Distributed event-triggered optimization for linear programming

Dean Richert Jorge Cortés

Abstract— **This paper considers a network of agents whose objective is for the aggregate of their states to converge to a solution of a linear program. We assume that each agent has limited information about the problem data and communicates with other agents at discrete times of its choice. Our main contribution is the development of a distributed continuous-time dynamics and a set of state-based rules, termed triggers, that an individual agent can use to determine when to broadcast its state to neighboring agents to ensure convergence. Our technical approach to the algorithm design and analysis overcomes a number of challenges, including establishing convergence in the absence of a common smooth Lyapunov function, ensuring that the triggers are detectable by agents using only local information, and accounting for the asynchronism in the state broadcasts of the agents. Simulations illustrate our results.**

I. Introduction

The global objective of many multi-agent systems can be formulated as an optimization problem where the individual agents' states are the decision variables. Due to the inherent network structure of these problems, much research has been devoted to developing local dynamics for each agent such that the aggregate of their states converge to a solution of the optimization problem. From an analysis viewpoint, the availability of powerful concepts and tools from stability analysis makes continuous-time coordination algorithms appealing. However, their implementation requires continuous information flow among agents. On the other hand, discrete-time algorithms are amenable to real-time implementation, but the selection of the stepsizes to guarantee convergence has to take into account worst-case situations, leading to an inefficient use of the network resources. In this paper we seek to combine the advantages of both approaches by designing a distributed continuous-time dynamics that relies on event-triggered communication to solve linear programs.

Literature review. The present work has connections with three main areas: distributed optimization, event-triggered control, and switched and hybrid systems. The literature on distributed optimization of multi-agent systems is vast and includes dual-decomposition [1], alternating direction method of multipliers [2], subgradient projection algorithms [3], [4], [5], auction algorithms [6], and saddle-point dynamics [7], [8]. The works [9], [10] propose algorithms specifically designed for distributed linear programming. In [9], the goal is for agents to agree on the global solution. In [10], instead, the goal is for the aggregate of agents' states to converge to a solution. Event-triggered control seeks to trade computation and decision-making for less communication, sensing or actuation effort while guaranteeing a desired level of performance, see e.g., [11], [12], [13] and references therein. A few works [14], [15], [16] have explored the design of distributed event-triggered optimization algorithms for multi-agent systems. A major difference between event-triggered stabilization and optimization is that in the former, the equilibrium is known a priori, whereas in the latter the determination of the equilibrium point is the objective itself. Finally, our work here is related to the literature on switched and hybrid systems [17], [18], [19]. To the authors' knowledge, this work is the first to consider an event-triggered implementation of a state-dependent switched dynamical system. A unique challenge that arises in this scenario is that the use of outdated state information by an event-triggered implementation may cause the system to miss a mode switch.

Statement of contributions. Our main contribution is the design of a distributed continuous-time dynamics and a set of distributed criteria to trigger state broadcasts among agents that enable a network of agents to collectively solve a linear program. Our starting point is an extension of the continuous-time dynamics for linear programming we introduced in [10]. We design a centralized event-triggered communication law and characterize its convergence properties. In doing so, we overcome the fact that a strict Lyapunov function is unknown for the original continuous-time dynamics by introducing a discontinuous potential function and examining its evolution during intervals where state-broadcasts do not occur. The fact that our potential function lacks some properties of a true Lyapunov function, such as continuity, complicates our proof of convergence. We draw on some concepts from switched and hybrid systems but, to our knowledge, the specific challenges we face have not been explored in the literature. Using the centralized design as motivation, we design a distributed event-triggered communication law. Because the centralized triggers are not locally detectable by individual agents, we are forced to relax some of them. We show that the relaxed triggers preserve the necessary decrease in the potential function and state a convergence result for the fully distributed implementation. In addition, we derive sufficient conditions for executions to be persistently flowing (that is, state broadcasts are separated by a uniform time infinitely often). We show that the asynchronous state broadcasts cannot be the cause of non-persistently flowing executions and conjecture that all executions are persistently flowing. For space reasons, proofs are omitted and will appear elsewhere.

II. Preliminaries

Here we introduce notation and notions of hybrid systems.

Notation: For $x \in \mathbb{R}^p$, $x \geq 0$ means that every component of

The authors are with the Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA 92093, USA, {drichert,cortes}@ucsd.edu

x is non-negative. The maximum component of x is $||x||_\infty$. For a matrix A, its i^{th} row is a_i, its (i,j)-element is $a_{i,j}$, and its spectral radius is $\rho(A)$. Given sets $S_1, S_2 \subseteq \mathbb{R}^p$, the elements that are in S_1 but not S_2 are $S_1 \setminus S_2$. A function $f : X \to \mathbb{R}$ is Lipschitz on X if, for some constant K and for all $x_1, x_2 \in X$, it holds that $||f(x_1)-f(x_2)|| \leq K||x_1-x_2||$. f is convex if for all $x_1, x_2 \in X$ and all $\lambda \in [0,1]$ it holds that $f(\lambda x_1+(1-\lambda)x_2) \leq \lambda x_1+(1-\lambda)x_2$. f is concave if $-f$ is convex. The generalized gradient of f at $\hat{x} \in X$, denoted $\partial f(\hat{x})$, is the set of v such that $f(x) - f(\hat{x}) \geq v^T(x - \hat{x})$ for all $x \in X$. If $g : X \times Y \to \mathbb{R}$, then $\partial_x g(x,y)$ (resp. $\partial_y g(x,y)$) is the generalized gradient of $x \mapsto g(x,y)$ (resp. $y \mapsto g(x,y)$). For $c \in \mathbb{R}$, $f^{-1}(\leq c)$ is the set of $x \in X$ such that $f(x) \leq c$. Given $V : \mathbb{R}^p \to \mathbb{R}$ and $F : \mathbb{R}^p \to \mathbb{R}^p$, the Lie-derivative of V along F at x is $\mathcal{L}_F V(x)$.

