde{T}}' \Gamma_i^{-1}(\tilde{T}) \mathbf{1}_{\tilde{T}} & \kappa_i^3(\tilde{T}) & := & 1 - q_i^{-1} e_{\tilde{T},\tilde{T}}' \Gamma_i^{-1}(\tilde{T}) e_{\tilde{T},\tilde{T}} \\ \boldsymbol{\kappa}^j & = & \mathbf{diag}\{\kappa_i^j,\, i = 1, \ldots, n\} & & & \end{array}$$

The LTV representation of Algorithm 1 is compelling for a few reasons. With this representation we are able to directly analyze the stability and convergence properties of the algorithm using tools from linear systems theory. In particular, we note that the only free parameter in (14) is the step-size $\alpha(t)$.[4] The proper choice for $\alpha(t)$ can then be cast as a stabilization and performance problem for the system in (14); this is the subject of a future work.

An interesting observation is that one of the terms driving the state $x(t)$ is given by $\boldsymbol{\kappa}^1(\tilde{T})R^{-1}(x(t) - \xi)$. In the absence of a consensus constraint, this term can be considered as the optimal state-feedback regulator. This term, however, is augmented with the term $-R^{-1}\boldsymbol{\kappa}^2(\tilde{T})Q^{-1}E\mu(t)$ which represents the influence of the multiplier over the state trajectory. It can be shown for sufficiently large $\tilde{T}$, that $\boldsymbol{\kappa}^2(\tilde{T}) \approx 0$ (see Appendix A). Therefore, when the remaining time horizon is large, there is almost no influence of the estimated multiplier on the physical system, and the state will track to its preference value. Only sufficiently close to the final time will the multiplier influence the trajectory, forcing the agents to move towards a consensus state. This gives a first intuition on how the algorithm will behave.

On the other hand, the multiplier update clearly reflects the distributed structure of the problem. Each component of the vector $\mu(t)$ is attached to an edge of the graph $\mathcal{G}$. The multiplier is then updated using only the state $x(t)$ and preference ξ that are incident to a particular edge. Furthermore, the multipliers attached to the edges that are adjacent to each other also influence the update in a weighted consensus like structure. Note, therefore, that $E'\boldsymbol{\kappa}^3(\tilde{T})Q^{-1}E$ is a *weighted edge Laplacian* [Zelazo and Mesbahi (2011)] which gives rise to the consensus like update rule.

4. PERFORMANCE AND CONVERGENCE ANALYSIS

Recall that the coupling constraint of the problem OCP requires all agents to agree upon a common state at the end of the horizon T, as stated in (3). The previous discussion suggested that for sufficiently long horizons T, the dynamic system (14) behaves as a state-feedback regulator, and each agent tracks to its preference. A reasonable measure for the performance of this algorithm, therefore, is how far the agents are from consensus at the horizon time T. This is captured by the norm of the consensus state,

$$\|E(\mathcal{G})'\mathbf{x}(T)\|_2. \tag{15}$$

Note that in the static case (e.g., problem $OCP(t, T, x_0)$), this quantity is precisely zero.

We can make this statement more explicit by first deriving an explicit expression for the multiplier associated with $OCP(t, T, x(t))$. We use the fact that the optimal control input and the optimal terminal state can be computed from the expressions (12) and (13) by replacing the multiplier $\mu(t)$ with the optimal multiplier $\overline{\mu}^t$, i.e.

$$(\overline{\mathbf{u}}^t(t), \overline{\mathbf{x}}^t(T)) = (\hat{\mathbf{u}}^t(t), \hat{\mathbf{x}}^t(T))|_{\mu(t)=\overline{\mu}^t}. \tag{16}$$

Corollary 3. The optimal multiplier values $\overline{\mu}^t$ corresponding to the problem $OCP(t, T, x(t))$ is given as

$$\overline{\mu}^t = \left(E'Q^{-1}\boldsymbol{\kappa}^3(\tilde{T})E\right)^{-1} E'\left(Q^{-1}\boldsymbol{\kappa}^2(\tilde{T})\,(x(t) - \xi) + \xi\right) \tag{17}$$

Proof: The primal feasibility of the problem $OCP(t, T, x(t))$ guarantees satisfaction of the terminal constraint. The matrix $E'Q^{-1}\boldsymbol{\kappa}^3(\tilde{T})E$ is a weighted edge-laplacian of a tree with positive weights, having only non-zero eigenvalues, and therefore invertible [Zelazo and Mesbahi (2011)]. □

The key feature, that we restate here, is at each time-horizon and state pair, there exists a *unique* optimal multiplier $\overline{\mu}^{(t,x(t))}$ leading to perfect consensus.

