A Unified Framework to Understand Decentralized and Federated Optimization Algorithms: A Multi-Rate Feedback Control Perspective

Xinwei Zhang, Mingyi Hong, Nicola Elia
Department of Electric and Computer Engineering
Minnesota University
zhan6234, mhong, nelia@umn.edu

Abstract

Distributed algorithms have been playing an increasingly important role in many applications such as machine learning, signal processing, and control. In this work, we provide a fresh perspective to understand, analyze, and design distributed optimization algorithms. Through the lens of multi-rate feedback control, we show that a wide class of distributed algorithms, including popular decentralized/federated schemes, can be viewed as discretizing a certain continuous-time feedback control system, possibly with multiple sampling rates, such as decentralized gradient descent, gradient tracking, and federated averaging. This key observation not only allows us to develop a generic framework to analyze the convergence of the entire algorithm class, more importantly, it also leads to an interesting way of designing new distributed algorithms. We develop the theory behind our framework and provide examples to highlight how the framework can be used in practice.

1. Introduction

Distributed computation has played an important role in machine learning, partly due to the dramatically increased size of the models and the datasets; see [3, 13] for a few recent surveys. Heterogeneous computational and communication resources in the distributed system create a number of different scenarios in distributed learning. For example, in a *decentralized optimization (DO)* setting, the communication and computation resources are equally important, so the algorithms alternatingly perform communication and communication steps [4, 17, 20, 32, 33]; In the *Federated Learning (FL)* setting, the communication is the bottleneck of the system, so the algorithms typically perform multiple local updates before one communication step [1, 10, 14, 34]; Additionally, in order to identify the *the optimal* decentralized algorithms that utilize the *minimum* computation and communication rounds, it is typically required to perform multiple communication steps before one local update [25, 31].

Despite the proliferation of distributed algorithms, there are a few concerns and challenges. First, for some hot applications, there are simply *too many algorithms* available, so much so that it becomes difficult to track all the technical details. Is it possible to establish some general guidelines to understand the relations between, and the fundamental principles of, those algorithms that provide similar functionalities? Second, much of the recent research on this topic appears to be *increasingly focused* on a specific setting (e.g., those mentioned in the previous paragraph). However, an algorithm developed for FL may have already been rigorously developed, analyzed, and tested for the DO setting; and vice versa. Since developing algorithms and performing analyses take sig-

nificant time and effort, it is desirable to have some mechanisms in place to reduce the possibility of reinventing the wheel.

Our main contribution is to build such a unified framework for distributed algorithms, using tools from multi-rate feedback control systems. Specifically, we first show that, a special continuous-time feedback control system is well-suited to capture a number of key properties of distributed algorithms. We then show that when such a continuous-time system is discretized appropriately (in which different parts of the system are discretized using different rates, hence the name "multi-rate" system), it recovers a wide range of decentralized/federated algorithms. Finally, we provide a generic convergence result that covers different feedback schemes as well as discretization patterns. The major benefits of our proposed framework can be summarized as:

1) Using our framework, we can establish the connection between different subclasses of distributed algorithms that are developed for different settings; in some sense, they can be viewed as applying different discretization schemes to certain continuous-time control system;

2) Our framework helps predict the algorithm performance, and facilitates algorithm design – as long as the continuous-time control system and the desired discretization pattern are identified, our framework readily provides the various system parameters that are needed to ensure algorithm convergence (an example is provided in the appendix to showcase how this can be done).

Note that there are many existing works which analyze optimization algorithms using control theory, but they mainly focus on some very special class of algorithms. For examples, [24, 27, 30] study a restrictive class of simple convex optimization algorithms; the paper [9, 12, 19] investigates the acceleration approaches for centralized problems in discrete time; [19, 30] focus on the continuous-time system and ignore the impact of the discretization to these algorithms; [5, 28, 29] investigate the connection between continuous-time system and discretized gradient descent algorithm, but their approaches and analyses do not generalize to federated/decentralized algorithms. It is also important to note that, to our knowledge, none of the above referred works provide any insights about relationship between difference classes of distributed algorithms (i.e., between DO and FL), nor do they facilitate the design of new algorithms.

2. Continuous-time System

In this section, we provide a general description of the continuous-time multi-agent feedback control system. We start by giving a general system structure and discuss the property of each controller and how the controllers are related to the discrete-time optimization algorithms. Then we provide the convergence properties of the system.

2.1. System Description

The Decentralized Optimization Problem. Consider a distributed system with N agents connected by a strongly connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, each optimizes a smooth and possibility non-convex local function $f_i(x)$. The global optimization problem is formulated as [30]

$$\min_{\mathbf{x}\in\mathbb{R}^{Nd_x}} f(\mathbf{x}) := \frac{1}{N}\sum_{i=1}^{N} f_i(x_i) \qquad \text{s.t. } (A\otimes I)\cdot\mathbf{x} = 0, \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^{N\times d_x}$ stacks N local variables $\mathbf{x} := [x_1; \ldots; x_N]$; $x_i \in \mathbb{R}^{d_x}$, $\forall\, i \in [N]$; $\otimes$ denotes the Kronecocker product; the incidence matrix A contains the graph connectivity pattern with the following definition: if edge $e(i,j) \in \mathcal{E}$ connects vertex i and j with $i > j$, then $A_{ei} = 1$, $A_{ej} = -1$

and $A_{ek} = 0,\ \forall k \neq i, j$. Let us use $\mathcal{N}_i \subset [N]$ denote the neighbors for agent i. For simplicity of notation, the Kronecocker products are ignored in the latter analyses.

The Continuous-Time Double-Feedback System.

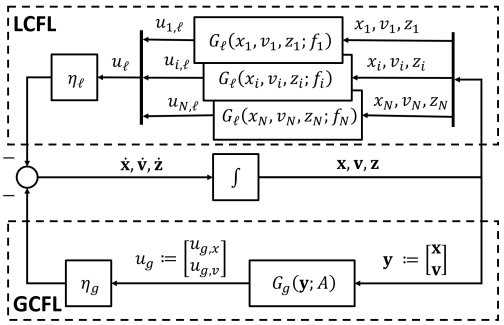


Figure 1: The proposed continuous-time double-feedback system for modeling the decentralized optimization problem (1).

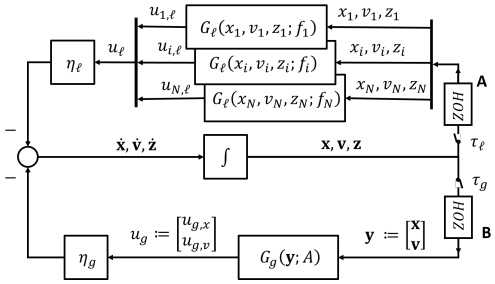


Figure 2: Discretized system using ZOH on both the GCFL and LCFL control loops with possibly different sampling times τ_g, τ_ℓ.

To optimize problem (1), our approach is to design a continuous-time feedback control system, in such a way that the system enters its stable state if and only if the state variables of the system precisely correspond to a first-order stationary solution of (1). Towards this end, let us use $\mathbf{x} \in \mathbb{R}^{N \times d_x}$ to denote the main state variable of the system. Let us introduce two feedback loops, referred to as the *global consensus feedback loop* (GCFL) and *local computation feedback loop* (LCFL), where the former incorporates the dynamics from multi-agent interactions, while the latter helps better stabilize the system. More specifically, these loops can be specified as below:

• **(The GCFL).** Define an auxiliary state variable $\mathbf{v} := [v_1; \ldots; v_N] \in \mathbb{R}^{Nd_v}$, with $v_i \in \mathbb{R}^{d_v},\ \forall\ i$; define $\mathbf{y} := [\mathbf{x}; \mathbf{v}] \in \mathbb{R}^{N(d_x+d_v)}$; define a feedback controller $G_g(\cdot; A) : \mathbb{R}^{N(d_x+d_v)} \to \mathbb{R}^{N(d_x+d_v)}$. The GCFL performs inter-agent communication based on the incidence matrix A, and it controls the consensus of the global variable $\mathbf{y}$. Specifically, at time t, define the output of the controller as $u_g(t) = G_g(\mathbf{y}(t); A)$, which can be further decomposed into two outputs $u_g(t) := [u_{g,x}(t); u_{g,v}(t)]$, one to control the consensus of $\mathbf{x}$ and the other for $\mathbf{v}$. After multiplied by the control gain $\eta_g(t) > 0$, the resulting signal will be combined with the output of the LCFL, and be fed back to local controllers.

