Technical Report: A Totally Asynchronous Algorithm for Tracking Solutions to Time-Varying Convex Optimization Problems

Gabriel Behrendt and Matthew Hale*

Abstract—This paper presents a decentralized algorithm for a team of agents to track time-varying fixed points that are the solutions to time-varying convex optimization problems. The algorithm is first-order, and it allows for total asynchrony in the communications and computations of all agents, i.e., all such operations can occur with arbitrary timing and arbitrary (finite) delays. Convergence rates are computed in terms of the communications and computations that agents execute, without specifying when they must occur. These rates apply to convergence to the minimum of each individual objective function, as well as agents’ long-run behavior as their objective functions change. Then, to improve the usage of limited communication and computation resources, we optimize the timing of agents’ operations relative to changes in their objective functions to minimize total fixed point tracking error over time. Simulation results are presented to illustrate these developments in practice and empirically assess their robustness to uncertainties in agents’ update laws.

I. INTRODUCTION

Time-varying convex optimization problems arise in machine learning, signal processing, robotics, power systems, and others [1]–[5]. Such problems can model, for example, time-varying demands in power systems [6], and uncertain, dynamic environments for real-time control of autonomous vehicles [7]. Often, these problems are solved with networks of agents, e.g., the robots that are given a time-varying task specification or a network of processors across which computations have been parallelized. Such instances then require a multi-agent algorithm to solve time-varying problems.

Many multi-agent systems face asynchrony in agents’ communications and computations. For example, asynchronous communications can result from environmental hazards that impede communications and/or interference among communication signals. Asynchronous computations onboard agents can result from clock mismatches and different computational hardware. As a result, agents may generate and share information with unpredictable timing. Asynchrony has been studied for static optimization problems in numerous ways, e.g., [8]–[12]. In particular, [13] establishes conditions for the convergence of static optimization under total asynchrony, namely communications and computations with arbitrary delays. To the best of our knowledge, total asynchrony remains unaddressed in time-varying optimization, and this paper fills this gap.

The algorithm we present uses totally asynchronous gradient descent to track time-varying fixed points that are the solutions to time-varying convex optimization problems. It is block-based, in the sense that each agent updates and communicates only a small subset of the decision variables in a problem. Due to total asynchrony, these operations can occur with arbitrary timing, which precludes the development of convergence rates that depend on delay bound parameters [14]–[17]. In this work, convergence rates are instead derived that depend upon how agents’ communications and computations are interleaved over time. These results enable the quantification of the quality of agents’ fixed point tracking in terms of what they execute, without specifying when operations must occur.

While it may be difficult to control the exact timing of all agents’ operations, it is easier for individual agents to control the sequence of their own local operations (without needing to control their exact timing). To leverage this capability, the aforementioned convergence rates are used to optimize the sequence of agents’ operations. This is done by optimizing (in a way made precise in Section V) the order of communications and computations, both relative to each other and relative to the timing of changes in objective functions. Specifically, a constrained optimization problem is formulated and solved in closed form to minimize agents’ convergence rates (through their dependence on agents’ operations) under constraints on communicating and computing.

Summary of Contributions: In summary, the contributions of this paper are:

- We show that the totally asynchronous gradient descent algorithm can track solutions to time-varying convex optimization problems (Section IV).
- We derive tracking bounds of the proposed algorithm in both the asymptotic and finite-time cases. These bounds quantify the dependence of agents’ performance upon their communications and computations, as well as problem parameters (Theorem 2 and Corollary 1).
- For problems in which agents can execute a limited number of computations and communications, we optimize the sequence of agents’ operations to achieve desired tracking error. Namely, a constrained optimization problem is solved in closed form to determine the optimal sequence of communications and computations to enforce a desired bound on fixed point tracking error (Theorems 4 and 8 and Corollary 2).
- We empirically demonstrate the robustness to noise of the totally asynchronous gradient descent algorithm in a time-varying problem (Section VI).

*Gabriel Behrendt and Matthew Hale are with the Department of Mechanical and Aerospace Engineering at the University of Florida, Gainesville, FL USA. Emails: {gbehrendt,matthewhale}@ufl.edu. This work was supported by AFOSR under grant FA9550-19-1-0169 and by ONR under grants N00014-19-1-2543 and N00014-21-1-2495.
Related Work There is a large corpus of research on multi-agent optimization, and we cite only a few representative examples. Asynchronous block-based algorithms include seminal work in [13], [18], [19] and recent developments in [20]–[24], which all consider static optimization problems. We differ by studying time-varying problems.

This paper is also closely related to a growing body of research on feedback optimization, which often includes time-varying optimization problems. Relevant work develops prediction-correction algorithms for problems with time-varying objective functions in both the centralized [25], [26] and distributed cases [27], [28]. Specifically, [27] considers an objective function that is the sum of locally available functions at each node and uses higher-order information to perform the prediction step. The present work considers objective functions of an arbitrary form and requires only first-order information. Moreover, the objective functions we consider change discretely in time and they need not be sampled from an objective function that varies continuously in time. Changes in the objective function thus cannot be predicted reliably, and we do not use a prediction step.

In [27, Section 1], it is noted that correction-only algorithms (which we consider) cannot effectively mitigate tracking error of the optimizer if the optimization problem and algorithm do not have sufficient timescale separation. In this work, some timescale separation is indeed required, in the sense that objective functions must not vary in time faster than agents can perform computations and communications (Section III provides a detailed discussion). We note that the need for timescale separation is motivated by feasibility rather than technical convenience: if objectives can change faster than agents can compute and communicate, then agents will be unable to track time-varying fixed points and we cannot hope to prove anything useful.

In [28, Remark 3], it is noted that distributed time-varying optimization by agents with different computational abilities is subject to ongoing research. To the best of our knowledge, this problem remains open, and we present a solution to it by allowing agents to both compute and communicate totally asynchronously.

The rest of the paper is organized as follows. Section II states the problems of interest. Section III states the algorithm for solving them. Then, Section IV develops convergence rates for agents’ short-term and asymptotic behavior. Section V optimizes agents’ communications and computations to improve tracking over time. After that, Section VI provides simulation results, and Section VII concludes.