If the multipliers generated by (14) are able to track the multipliers $\overline{\mu}^t$ associated with a sequence of OCP problems varying along the state trajectories of (14), then the terminal constraint can be met exactly. We are now prepared to consider the *multiplier estimation error* between the system (14) and the optimal multiplier values $\overline{\mu}^t$,

$$\epsilon(\tilde{T}, x(t)) = \mu(t) - \overline{\mu}^{(t,x(t))}. \tag{18}$$

For notational convenience we will sometimes write $\epsilon(t)$ in place of $\epsilon(\tilde{T}, x(t))$.

This highlights a significant difference between an implicit objective of the shrinking horizon agreement algorithm, and the static dual decomposition sub-gradient algorithm used to solve $OCP(t_0, T, x_0)$. In particular, for the dual decomposition sub-gradient algorithm, it is desired that the multiplier $\mu^{[k]}$ converges to the multiplier for the centralized problem, $\overline{\mu}^{t_0} := \overline{\mu}^{(t_0,x_0)}$ ($\lim_{k\to\infty} \|\mu^{[k]} - \overline{\mu}^{t_0}\| \to 0$). In contrast, with the shrinking horizon agreement algorithm, we want the multiplier estimate to satisfy

$$\lim_{t\to T} \|\mu(t) - \overline{\mu}^{(t,x(t))}\| \to 0.$$

We can now analyze how the error $\epsilon(t)$ evolves along the trajectories of the system (10), (11). A main result of this work, therefore, is the observation that the error evolves as a time-varying linear system, with the step-size $\alpha(t)$ as a parameter. We summarize the result in the following theorem.

Theorem 4. The error $\epsilon(t) = \mu(t) - \overline{\mu}^{(t,x(t))}$ evolves according to the time-varying linear dynamics

$$\epsilon(t+1) = \left(E'Q^{-1}\boldsymbol{\kappa}^3(\tilde{T} - 1)E^{-1} - \alpha(t)I\right)E'Q^{-1}\boldsymbol{\kappa}^3(\tilde{T})E\epsilon(t). \tag{19}$$

The proof relies on (13), Corollary 3, properties of the constants $\boldsymbol{\kappa}^i$, and application of the principal of optimality for dynamic programming. Due to space constraints the details are omitted.

The above results relate to the performance of the error system for the multiplier values. It is also worth investigating how this impacts the error of the primal system, and in particular, the error of the terminal state as described in (15).

Recall that the algorithm computes at each time step a prediction of the terminal state $\hat{\mathbf{x}}^t(T)$ and uses the next-step optimal control to propagate the state forward at each time. Therefore, the terminal state $\mathbf{x}(T)$ is precisely equal to the predicted state $\hat{\mathbf{x}}^{T-1}(T)$ at the last step in the algorithm. This motivates a study of the "predicted disagreement" for the system, $e(t) = E'\hat{\mathbf{x}}^t(T)$. It is clear that $E'x(T) = \mathbf{e}(T-1)$.

Theorem 5. The predicted disagreement $\mathbf{e}(t) = E'\hat{\mathbf{x}}^t(T)$ evolves according to the linear time-varying dynamics

$$\mathbf{e}(t+1) = \left(I + \alpha(t)E'Q^{-1}\boldsymbol{\kappa}^3(\tilde{T} - 1)E\right)\mathbf{e}(t).$$

Proof: Using (13), we can express the predicted disagreement at time t, $\mathbf{e}(t) = E'\hat{\mathbf{x}}^t(T)$, in terms of the multiplier value $\mu(t)$.

[4] Although specified as a time-varying function, we note that a constant step-size rule will also suffice.