Hybrid systems: These basic notions on hybrid systems follow closely the exposition found in [19]. A hybrid system $H = (f, g, C, D)$ is a dynamical system of the form

$$\dot{x} = f(x), \quad x \in C, \tag{1a}$$
$$x^+ = g(x), \quad x \in D. \tag{1b}$$

A function $\psi(t,j)$ is a solution to the hybrid system (1) if

(i) for all $j \in \mathbb{N}$ such that $I^j := \{t : (t,j) \in \operatorname{dom}\psi\}$ has non-empty interior

$$\psi(t,j) \in C, \qquad \forall t \in \operatorname{int}(I^j),$$
$$\dot{\psi}(t,j) = f(\psi(t,j)), \quad \text{for almost all } t \in I^j.$$

(ii) for all $(t,j) \in \operatorname{dom}\psi$ such that $(t,j+1) \in \operatorname{dom}\psi$

$$\psi(t,j) \in D,$$
$$\psi(t,j+1) = g(\psi(t,j)).$$

We call ψ *persistently flowing* if

(i) $([t_J, \infty), J) \subset \operatorname{dom}\psi$ for some $J \in \mathbb{N}$, or
(ii) there exists a $\tau_P > 0$ and an infinite increasing sequence $\{j_k\}_{k=0}^\infty \subset \mathbb{N}$ such that $([t_{j_k}, t_{j_k}+\tau_P], j_k) \subset \operatorname{dom}\psi$ for each $k \in \mathbb{N}$.

III. Problem statement and network model

Our main motivation for this paper is to solve, in a multi-agent setting, a standard form linear program

$$\min\{c^T x : Ax = b, \quad x \geq 0\}, \tag{2}$$

where $c, x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and $A \in \mathbb{R}^{m \times n}$ for $n, m \in \mathbb{N}$. We assume that (2) is feasible with a finite optimal value and let $\mathcal{X} \subseteq \mathbb{R}^n$ denote the set of its solutions. We also make the following standing assumption of the matrix A:

SA #1: $\rho(A^T A) \leq 1$.

This assumption is made without loss of generality and the results of the paper can easily be extended to the case when $\rho(A^T A) > 1$; its purpose is simply to ease notation.

The multi-agent setting that we consider is described at a high-level as follows: Consider a collection of agents, each with a unique identifier $i \in \{1, \ldots, n\}$ and associated state x_i. The goal of the network is for the aggregate of the agents' states, $x = (x_1, \ldots, x_n)$, to converge to a solution of (2). However, each agent i knows only its own state and the following limited data: c_i and the non-zero elements of any row a_ℓ (and the associated b_ℓ) of A where $a_{\ell,i} \neq 0$. In addition, we assume that if x_i and x_j appear in a common constraint then i and j can communicate their states to each other. Even though two agents can communicate, we assume that messages can only be sent at discrete instances. We model this setup using a hybrid system, and the essence of the problem we solve is captured in the following.

Problem Statement III.1 (Distributed event-triggered linear programming). *Design a hybrid system of the form: for each* $i \in \{1, \ldots, n\}$

$$\dot{x}_i = g_i(\hat{x}), \quad \text{if } (x, \hat{x}) \notin \mathcal{T}_i, \tag{3a}$$
$$\hat{x}_i^+ = x_i, \qquad \text{if } (x, \hat{x}) \in \mathcal{T}_i, \tag{3b}$$

such that

(i) *g_i is computable by agent i and the inclusion $(x, \hat{x}) \in \mathcal{T}_i$ is detectable by agent i using local information and communication, and*
(ii) *the aggregate of the agents' states converge to a solution of* (2)*.*

The interpretation of Problem III.1 is as follows: agents use the last broadcast state $\hat{x}$ of their neighbors' states to compute their continuous-time flow as modeled by (3a). While flowing, there is no dynamics for $\hat{x}_i$, i.e., $\dot{\hat{x}}_i = 0$ when $(x, \hat{x}) \notin \mathcal{T}_i$. The condition $(x, \hat{x}) \in \mathcal{T}_i$ is a state-based trigger used by agent i to know when it should broadcast its current state x_i to its neighbors. The broadcast procedure is modeled by the jump relation (3b). Similarly, it is implied that $x_i^+ = x_i$ when $(x, \hat{x}) \in \mathcal{T}_i$. Note that there is dynamical coupling between agents through the broadcast state $\hat{x}$ only.