Problem 1. With $N$ agents, over a time horizon $T$,
\[
\begin{align*}
\text{minimize} & \quad f(u, t) \\
\text{subject to} & \quad u \in U,
\end{align*}
\]
where $U \subseteq \mathbb{R}^n$, $f : \mathbb{R}^n \times \mathbb{N} \to \mathbb{R}$, and $t \in T$ is time.

We make the following assumptions about Problem 1.

Assumption 1. For all $t \in T$, the function $f(\cdot, t)$ is twice continuously differentiable.

Assumption 2 guarantees the existence and continuity of both the gradient and the Hessian of $f(\cdot, t)$. Many functions satisfy this assumption, such as polynomials of all orders.

Assumption 2. The constraint set $U$ can be decomposed via
\[
U = U_1 \times U_2 \times \cdots \times U_N,
\]
where $U_i \subseteq \mathbb{R}^{n_i}$, $n_i \in \mathbb{N}$, is non-empty, compact, and convex.

Assumption 2 permits constraints such as box constraints, and it implies that $U$ is also non-empty, compact, and convex.

It also ensures that each agent can update its block of decision variables independently, which will be used to develop a distributed gradient projection law for enforcing the set constraint $u \in U$. Similarly, $u \in U$ can be decomposed via
\[
u = \left(\begin{array}{c} u_1 \\ \vdots \\ u_N \end{array}\right),
\]
where $u_i \in U_i \subseteq \mathbb{R}^{n_i}$. We define
\[
p_u = [n_1, n_2, \ldots, n_N] \in \mathbb{N}^N
\]  
(1)

to be the vector of dimensions of the blocks into which $u$ is partitioned. From Assumptions 1, 2 we have that $H(\cdot, t) := \nabla^2 f(\cdot, t)$ is continuous and $U$ is compact. Therefore, $\nabla f(\cdot, t)$ is Lipschitz continuous. For each $t \in T$, we use $L_t$ to denote the bound on $\|H(\cdot, t)\|_2$ over $U$, and this $L_t$ is the Lipschitz constant of $\nabla f(\cdot, t)$ over $U$.

The next assumption uses strict block diagonal dominance of partitioned matrices. Suppose a matrix $A \in \mathbb{R}^{n \times n}$ is partitioned into blocks according to the partition vector $p_u = [n_1, \ldots, n_N]$, where $\sum_{i=1}^N n_i = n$ (we allow $n_i \neq n_j$ for all $i \neq j$). Let $[A]_{p_u}$ denote the partitioned matrix, i.e.,
\[
[A]_{p_u} = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1N} \\
A_{21} & A_{22} & \cdots & A_{2N} \\
\vdots & \vdots & \ddots & \vdots \\
A_{N1} & A_{N2} & \cdots & A_{NN} \end{bmatrix}.
\]

The $i^{th}j^{th}$ block, denoted $A_{ij} \in \mathbb{R}^{n_i \times n_j}$, is the matrix block of the rows of $[A]_{p_u}$ with indices $\sum_{k=1}^{i-1} n_k + 1$ through $\sum_{k=1}^{i} n_k$ and the columns of $[A]_{p_u}$ with indices $\sum_{k=1}^{j-1} n_k + 1$ through $\sum_{k=1}^{j} n_k$, i.e., $A_{1i}$ is the first $n_1$ columns of the first $n_1$ rows, $A_{12}$ is the next $n_2$ columns of the first $n_1$ rows, etc.

Assumption 3. The matrix $H(u, t)$ can be partitioned with respect to $p_u$ from Equation 1, and $H_{ii}(u, t)$ is symmetric.
and positive definite for all $u \in U$ and each $t \in T$, i.e., $H_{ii}(u, t) = H_{ii}(u, t)^T > 0$. Moreover, for each $t \in T$, $H(u, t)$ is $\beta_t$-strictly block diagonally dominant, i.e., for some $\beta_t > 0$

$$
\left(\|H_{ii}(u, t)^{-1}\|_2\right)^{-1} \geq \sum_{j=1, j \neq i}^n \|H_{ij}(u, t)\|_2 + \beta_t.
$$

As noted in [13, Section 6.3.2], Assumption 3 is sufficient to ensure that agents’ iterates contract towards the minimizer of $f(\cdot, t)$ (in an appropriate norm) for fixed $t$. And, in a way made precise in [13, Section 6.3.1], a problem is unlikely to be solvable by a totally asynchronous algorithm if it fails to satisfy Assumption 3. Thus, we enforce it here. One consequence of Assumption 3 is given next.

**Lemma 1.** Let Assumptions 2 and 3 hold. Then $f(\cdot, t)$ is $\beta_t$-strongly convex and it has a unique minimizer, denoted $\hat{u}^t$.

**Proof:** From [21, Lemma 4], it immediately follows that

$$
\lambda_{\min}[H(u, t)] \geq \left(\|H_{ii}(u, t)^{-1}\|_2\right)^{-1} - \sum_{j=1, j \neq i}^n \|H_{ij}(u, t)\|_2
$$

for all $u \in U$ and $t \in T$. Therefore by Assumption 3

$$
\lambda_{\min}[H(u, t)] \geq \beta_t
$$

for all $u \in U$ and $t \in T$. This implies that $f(\cdot, t)$ is $\beta_t$-strongly convex [29, Section 9.1.2], from which the uniqueness of the minimum $\hat{u}^t := \arg\min_{u \in U} f(u, t)$ follows.

Regarding changes over time, the following is assumed.

**Assumption 4.** For each $t \in T$, there exists a constant $\sigma_t \geq 0$ such that $\|\hat{u}^{t+1} - \hat{u}^t\|_2 \leq \sigma_t$.

Without Assumption 4, a problem could change by arbitrarily large amounts, and it would not be possible to track the solution of Problem 1

### III. Asynchronous Update Law

In this section we develop the totally asynchronous algorithm for tracking time-varying fixed points. First, we describe the relationship between the times at which the objective function changes and the times at which agents execute communications and computations.