Alternatively, we can express this term as a function of the estimation error $\epsilon(t)$ as

$$\mathbf{e}(t) = E'\boldsymbol{\kappa}^2(\tilde{T})(x(t) - \xi) - E'Q^{-1}\boldsymbol{\kappa}^3(\tilde{T})E\overline{\mu}^{(\tilde{T},x(t))} + E'\xi - E'Q^{-1}\boldsymbol{\kappa}^3(\tilde{T})E\epsilon(t). \quad (20)$$

Note that if the optimal multiplier $\overline{\mu}^{(\tilde{T},x(t))}$ is used to compute the state trajectories at time t, then the final consensus error will be identically zero (this is equivalent to the centralized solution to $OCP(t, T, x(t))$). Therefore, all terms except the last one in (20) vanish, and what remains is an expression relating the predicted terminal state error as a function of the multiplier error,

$$\mathbf{e}(t) = -E'Q^{-1}\boldsymbol{\kappa}^3(\tilde{T})E\epsilon(\tilde{T}, x(t)). \quad (21)$$

Propagating the error state forward and using the dynamics for the multiplier error dynamics in (19) leads to the desired result. □

This result can be used to derive an error bound on the final disagreement.

Lemma 6. The final disagreement (15) is bounded by a function of the communication graph $\mathcal{G}$, the weights Q, R, and the estimation error $\epsilon(T-1)$ as

$$\|E'\mathbf{x}(T)\|_2 \leq \|E'Q^{-1}\boldsymbol{\kappa}^3(1)E\|_2\|\epsilon(T-1)\|_2. \quad (22)$$

Proof: This result follows directly from (20) and the observation that the physical state at the terminal horizon is equal to the predicted state at the previous time. □

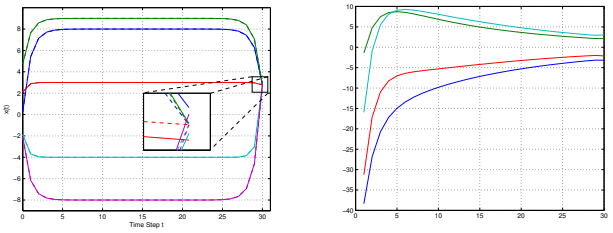
This result illustrates that the quality of the multiplier estimate by the system (14) affects the consensus error for the physical state. This also highlights the structure of the graph as an important influence on the error bound. Finally, the choice of the step-size α has the most important effect, as it will dictate the size of the multiplier error norm.

5. SIMULATION RESULTS

We illustrate our results with an example consisting of 5 agents over a time horizon of 30 steps with a constant step-size of $\alpha = 8.76$. Each agent has a unique initial condition, preference state, and weights for its state and control. Figure 1(a) shows the resulting trajectories computed by the centralized control problem $OCP(t_0, T, x_0)$ (dashed lines), and the system (14) (solid lines). Note that while the agents perform well at the early stages of the algorithm (e.g., they all track to their preference), there is still an error for their negotiated final state. We note that this result is expected, as discussed in Lemma 6. In Figure 1(b) we plot the trajectories of the multiplier estimation error, as in Theorem 4. This shows that the error is non-zero at all times, reinforcing the observation that the terminal consensus constraint is not met identically.

6. CONCLUDING REMARKS

A real-time implementation of a distributed dual sub-gradient algorithm for a multi-agent optimal control problem was presented. This method used a shrinking horizon formulation and was shown to be equivalent to a linear time-varying dynamical system. The algorithm is also inherently distributed. We also have shown that the Lagrange multipliers associated with the terminal consensus constraint evolve dynamically in the algorithm. This led to another linear and time-varying dynamical system representation for the error between the optimal multipliers and the algorithm multipliers. We additionally provided an error bound for the final state of the agents that is a function of the communication graph and the state and control weights for each agent.



(a) (b)

Fig. 1. Simulations of SHPA algorithm. In (a), solid lines are trajectories of (14) and dashed line is the solution of $OCP(t_0, T, x_0)$. Subfigure (b) shows the multiplier estimation error, as in Theorem 4.

Appendix A. ANALYSIS OF VARIABLES $\boldsymbol{\kappa}$

The functions $\boldsymbol{\kappa}^j(\tilde{T})$ defined in Table 1 play a crucial role in the evolution of the system (14). This appendix provides a more detailed analysis of these functions. The analytic solution of each subproblem $QP_i(\tilde{T})$ involves the inverse of the matrix $\Gamma_i(\tilde{T}) = q_i^{-1}I_{\tilde{T}} + r^{-1}B_{\tilde{T}}B'_{\tilde{T}}$.