IV. Continuous-time discontinuous dynamics

This section begins the process of solving Problem III.1 by first designing the local continuous-time dynamics for each agent (i.e., functions g_i). The design rationale we employ is to design $g = (g_1, \ldots, g_n)$ such that (i) each g_i is computable by agent i and (ii) the trajectories of $\dot{x} = g(x)$ converge to a solution of (2). A dynamics of this type has been proposed in [10], however it does not lend itself well to an event-triggered implementation, which is our ultimate goal. Nevertheless, the design approach that we employ is analogous to the one found in that paper.

A. Continuous-time saddle-point dynamics

The general approach we use for designing the continuous-time dynamics is to define an augmented Lagrangian function based on some optimization problem and then derive the natural saddle-point dynamics for that function. We start by introducing the aforementioned optimization problem,

$$\min\{c^T x + \tfrac{1}{2} x^T x : Ax = b, \quad x \geq 0\}. \tag{4}$$

We use this regularization rather than (2) itself because the resulting algorithm is amenable to an event-triggered implementation (we will revisit this fact in Section VI). We make the following standing assumption regarding (4)

SA #2: the solution of (4) is a solution of (2).

It is well understood in the optimization literature that **SA #2** can be made true by appropriately scaling the objective direction c (see e.g., [20]). Thus, **SA #2** is again made without loss of generality. Next, we establish a correspondence between the solution of (4) and the saddle points of an associated augmented Lagrangian function.

Lemma IV.1 (Solutions of (4) as saddle points). *For $K \geq 0$, consider the function*

$$L^K(x,z) = c^T x + \tfrac{1}{2}x^T x + \tfrac{1}{2}(Ax-b)^T(Ax-b) + z^T(Ax-b) + K\mathbf{1}_n^T \max\{0,-x\}.$$

Then, L^K is convex in x and concave in z. Suppose that $x_* \in \mathbb{R}^n$ (resp. $z_* \in \mathbb{R}^m$) is the solution to (4) (resp. a solution to the dual of (4)). Then, for $K > \|c + x_* + A^T z_*\|_\infty$,

(i) *(x_*, z_*) is a saddle point of L^K,*
(ii) *if $(\bar{x}, \bar{z})$ is a saddle point of L^K, $\bar{x}$ is a solution of (2).*

As a way of solving (2), the above result suggests using the saddle-point dynamics associated to the Lagrangian function,

$$\dot{x} \in -\partial_x L^K(x,z), \tag{5a}$$
$$\dot{z} = \partial_z L^K(x,z). \tag{5b}$$

This saddle-point dynamics is well-defined since L^K is Lipschitz. In fact, one can show that any bounded trajectory of the following (continuous-time discontinuous) dynamics

$$\dot{x}_i = \begin{cases} f_i(x,z), & \text{if } x_i > 0, \\ \max\{0, f_i(x,z)\}, & \text{if } x_i = 0, \end{cases} \quad i \in \{1,\dots,n\}, \tag{6a}$$
$$\dot{z} = Ax - b, \tag{6b}$$

where $f(x,z) = -(A^T z + c + x) - A^T(Ax-b)$, is a trajectory of (5). This is the dynamics that we design an event-triggered implementation for in the next sections.

B. Distributed implementation

Let us now explore the distributed implementation of (6) based on the network model alluded to in Section III. The dynamics of the auxiliary variables $z \in \mathbb{R}^m$ can be interpreted as internal to the system. For purposes of analysis, we consider m virtual agents with identifiers $\{n+1,\dots,n+m\}$ where virtual agent $n+\ell$ has state z_ℓ and knows the data a_ℓ and b_ℓ (in actual implementation, the state and dynamics of a virtual agent can be stored and implemented by one of the real agents). For each $\ell \in \{1,\dots,m\}$, the set of agents

$$\{i \in \{1,\dots,n\} : a_{\ell,i} \neq 0\} \cup \{n+\ell\},$$

can communicate their state information to each other. In words, if x_i and x_j appear in constraint ℓ, then agents i, j, and $n+\ell$ can communicate with each other. The set of all agents that i can communicate with is denoted $\mathcal{N}_i$ (the *neighbor* set of i). The set of real (resp. virtual) neighbors of i is $\mathcal{N}_i^x := \mathcal{N}_i \cap \{1,\dots,n\}$ (resp. $\mathcal{N}_i^z := \mathcal{N}_i \cap \{n+1,\dots,n+m\}$). Under these assumptions, it is straight forward to verify that real agent $i \in \{1,\dots,n\}$ can compute $f_i(x,z)$ using local information and can thus implement $\dot{x}_i = f_i(x,z)$. Likewise, a virtual agent $n+\ell \in \{n+1,\dots,n+m\}$ can compute and implement the dynamics $\dot{z}_\ell = a_\ell^T x - b_\ell$.

V. Centralized event-triggered design

Now that we have defined the agents' dynamics during continuous-time flow, we turn our attention to the design of the state-based broadcast trigger sets, $\mathcal{T}_i$ (cf. Problem III.1). However, in this preliminary design, we only concern ourselves with a centralized trigger. The next section will design a distributed version of the design here. In light of the developments thus far, it is helpful to formally state the problem we solve here in terms of the dynamics (6).