#### A. Time and Timescale Separation

Problem 1 evolves over time, indexed by the discrete time index $t$, and, in defining agents’ algorithm, we will index their operations over the discrete time index $k$. The reason is that the timing of agents’ computations and communications need not coincide with changes in their objective functions. For example, agents may execute several communications and computations when minimizing $f(\cdot, t)$ for each $t \in T$. If this were not done, then the problem could evolve faster than agents compute and communicate, and tracking a time-varying fixed point would be infeasible. Thus, it is necessary to adopt a formalism that enables agents to perform multiple operations to minimize $f(\cdot, t)$ for each $t \in T$.

In particular, we require some timescale separation between the ticks of $t$, the time index for changes in the objective function, and the ticks of $k$, the index of agents’ operations. For for each $t \in T$, some number of ticks $\kappa_t \geq 1$ of $k$ elapse before $t$ increments to $t + 1$. Agents may compute and/or communicate something at each tick of $k$, though this is not guaranteed due to asynchrony. In other words, timescale separation allows for the possibility that agents perform multiple communications and computations for each $t$, though we by no means assume that they do so. As will be shown below, our analysis allows for (and quantifies the effects of) scenarios in which agents perform no operations at all. Of course, timescale separation is not new to this work, and has been used, e.g., in optimization and model predictive control in [30]–[32].

#### B. Formal Algorithm Statement

We consider a block-based gradient algorithm with totally asynchronous computations and communications to solve Problem 1 over a network of $N$ agents. Each agent updates only a subset of the entire decision vector and communicates the values of their sub-vector with the rest of the network over time. Asynchrony implies that agents receive different information at different times, and thus we expect them to have differing values for decision variables onboard. At any time $k$, agent $i \in [N]$ has a local copy of its decision vector, denoted $u^t_i(k) \in \mathbb{R}^n_i$. Due to computation and communication delays, we allow $u^t_i(k) \neq u^t_j(k)$ for $i \neq j$. Within the vector $u^t_i(k)$, agent $i$ computes updates only to $u^t_j(k) \in \mathbb{R}^{n_j}$. At any time $k$, agent $i$ has (possibly old) values for agent $j$’s sub-vector of the decision variables, denoted $u^t_j(k) \in \mathbb{R}^{n_j}$. These are updated only by agent $j$ and then shared with the rest of the network.

At time $k$, if agent $i$ computes an update to $u^t_j(k)$, its computations are in terms of the decision vector $u^t_i(k)$. The entries of $u^t_j(k)$ for $j \neq i$ are obtained by communications from agent $j$, which may be subject to delays. Therefore, agent $i$ may (and often will) compute updates to $u^t_i$ based on outdated information from other agents.

Let $K^t$ be the set of times at which agent $i$ computes an update to $u^t_i$. Similarly, let $C^t_i$ be the set of times at which agent $j$ receives a transmission of $u^t_i$ from agent $i$; due to communication delays, these transmissions can be received at any time after they are sent, and they can be received at different times by different agents. We emphasize that the sets $K^t$ and $C^t_i$ are only defined to simplify discussion and analysis. Agents do not know and do not need to know $K^t$ and $C^t_i$. We define $\tau^t_j(k)$ to be the time at which agent $j$ originally computed the value of $u^t_j$ that agent $i$ has onboard at time $k$; agents $i$ and $j$ also do not need to know $\tau^t_j$. Formally,

$$
u^t_j(k) = u^t_j(\tau^t_j(k)).
$$

Then $k - \tau^t_j(k)$ is the length of communication delay, and we emphasize that we do not assume any bound on agents’ delays. Instead, only the following is enforced.
For all $i \in [N]$, the set $K^i$ is infinite. Let $\{k_d\}_{d \in \mathbb{N}}$ be a non-decreasing sequence in $K^i$ tending to infinity. Then for all $i \in [N]$ and $j \in [N]\{i\}$, we have
\[
\lim_{d \to \infty} \tau^j_d(k_d) = \infty.
\]

And messages are received in the order they were sent.

Assumption 5 ensures that (i) no agent ever stops computing updates (because $K^i$ is infinite), (ii) no agent ever stops sharing its updates with other agents (because $\tau^j_d(k_d)$ tends to infinity), and (iii) arbitrarily old information cannot be received arbitrarily late into the run of an algorithm. This assumption is one of feasibility: there is no tracking of fixed points (and nothing to prove) if any agent ever permanently stops computing and/or communicating, or if arbitrarily old information can be received at any time.

We propose to use a gradient-based update law for agent $i$, which takes the form
\[
u^i_t(k + 1) = \left\{ \begin{array}{ll}
\Pi_{U_i} \left[ u^i_t(k) - \gamma_t \left( \nabla u_i f(u^i_t(k), t) \right) \right] & k \in K^i \\
n_i^i(k) & k \notin K^i
\end{array} \right.
\]
\[
u^j_t(k + 1) = \left\{ \begin{array}{ll}
\nu^j_t(\tau^j_t(k)) & k \in C^j_t \\
n_i^j(k) & k \notin C^j_t
\end{array} \right.
\]

where $\gamma_t > 0$ is the stepsize used when minimizing $f(\cdot, t)$. Bounds on $\gamma_t$ are established later in Proposition 1. Algorithm 1 provides pseudocode for agents’ updates; for notational convenience, we set $\kappa_{-1} = 0$.

**Algorithm 1: Totally Asynchronous Algorithm for Tracking Time-Varying Fixed Points**

| Input: $u^i_t(0)$ for all $i, j \in [N]$, $T \in \mathbb{N}$ |
| $k_{loop, -1} = -1$ |
| $k_{loop, t} = k_{loop, t-1} + \kappa_{t-1}$ |
| $k_{loop, t} = k_{loop, t-1} + \kappa_{t-1}$ |
| for $k = k_{loop, t} + 1 : k_{loop, t} + \kappa_{t}$ do |
| for $i=1:N$ do |
| for $j=1:N$ do |
| if $k \in C^j_t$ then |
| $u^j_t(k + 1) = u^j_t(\tau^j_t(k))$ |
| end |
| else |
| $u^j_t(k + 1) = u^j_t(k)$ |
| end |
| end |
| end |
| end |

The remainder of the paper focuses on analyzing and optimizing the execution of Algorithm 1.