• **(The LCFL).** Define an auxiliary state variable $\mathbf{z} := [z_1; \ldots; z_N] \in \mathbb{R}^{Nd_z}$, with $z_i \in \mathbb{R}^{d_z},\ \forall\ i$; define a local feedback controller $G_\ell(\cdot; f_i) : \mathbb{R}^{d_x+d_v+d_z} \to \mathbb{R}^{d_x+d_v+d_z}$, one for each agent i and parameterized by different f_i's. The LCFL optimizes the local function $f_i(\cdot)$'s for each agent. At time t, the ith local controller takes the local variables $x_i(t), v_i(t), z_i(t)$ as inputs and produces a local control signal. To describe the system, let us denote the output of the local controllers as $u_{i,\ell}(t) = G_\ell(x_i(t), v_i(t), z_i(t); f_i),\ \forall\ i \in [N]$; further decompose it into three parts:

$$u_{i,\ell}(t) := [u_{i,\ell,x}(t); u_{i,\ell,v}(t); u_{i,\ell,z}(t)].$$

Denote the concatenated local controller outputs as: $u_{\ell,x}(t) := [u_{1,\ell,x}(t); \ldots; u_{N,\ell,x}(t)]$, and define $u_{\ell,v}(t), u_{\ell,z}(t)$ similarly. After multiplied by the control gain $\eta_\ell(t) > 0$, the resulting signal will be combined with the output of GCFL, and be fed back to the local controllers.

The overall system is described in Fig. 1. The GCFL controller $G_g(\cdot; A)$ is designed to have the following properties:

P 1 (Control Signal Direction) *The output of the controller G_g aligns with the direction that reduces the consensus error, that is:*

$$\langle (I-R)\cdot \mathbf{y}, G_g(\mathbf{y};A)\rangle \geq C_g \cdot \|(I-R)\cdot \mathbf{y}\|^2, \quad \forall\, \mathbf{y},$$

for some constant $C_g > 0$. Further, the controller G_g satisfies:

$$\langle \mathbb{1}, G_g(\mathbf{y};A)\rangle = 0, \quad \forall\, \mathbf{y}, \text{ which implies } \langle \mathbb{1}, u_g(t)\rangle = 0, \quad \forall\, t.$$

P 2 (Linear Operator) *The controller G_g is a linear operator of $\mathbf{y}$, that is, we have $G_g(\mathbf{y};A) = W_A\mathbf{y}$ for some matrix $W_A \in \mathbb{R}^{N(d_x+d_v)}$ parameterized by A, and its eigenvalues satisfy: $|\lambda(W_A)| \in [0,1]$.*

Combining P1 and P2, we have:

$$C_g^2 \|(I-R)\cdot \mathbf{y}\|^2 \leq \|G_g(\mathbf{y};A)\|^2 \leq \|(I-R)\cdot \mathbf{y}\|^2, \text{ and } R\cdot W_A = 0. \tag{2}$$

The local controllers are designed to have the following properties:

P 3 (Lipschitz Smoothness) *The controller is Lipschitz continuous, that is:*

$$\begin{aligned}&\left\|G_\ell(x_i,v_i,z_i;f_i) - G_\ell(x_i',v_i',z_i';f_i)\right\| \leq L\left\|[x_i;v_i;z_i]-[x_i';v_i';z_i']\right\|,\\ &\forall\, i\in[N],\ x_i,x_i'\in\mathbb{R}^{d_x},\ v_i,v_i'\in\mathbb{R}^{d_v},\ z_i,z_i'\in\mathbb{R}^{d_z}.\end{aligned}$$

P 4 (Control Signal Direction and Size) *The local controllers are designed such that there exist initial values $x_i(t_0)$, $v_i(t_0)$ and $z_i(t_0)$ ensuring that the following holds:*

$$\langle \nabla f_i(x_i(t)), u_{i,\ell,x}(t)\rangle \geq \alpha(t)\cdot \|\nabla f_i(x_i(t))\|^2, \quad \forall\, t\geq t_0,$$

where $\alpha(t) > 0$ satisfies $\lim_{t\to\infty}\int_{t_0}^t \alpha(\tau)\mathrm{d}\tau \to \infty$.

Further, for any given x_i, v_i, z_i, the sizes of the control signals are upper bounded by those of the local gradients. That is, for some positive constants C_x, C_v and C_z:

$$\|u_{i,\ell,x}\| \leq C_x\|\nabla f_i(x_i)\|, \quad \|u_{i,\ell,v}\| \leq C_v\|\nabla f_i(x_i)\|, \quad \|u_{i,\ell,z}\| \leq C_z\|\nabla f_i(x_i)\|.$$

The more detailed description of properties of different controllers are given in Appendix A. To close this subsection, we note that the continuous-time system we have presented so far (cf. Figure 1) can be described using the following dynamics:

$$\begin{aligned}\dot{\mathbf{v}}(t) &= -\eta_g(t)\cdot u_{g,v}(t) - \eta_\ell(t)\cdot u_{\ell,v}(t)\\ \dot{\mathbf{x}}(t) &= -\eta_g(t)\cdot u_{g,x}(t) - \eta_\ell(t)\cdot u_{\ell,x}(t), \quad \dot{\mathbf{z}}(t) = -\eta_\ell(t)\cdot u_{\ell,z}(t).\end{aligned} \tag{3}$$

Additionally, throughout the paper, we will use u_g and G_g, u_ℓ and G_ℓ interchangeably.

2.2. Convergence Properties

We proceed to analyze the convergence of the continuous-time system. Towards this end, we define an energy-like function:

$$\mathcal{E}(t) := f(\bar{\mathbf{x}}(t)) - f^\star + \frac{1}{2}\left\|(I-R)\cdot\mathbf{y}(t)\right\|^2. \tag{4}$$

Note that $\mathcal{E}(t) \geq 0$ for all $t \geq 0$. It follows that its derivative can be expressed as:

$$\dot{\mathcal{E}}(t) = -\left\langle \nabla f(\bar{\mathbf{x}}(t), \eta_\ell(t)\cdot\frac{\mathbb{1}^T}{N}u_{\ell,x}(t)\right\rangle + \left\langle (I-R)\cdot\mathbf{y}(t), \eta_g(t)u_g(t) + \eta_\ell(t)u_{\ell,y}(t)\right\rangle. \tag{5}$$

To proceed, we require that the system satisfies the following property:

P 5 (Energy Function Reduction) *The derivative of the energy function, $\dot{\mathcal{E}}(\cdot)$ as expressed in (5), satisfies the following:*

$$\begin{aligned}
&-\int_0^t \left(\left\langle \nabla f(\bar{\mathbf{x}}(\tau), \eta_\ell(\tau)\cdot\frac{\mathbb{1}^T}{N}u_{\ell,x}(\tau)\right\rangle + \left\langle (I-R)\cdot\mathbf{y}(\tau), \eta_g(\tau)u_g(\tau) + \eta_\ell(\tau)u_{\ell,y}(\tau)\right\rangle\right)\mathrm{d}\tau \\
&\leq -\int_0^t \left(\gamma_1(\tau)\cdot\left\|\nabla f(\bar{\mathbf{x}}(\tau))\right\|^2 + \gamma_2(\tau)\cdot\left\|(I-R)\cdot\mathbf{y}(\tau)\right\|^2\right)d\tau,
\end{aligned} \tag{6}$$

where $\gamma_1(\tau), \gamma_2(\tau) > 0$ are some time-dependent coefficients.

P5 is a property about the entire continuous-time system. Although one could show that by using P1 - P4, and by selecting $\eta_g(t)$ and $\eta_\ell(t)$ appropriately, this property can be satisfied with some *specific* $\gamma_1(\tau)$ and $\gamma_2(\tau)$ (cf. Corollary 4 in Appendix A.), here we still list it as an independent property, because at this point we want to keep the choice of $\gamma_1(\tau)$, $\gamma_2(\tau)$ general. Under P5, the continuous-time system will converge to the set of stationary points, and that $\mathbf{x}$ will converge to the set of stationary solutions of problem (1) and the following holds:

$$\min_t \left\{\left\|\nabla f(\bar{\mathbf{x}}(t))\right\|^2 + \left\|(I-R)\cdot\mathbf{y}(t)\right\|^2\right\} = \mathcal{O}\left(\max\left\{\frac{1}{\int_0^T \gamma_1(\tau)\mathrm{d}\tau}, \frac{1}{\int_0^T \gamma_2(\tau)\mathrm{d}\tau}\right\}\right). \tag{7}$$

The above result indicates that if P5 is satisfied, not only will the system asymptotically converge to the set of stationary points, but more importantly, we can use $\{\gamma_1(t), \gamma_2(t)\}$ to characterize the rate in which the stationary gap of problem (1) shrinks. Please see Appendix A for more detailed discussion on the convergence property and the proofs for the above results.