Problem Statement V.1 (Centralized event-triggered linear programming). *Design a hybrid system of the form: if $(x,z,\hat{x},\hat{z}) \notin \mathcal{T}^c$ then*

$$\dot{x}_i = \begin{cases} f_i(\hat{x},\hat{z}), & \hat{x}_i > 0, \\ \max\{0, f_i(\hat{x},\hat{z})\}, & \hat{x}_i = 0, \end{cases} \quad i \in \{1,\dots,n\}, \tag{7a}$$
$$\dot{z} = A\hat{x} - b, \tag{7b}$$

and, if $(x,z,\hat{x},\hat{z}) \in \mathcal{T}^c$, then

$$(\hat{x}^+, \hat{z}^+) = (x,z), \tag{7c}$$

such that the aggregate of the real agents' states, x, converges to a solution of the linear program (2).

The set $\mathcal{T}^c$ is called the *centralized trigger set* and we do not put any restrictions on individual agents being able to detect the inclusion $(x,z,\hat{x},\hat{z}) \in \mathcal{T}^c$. Also, note that in our centralized event-triggered setup the state broadcasts are performed synchronously as modeled by (7c). The organization of this section is as follows. We first introduce a compact notation to represent (7) that will be useful in our analysis. Then we introduce a certain potential function and design $\mathcal{T}^c$ such that its evolution is decreasing along solutions of (7). We conclude with a convergence result.

A. Compact notation to represent (7)

In the analysis that follows, we require a more compact notation to represent the hybrid system (7). Denote by $\sigma(\hat{x},\hat{z})$ the set of i for which $\dot{x}_i = f_i(\hat{x},\hat{z})$ in (7). Formally,

$$\sigma(\hat{x},\hat{z}) = \big\{i \in \{1,\dots,n\} : f_i(\hat{x},\hat{z}) \geq 0 \text{ or } \hat{x}_i > 0\big\}.$$

The matrix $I_{\sigma(\hat{x},\hat{z})} \in \mathbb{R}^{n\times n}$ is an identity-like matrix with a zero (i,i)-element if $i \notin \sigma(\hat{x},\hat{z})$. Defined component-wise,

$$(I_{\sigma(\hat{x},\hat{z})})_{i,j} = \begin{cases} 0, \text{ if } i \neq j \text{ or } i \notin \sigma(\hat{x},\hat{z}), \\ 1, \text{ otherwise.} \end{cases}$$

Then, a compact representation of (7a)-(7b) is

$$(\dot{x}, \dot{z}) = F(\hat{x}, \hat{z}) := (I_{\sigma(\hat{x},\hat{z})} f(\hat{x}, \hat{z}), A\hat{x} - b),$$

where $F = (F_x, F_z) : \mathbb{R}^n_{\geq 0} \times \mathbb{R}^m \to \mathbb{R}^n \times \mathbb{R}^m$.

B. Potential function and trigger set design

Now let us define and analyze the potential function, V, that we use to design the trigger set $\mathcal{T}^c$. We introduce V as

$$V(x, z) = (V_1 + V_2)(x, z),$$

where V_1 is the Lyapunov function used in [10] and V_2 is an additional function required for our design. Neither V_1 or V_2 by themselves are sufficient to design an event-triggered implementation of (6) which is why we require both. To define V_1, fix $\mathcal{K} > \|c + x_* + A^T z_*\|_\infty$ where x_* (resp. z_*) is the solution to (4) (resp. any solution of the dual of (4)) and let $(\bar{x}, \bar{z})$ be a saddle-point of $L^{\mathcal{K}}$. Then

$$V_1(x, z) = \tfrac{1}{2}(x - \bar{x})^T(x - \bar{x}) + \tfrac{1}{2}(z - \bar{z})^T(z - \bar{z}).$$

Note that $V_1 \geq 0$ is smooth with compact sublevel sets. Next,

$$V_2(x, z) = \tfrac{1}{2} f(x, z)^T I_{\sigma(x,z)} f(x, z) + \tfrac{1}{2}(Ax - b)^T(Ax - b).$$

Note that V_2 is non-negative but, due to the state-dependent matrix $I_{\sigma(x,z)}$, is only piecewise smooth. In this sense V_2 can be viewed as a collection of multiple (smooth) Lyapunov functions, each defined on a neighborhood where $\sigma(x, z)$ is constant. Also, $V_2^{-1}(0)$ is, by definition, the set of saddle-points of $L^{\mathcal{K}}$. The following result reveals an upper bound on $\mathcal{L}_F V$ in terms of the state errors

$$e_x = x - \hat{x}, \quad e_z = z - \hat{z}.$$

Proposition V.2 (Evolution of V). *Let $X \times Z \subseteq \mathbb{R}^n_{\geq 0} \times \mathbb{R}^m$ be compact and suppose that $(x, z, \hat{x}, \hat{z}) \in X \times Z \times X \times Z$ is such that $\sigma(\hat{x}, \hat{z}) \subseteq \sigma(x, z)$ and*

$$\sigma(x, z) = \lim_{\alpha \to 0} \sigma(x + \alpha F_x(\hat{x}, \hat{z}), z + \alpha F_z(\hat{x}, \hat{z})).$$