**IV. CONVERGENCE RESULTS**

This section establishes error bounds for Algorithm 1 by first deriving convergence rates for time-invariant problems. Then, those results are extended to the time-varying case to bound error when tracking time-varying fixed points. First, we define block-maximum norms.

**Definition 1 (Block-maximum norm).** Let $u \in \mathbb{R}^n$ be partitioned according to $p_u$ as in Equation 1. Then define
\[
\|u\|_{2,\infty} := \max_{i \in [N]} \|u_i\|_2.
\]

This definition allows for the quantification of the worst-performing agent in the network, which will be considered in the convergence analysis.

For a time-invariant objective function, it has been shown in [13, Section 6.2, Prop. 2.1] that if one can construct a sequence of sets $\{U(s)\}_{s \in \mathbb{N}}$ that satisfy the properties of Definition 2 below, then Algorithm 1 asymptotically converges to a minimizer of that objective function. We will leverage that analysis for each fixed $t$ and each fixed $f(\cdot, t)$ individually. Then we will use these analyses to bound tracking error when $f$ evolves over time. For any fixed $t$, we have the following.

**Definition 2.** Fix $t \in T$ and fix $f(\cdot, t)$. A collection of sets $\{U(s)\}_{s \in \mathbb{N}}$ is called admissible if it satisfies:

1) For all $s \in \mathbb{N}$, $U(s)$ can be decomposed as $U(s) = U_1(s) \times \cdots \times U_N(s)$, where $U_i(s) \subseteq U_i$.
2) $U(s + 1) \subseteq U(s)$ for all $s \in \mathbb{N}$
3) $\lim_{s \to \infty} U(s) = \{\hat{u}^i\}$
4) For all $y \in U(s)$ and $i \in [N]$, the update $z_i = \Pi_{U_i} [y_i - \gamma \nabla y_i f(y, t)]$ satisfies $z_i \in U_i(s + 1)$.

Definition 2 allows for the agents to independently update their local sub-vectors asynchronously without violating set constraints. Definitions 2 and 2.3 guarantee that the collection of sets $\{U(s)\}_{s \in \mathbb{N}}$ is nested and converges to a singleton containing $\hat{u}^i$. Definition 2.4 guarantees that each update of an agent’s sub-vector makes progress toward $\hat{u}^i$.

In the analysis of Proposition 1 below, we fix a $t \in T$ in order to only analyze the convergence of Algorithm 1 for $f(\cdot, t)$. After proving convergence to the optimizer of a fixed $f(\cdot, t)$, we will extend these results to the time-varying case. We introduce the symbol $\eta_t \in \mathbb{N}$ for this analysis, and it denotes the total number of ticks of $k$ that elapse before $t$ increments to $t + 1$, i.e., $\eta_t = \sum_{t=0}^{\infty} \kappa_t$.

**Proposition 1.** Let Assumptions 1-4 hold. Fix $t \in T$ and fix $f(\cdot, t)$. Consider the collection $\{U(s)\}_{s \in \mathbb{N}}$ with
\[
U(s) = \left\{ y \in \mathbb{R}^n : \|y - \hat{u}^i\|_{2,\infty} \leq q^\infty \max_{i \in [N]} \|u^i(\eta_t - 1) - \hat{u}^i\|_{2,\infty} \right\},
\]
where the stepsize $\gamma_t > 0$ satisfies

$$\gamma_t < \frac{1}{\max\max_{u \in U} \sum_{i = 1}^{n} \|H_{ij}(u, t)\|_2},$$

and $q_t := \max\{1 - \gamma_t \beta_t, [1 - \gamma_t L_t]\}$, which satisfies $q_t \in (0, 1)$. This collection is admissible in the sense of Definition 3.

**Proof:** See Appendix A.

This construction of sets will be used to derive a convergence rate in terms of the number of communication cycles agents complete, defined next.

**Definition 3** ([33, Definition 4]). A communication cycle has occurred after (a) for all $i$, agent $i$ has computed at least one update to $x_i^t$, and then (b) that updated value has been sent to and received by the other agents that need it. The number of communication cycles started and finished when minimizing $f(\cdot, t)$ before switching to $f(\cdot, t+1)$ is denoted by $c(t) \in \mathbb{N}$.

Note that some agents may compute and share many updates within one cycle because a cycle is only completed after all agents have computed and shared at least one update.

The following constants are used in the forthcoming analysis:

$$q_{\max} := \max_{t \in \mathcal{T}} q_t \tag{2}$$

$$D_0 := \max_{i \in [N]} \|u^i(0) - \hat{u}^0\|_{2,\infty} \tag{3}$$

$$D_t := q_t^{c(t-1)} D_{t-1} + \sigma_t, \quad \text{for all } t \geq 1$$

$$B := \max \left\{ \max_{t \in \mathcal{T}} \sigma_t, \quad D_0 \right\}.$$

We first analyze convergence for $f(\cdot, 0)$ held fixed, then iterate this analysis forward in time to account for time-varying problems. The convergence rate for $f(\cdot, 0)$ is as follows.

**Theorem 1** (Time-Invariant Convergence Bound). Let Assumptions 4 and 5 hold. Then, for $N$ agents executing Algorithm 7 just before $f$ switches from $f(\cdot, 0)$ to $f(\cdot, 1)$, for all $i \in [N]$ we have

$$\|u^i(\eta_t) - \hat{u}^t\|_{2,\infty} \leq q_0^{c(0)} D_0. \tag{4}$$

**Proof:** This follows from applying [21, Theorem 1] to $f(\cdot, 0)$ when agent $i$ has the initial condition $u^i(0)$. ■

Using Theorem 1 the next result accounts for changes in the objective function and bounds tracking error across finite time horizons under such changes.