So far, we have completed the setup of the continuous-time feedback control system, by specifying the state variables, the feedback loops, and by introducing a few desired properties of the local controllers and the entire system. In particular, we show that property P5 is instrumental in ensuring that the system converges to the set of stationary points. However, there is a key question remain to be answered: How to map the continuous-time system to a distributed optimization algorithm, and to transfer the convergence guarantees of the former to the latter? This question will be addressed in the main technical part of this work to be presented shortly.

3. System Discretization

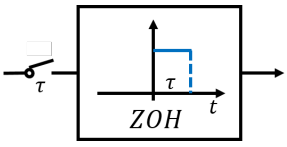


Figure 3: The discretization block that has a switch and a Zero-Order Hold.

In this section, we discuss the impact of system discretization on the proposed double-feedback control system. Since our continuous-time control system involves *two* different feedback loops GCFL and LCFL, special attention will be given to the two loops that are discretized differently. Interestingly, we will show that many state-of-the-art algorithms for decentralized learning can be precisely mapped to some versions of discretized double-feedback control system, by properly choosing a specific discretization scheme, and by specializing the global and local controllers.

3.1. Modeling the Discretization

Typically, a continuous-time system is discretized by using a switch that samples the input with sample time τ, followed by a Zero-Order Hold (ZOH) that keeps the sampled signal as constant between the consecutive sampling instances [11]; see Figure 3 for an illustration. Note that the continuous-time system is equivalent to the case where the sampling time $\tau = 0$ and the switch is constantly on.

Let us begin by using such a block to discretize the proposed double-feedback continuous-time control system. By examining Fig. 2, we see that the discretization can happen at the two points A and B, where the local states are about to enter the controllers.

It is important to observe that, depending on which place (or places) that the discretization blocks are implemented, and depending on the actual sampling rate for each of the discretization block, the original continuous system can be discretized in many different ways. In a high-level, each of these discretization scheme will corresponds to a *multi-rate* control system, in which different parts of the system runs on different sampling rates. To precisely understand the effect of such kind of multi-rate system, let us define the *sampling intervals* for the GCFL and LCFL as τ_g and τ_ℓ, respectively.

3.2. Decentralized Algorithms as Multi-Rate Discretized Systems

In this section, we make the connection between different *classes* of decentralized algorithms and different discretization patterns. For convenience, let t_k denote the times at which the inputs of the ZOHs get sampled by *both* the global and local controllers. Note that when the sampling interval is zero, the corresponding ZOH samples all the time.

Case 1 ($\tau_g > 0, \tau_\ell = 0$): In this case, the local updates are continuous and the global consensus signal is updated every τ_g time interval. The system becomes

$$\begin{aligned}\dot{\mathbf{v}}(t) &= -\eta_g(t) \cdot u_{g,v}(t_k) - \eta_\ell(t) \cdot u_{\ell,v}(t)\\ \dot{\mathbf{x}}(t) &= -\eta_g(t) \cdot u_{g,x}(t_k) - \eta_\ell(t) \cdot u_{\ell,x}(t)\\ \dot{\mathbf{z}}(t) &= -\eta_\ell(t) \cdot u_{\ell,z}(t).\end{aligned} \tag{8}$$

By A4, we know that with only u_ℓ, the dynamic system finds a stationary point of the local problem that $\|\nabla f_i(x_i)\|^2 = 0$. So with (8), the dynamic system finds a stationary point that $u_\ell + \frac{\eta_g(t)}{\eta_\ell(t)} u_g = 0$, which is the stationary solution of the following problem for each agent:

$$f_i'(x_i(t)) := f_i(x_i(t)) + \frac{\eta_g(t)}{\eta_l(t)} \langle u_{i,g}(t_k), y_i(t) \rangle .$$

Therefore, from t_0 to $t_0+\tau_g$, each agent minimizes a perturbed local problem to $\gamma(\tau)$ accuracy. This system has the same form as the distributed algorithms that require to solve some local problems to a given accuracy, before any local communication steps take place; see for examples FedProx [14], FedPD [34] and NEXT [4].

Case 2 ($\tau_g = 0, \tau_\ell > 0$): In this case, the global consensus signal is continuous and the local updates are updated every τ_ℓ time interval. The system becomes

$$\begin{aligned}\dot{\mathbf{v}}(t) &= -\eta_g(t)\cdot u_{g,v}(t) - \eta_\ell(t)\cdot u_{\ell,v}(t_k)\\ \dot{\mathbf{x}}(t) &= -\eta_g(t)\cdot u_{g,x}(t) - \eta_\ell(t)\cdot u_{\ell,x}(t_k)\\ \dot{\mathbf{z}}(t) &= -\eta_\ell(t)\cdot u_{\ell,z}(t_k).\end{aligned} \tag{9}$$

Similar to Case I, between τ_ℓ time interval, the dynamic system finds the first-order stationary point the following problem

$$\left\|(I-R)\mathbf{y} + \frac{\eta_\ell(t)}{\eta_g(t)} u_{\ell,y}(t_k)\right\|^2$$

to certain accuracy. This system has the same form as the algorithms that require to solve some networked problems to a certain accuracy, see for example [7, 8, 25].

Case 3 ($\tau_g = \tau_\ell > 0$): In this case, the system is discretized with the same sampling time. The update of the system can be written as:

$$\begin{aligned}\mathbf{x}(t_{k+1}) &= \mathbf{x}(t_k) - \eta'_\ell(t_k)\cdot u_{\ell,x}(t_k) - \eta'_g(t_k)\cdot u_{g,x}(t_k),\\ \mathbf{v}(t_{k+1}) &= \mathbf{v}(t_k) - \eta'_\ell(t_k)\cdot u_{\ell,v}(t_k) - \eta'_g(t_k)\cdot u_{g,v}(t_k),\\ \mathbf{z}(t_{k+1}) &= \mathbf{z}(t_k) - \eta'_g(t_k)\cdot u_{\ell,z}(t_k),\end{aligned} \tag{10}$$

where $\eta'_\ell(t_k) = \int_{t_k}^{t_k+\tau_g} \eta_\ell(t)\mathrm{d}t$, $\eta'_g(t_k) = \int_{t_k}^{t_k+\tau_g} \eta_g(t)\mathrm{d}t$. The above updates can be shown to be equivalent to many existing DO algorithms, such as DGD, D^2 [16], DLM, which perform one step local update, followed by one step of communication.

Case 4 ($\tau_g > \tau_l > 0$): In this case, we assume that $\tau_g = Q\cdot\tau_l$, which means that each agent performs Q times local computation steps between two communication steps. This update strategy has the same spirit as the class of (horizontal) FL algorithms [1].

Case 5 ($\tau_l > \tau_g > 0$): In this case, we assume that $\tau_l = K\cdot\tau_g$, which means that the agents perform K times communication steps before two local computation steps. This update strategy is similar to the line of works that are trying to achieve optimal communication complexities [15, 25].

We summarize the above discussion in Table 1 and provide some examples of the algorithms. Note that the above discussion about relations of algorithms and discretization settings is still a bit vague, but later in Appendix B and C, we will provide specific examples to showcase how one can precisely map an existing algorithm into a discretization setting.

Such kinds of connections are useful in the following sense: First, it suggests that *different* classes of algorithms may be rooted in *the same* continuous-time system, so they are likely to share some common properties, and it is plausible that they can be covered by a single analysis framework. Second, by properly utilizing such kinds of connections, we can develop a systematic way of designing new, and application-specific algorithms. More specifically, we can begin by designing and analyzing a continuous-time control system (say, with a specific controller), then choose a desirable discretization scheme, and transfer the theoretical results to such a particular setting. Therefore, it will be important to have a systematic way of transferring the theoretical results from the continuous-time system to different discretization settings.