The structure of $B_{\tilde{T}}$ allows us to precisely characterize $\Gamma_i^{-1}(\tilde{T})$. To begin, we employ the matrix inverse identity $(I + D^{-1})^{-1} = D(I + D)^{-1}$ [Petersen and Pedersen (2008)] to obtain

$$\Gamma_i^{-1}(\tilde{T}) = r_i\left(B_{\tilde{T}}B'_{\tilde{T}}\right)^{-1}\left(I_{\tilde{T}} + \frac{r_i}{q_i}\left(B_{\tilde{T}}B'_{\tilde{T}}\right)^{-1}\right)^{-1}. \quad (A.1)$$

We now note that $\left(B_{\tilde{T}}B'_{\tilde{T}}\right)^{-1}$ is a tridiagonal matrix with the value -1 on the lower and upper diagonal, and 2 on the diagonal, except for the last element that takes the value 1 [Usmani (1994)]. This can be used to write

$$[F_i(\tilde{T})]_{ij} = [I_{\tilde{T}} + \frac{r_i}{q_i}(B_{\tilde{T}}B'_{\tilde{T}})^{-1}]_{ij} = \begin{cases} 1 + 2\frac{r_i}{q_i}, & i = j < \tilde{T} \\ 1 + \frac{r_i}{q_i}, & i = j = \tilde{T} \\ -\frac{r_i}{q_i}, & j = i \pm 1 \\ 0, & \text{otherwise} \end{cases} \quad (A.2)$$

In [Usmani (1994)], a concise expression for the inverse of a tridiagonal matrix is derived based upon a recurrence relation. The recursion generates two sequences each containing $\tilde{T}$ elements. As each agent must compute its own sequence at each time t, we denote the sequences as $\theta_i(\tilde{T})$ and $\phi_i(\tilde{T})$. The kth element of the sequence is denoted as $\theta_i^{[k]}(\tilde{T})$ and $\phi_i^{[k]}(\tilde{T})$. The initial conditions are $\theta_i^{[0]}(\tilde{T}) = 1$, $\theta_i^{[1]}(\tilde{T}) = 1 + 2\frac{r_i}{q_i}$, $\phi_i^{[\tilde{T}+1]}(\tilde{T}) = 1$, and $\phi_i^{[\tilde{T}]}(\tilde{T}) = 1 + \frac{r_i}{q_i}$.

$$\theta_i^{[k]}(\tilde{T}) = \left(1 + 2\frac{r_i}{q_i}\right)\theta_i^{[k-1]}(\tilde{T}) - \left(\frac{r}{q}\right)^2\theta_i^{[k-2]}(\tilde{T}),\ k = 2, \ldots, \tilde{T} - 1$$

$$\theta_i^{[\tilde{T}]}(\tilde{T}) = \left(1 + \frac{r_i}{q_i}\right)\theta_i^{[\tilde{T}-1]}(\tilde{T}) - \left(\frac{r}{q}\right)^2\theta_i^{[\tilde{T}-2]}(\tilde{T})$$

$$\phi_i^{[k]}(\tilde{T}) = \left(1 + 2\frac{r_i}{q_i}\right)\phi_i^{[k+1]}(\tilde{T}) - \left(\frac{r}{q}\right)^2\phi_i^{[k+2]}(\tilde{T}),\ k = \tilde{T} - 1, \ldots, 1$$

Observe that this recursion also computes the determinant of $F_i(\tilde{T})$, $\det[F_i(\tilde{T})] = \theta_i^{[\tilde{T}]}(\tilde{T})$.

Proposition 7. For each agent i and horizon window $\tilde{T}$, $\theta_i^{[k]}(\tilde{T}) = \theta_i^{[k]}(\tilde{T}-1)$ for $k = 0, 1, \ldots, \tilde{T}-2$.

Proof: The proof follows directly from the recursion. □

Proposition 7 is important from both a computational and memory storage standpoint. The first step in Algorithm 1 must implicitly computes the sequence $\theta_i(T)$, and as time progresses, each new sequence $\theta_i(T-t)$ can use the first $T-t-2$ elements of $\theta_i(T)$. We now present some properties for the sequence $\theta_i(\tilde{T})$.

Proposition 8. For each agent i and time $t \in [0,\ T]$, the sequence $\theta_i(\tilde{T})$ satisfies $0 < \theta_i^{[\tilde{T}-1]}(\tilde{T}) < \theta_i^{[\tilde{T}]}(\tilde{T})$.

Proof: From Geršgorin's circle theorem [Horn and Johnson (1991)] and noting that $F_i(\tilde{T})$ is a real and symmetric matrix, we conclude that all the eigenvalues of $F_i(\tilde{T})$ lie in interval $[1,\ 1+4\frac{r_i}{q_i}]$. This implies that $\theta_i^{[\tilde{T}]}(\tilde{T}) \geq 1$ for all $t \in [0,\ T]$.