Then $\mathcal{L}_F V(x, z)$ exists and

$$\begin{aligned} \mathcal{L}_F V(x, z) \leq & -\tfrac{1}{4}(A\hat{x} - b)^T(A\hat{x} - b) + 20 e_z^T e_z && \text{(8a)} \\ & - \tfrac{1}{2} f(\hat{x}, \hat{z})^T I_{\sigma(\hat{x},\hat{z})} f(\hat{x}, \hat{z}) + 40 e_x^T e_x && \text{(8b)} \\ & + 15 f(x, z)^T I_{\sigma(x,z) \setminus \sigma(\hat{x},\hat{z})} f(x, z). && \text{(8c)} \end{aligned}$$

We now show how V can be used to design the trigger set $\mathcal{T}^c$. We incrementally design subsets of $\mathcal{T}^c$ and combine them at the end to define $\mathcal{T}^c$. The main observation that we base our design on is that, after a state broadcast, $e_x = e_z = 0$. Thus, to ensure that (8a) and (8b) we design

$$\begin{aligned} (x, z, \hat{x}, \hat{z}) \in \mathcal{T}^{c,e} := \{(x, z, \hat{x}, \hat{z}) : & 20 e_z^T e_z + 40 e_x^T e_x \\ & > \tfrac{1}{8}(A\hat{x} - b)^T(A\hat{x} - b)) \\ & + \tfrac{1}{4} f(\hat{x}, \hat{z})^T I_{\sigma(\hat{x},\hat{z})} f(\hat{x}, \hat{z})\}. \end{aligned}$$

Likewise, when there is a mismatch between the modes $\sigma(x, z)$ and $\sigma(\hat{x}, \hat{z})$ the term (8c) is positive. However, after a state-broadcast, $\sigma(x, z) = \sigma(\hat{x}, \hat{z})$ and $I_{\sigma(x,z) \setminus \sigma(\hat{x},\hat{z})} = 0$ which means that (8c) is also zero. For this reason, define

$$\mathcal{T}^{c,\sigma} := \{(x, z, \hat{x}, \hat{z}) : \sigma(x, z) \neq \sigma(\hat{x}, \hat{z})\},$$

which demands a state-broadcast when the mode σ changes. We require one final trigger set for the following reason: In the (6), the set $\mathbb{R}^n_{\geq 0} \times \mathbb{R}^m$ is invariant, but in the event-triggered case this may not be the case because agents use outdated state information. To preserve invariance of this set,

$$\mathcal{T}^{c,0} := \{(x, z, \hat{x}, \hat{z}) : \exists i \in \{1, \dots, n\} \text{ s.t. } \hat{x}_i > 0, x_i = 0\}.$$

If this trigger is activated by some agent i's state becoming zero, then it is easy to see from the definition of the dynamics (7a) that $\dot{x}_i \geq 0$ after the state broadcast and thus x_i remains non-negative. Finally, the overall trigger set is

$$\mathcal{T}^c := \mathcal{T}^{c,e} \cup \mathcal{T}^{c,\sigma} \cup \mathcal{T}^{c,0}. \tag{9}$$

We now state a convergence result of the centralized design.

Theorem V.3 (Convergence of the centralized event-triggered design). *If $\psi(t, j)$ is a persistently flowing solution of (7) with $\mathcal{T}^c$ defined as in (9) then there exists a point $(x_*, z') \in \mathcal{X} \times \mathbb{R}^m$ such that,*

$$\psi(t, j) \to (x_*, z', x_*, z') \quad \textit{as} \quad (t + j) \xrightarrow{\substack{(t,j) \in \\ \operatorname{dom} \psi}} \infty.$$

In the next section, we focus on decomposing $\mathcal{T}^c$ such that the triggers can be implemented in a distributed way as well as addressing the issue of persistently flowing solutions.

VI. Distributed event-triggered design

Using the centralized trigger set $\mathcal{T}^c$ as motivation, in this section we reveal the main contribution of this paper: a distributed event-triggered dynamics for linear programming. The two main challenges that we face in this section are that (i) there is no obvious way to decompose $\mathcal{T}^{c,\sigma}$ and (ii) the asynchronism in the state broadcasts add to the possibility of non-persistently flowing solutions. Our design assumes that each agent i keeps track of (a) the time elapsed since it last *sent* a state broadcast, denoted s_i, and (b) the time elapsed between its last state broadcast and the first broadcast it *receives* from its neighbor j, denoted $r_{i \leftarrow j}$. Let $\xi = (x, z, s, r, \hat{x}, \hat{z})$ be the total state vector and we restate the problem to reflect these developments.