Case	$\tau_\ell, \tau_\mathbf{g}$	Communication	Computation	Related Algorithm
I	$\tau_g > 0, \tau_l = 0$	Slow	Continuous	NEXT [4], FedProx [14]
II	$\tau_g = 0, \tau_l > 0$	Continuous	Slow	GPDA [8]
III	$\tau_g = \tau_l > 0$	Same rate		DGD [32], DGT [33]
IV	$\tau_g > \tau_l > 0$	Slow	Fast	Scaffold [10], NIDS [15]
V	$\tau_l > \tau_g > 0$	Fast	Slow	AGD [31], xFilter [25]

Table 1: Summary of different discretization settings, and their corresponding distributed learning algorithms.

3.3. Convergence Result of Discretized Multi-Rate Systems

For a given distributed algorithm, we can first find its continuous-time counterpart, then perform discretization based on the requirements of each part of the system. This procedure allows the new algorithm to share some desirable properties of the original algorithm. However, the discretization procedure will introduce instability to the system as the sampled control signal will deviate from the continuous-time control signal. As the sampling interval increases, the deviation also increases. Understanding how the deviations introduced by different discretization schemes affect the system is crucial to transferring the theoretical results from the continuous-time system to discretized systems. The following result provides a way to analyze the convergence for all different discretization schemes.

Theorem 1 (Dynamics of $\mathcal{E}$ for discretized system) *Suppose GCFL and LCFL satisfies properties P1-P5 and consider the discretize-time system with $\tau_\ell \geq 0, \tau_g \geq 0$. Then we have the following:*

$$\dot{\mathcal{E}}(t) \leq -\left(\frac{\gamma_1(t)}{2} - C_3(t)\right)\left\|\sum_{i=1}^{N}\nabla f_i(\bar{\mathbf{x}}(t))\right\|^2 - \left(\frac{\gamma_2(t)}{2} - C_4(t)\right)\|(I-R)\mathbf{y}(t)\|^2, \tag{11}$$

where $C_3(t)$ and $C_4(t)$ are some constants depending on $N, C_g, L, \eta_\ell(t), \eta_g(t), \tau_\ell, \tau_g, K, Q$.

This result indicates that by proper choice of τ_ℓ, τ_g, K, Q, such that $\left(\frac{\gamma_1(t)}{2} - C_3(t)\right) > 0$ and $\left(\frac{\gamma_2(t)}{2} - C_4(t)\right) > 0$, the discretized algorithm preserves the convergence rate of the continuous-time system and only slows down by a constant factor

$$\max\left\{\gamma_1(t)\Big/\left(\frac{\gamma_1(t)}{2} - C_3(t)\right), \gamma_2(t)\Big/\left(\frac{\gamma_2(t)}{2} - C_4(t)\right)\right\}.$$

The detailed convergence analyses and discussions are omitted due to page limitation.

4. Summary

In this work, we have designed a framework to understand distributed optimization algorithms from a control perspective. We have shown that a multi-rate double-feedback control system can represent a wide range of deterministic distributed optimization algorithms. Furthermore, we have provided a set of properties the system should satisfy and established the theoretical guarantees for both continuous-time and discretized systems. We use a few example algorithms in Appendix B to demonstrate how the proposed framework can help understand the connection between algorithms, and further provide the new algorithm design procedure with theoretical analysis and numerical experiments in Appendix C.

References

[1] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems*, 1:374–388, 2019.

[2] Sébastien Bubeck, Yin Tat Lee, and Mohit Singh. A geometric alternative to nesterov's accelerated gradient descent. *arXiv preprint arXiv:1506.08187*, 2015.

[3] Tsung-Hui Chang, Mingyi Hong, Hoi-To Wai, Xinwei Zhang, and Songtao Lu. Distributed learning in the nonconvex world: From batch data to streaming and beyond. *IEEE Signal Processing Magazine*, 37(3):26–38, 2020.

[4] Paolo Di Lorenzo and Gesualdo Scutari. Next: In-network nonconvex optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 2(2):120–136, 2016.

[5] Guilherme França, Daniel P Robinson, and Rene Vidal. A dynamical systems perspective on nonsmooth constrained optimization. *arXiv preprint arXiv:1808.04048*, 2018.

[6] Euhanna Ghadimi, Mikael Johansson, and Iman Shames. Accelerated gradient methods for networked optimization. In *Proceedings of the 2011 American Control Conference*, pages 1668–1673. IEEE, 2011.

[7] Mingyi Hong, Davood Hajinezhad, and Ming-Min Zhao. Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks. In *International Conference on Machine Learning*, pages 1529–1538, 2017.

[8] Mingyi Hong, Meisam Razaviyayn, and Jason Lee. Gradient primal-dual algorithm converges to second-order stationary solution for nonconvex distributed optimization over networks. In *International Conference on Machine Learning*, pages 2009–2018. PMLR, 2018.

[9] Bin Hu and Laurent Lessard. Control interpretations for first-order optimization methods. In *2017 American Control Conference (ACC)*, pages 3114–3119. IEEE, 2017.

[10] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[11] Benjamin C Kuo. Digital control systems, 1980.

[12] Laurent Lessard, Benjamin Recht, and Andrew Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.

[13] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[14] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

[15] Zhi Li, Wei Shi, and Ming Yan. A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates. *IEEE Transactions on Signal Processing*, 67(17):4494–4506, 2019.

[16] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5336–5346, 2017.

[17] Qing Ling, Wei Shi, Gang Wu, and Alejandro Ribeiro. DLM: Decentralized linearized alternating direction method of multipliers. *IEEE Transactions on Signal Processing*, 63(15): 4051–4064, 2015.

[18] Songtao Lu, Xinwei Zhang, Haoran Sun, and Mingyi Hong. GNSD: A gradient-tracking based nonconvex stochastic algorithm for decentralized optimization. In *2019 IEEE Data Science Workshop (DSW)*, pages 315–321. IEEE, 2019.

[19] Michael Muehlebach and Michael Jordan. A dynamical systems perspective on nesterov acceleration. In *International Conference on Machine Learning*, pages 4656–4662, 2019.

[20] Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[21] Alex Olshevsky and John N Tsitsiklis. Convergence speed in distributed consensus and averaging. *SIAM journal on control and optimization*, 48(1):33–55, 2009.

[22] Antonio Orvieto and Aurelien Lucchi. Continuous-time models for stochastic optimization algorithms. *Advances in Neural Information Processing Systems*, 32, 2019.

[23] Sashank J Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2020.

[24] Riccarda Rossi and Giuseppe Savaré. Gradient flows of non convex functionals in hilbert spaces and applications. *ESAIM: Control, Optimisation and Calculus of Variations*, 12(3): 564–614, 2006.

[25] Haoran Sun and Mingyi Hong. Distributed non-convex first-order optimization and information processing: Lower complexity bounds and rate optimal algorithms. *IEEE Transactions on Signal processing*, 67(22):5912–5928, 2019.

[26] Haoran Sun, Songtao Lu, and Mingyi Hong. Improving the sample and communication complexity for decentralized non-convex optimization: Joint gradient estimation and tracking. In *International Conference on Machine Learning*, pages 9217–9228. PMLR, 2020.

[27] Akhil Sundararajan. *Analysis and Design of Distributed Optimization Algorithms*. The University of Wisconsin-Madison, 2021.

[28] Brian Swenson, Ryan Murray, H Vincent Poor, and Soummya Kar. Distributed gradient descent: Nonconvergence to saddle points and the stable-manifold theorem. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 595–601. IEEE, 2019.

[29] Brian Swenson, Ryan Murray, H Vincent Poor, and Soummya Kar. Distributed gradient flow: Nonsmoothness, nonconvexity, and saddle point evasion. *IEEE Transactions on Automatic Control*, 2021.

[30] Jing Wang and Nicola Elia. A control perspective for centralized and distributed convex optimization. In *2011 50th IEEE conference on decision and control and European control conference*, pages 3800–3805. IEEE, 2011.

[31] Haishan Ye, Luo Luo, Ziang Zhou, and Tong Zhang. Multi-consensus decentralized accelerated gradient descent. *arXiv preprint arXiv:2005.00797*, 2020.

[32] Kun Yuan, Qing Ling, and Wotao Yin. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization*, 26(3):1835–1854, 2016.

[33] Kun Yuan, Wei Xu, and Qing Ling. Can primal methods outperform primal-dual methods in decentralized dynamic optimization? *IEEE Transactions on Signal Processing*, 68:4466–4480, 2020.

[34] Xinwei Zhang, Mingyi Hong, Sairaj Dhople, Wotao Yin, and Yang Liu. FedPD: A federated learning framework with adaptivity to non-iid data. *IEEE Transactions on Signal Processing*, 69:6055–6070, 2021.