We now note that $\theta_i^{[\tilde{T}-1]}(\tilde{T})$ is the determinant of a principle minor of $F_i(\tilde{T})$ obtained by deleting the last row and column, denoted as $\{F_i(\tilde{T})\}_{\tilde{T}}$. Consequently, $F_i(\tilde{T})$ can be viewed as the bordered matrix,

$$F_i(\tilde{T}) = \begin{bmatrix} \{F_i(\tilde{T})\}_{\tilde{T}} & \begin{bmatrix} \mathbf{0} \\ -\frac{r}{q} \end{bmatrix} \\ \begin{bmatrix} \mathbf{0} & -\frac{r}{q} \end{bmatrix} & 1+\frac{r}{q} \end{bmatrix} \tag{A.3}$$

Denoting the eigenvalues of $F_i(\tilde{T})$ and $\{F_i(\tilde{T})\}_{\tilde{T}}$ as λ_j and μ_k respectively, we can invoke Cauchy's interlacing theorem for symmetric tridiagonal matrices [Golub and Van Loan (1996)] to order the eigenvalues as $1 < \lambda_1 < \mu_1 < \cdots < \mu_{\tilde{T}-1} < \lambda_{\tilde{T}}$. Since the determinant of a matrix is the product of its eigenvalues, and as all eigenvalues are greater than 1, we can conclude that $\theta_i^{[\tilde{T}]}(\tilde{T}) > \theta_i^{[\tilde{T}-1]}(\tilde{T})$. □

Proposition 9. For each agent i and time $t \in [0,\ T]$, the sequences $\theta_i^{[\tilde{T}]}(\tilde{T})$ and $\theta_i^{[\tilde{T}-1]}(\tilde{T}-1)$ satisfies

$$\frac{\theta_i^{[\tilde{T}]}(\tilde{T})}{\theta_i^{[\tilde{T}-1]}(\tilde{T}-1)} > \frac{r_i}{q_i}. \tag{A.4}$$

Proof: From Proposition 7 we have that $\theta_i^{[\tilde{T}-2]}(\tilde{T}) = \theta_i^{[\tilde{T}-2]}(\tilde{T}-1)$. Therefore, we have $\theta_i^{[\tilde{T}-1]}(\tilde{T}-1) = \theta_i^{[\tilde{T}-1]}(\tilde{T}) - \frac{r_i}{q_i}\theta_i^{[\tilde{T}-2]}(\tilde{T})$. Using the recursion formula and Proposition 8, it can be shown that $\frac{\theta_i^{[\tilde{T}]}(\tilde{T})}{\theta_i^{[\tilde{T}-1]}(\tilde{T}-1)} = \frac{\theta_i^{[\tilde{T}-1]}(\tilde{T})}{\theta_i^{[\tilde{T}-1]}(\tilde{T}-1)} + \frac{r_i}{q_i}$. All quantities on the right-hand side are strictly positive which concludes the proof. □

The inverse of $F_{\tilde{T}}$ can be computed as [Usmani]

$$\left[F_{\tilde{T}}^{-1}\right]_{uv} = \begin{cases} (-1)^{u+v}\displaystyle\prod_{k=u}^{v-1}\left(\frac{-r}{q}\right)\frac{\theta_i^{[u-1]}(\tilde{T})\phi_i^{[v+1]}(\tilde{T})}{\theta_i^{[\tilde{T}]}(\tilde{T})} & if\ u \leq v \\ (-1)^{u+v}\displaystyle\prod_{k=v}^{u-1}\left(\frac{-r}{q}\right)\frac{\theta_i^{[v-1]}(\tilde{T})\phi_i^{[u+1]}(\tilde{T})}{\theta_i^{[\tilde{T}]}(\tilde{T})} & if\ u > v. \end{cases} \tag{A.5}$$

Algorithm 1 requires that each agent compute the values $\hat{u}_i^{[1]}(t)$ and $\hat{x}_i^{[\tilde{T}]}(t)$ at time t (see Lemma 1). We can now use the above results to derive expressions for $\kappa_i^1(\tilde{T}), \kappa_i^2(\tilde{T})$, and $\kappa_i^3(\tilde{T})$.