**Theorem 2** (Finite-Time Tracking Bound). Let Assumptions 4 and 5 hold. Then, for any $t \in \mathcal{T}$ and $\eta_t = \sum_{t=0}^{t} \kappa_t$, the iterates produced by $N$ agents executing Algorithm 7 satisfy

$$\|u^i(\eta_t) - \hat{u}^t\|_{2,\infty} \leq D_0 \Pi_{\theta=0}^{t} q_{\max}^{c(\theta)} + \sum_{p=1}^{t} \sigma_{p-1} \Pi_{r=p}^{t} q_{\max}^{c(r)}. \tag{5}$$

**Proof:** For all $i \in [N]$, Theorem 1 gives

$$\|u^i(\eta_t) - \hat{u}^0\|_{2,\infty} \leq q_0^{c(0)} D_0.$$

After the objective function changes from $f(\cdot, 0)$ to $f(\cdot, 1)$ (and before the next tick of $k$), the tracking error for agent $i$ can be written as

$$\|u^i(\eta_t) - \hat{u}^t\|_{2,\infty} \leq \|u^i(\eta_t) - \hat{u}^0\|_{2,\infty} + \|\hat{u}^1 - \hat{u}^0\|_{2,\infty}$$

$$\leq q_0^{c(0)} D_0 + \sigma_0,$$

which follows from the triangle inequality, Theorem 1 Assumption 4 and the fact that $\|v\|_{2,\infty} \leq \|v\|_{2}$ for $v \in \mathbb{R}^n$. Next, $\kappa_1$ ticks of $k$ will elapse before $f(\cdot, 1)$ switches to $f(\cdot, 2)$, during which time agents will complete some number of cycles, denoted $c(1)$. Then, just before $f(\cdot, 1)$ switches to $f(\cdot, 2)$, for all $i \in [N]$ we find

$$\|u^i(\eta_t) - \hat{u}^t\|_{2,\infty} \leq q_1^{c(1)} \left(q_0^{c(0)} D_0 + \sigma_0\right).$$

Applying this recursively to all $t \in \mathcal{T}$ completes the proof. ■

Increasing the number of communication cycles performed to minimize $f(\cdot, t)$, namely $c(t)$, will decrease the value of $q_t^{c(0)}$, shrinking the error on the right-hand side of Equation 4. Therefore, the completion of more communication cycles will lead to better tracking of the minimizer $\hat{u}^t$. Indeed, the long-term tracking error depends on the value of $c(t)$ for every previous $t \in \mathcal{T}$, which is intuitive because cycles count agents’ total operations over time.

The following result bounds tracking error as the length of time horizon $T$ grows arbitrarily large. It assumes that $c(t) \geq 1$, i.e., that agents complete at least one communication cycle per objective function. Without such an assumption, it is possible for agents to perform no computations and communications, and no tracking would be possible.

**Corollary 1** (Asymptotic Tracking Bound). Let Assumptions 4 and 5 hold and suppose that $N$ agents are executing Algorithm 7 with $c(t) \geq 1$ for all $t$. Then

$$\lim_{t \to \infty} \|u^i(\eta_t) - \hat{u}^t\|_{2,\infty} \leq B \frac{q_{\max}}{1 - q_{\max}}.$$

**Proof:** Equations 2 and 3 imply that

$$B \geq D_0, \quad B \geq \sigma_t \quad \text{for all } t \in \mathcal{T}, \quad \text{and} \quad q_{\max} \geq q_1 \quad \text{for all } t \in \mathcal{T}.$$

Then the result in Theorem 2 can be upper bounded via

$$\|u^i(\eta_t) - \hat{u}^t\|_{2,\infty} \leq B \Pi_{\theta=0}^{t} q_{\max}^{c(\theta)} + \sum_{p=1}^{t} B \Pi_{r=p}^{t} q_{\max}^{c(r)}$$

$$= B \left(\Pi_{\theta=0}^{t} q_{\max}^{c(\theta)} + \sum_{p=1}^{t} \Pi_{r=p}^{t} q_{\max}^{c(r)}\right).$$
With \( c(t) \geq 1 \) for all \( t \in \mathcal{T} \), we have
\[
\|u^t(\eta_t) - \hat{u}^t\|_{2,\infty} \leq B \left( \prod_{\theta=0}^{t} q_{\max}^{\theta} + \sum_{p=1}^{t} \prod_{r=p}^{t} q_{\max}^{r} \right).
\]

The results follows by taking the limit as \( t \to \infty \) and using the fact \( \sum_{k=1}^{\infty} r^k = \frac{r}{r^2} \) for \( |r| < 1 \).

**Remark 1.** A similar result to Corollary 7 can be found in [28, Theorem 2], which considers a synchronous model of computation. Corollary 1 thus shows that, under mild conditions, one can achieve similar tracking performance with a computational model that permits total asynchrony in computations and communications among agents. Thus, total asynchrony can be permitted without sacrificing performance.

Resource-constrained agents may only be able to complete a limited number of total communications and computations, e.g., due to limited fuel or battery power onboard a mobile robot. The long-run dependence upon all values of \( c(t) \) then leads to the question of how communications and computations should be interleaved over time to tune the values of each \( c(t) \) to lead to optimal tracking overall. This is the subject of the next section.

**V. Cycle Optimization**

In this section, we determine when agents should compute and communicate in order to minimize the error bound in Theorem 2. In particular, we optimize the number of communication cycles \( c(t) \) that agents should perform when minimizing \( f(\cdot, t) \). This is done under constraints on the number of communications and computations that agents can perform, which can represent limits on total energy and/or bandwidth. This is done for two cases: (i) when \( c(t) \equiv c \), i.e., when the number of cycles is the same for all \( t \), and (ii) when the number of cycles can differ for each value of \( t \). We emphasize that the completion of cycles is determined by the order of agents’ operations, and each agent can locally control the sequence of its own operations. Hence, cycles can be optimized without making assumptions about controlling the exact timing of agents’ operations.

**A. Optimizing over Constant \( c(t) \)**

Suppose we wish to drive agents’ iterates within a ball centered on the minimizer, \( \hat{u}^t \), for all \( t \in \mathcal{T} \). In particular, for all \( t \), we would like agents’ tracking error to be bounded by a given \( \rho > 0 \) before the objective function changes from \( f(\cdot, t) \) to \( f(\cdot, t+1) \). This is first done for \( c(t) \equiv c \) for all \( t \in \mathcal{T} \). This corresponds to the case in which agents are able to do the same amount of communicating and computing for each objective function, as could happen if changes in the objective function were evenly spaced in time.