Appendix A. Additional Discussions for Section 2

In this section, we describe the details of the controllers in the continuous-time system described in Fig. 1.

To have a rough idea of how the GCFL and LCFL loops can be mapped to a distributed algorithm, let us consider the NEXT algorithm [4], whose updates (in discrete time) are:

$$\begin{aligned}\mathbf{x}(k+1/2) &= \arg\min_{\mathbf{x}} \tilde{f}(\mathbf{x};\mathbf{x}(k)) + \langle N\mathbf{v}(k) - \nabla f(\mathbf{x}(k)), \mathbf{x}-\mathbf{x}(k)\rangle\,,\\ \mathbf{x}(k+1) &= W\left(\mathbf{x}(k) + \alpha\cdot(\mathbf{x}(k+1/2)-\mathbf{x}(k))\right),\\ \mathbf{v}(k+1) &= W\mathbf{v}(k) + \nabla f(\mathbf{x}(k+1)) - \mathbf{z}(k), \quad \mathbf{z}(k+1) = \nabla f(\mathbf{x}(k+1)),\end{aligned}$$

where $\tilde{f}$ is some surrogate function; k indicates the iteration index; $\alpha > 0$ and $c > 0$ are some stepsize parameters. By using the common choice that $\tilde{f}(\mathbf{x};\mathbf{x}(k)) = \langle \nabla f(\mathbf{x}(k)), \mathbf{x}-\mathbf{x}(k)\rangle + \frac{\eta}{2}\|\mathbf{x}-\mathbf{x}(k)\|^2$, (where $\eta > 0$ are some constant) the algorithm can be simplify as:

$$\begin{aligned}\mathbf{x}(k+1) &= W\mathbf{x}(k) - N\alpha/\eta\cdot\mathbf{v}(k), \quad \mathbf{z}(k+1) = \mathbf{x}(k+1),\\ \mathbf{v}(k+1) &= W\mathbf{v}(k) + \nabla f(\mathbf{x}(k+1)) - \nabla f(\mathbf{z}(k)).\end{aligned} \tag{12}$$

Here, $\mathbf{x}$ is the optimization variable, $\mathbf{v}$ tracks the average of the gradients, $\mathbf{z}$ records the one-step-behind state of $\mathbf{x}$. The corresponding controllers are given by:

$$G_g(\mathbf{x},\mathbf{v};A) := \begin{bmatrix}(I-W)\cdot\mathbf{x}\\ (I-W)\cdot\mathbf{v}\end{bmatrix}, \quad G_\ell(x_i,v_i,z_i;f_i) := \begin{bmatrix}v_i\\ \nabla f_i(z_i) - \nabla f_i(x_i)\\ z_i - x_i\end{bmatrix}. \tag{13}$$

Next, we describe in detail the properties of the two feedback loops.

A.1. Global Consensus Feedback Loop

The GCFL performs inter-agent communication based on the incidence matrix A, and it controls the consensus of the global variable $\mathbf{y} := [\mathbf{x};\mathbf{v}]$. Specifically, at time t, define the output of the controller as $u_g(t) = G_g(\mathbf{y}(t);A)$, which can be further decomposed into two outputs $u_g(t) := [u_{g,x}(t); u_{g,v}(t)]$, one to control the consensus of $\mathbf{x}$ and the other for $\mathbf{v}$. After multiplied by the control gain $\eta_g(t) > 0$, the resulting signal will be combined with the output of the LCFL, and be fed back to local controllers.

We require that the global controller $G_g(\cdot;A)$ to have the following properties:

P 6 (Control Signal Direction) *The output of the controller G_g aligns with the direction that reduces the consensus error, that is:*

$$\langle (I-R)\cdot\mathbf{y}, G_g(\mathbf{y};A)\rangle \geq C_g\cdot\|(I-R)\cdot\mathbf{y}\|^2, \quad \forall\,\mathbf{y},$$

for some constant $C_g > 0$. Further, the controller G_g satisfies:

$$\langle \mathbb{1}, G_g(\mathbf{y};A)\rangle = 0, \quad \forall\,\mathbf{y}, \text{ which implies } \langle \mathbb{1}, u_g(t)\rangle = 0, \quad \forall\, t.$$

P 7 (Linear Operator) *The controller G_g is a linear operator of $\mathbf{y}$, that is, we have $G_g(\mathbf{y};A) = W_A\mathbf{y}$ for some matrix $W_A \in \mathbb{R}^{N(d_x+d_v)}$ parameterized by A, and its eigenvalues satisfy: $|\lambda(W_A)| \in [0,1]$.*

Combining P1 and P2, we have:

$$C_g^2 \left\|(I-R)\cdot \mathbf{y}\right\|^2 \leq \left\|G_g(\mathbf{y};A)\right\|^2 \leq \left\|(I-R)\cdot \mathbf{y}\right\|^2, \quad \text{and} \quad R\cdot W_A = 0. \tag{14}$$

It is easy to check that both P1 and P2 hold in most of the existing consensus-based algorithms. For example, when the communication graph is strongly connected, we can choose $G_g(\mathbf{y};A) = (I-W)\cdot\mathbf{y}$. It is easy to verify that, $C_g = 1-\lambda_2(W)$ where $\lambda_2(\cdot)$ denotes the eigenvalue withe the second largest magnitude [3, 32]. As another example, consider the accelerated averaging algorithms [6], where we have

$$G_g(\mathbf{y},A) = \begin{bmatrix} I-(c+1)\cdot W & c\cdot I \\ -I & I \end{bmatrix}\begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix}, \text{ with } c := \frac{1-\sqrt{1-\lambda_2(W)}}{1+\sqrt{1-\lambda_2(W)^2}}.$$

In this case, one can verify that $C_g = 1-\frac{\lambda_2(W)}{1+\sqrt{1-\lambda_2(W)^2}} \geq 1-\lambda_2(W)$.

By using P1, we can follow the general analysis of averaging systems [21], and show that the GCFL will behave *as expected*, that is, if the system *only* performs GCFL and shuts off the LCFL, then the consensus can be achieved. More precisely, assuming that $\eta_\ell(t)=0, \eta_g(t)=1$, then under P1, the local state $\mathbf{y}$ converges to the average of the initial states linearly:

$$\left\|(I-R)\cdot\mathbf{y}(t)\right\|^2 \leq \mathrm{e}^{-2C_g t}\left\|(I-R)\cdot\mathbf{y}(0)\right\|^2. \tag{15}$$

A.2. The Local Computation Feedback Loop

The LCFL optimizes the local function $f_i(\cdot)$'s for each agent. At time t, the ith local controller takes the local variables $x_i(t), v_i(t), z_i(t)$ as inputs and produces a local control signal. To describe the system, let us denote the output of the local controllers as $u_{i,\ell}(t) = G_\ell(x_i(t),v_i(t),z_i(t);f_i),\ \forall\, i\in[N]$; further decompose it into three parts:

$$u_{i,\ell}(t) := [u_{i,\ell,x}(t); u_{i,\ell,v}(t); u_{i,\ell,z}(t)].$$

Denote the concatenated local controller outputs as: $u_{\ell,x}(t) := [u_{1,\ell,x}(t);\ldots;u_{N,\ell,x}(t)]$, and define $u_{\ell,v}(t), u_{\ell,z}(t)$ similarly. Note that we have assumed that all the agents use the same local controller $G_\ell(\cdot;\cdot)$, but they are parameterized by different f_i's. After multiplied by the control gain $\eta_\ell(t)>0$, the resulting signal will be combined with the output of GCFL, and be fed back to the local controllers.

The local controllers are designed to have the following properties:

P 8 (Lipschitz Smoothness) *The controller is Lipschitz continuous, that is:*

$$\begin{aligned}&\left\|G_\ell(x_i,v_i,z_i;f_i) - G_\ell(x_i',v_i',z_i';f_i)\right\| \leq L\left\|[x_i;v_i;z_i]-[x_i';v_i';z_i']\right\|,\\ &\forall\, i\in[N],\ x_i,x_i'\in\mathbb{R}^{d_x},\ v_i,v_i'\in\mathbb{R}^{d_v},\ z_i,z_i'\in\mathbb{R}^{d_z}.\end{aligned}$$

P 9 (Control Signal Direction and Size) *The local controllers are designed such that there exist initial values $x_i(t_0)$, $v_i(t_0)$ and $z_i(t_0)$ ensuring that the following holds:*

$$\langle \nabla f_i(x_i(t)), u_{i,\ell,x}(t)\rangle \geq \alpha(t)\cdot\left\|\nabla f_i(x_i(t))\right\|^2, \quad \forall\, t\geq t_0,$$

where $\alpha(t)>0$ satisfies $\lim_{t\to\infty}\int_{t_0}^t \alpha(\tau)\mathrm{d}\tau\to\infty$.