Problem Statement VI.1 (Distributed event-triggered linear programming - take 2). *Design a hybrid system of the form: for each agent $i \in \{1, \dots, n+m\}$, if $\xi \notin \mathcal{T}_i$ then*

$$\dot{x}_i = \begin{cases} f_i(\hat{x}, \hat{z}), & \text{if } \hat{x}_i > 0, \\ \max\{0, f_i(\hat{x}, \hat{z})\}, & \text{if } \hat{x}_i = 0, \end{cases} \quad \text{if } i \leq n, \tag{10a}$$

$$\dot{z}_{i-n} = a_{i-n}^T \hat{x} - b_{i-n}, \quad \text{if } i \geq n+1 \tag{10b}$$

$$\dot{s}_i = \begin{cases} 1, & \text{if } s_i < s_i^{\max}, \\ 0, & \text{if } s_i \geq s_i^{\max}, \end{cases} \quad \text{for all } i, \tag{10c}$$

and, if $\xi \in \mathcal{T}_i$, then

$$\hat{x}_i^+ = x_i, \qquad \text{if } i \le n, \tag{10d}$$
$$\hat{z}_{i-n}^+ = z_{i-n}, \qquad \text{if } i \ge n+1, \tag{10e}$$
$$s_i^+ = 0, \qquad \text{for all } i, \tag{10f}$$
$$(r_{i\leftarrow j}^+, r_{j\leftarrow i}^+) = (-1, s_j), \qquad \text{for all } i \text{ \& all } j \in \mathcal{N}_i, \tag{10g}$$

such that

(i) *the inclusion $\xi \in \mathcal{T}_i$ is detectable by agent i, and*
(ii) *the aggregate of the real agents' states, x, converges to a solution of* (2)*.*

In the above, $r_{i\leftarrow j} = -1$ after an agent i broadcasts its state to indicate that it has not yet received a state broadcast from j. Some of the centralized trigger sets designed in the previous section are easily distributed, and we show how here. First, consider the following decomposition of $\mathcal{T}^{c,e}$,

$$\mathcal{T}_i^e := \begin{cases} \{\xi : (e_x)_i^2 > \mu_i f_i(\hat{x},\hat{z})^2\}, & \text{if } i \le n, \\ \{\xi : (e_z)_{i-n}^2 > \mu_i (a_{i-n}^T \hat{x} - b_{i-n})^2\}, & \text{if } i \ge n+1, \end{cases}$$

where the ranges of each μ_i to ensure convergence appear in Theorem VI.3. Likewise a decomposition for $\mathcal{T}^{c,0}$ is,

$$\mathcal{T}_i^0 := \begin{cases} \{\xi : \hat{x}_i > 0 \text{ but } x_i = 0\}, & \text{if } i \le n, \\ \emptyset, & \text{if } i \ge n+1. \end{cases}$$

A. Distributing the trigger set $\mathcal{T}^{c,\sigma}$

The challenge when designing a distributed version of $\mathcal{T}^{c,\sigma}$ is that an agent cannot detect a change in $\sigma(x,z)$ without updated state information from its neighbors. The specific scenario we refer to is when, for some real agent, $x_i = 0$ and $f_i(\hat{x},\hat{z}) < 0$ but, since the last broadcast, it happens that $f_i(x,z) \ge 0$ which is undetectable by i. In such a case, $i \notin \sigma(\hat{x},\hat{z})$ but $i \in \sigma(x,z)$ meaning we cannot enforce $\sigma(x,z) = \sigma(\hat{x},\hat{z})$. Thus, we explore the effects on the evolution of V when there is such a mismatch in the modes.

Proposition VI.2 (A bound when there is a mode switch). *Suppose that $(\hat{x},\hat{z}) \in \mathbb{R}^n_{\ge 0} \times \mathbb{R}^m$ is such that $i \notin \sigma(\hat{x},\hat{z})$ for some $i \in \{1,\dots,n\}$ and let $(x(t),z(t))$ be the solution to*

$$(\dot{x},\dot{z}) = F(\hat{x},\hat{z}),$$

from $(\hat{x},\hat{z})$. Let $T > 0$ be the minimum time such that $i \in \sigma(x(T),z(T))$. Then, for any $\nu > 0$, and all $T \le t < \frac{\nu}{2\sqrt{2}}$

$$f_i(x(t),z(t))^2 \le \nu^2 f(\hat{x},\hat{z})^T I_{\sigma(\hat{x},\hat{z})\cap\mathcal{N}_i^x} f(\hat{x},\hat{z}) + \nu^2 (A\hat{x}-b)^T I_{\mathcal{N}_i^z}(A\hat{x}-b).$$

The important consequence of the above result is that, for an appropriately chosen value of ν, the negative terms in $\mathcal{L}_F V$ can compensate (8c) if even though $i \in \sigma(x,z) \setminus \sigma(\hat{x},\hat{z})$ for some time. This motivates the following trigger sets which cause neighbors to send synchronized broadcasts periodically to an agent if its state is zero. First, if an agent's state is zero and it has not sent a broadcast for τ_i time, it triggers a broadcast to notify its neighbors that it requires new states,

$$\mathcal{T}_i^{\text{request}} := \begin{cases} \{\xi : x_i = 0 \text{ and } s_i \ge \tau_i\}, & \text{if } i \le n, \\ \emptyset, & \text{if } i \ge n+1. \end{cases}$$

On the receiving end, if i receives a broadcast from neighbor j with $\hat{x}_j = 0$, then it should also broadcast immediately,

$$\mathcal{T}_i^{\text{send}} := \{\xi : \exists j \in \mathcal{N}_i^x \text{ s.t. } \hat{x}_j = 0 \text{ and } r_{i\leftarrow j} \ge 0\}.$$