Proposition 10.

$$\kappa_i^1(\tilde{T}) = \frac{r_i}{\theta_i^{[\tilde{T}]}(\tilde{T})}\sum_{j=2}^{\tilde{T}+1}\left(\frac{r_i}{q_i}\right)^{j-2}\phi_i^{[j]}(\tilde{T}). \tag{A.6}$$

Proof: Observe that $\mathbf{1}'_{\tilde{T}}(B_{\tilde{T}}B'_{\tilde{T}})^{-1} = [\ 1\ \mathbf{0}_{\tilde{T}-1}\ ]$. Applying (A.5) and (A.1) yields the desired result. □

Proposition 11.

$$\kappa_i^2(\tilde{T}) = \frac{r_i}{\theta_i^{[\tilde{T}]}(\tilde{T})}\left(\frac{r_i}{q_i}\right)^{\tilde{T}-1}. \tag{A.7}$$

Proof: This follows directly from (A.5). □

Proposition 12. For each agent i and $t \in [0,\ T]$, $\kappa_i^2(\tilde{T})$ is strictly monotonically decreasing as $t \to 0$.

Proof: From Proposition 8 we can conclude that $\frac{\kappa_i^2(\tilde{T})}{\kappa_i^2(\tilde{T}-1)} < 1$. Since $\kappa_i^2(\tilde{T}) > 0$ for all t we conclude that $\kappa_i^2(\tilde{T}) < \kappa_i^2(\tilde{T}-1)$ and in fact $\lim_{\tilde{T}\to\infty}\kappa_i^2(\tilde{T}) = 0$. □

Proposition 13.

$$\kappa_i^3(\tilde{T}) = \frac{\theta_i^{[\tilde{T}-1]}(\tilde{T})}{\theta_i^{[\tilde{T}]}(\tilde{T})} \tag{A.8}$$

Proof: This follows directly from (A.5). □

REFERENCES

Bertsekas, D.P. and Tsitsiklis, J. (1989). *Parallel and distributed computation*. Prentice Hall Inc., Old Tappan, NJ.

Boyd, S.P. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press, Cambridge.

Buerger, M., Schmidt, G.S., and Allgower, F. (2010). Preference Based Group Agreement in Cooperative Control. In *Proc. of the 8th IFAC Symposium on Nonlinear Control Systems*, 149–154.

Godsil, C. and Royle, G. (2001). *Algebraic graph theory*. Springer.

Golub, G.H. and Van Loan, C.F. (1996). *Matrix Computations*. John Hopkins University Press, Baltimore, MD, USA, 3rd edition.

Horn, R.A. and Johnson, C.R. (1991). *Matrix Analysis*. Cambridge University Press, New York, NY.

Johansson, B., Speranzon, A., Johansson, M., and Johansson, K. (2008). On decentralized negotiation of optimal consensus. *Automatica*, 44(4), 1175–1179.

Johansson, B., Rabi, M., and Johansson, M. (2009). A randomized incremental subgradient method for distributed optimization in networked systems. *SIAM Journal on Optimization*, 20(3), 1157–1170.

Mesbahi, M. and Egerstedt, M. (2010). *Graph Theoretic Methods in Multiagent Networks*. Princeton University Press, Princeton, NJ.

Nedic, A. and Ozdaglar, A. (2009). Distributed Subgradient Methods for Multi-Agent Optimization. *IEEE Transactions on Automatic Control*, 54(1), 48–61.

Petersen, K.B. and Pedersen, M.S. (2008). The matrix cookbook. Version 20081110.

Rantzer, A. (2008). Using game theory for distributed control engineering. Technical Report ISRN LUTFD2/TFRT--7620--SE, Department of Automatic Control, Lund University, Sweden.

Ruszczynski, A. (2006). *Nonlinear Optimization*. Princeton University Press.

Skaf, J., Boyd, S., and Zeevi, A. (2010). Shrinking-horizon dynamic programming. *International Journal of Robust and Nonlinear Control*.

Tsitsiklis, J.N. (1984). *Problems in Decentralized Decision Making and Computation*. Ph.D. thesis, Laboratory for Information and Decision Systems, MIT.

Usmani, R. (1994). Inversion of a Tridiagonal Jacobi Matrix. *Linear Algebra and its Applications*, 212, 413–414.

Zelazo, D. and Mesbahi, M. (2011). Edge Agreement: Graph-theoretic Performance Bounds and Passivity Analysis. *IEEE Transactions on Automatic Control*, 56(3), 544–555.

Zhu, M. and Martínez, S. (2011). On distributed convex optimization under inequality and equality constraints via primal-dual subgradient methods. *IEEE Transactions on Automatic Control (to appear)*.