Further, for any given x_i, v_i, z_i, the sizes of the control signals are upper bounded by those of the local gradients. That is, for some positive constants C_x, C_v and C_z:

$$\|u_{i,\ell,x}\| \leq C_x \|\nabla f_i(x_i)\|, \quad \|u_{i,\ell,v}\| \leq C_v \|\nabla f_i(x_i)\|, \quad \|u_{i,\ell,z}\| \leq C_z \|\nabla f_i(x_i)\|.$$

Let us comment on these properties. P3 is easy to verify for a given realizations of the local controllers; P4 abstracts the convergence property of the local optimizer. This property implies that the update direction $-u_{i,\ell,x}(t)$ points to a direction that decreases the local objective. Note that it is postulated that x_i, v_i and z_i are initialized properly, because in some of the cases, improper initial values lead to non-convergence of the local controllers (or equivalently, the local algorithm). For example, for accelerated gradient descent method [2, 31], $z_i(t_0)$ should be initialized as $\nabla f_i(x_i(t_0))$.

By using P4, we can follow the general analysis of the gradient flow algorithms (e.g., [22]), and show that the LCFL will behave *as expected*, in the sense that the agents can properly optimize their local problems. More precisely, assume that $\eta_g(t) = 0, \eta_\ell(t) = 1$, that is, the system shuts off the GCFL. Assume that $G_\ell(\cdot;\cdot)$ satisfies P4, then each local system produces $x_i(t)$'s that satisfy:

$$\min_{\tau} \|\nabla f_i(x_i(t+\tau))\|^2 \leq \gamma(\tau) \cdot (f_i(x_i(t)) - \underline{f}_i), \tag{16}$$

where $\{\gamma(\tau)\}$ is a sequence of positive constants satisfying:

$$\gamma(\tau) = \frac{1}{\int_0^t \alpha(\tau)\mathrm{d}\tau} \to 0, \quad \text{as } \tau \to \infty. \tag{17}$$

A.3. Convergence Property

In the following, we study the convergence of $\mathcal{E}(t)$ and characterize the set of stationary points that the states satisfy $\dot{\mathcal{E}}(t) = 0$. We do not attempt to analyze the stronger property of *stability*, not only because such kind of analysis can be challenging due to the non-convexity of the local functions $f_i(\cdot)$'s, but more importantly, analyzing the convergence of $\mathcal{E}(t)$ is already sufficient for us to understand the convergence of the state variable $\mathbf{x}$ to the set of stationary solutions of problem (1), as we will show shortly.

First, we define the first order stationary point of a problem as:

Definition 2 (First-order Stationary Point) *We define the first-order stationary solution and the ϵ-stationary solution respectively, as:*

$$\sum_{i=1}^N \nabla f_i\left(\frac{1}{N}\sum_{i=1}^N x_i\right) = 0, \quad \mathbf{x} - \frac{\mathbb{1}\mathbb{1}^T}{N}\mathbf{x} = 0, \tag{18a}$$

$$\left\|\frac{1}{N}\sum_{i=1}^N \nabla f_i\left(\frac{1}{N}\sum_{i=1}^N x_i\right)\right\|^2 + \left\|\mathbf{x} - \frac{\mathbb{1}\mathbb{1}^T}{N}\mathbf{x}\right\|^2 \leq \epsilon. \tag{18b}$$

We refer to the left hand side (LHS) of (18b) as the stationarity gap of (1).

Recall that the energy-like function $\mathcal{E}(t)$ is defined as:

$$\mathcal{E}(t) := f(\bar{\mathbf{x}}(t)) - f^\star + \frac{1}{2}\|(I-R)\cdot\mathbf{y}(t)\|^2. \tag{19}$$

Note that $\mathcal{E}(t) \geq 0$ for all $t \geq 0$. It follows that its derivative can be expressed as:

$$\dot{\mathcal{E}}(t) = -\left\langle \nabla f(\bar{\mathbf{x}}(t), \eta_\ell(t) \cdot \frac{\mathbb{1}^T}{N} u_{\ell,x}(t) \right\rangle + \langle (I-R)\cdot \mathbf{y}(t), \eta_g(t) u_g(t) + \eta_\ell(t) u_{\ell,y}(t) \rangle \,. \tag{20}$$

Next, we will show that under P5, the continuous-time system will converge to the set of stationary points, and that $\mathbf{x}$ will converge to the set of stationary solutions of problem (1).

Theorem 3 *Suppose P5 holds true. Then we have the following results:*
1) *Further, suppose that P1, P2 and P4 hold, then $\dot{\mathcal{E}} = 0$ implies that the corresponding state variable $\mathbf{x}_s$ is bounded, and the following holds:*

$$\dot{\mathbf{x}}_s = 0, \quad \dot{\mathbf{v}}_s = 0, \quad \dot{\mathbf{z}}_s = 0, \quad u_g = 0, \quad u_\ell = 0. \tag{21}$$

Additionally, let us define the set $\mathbf{S}$ as below:

$$\mathbf{S} := \left\{ \mathbf{v}, \mathbf{z} \,\middle|\, \eta_\ell u_{\ell,v} + \eta_g u_{g,v} = 0,\ u_{\ell,z} = 0,\ \eta_\ell u_{\ell,x} + \eta_g u_{g,x} = 0 \right\}.$$

If we assume that $\mathbf{S}$ is compact for any state variable $\mathbf{x}$ that satisfies the stationarity condition (18a), then the auxiliary state variables $\{\mathbf{v}(t)\}$ and $\{\mathbf{z}(t)\}$ are also bounded.
2) *The control system asymptotically converges to the set of stationary points, in that $\mathbf{x}(t)$ is bounded $\forall t \in [0, \infty)$, and $\dot{\mathcal{E}} \to 0$. Further, the stationary gap can be upper bounded by the following:*

$$\min_t \left\{ \|\nabla f(\bar{\mathbf{x}}(t))\|^2 + \|(I-R)\cdot \mathbf{y}(t)\|^2 \right\} = \mathcal{O}\left(\max\left\{ \frac{1}{\int_0^T \gamma_1(\tau)\mathrm{d}\tau}, \frac{1}{\int_0^T \gamma_2(\tau)\mathrm{d}\tau} \right\} \right). \tag{22}$$

Proof To show part (1), consider a set of states $\mathbf{x}_s, \mathbf{v}_s, \mathbf{z}_s$ in which $\dot{\mathcal{E}}(\mathbf{x}_s, \mathbf{v}_s) = 0$. P5 implies that $\nabla f(\bar{\mathbf{x}}_s) = 0$, and P4 implies $\|u_\ell\| \leq (C_x + C_v + C_z)\|\nabla f(\bar{\mathbf{x}}_s)\| = 0$. Similarly, with P1 and P2 we have that $\langle u_g, (I-R)\mathbf{y}_s \rangle = 0$ and $\mathbb{1}^T u_g = 0$ so $u_g = 0$. Therefore $\dot{\mathbf{x}}_s = 0, \dot{\mathbf{v}}_s = 0, \dot{\mathbf{z}}_s = 0$. Combining $\nabla f(\bar{\mathbf{x}}_s) = 0$ and the coercive assumption on the problem implies that $\mathbf{x}_s$ is bounded. Note that the value of $\mathbf{v}(t), \mathbf{z}(t)$ may not be bounded, even if the system converges to a stationary solution. Using the compactness assumption on the set $\mathbf{S}$, it is easy to show that $\mathbf{v}(t), \mathbf{z}(t)$ are also bounded.