B. Accounting for asynchronism

In this section we introduce an additional trigger set so as to prevent non-persistently flowing solutions occurring due to the asynchronism of state broadcasts. The intuition behind the additional trigger is the following. If an agent broadcasts its state and in turn receives a state broadcast from a neighbor faster than some tolerated rate, that agent should broadcast its state immediately again. The effect of this trigger is that, if broadcasts start occurring too frequently in the network, neighboring agents' broadcasts will synchronize. Formally,

$$\mathcal{T}_i^{\text{synch}} := \{\xi : \exists j \in \mathcal{N}_i \text{ s.t. } 0 \le r_{i\leftarrow j} \le r_i^{\min}\}.$$

The threshold $r_i^{\min}$ is designed to be small. Finally, we define

$$\mathcal{T}_i := \mathcal{T}_i^e \cup \mathcal{T}_i^0 \cup \mathcal{T}_i^{\text{request}} \cup \mathcal{T}^{\text{send}} \cup \mathcal{T}_i^{\text{synch}}. \tag{11}$$

We now state the main convergence result of this paper.

Theorem VI.3 (Distributed triggers - convergence and persistently flowing solutions). *Suppose $0 < \mu_i \le \frac{1}{180}$ and*

$$0 < r_i^{\min} \le \tau_i \le \frac{1}{6\sqrt{2|N_i|\max_{j\in N_i}|N_j|}} \le s_i^{\max},$$

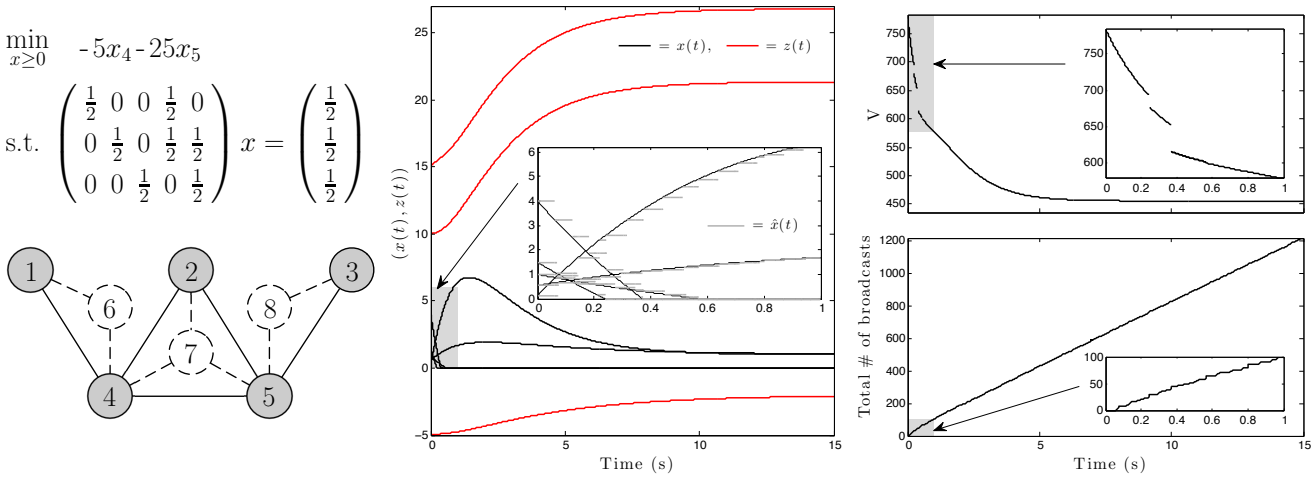
for each $i \in \{1,\dots,n+m\}$. Let $\psi(t,j)$ be a solution of (10) *with each $\mathcal{T}_i$ defined as in* (11)*. Then,*

(i) *if ψ is persistently flowing, there exists a point $(x_*,z) \in \mathcal{X} \times \mathbb{R}^m$ such that,*

$$(x(t,j),z(t,j)) \to (x_*,z') \quad \text{as} \quad (t+j) \xrightarrow{\substack{(t,j)\in \\ \text{dom}\,\psi}} \infty,$$

(ii) *if there exists a $\tau_p > 0$ such that, for any time $(t',j') \in \text{dom}\,\psi$ where $\psi(t',j') \in \mathcal{T}_i^0$ for some i it holds that $\psi(t,j) \notin \mathcal{T}_i^0$ for all $(t,j) \in ([t',t'+\tau_p]\times\mathbb{N}) \cap \text{dom}\,\psi$, the solution ψ is persistently flowing.*

The meaning of (ii) in the above result is that triggers $\mathcal{T}_i^e, \mathcal{T}_i^\tau, \mathcal{T}_i^{\text{request}}, \mathcal{T}_i^{\text{send}}, \mathcal{T}_i^{\text{synch}}$ cannot be the cause of non-persistently flowing solutions. If we had used (2) in our derivation instead of (4), the resulting design would not have enjoyed this attribute. In our experience, τ_P (as defined in (ii) above) is always ∞. Thus, we conjecture that all solutions to (10) are persistently flowing. Figure 1 shows simulation results, which support our claim.