**Theorem 3** (Finite Time Communication Cycle Requirement). Let Assumptions 1-5 hold and let \( \rho > 0 \). Suppose that \( N \) agents are executing Algorithm 1 with \( c(t) \equiv c \) for all \( t \in \mathcal{T} \). If
\[
c \geq \frac{\ln \left( \frac{q_{\max}^{(T+2)c} - \rho}{B q_{\max}^c} \right)}{\ln (q_{\max}^c)}.
\]
then \( \|u^t(\eta_t) - \hat{u}^t\|_{2,\infty} \leq \rho \) for all \( t \in \mathcal{T} \).

**Proof:** From Theorem 2 we have
\[
\|u^t(\eta_t) - \hat{u}^t\|_{2,\infty} \leq D_0 \prod_{\theta=0}^{t} q_{\max}^{\theta} + \sum_{p=1}^{t} \prod_{r=p}^{t} q_{\max}^{r},
\]
where \( \eta_t = \sum_{\ell=0}^{t} \kappa_{\ell} \). For \( c(t) \equiv c \), this bound simplifies to
\[
\|u^t(\eta_t) - \hat{u}^t\|_{2,\infty} \leq D_0 \prod_{\theta=0}^{t} q_{\max}^{\theta} + \sum_{p=1}^{t} \prod_{r=p}^{t} q_{\max}^{c}.
\]
Equation 5 can be upper bounded via
\[
\|u^t(\eta_t) - \hat{u}^t\|_{2,\infty} \leq \left( B \prod_{\theta=0}^{t} q_{\max}^{\theta} + \sum_{p=1}^{t} \prod_{r=p}^{t} q_{\max}^{c} \right) B q_{\max}^c \sum_{\theta=0}^{t} \frac{\theta c}{q_{\max}^c},
\]
where the first line follows from the definition of \( B \), the second line expands the sum and products, and the third line combines like terms. To bound the left-hand side by \( \rho \), it is sufficient to have
\[
B q_{\max}^c \sum_{\theta=0}^{t} \frac{\theta c}{q_{\max}^c} \leq \rho. \tag{6}
\]
From the fact that \( \sum_{m=0}^{n-1} a_m n^r = a \left( \frac{r^{n+1}}{r+1} \right) \) when \( r < 1 \), the bound in Equation 6 holds if and only if
\[
B q_{\max}^c \left( \frac{(t+1)c}{q_{\max}^c} - 1 \right) \leq \rho.
\]
Setting \( t = T \) maximizes the left-hand side, and then solving for \( c \) provides a bound that holds for all \( t \in \mathcal{T} \).

Extending this bound to the asymptotic case gives the following.

**Corollary 2** (Asymptotic Communication Cycle Requirement). Let Assumptions 1,2 hold and let \( \rho > 0 \). Suppose that \( N \) agents are executing Algorithm 1 with \( c(t) \equiv c \) for all \( t \in \mathcal{T} \). If
\[
c \geq \frac{\ln \left( \frac{1}{1+c B^c} \right)}{\ln (q_{\max}^c)} \tag{7}
\]
then \( \|u^t(\eta_t) - \hat{u}^t\|_{2,\infty} \leq \rho \) for all \( t \in \mathbb{N} \).

**Proof:** In Theorem 2 use \( \lim_{T \to \infty} q_{\max}^{(T+2)c} = 0 \).
B. Optimizing over Time-Varying \( c(t) \)

Lastly, we optimize the number of communication cycles the network should complete for each objective function to minimize dynamic regret subject to a constraint on the number of cycles agents can complete. This can be modeled, for example, the limits of finite battery life upon the operations an agent can execute over the time horizon \( T \). Although exact tracking error is not known in general, we can minimize the bound on tracking error in Equation (4) because that bound is computable, and this is what we will do. In particular, we solve the following optimization problem to calculate the optimal number of cycles to complete for \( f(\cdot, t) \), denoted \( c^*(t) \).

**Problem 2.** For \( N \) agents running Algorithm 2 over a time horizon \( T \),

\[
\begin{align*}
\min_{\{c(t)\}_{t=0}^{T}} & \sum_{t=0}^{T} q_i^{(t)} D_t \\
\text{subject to} & \sum_{t=0}^{T} c(t) \leq K,
\end{align*}
\]

where \( K \in \mathbb{N} \) is the total number of communication cycles agents can complete over the time horizon \( T \).

Problem 2 is a constrained convex optimization problem whose solution is derived next.

**Theorem 4 (Optimal Number of Communication Cycles).** Let Assumptions 15 hold, and define

\[
F_t := \sum_{t=0}^{T} \left( \prod_{i=0}^{t-1} T_{j}^{(i)} \prod_{j=t+1}^{\theta} q_j^{(j)} \right) q_i^{(t)} D_t
\]

\[
E_t := \sum_{t=0}^{T} \sigma_i \sum_{j=1}^{T-t+1} \prod_{k=t+1}^{T-j+1} q_k \prod_{l=t+1}^{T-j-1} q_l.
\]

Then, for each \( t \in T \), the optimal number of cycles for \( N \) agents executing Algorithm 2 to complete when minimizing \( f(\cdot, t) \) is

\[
c^*(t) = \ln \left( \frac{\mu^*}{\ln (q_i)} \right).
\]

The value of \(\mu^*\) can be computed along with \( c^*(t) \) by solving the system of equations we derive at the end of the following proof.

**Proof:** Problem 2’s objective function is twice continuously differentiable in each \( c(t) \). And, for each \( c(t) \), we find

\[
\frac{\partial^2}{\partial c(t)^2} q_i^{(t)} D_t = \left( \ln (q_i) \right)^2 c_i^{(t)} D_t > 0,
\]

and thus the objective function in Problem 2 is strictly convex. The Lagrangian associated with Problem 2 is

\[
L(c(t), \mu) = \sum_{t=1}^{T} q_i^{(t)} D_t + \mu \left( \sum_{t=1}^{T} c(t) - K \right),
\]

where \(\mu \geq 0\) is the dual variable. Because the constraint is affine, it satisfies the Linearity Constraint Qualification. Combined with the strict convexity of the cost, this implies that the Karush-Kuhn-Tucker (KKT) conditions are sufficient to find the unique minimizer for Problem 2.