To show part (2), we can integrate $\dot{\mathcal{E}}(t)$ from $t = 0$ to T to obtain:

$$\int_0^T \gamma_2(t)\,\|(I-R)\cdot \mathbf{y}(t)\|^2\,\mathrm{d}t + \int_0^T \gamma_1(t)\,\|\nabla f(\bar{\mathbf{x}}(t))\|^2\,\mathrm{d}t \leq \mathcal{E}(0) - \mathcal{E}(T),$$

divide both sides by $\int_0^T \gamma_1(t)\mathrm{d}t$ or $\int_0^T \gamma_2(t)\mathrm{d}t$, we obtain(7). By P5 we know $\int_0^t \dot{\mathcal{E}}(\tau)\mathrm{d}\tau \leq 0,\ \forall t$, but since $\mathcal{E}(t) \geq 0$, it follows that $\lim_{t\to\infty} \dot{\mathcal{E}}(t) = 0$. ■

Note that without the compactness assumption, $\mathbf{v}$ and $\mathbf{z}$ can be unbounded. As an example, FedYogi uses AdaGrad for LCFL [23] where $\mathbf{v}(t)$ accumulates the norm of the gradients and does not satisfy the compactness assumption, so $\lim_{t\to\infty} \mathbf{v}(t) \to \infty$. Although such unboundedness does not affect the convergence of the main state variable in part (2), from the control perspective it is still desirable to have a sufficient condition to guarantee the boundedness of all state variables.

Part (2) of the above result indicates that if P5 is satisfied, not only will the system asymptotically converge to the set of stationary points, but more importantly, we can use $\{\gamma_1(t), \gamma_2(t)\}$ to

characterize the rate in which the stationary gap of problem (1) shrinks. This result, although rather simple, will serve as the basis for our subsequent system discretization analysis.

Below we show that for a *generic* system that satisfies properties P1 – P4, when the control gains $\eta_g(t), \eta_\ell(t)$ are selected appropriately, then P5 will be satisfied.

Corollary 4 *Suppose that P1, P3, P4 are satisfied. By choosing* $\eta_g(t) = 1, \eta_\ell(t) = \mathcal{O}(1/\sqrt{T})$*, P5 holds true with* $\gamma_1(t) = \mathcal{O}(\eta_\ell(t))$*,* $\gamma_2(t) = \mathcal{O}(1)$ *Further,*

$$\min_t \left\{ \|\nabla f(\bar{\mathbf{x}}(t))\|^2 + \|(I-R)\cdot \mathbf{y}(t)\|^2 \right\} = \mathcal{O}\left(\frac{1}{\int_0^T \eta_\ell(\tau)\mathrm{d}\tau} \right) = \mathcal{O}\left(\frac{1}{\sqrt{T}} \right).$$

The proof of the above result follows the steps used in analyzing distributed gradient flow algorithm [28] and is omitted in this section due to page limitation.

Additionally, one can also verify P5 in a case-by-case manner for individual systems. In this way, it is possible that one can obtain larger gains $\eta_\ell(t), \eta_g(t)$, hence larger coefficients $\gamma_1(t)$ and $\gamma_2(t)$ to further improve the convergence rate estimate. In fact, verifying property P5, and computing the corresponding coefficients is a key step in our proposed analysis framework for distributed algorithms. Shortly in Appendix C, we will provide an example to showcase how to verify that the continuous-time system which corresponds to the DGT algorithm satisfies P5 with $\gamma_1(t) = \mathcal{O}(1)$ and $\gamma_2(t) = \mathcal{O}(1)$, leading to a convergence rate of $\mathcal{O}(1/T)$.

Appendix B. Discussions on the Existing Algorithms

In this section, we discuss some of the applications of the proposed framework. We first show that by properly choosing the two controllers and the discretization scheme, the proposed framework can be specialized to a number of popular decentralized learning algorithms. Second, we show how the proposed framework can help us identify the relationship between different algorithms, as well as facilitate development of new algorithms.

B.1. Existing Decentralized Algorithms as Discretized Multi-Rate Systems

We map some of the existing distributed algorithms into the discretized multi-rate system with specific GCFL and LCFL.

First let us begin with the DO algorithms:

DGD [20]: The update step of DGD is:

$$\mathbf{x}(k+1) = W\mathbf{x}(k) - c\nabla f(\mathbf{x}(k)),$$

where $c > 0$ is the stepsize. So it is the discretization Case III of the system with the corresponding continuous-time controllers:

$$u_{g,x} = (I-W)\mathbf{x}, \quad u_{\ell,x} = \nabla f(\mathbf{x}).$$

DLM [17]: The update step of Decentralized Linearized-ADMM (DLM) algorithm is:

$$\begin{aligned}\mathbf{x}(k+1) &= \mathbf{x}(k) - \eta\left(\nabla f(\mathbf{x}(k)) + c(I-W)\mathbf{x}(k) + \mathbf{v}(k)\right),\\ \mathbf{v}(k+1) &= \mathbf{v}(k) + c(I-W)\mathbf{x}(k+1).\end{aligned}$$

So it is the discretization Case III of the system with the corresponding continuous-time controllers:

$$u_{g,x} = c(I-W)\mathbf{x} + \mathbf{v}, \quad u_{g,v} = (I-W)\mathbf{x},$$
$$u_{\ell,x} = \nabla f(\mathbf{x}), \quad u_{\ell,v} = 0.$$

Then we list some of the FL algorithms:

FedAvg [1]: The update step of FedAvg and Local GD is:

$$\mathbf{x}(k+1) = \begin{cases} \mathbf{x}(k) - \eta\nabla f(\mathbf{x}(k)),\ k \mod Q \neq 0, \\ R\mathbf{x}(k) - \eta\nabla f(\mathbf{x}(k)), k \mod Q = 0. \end{cases}$$

We can see that FedAvg cannot be translated into a continuous-time system as it does not have a persistent GCFL:

$$u_{g,x} = \begin{cases} 0,\ t \neq k\tau_g, \\ (I-R)\mathbf{x}(t)\delta(t), t = k\tau_g = 0, \end{cases}$$

where $\delta(t)$ denotes the Dirac delta function.

FedProx [14]: By assuming the local problem of FedProx is solve by gradient descent, the update step of FedProx is:

$$\mathbf{x}(k+1) = \begin{cases} \mathbf{x}(k) - \eta_1\nabla f(\mathbf{x}(k)) - \eta_2(\mathbf{x}(k) - \mathbf{x}(k_0)),\ k \mod Q \neq 0,\ k_0 = k - (k \mod Q), \\ R\mathbf{x}(k) - \eta_1\nabla f(\mathbf{x}(k)) - \eta_2(\mathbf{x}(k) - \mathbf{x}(k_0)), k \mod Q = 0,\ k_0 = k. \end{cases}$$

So it is the discretization Case I or IV of the system with the corresponding continuous-time controllers:

$$u_{g,x} = (I-R)\mathbf{x}, \quad u_{\ell,x} = \nabla f(\mathbf{x}).$$

FedPD [34]: By assuming the local problem of FedPD is solve by gradient descent, the update step of FedPD is:

$$\mathbf{x}(k+1) = \mathbf{x}(k) - \eta_1(\nabla f(\mathbf{x}(k)) + \mathbf{z}(k) + \eta_2(\mathbf{x}(k_0) - R\mathbf{x}(k_0))),\ k_0 = k - (k \mod Q),$$
$$\mathbf{v}(k+1) = \begin{cases} R\mathbf{x}(k), k \mod Q = 0 \\ \mathbf{v}(k), k \mod Q \neq 0, \end{cases}$$
$$\mathbf{z}(k+1) = \begin{cases} \mathbf{z}(k) + \frac{1}{\eta_2}(\mathbf{x}(k) - \mathbf{v}(k)), k \mod Q = 0 \\ \mathbf{z}(k), k \mod Q \neq 0, \end{cases}$$

So it is the discretization Case I or IV of the system with the corresponding continuous-time controllers:

$$u_{g,x} = \mathbf{x} - \mathbf{v}, \quad u_{g,v} = \mathbf{v} - R\mathbf{x},$$
$$u_{\ell,x} = \nabla f(\mathbf{x}) + \mathbf{z}, \quad u_{\ell,v} = 0, \quad u_{\ell,z} = -(\mathbf{x} - \mathbf{v}).$$

We can observe that $\mathbf{v}$ is tracking $R\mathbf{x}$. Replacing $\mathbf{v}$ with $R\mathbf{x}$ we have the following controllers:

$$\begin{aligned} u_{g,x} &= (I-R)\mathbf{x}, \\ u_{\ell,x} &= \nabla f(\mathbf{x}) + \mathbf{z} \quad u_{\ell,z} = -(I-R)\mathbf{x}. \end{aligned}$$