(a) Linear program and network model (b) State trajectories (c) Evolution of V and number of broadcasts

Fig. 1. Simulation results of agents implementing (10) to solve the linear program in (a). Virtual agents are denoted by transparent nodes. The state trajectories are shown in (b) with an inlay displaying the transient response in detail. This inlay also shows the evolution of the broadcast states, $\hat{x}$ in grey. The aggregate of the agents' states converge to the solution $\mathcal{X} = (1, 0, 0, 0, 1)$. The function V is discontinuous but decreasing as evidenced in (c). The accumulation of broadcasts appears linear, suggesting that executions are persistently flowing. Each inlay shows the first second of the simulation in detail.

VII. Conclusions and Future Work

We have studied the design of distributed algorithms for networks of agents that seek to collectively solve linear programs in standard form. Our algorithmic solution has agents executing a distributed continuous-time dynamics and deciding in an opportunistic and autonomous way when to broadcast updated state information to their neighbors. Our design methodology combines elements from switched and hybrid systems, event-triggered control, and stability theory. We have rigorously characterized the asymptotic convergence of persistently flowing executions to a solution of the linear program. Based on a sufficient condition for executions to be persistently flowing, we conjecture that they all are. Future work will be devoted to establish that all solutions are persistently flowing, study the trade-off between the number of communication events and the rate of convergence, characterize the algorithm robustness against disturbances, extend our results to event-triggered implementations of general switched systems, and experimental implementations on a multi-agent testbed.

Acknowledgments

This research was supported by Award FA9550-10-1-0499 and NSF Award ECCS-1307176.

References

[1] S. Samar, S. Boyd, and D. Gorinevsky, "Distributed estimation via dual decomposition," in *European Control Conference*, (Kos, Greece), pp. 1511–1516, July 2007.

[2] E. Wei and A. Ozdaglar, "Distributed alternating direction method of multipliers," in *IEEE Conf. on Decision and Control*, (Maui, HI), pp. 5445–5450, 2012.

[3] B. Johansson, T. Keviczky, M. Johansson, and K. H. Johansson, "Subgradient methods and consensus algorithms for solving convex optimization problems," in *IEEE Conf. on Decision and Control*, (Cancun, Mexico), pp. 4185–4190, 2008.

[4] M. Zhu and S. Martínez, "An approximate dual subgradient algorithm for distributed non-convex constrained optimization," *IEEE Transactions on Automatic Control*, vol. 58, no. 6, pp. 1534–1539, 2013.

[5] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[6] D. P. Bertsekas and D. A. Castañón, "Parallel synchronous and asynchronous implementations of the auction algorithm," *Parallel Computing*, vol. 17, pp. 707–732, 1991.

[7] D. Feijer and F. Paganini, "Stability of primal-dual gradient dynamics and applications to network optimization," *Automatica*, vol. 46, pp. 1974–1981, 2010.

[8] B. Gharesifard and J. Cortés, "Distributed continuous-time convex optimization on weight-balanced digraphs," *IEEE Transactions on Automatic Control*, vol. 59, no. 3, pp. 781–786, 2014.

[9] M. Burger, G. Notarstefano, F. Bullo, and F. Allgower, "A distributed simplex algorithm for degenerate linear programs and multi-agent assignment," *Automatica*, vol. 48, no. 9, pp. 2298–2304, 2012.

[10] D. Richert and J. Cortés, "Robust distributed linear programming," *IEEE Transactions on Automatic Control*, 2013. Submitted. Available at `http://carmenere.ucsd.edu/jorge`.

[11] W. P. M. H. Heemels, K. H. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," in *IEEE Conf. on Decision and Control*, (Maui, HI), pp. 3270–3285, 2012.

[12] X. Wang and M. D. Lemmon, "Event-triggering in distributed networked control systems," *IEEE Transactions on Automatic Control*, vol. 56, no. 3, pp. 586–601, 2011.

[13] M. Mazo Jr. and P. Tabuada, "Decentralized event-triggered control over wireless sensor/actuator networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 10, pp. 2456–2461, 2011.

[14] P. Wan and M. D. Lemmon, "Event-triggered distributed optimization in sensor networks," in *Symposium on Information Processing of Sensor Networks*, (San Francisco, CA), pp. 49–60, 2009.

[15] S. S. Kia, J. Cortés, and S. Martínez, "Distributed convex optimization via continuous-time coordination algorithms with discrete-time communication," *Automatica*, 2014. Submitted.

[16] M. Zhong and C. G. Cassandras, "Asynchronous distributed optimization with event-driven communication," *IEEE Transactions on Automatic Control*, vol. 55, no. 12, pp. 2735–2750, 2010.

[17] D. Liberzon, *Switching in Systems and Control*. Systems & Control: Foundations & Applications, Birkhäuser, 2003.

[18] J. P. Hespanha, "Uniform stability of switched linear systems: Extensions of LaSalle's Invariance Principle," *IEEE Transactions on Automatic Control*, vol. 49, no. 4, pp. 470–482, 2004.

[19] R. Goebel, R. G. Sanfelice, and A. R. Teel, *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. Princeton University Press, 2012.

[20] O. L. Mangasarian and R. R. Meyer, "Nonlinear perturbation of linear programs," *SIAM Journal on Control and Optimization*, vol. 17, no. 6, pp. 745–752, 1979.