To solve for the optimal primal-dual pair \((c^*(t), \mu^*)\), we use the KKT first-order optimality conditions, \(\nabla_c p(c^*(t), \mu^*) = 0\) and \(\nabla_{\mu} L(c^*(t), \mu^*) = 0\). For a fixed \( t \in T \), computing the derivatives of \( L(c(t), \mu) \) with respect to \( c(t) \) and \(\mu\) gives

\[
\frac{\partial L}{\partial c(t)} = q_i^{(t)} \ln(q_i) [D_0 F_t + E_t] + \mu
\]

\[
\frac{\partial L}{\partial \mu} = -\sum_{t=0}^{T} c(t) - K.
\]

Using the KKT first-order optimality conditions, we set these derivatives equal to 0 for each \( t \in T \) to find

\[
0 = q_0^{c^*(0)} \ln(q_0) [D_0 F_0 + E_0] + \mu^*
\]

\[
0 = q_T^{c^*(T)} \ln(q_T) [D_T F_T + E_T] + \mu^*
\]

\[
0 = \sum_{t=0}^{T} c^*(t) - K.
\]

This is a system of \( T + 2 \) equations in \( T + 2 \) unknowns (namely, \( c^*(t) \) for \( t \in T \) and \( \mu^* \in \mathbb{R} \)). Solving for \( c^*(t) \) and \(\mu^* \) completes the proof.

\[\square\]

VI. NUMERICAL EXAMPLE

This section considers two classes of simulations. First, two time-varying quadratic programs are considered to illustrate the results in Section [V]. Second, because the results of this paper are motivated in part by ongoing work in feedback optimization (see the Introduction for discussion), the performance of Algorithm 1 on a noisy feedback optimization problem is evaluated empirically.

A. Time-Varying Quadratic Programs

We consider two examples with \( N = 15 \) agents executing Algorithm 1 as their objective switches 10 times. The objective function takes the form

\[
f(u, t) = \frac{1}{2} u^T Q(t) u + q(t)^T u,
\]

where \( 0 \prec Q(t) = Q^T(t) \in \mathbb{R}^{n \times n} \) and \( q(t) \in \mathbb{R}^n \) vary over \( t \in T \). Both \( Q(t) \) and \( q(t) \) are randomly generated in this example. At each tick of \( k \), agent \( i \) communicates with others with probability \( p_i(k) \in (0, 1) \), for which a new value is generated randomly from a uniform distribution at each \( k \). The objective function switches every 50 ticks of \( k \). Figure 1 shows how the distance between the current iterate \( \bar{u}^t(k) \) and the unique minimizer \( \bar{u}^t \) changes for all \( i \in [N] \) and for each \( t \in T \). It can be seen that the individual error of each agent decreases as more iterations (and therefore more communication cycles) occur. Figure 2 is a zoomed-in plot of all agents’ errors, and it shows the effects of total asynchrony on the convergence of Algorithm 1. In particular, the areas of descent in Figure 2 are due to agents computing...
and communicating, leading to completed communication cycles and hence progress towards fixed points. Conversely, flat areas are where agents either do not compute or communicate, thereby completing no cycles and making no progress in reducing tracking error.

After every 50 iterations there is a sudden increase in error because those are the times at which the objective function switches from $f(\cdot, t)$ to $f(\cdot, t + 1)$. Thus, the latest copy of each agents’ decision vector becomes the initial guess of the minimum for $f(\cdot, t + 1)$. Because $\hat{u}^{t+1} \neq \hat{u}^t$, this initial guess is typically far away, leading to an abrupt increase in error at switching times.

**B. Feedback Optimization with Measurement Noise**

Secondly, we consider an example from the study of feedback optimization [34]–[36] in which the cost is on the deviation of the output of a dynamical system from a reference trajectory. Consider the objective function

$$f(u, t) = \frac{1}{2} u^T Q(t) u + \frac{1}{2} (y(u) - r(t))^2,$$

where $Q(t) \in \mathbb{R}^{n \times n}$, $y : \mathbb{R}^n \to \mathbb{R}$ is the input-output map of the system, and $r(t) \in \mathbb{R}$ is the time-varying reference signal that $y$ must track. In feedback optimization we only have knowledge of an output measurement affected by noise. Therefore, the objective function that we can compute is

$$f(u, t) = \frac{1}{2} u^T Q(t) u + \frac{1}{2} (\hat{y}(u) - r(t))^2,$$

where $\hat{y}(u) = y(u) + n(t)$ is the noisy measurement of the input-output map and $n(t) \sim \mathcal{N}(0, 1000^2)$ is i.i.d.

Figure 3 shows that, even with measurements corrupted by noise, Algorithm 1 can still track time-varying fixed points to within a ball centered on them. For example, just before the 8th switch, agents’ average tracking error is 1.192, which indicates close tracking even under noise. Similar to the previous example, the switching of objective functions causes a noticeable error that is quickly mitigated by subsequent communication cycles. This example goes beyond the analytical tracking guarantees provided above, and further characterization of Algorithm 1 with measurement noise is a direction for future research.

**VII. CONCLUSIONS**

In this paper, we have proposed a totally asynchronous multi-agent algorithm for tracking time-varying fixed points that are solutions to time-varying optimization problems. We further optimized agents’ computations and communications to minimize dynamic regret when they are able to perform only a limited number of each type of operation. Next steps in this work include integrating the dynamics of agents into the update law for feedback optimization and analytically bounding tracking error when measurement noise is included in a feedback optimization setup.

**REFERENCES**

[1] S. M. Fosson, “Online optimization in dynamic environments: a regret analysis for sparse problems,” in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 7225–7230.
We prove that each of the four properties of Definition 2 is satisfied. First, consider the sets

\[ U_i(s) = \{ y_i \in \mathbb{R}^N : \| y_i - \bar{u}_i \|_2 \leq q_i \max_{i \in [N]} \| u^i(\eta_{t-1}) - \bar{u}_i \|_{2,\infty} \} . \]

Then, for \( y = (y_1, \ldots, y_N)^T \in U_1(s) \times \cdots \times U_N(s) \), we have \( N \) inequalities

\[ \| y_1 - \bar{u}_1 \|_2 \leq q_1 \max_{i \in [N]} \| u^i(\eta_{t-1}) - \bar{u}_1 \|_{2,\infty} , \]

\[ \vdots \]

\[ \| y_N - \bar{u}_N \|_2 \leq q_N \max_{i \in [N]} \| u^i(\eta_{t-1}) - \bar{u}_N \|_{2,\infty} ; \]

from which it follows that

\[ \| y - \bar{u} \|_{2,\infty} := \max_{i \in [N]} [ \| y_i - \bar{u}_i \|_2 \leq q_i \max_{i \in [N]} \| u^i(\eta_{t-1}) - \bar{u}_i \|_{2,\infty} ] ; \]

Thus \( y \in U_1(s) \times \cdots \times U_N(s) \) implies \( y \in U(s) \) and

\[ U_1(s) \times U_2(s) \times \cdots \times U_N(s) \subseteq U(s). \]
Now, consider $y \in U(s)$. Then,

$$\|y - \hat{u}^t\|_{2,\infty} \leq q_t^* \max_{i \in [N]} \|u^i(\eta_{t-1}) - \hat{u}^t\|_{2,\infty}$$

and

$$\max_{i \in [N]} \|y_i - \hat{u}^t_i\|_2 \leq q_t^* \max_{i \in [N]} \|u^i(\eta_{t-1}) - \hat{u}^t\|_{2,\infty}. \tag{8}$$

This implies that, for all $i$, we have $\|y_i - \hat{u}^t_i\|_2 \leq q_t^* \max_{i \in [N]} \|u^i(\eta_{t-1}) - \hat{u}^t\|_{2,\infty}$. Then $y \in U(s)$ implies that $y_i \in U_i(s)$ for all $i$, and thus $U(s) \subseteq U_1(s) \times U_2(s) \times \cdots \times U_N(s)$. This, combined with Equation (7), implies that Definition 2.1 holds.

For Definition 2.2, consider $y \in U(s+1)$. Then

$$\|y - \hat{u}^t\|_{2,\infty} \leq q_t^{s+1} \max_{i \in [N]} \|u^i(\eta_{t-1}) - \hat{u}^t\|_{2,\infty}.$$

Since $q_t \in (0, 1)$, we have

$$\|y - \hat{u}^t\|_{2,\infty} \leq q_t^{s+1} \max_{i \in [N]} \|u^i(\eta_{t-1}) - \hat{u}^t\|_{2,\infty} < q_t^* \max_{i \in [N]} \|u^i(\eta_{t-1}) - \hat{u}^t\|_{2,\infty},$$

which implies $y \in U(s)$. Then $U(s+1) \subseteq U(s)$.

For Definition 2.3, we note that, because $q_t \in (0, 1)$, we have $\lim_{s \to \infty} q_t^s = 0$. It immediately follows from the definition of $U(s)$ that $\lim_{s \to \infty} U(s) = \{\hat{u}^t\}$.

Towards satisfying Definition 2.4, we first observe that [21, Lemma 1] implies

$$\|I_n - \gamma_t H(u, t)\|_{2,\infty} \leq \max_{i \in [N]} \left[\|I_{n_i} - \gamma_t H_{ii}(u, t)\|_2 + \gamma_t \sum_{j=1}^n \|H_{ij}(u, t)\|_2\right]. \tag{8}$$

Now consider the $i^{th}$ diagonal block of $I_n - \gamma_t H(u, t)$, which satisfies $[I_n - \gamma_t H(u, t)]_{ii} = I_{n_i} - \gamma_t H_{ii}(u, t)$. By Assumption 3, $H_{ii}(t)$ is symmetric. Then

$$\|I_{n_i} - \gamma_t H_{ii}(u, t)\|_2 = \max\{\lambda_{\min}[I_{n_i} - \gamma_t H_{ii}(u, t)], \lambda_{\max}[I_{n_i} - \gamma_t H_{ii}(u, t)]\} = \max\{1 - \gamma_t \lambda_{\min}[H_{ii}(u, t)], 1 - \gamma_t \lambda_{\max}[H_{ii}(u, t)]\}.$$

The upper bound on $\gamma_t$ gives $1 - \gamma_t \lambda_{\min}[H_{ii}(u, t)] \geq 1 - \gamma_t \lambda_{\max}[H_{ii}(u, t)]$. Then

$$\|I_{n_i} - \gamma_t H_{ii}(u, t)\|_2 = 1 - \gamma_t \lambda_{\min}[H_{ii}(u, t)].$$

Then we have

$$\max_{i \in [N]} \|I_{n_i} - \gamma_t H_{ii}(u, t)\|_2 + \gamma_t \sum_{j=1}^n \|H_{ij}(u, t)\|_2 = \max_{i \in [N]} \left[1 - \gamma_t \left(\lambda_{\min}[H_{ii}(u, t)] - \sum_{j=1}^n \|H_{ij}(u, t)\|_2\right)\right] \leq 1 - \gamma_t \beta_t.$$

Using Equation (8), $\|I_n - \gamma_t H(u, t)\|_{2,\infty} \leq 1 - \gamma_t \beta_t$.

For Definition 2.4, consider $\hat{u}^t = \Pi_U [\hat{u}^t - \gamma_t \nabla u_f(\hat{u}^t, t)]$, i.e., $\hat{u}^t$ is a fixed point of the projected gradient descent mapping. Using the non-expansive property of the projection operator $\Pi_U[.]$, we find

$$\|z_i - \hat{u}^t\|_2 = \|\Pi_U[y_i - \gamma_t \nabla u_f(y_i, t)] - \Pi_U[\hat{u}^t - \gamma_t \nabla u_f(\hat{u}^t, t)]\|_2 \leq \|y_i - \gamma_t \nabla u_f(y_i, t) - \hat{u}^t + \gamma_t \nabla u_f(\hat{u}^t, t)\|_2.$$