Finally, we give an example of the rate-optimal algorithms:
Scaffold [10]: The update step of Scaffold is:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{x}(k) - \eta_1(\nabla f(\mathbf{x}(k)) - \mathbf{z}(k) + \mathbf{v}_2(k_0)),\ k_0 = k - (k \mod Q), \\ \mathbf{v}_1(k+1) &= \begin{cases} \mathbf{v}_1(k) + \eta_2 R(\mathbf{x}(k) - \mathbf{v}_1(k)), & k \mod Q = 0 \\ \mathbf{v}_1(k), & k \mod Q \neq 0, \end{cases} \\ \mathbf{v}_2(k+1) &= \begin{cases} \mathbf{v}_2(k) - R(\mathbf{v}_2(k) - \frac{1}{Q\eta_1}(\mathbf{v}_1(k) - \mathbf{x}(k))), & k \mod Q = 0 \\ \mathbf{v}_2(k), & k \mod Q \neq 0, \end{cases} \\ \mathbf{z}(k+1) &= \mathbf{z}(k) - \frac{1}{Q}\mathbf{v}_2(k) - \frac{1}{Q\eta_1}(\mathbf{x}(k+1) - \mathbf{x}(k)), \end{aligned}$$

So it is the discretization Case IV of the system with the corresponding continuous-time controllers:

$$\begin{aligned} u_{g,x} &= \mathbf{v}_2, \quad u_{g,v} = [R(\mathbf{v}_1 - \mathbf{x}); R\mathbf{v}_2 - \frac{1}{\eta_1}R(\mathbf{v}_1 - \mathbf{x})], \\ u_{\ell,x} &= \nabla f(\mathbf{x}) - \mathbf{z}, \quad u_{\ell,v} = 0, \quad u_{\ell,z} = \mathbf{v}_2 + \frac{1}{\eta_1}\dot{\mathbf{x}}. \end{aligned}$$

xFilter [25]: The update step of xFilter is:

$$\begin{aligned} \mathbf{x}(k+1) &= \eta_1((1-\eta_2)I - \eta_2(I-W))\mathbf{x}(k) + (1-\eta_1)\mathbf{x}(k-1) + \eta_2\eta_1\mathbf{v}(k_0), k_0 = k - (k \mod K) \\ \mathbf{v}(k+1) &= \begin{cases} \mathbf{v}(k) + (\mathbf{z}_1(k) - \mathbf{z}_2(k)) - (I-W)\mathbf{x}(k), & k \mod K = 0 \\ \mathbf{v}(k), & k \mod K \neq 0, \end{cases} \\ \mathbf{z}_1(k+1) &= \begin{cases} \mathbf{x}(k) - \eta_3\nabla f(\mathbf{x}(k)), & k \mod K = 0 \\ \mathbf{z}_1(k), & k \mod K \neq 0, \end{cases} \\ \mathbf{z}_2(k+1) &= \begin{cases} \mathbf{z}_1(k), & k \mod K = 0 \\ \mathbf{z}_2(k), & k \mod K \neq 0, \end{cases} \end{aligned}$$

So it is the discretization Case V of the system with the corresponding continuous-time controllers:

$$\begin{aligned} u_{g,x} &= \frac{\eta_2}{2-\eta_2}W\mathbf{x}, \quad u_{g,v} = W\mathbf{x}, \\ u_{\ell,x} &= \frac{\eta_2}{2-\eta_2}\mathbf{v}, \quad u_{\ell,v} = (\mathbf{z}_1 - \mathbf{z}_2), \quad u_{\ell,z} = [\eta_3\nabla f(\mathbf{x}); -(\mathbf{z}_1 - \mathbf{z}_2)]. \end{aligned}$$

B.2. Existing Algorithms Connections

From the previous section, we have identified the controllers used by each algorithm in their continuous-time counterparts.

Algorithm	Global Consensus	Local Computation	FL	RO	DO
DGD	$(I-W)\mathbf{y}$	$\nabla f(\mathbf{x})$	FedProx	–	DGD
DLM	$c(I-W)\mathbf{x}+\mathbf{v}$	$\nabla f(\mathbf{x})$	FedPD	–	DLM
xFilter	$(I-W)\mathbf{x}+\mathbf{v}$	$u_{\ell,v}=-\dot{\nabla} f(\mathbf{x})$	Scaffold	xFilter	–

Table 2: The summary of the controllers used in different algorithms. In GCFL and LCFL we abstract the most important steps of the controller.

We can summarize the above interpretation of the existing algorithm into Table 2. From the table, we can see that some of the algorithms corresponds to the same continuous-time dynamic system and the only difference in the way we discretize the system. For examples, FedPD and DLM have the same continuous-time dynamics while DLM corresponds to Case III that GCFL and LCFL have the same sampling intervals, while FedPD correspond to Case I,IV that $\tau_g = Q\tau_\ell$; Scaffold and xFilter also have the same continuous-time dynamics.

From the table we can see that there are many missing blanks. Each of these blanks represents a new algorithm. Also, we cam combine different GCFL and LCFL to create new algorithms that are not in this table. In the next section, we use DGT as an example to show how it can be extended to Case I,IV and to a different GCFL.

Appendix C. Example: Analysis and Extensions of Gradient Tracking

In this section, we use the well-known gradient tracking algorithm as an example to illustrate how our proposed framework can be used in practice to analyze algorithm behavior, and to facilitate the development of new algorithms.

C.1. The Gradient Tracking Algorithm

The iteration of the original gradient tracking algorithm is given below:

$$\begin{aligned}\mathbf{x}(k+1) &= W\mathbf{x}(k) - c\mathbf{v}(k),\\ \mathbf{v}(k+1) &= W\mathbf{v}(k) + \nabla f(\mathbf{x}(k+1)) - \nabla f(\mathbf{x}(k)),\end{aligned} \tag{23}$$

where $c>0$ is some stepsize. Note that the algorithm only has one auxiliary consensus state $\mathbf{v}$.

Under the assumption that a) W is symmetric and doubly stochastic; b) f_i's has Lipschitz gradients and non-convex; c) $\sum_i f_i$ is lower bounded, this algorithm converges to the stationary point of the problem at a rate of $\mathcal{O}(1/T)$ [18, 26].

Our approach is to first analyze the corresponding continuous-time double-feedback system, and apply appropriate discretization schemes and utilize the corresponding convergence results.

C.2. Continuous-time Analysis

We begin by analyzing the continuous-time counterpart of the gradient tracking algorithm. First, notice that the gradient tracking algorithm falls into the case that $\tau_g = \tau_\ell$, because communication and computation happen at the same time-scale. By letting $\tau_g = \tau_\ell \to 0$, we obtain the following continuous-time dynamic:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= -\eta_g(t)(I-W)\mathbf{x}(t) - \eta_\ell(t)(c\mathbf{v}(t)),\\ \dot{\mathbf{v}}(t) &= -\eta_g(t)(I-W)\mathbf{v}(t) + \eta_\ell(t)(\nabla \dot{f}(\mathbf{x})).\end{aligned} \tag{24}$$

where $\eta_g(t) = 1, \eta_\ell(t) = 1, \forall\, t$. In the above system, the global controllers given by:

$$u_{g,x} = (I - W)\mathbf{x}(t), \quad u_{g,v} = (I - W)\mathbf{v}(t),$$

and local update controllers given by:

$$u_{\ell,x} = \eta\mathbf{v}(t), \quad u_{\ell,v} = -\nabla\dot{f}(\mathbf{x}(t)),$$

where $\nabla\dot{f}(\mathbf{x}(t)) := \left\langle \nabla^2 f(\mathbf{x}(t)), \dot{\mathbf{x}}(t) \right\rangle$.

Next, let us verify the properties P1-P5. First, it is straightforward to prove P2 with the definition of u_g. For P1, we know that W is doubly-stochastic and symmetric and the communication graph is connected and $\mathbb{1}$ is an eigenvector of $I - W$. Therefore $\mathbf{y}$ is an averaging system with linear convergence rate C_g equals to the eigenvalue of $I - W$ with the second smallest magnitude.

For P3, we can verify it by the following steps:

$$\begin{aligned}
\left\|G_{\ell,x}(x, v; f_i) - G_{\ell,x}(x', v'; f_i)\right\| &= \eta \left\|v - v'\right\| \\
\left\|G_{\ell,v}(x, v; f_i) - G_{\ell,x}(x', v'; f_i)\right\| &= \left\|\left\langle \nabla^2 f_i(x), \dot{x}\right\rangle - \left\langle \nabla^2 f_i(x'), \dot{x}'\right\rangle\right\| \\
&\leq \left(\left\|\nabla^2 f_i(x)\right\| + \left\|\nabla^2 f_i(x')\right\|\right) \left\|\dot{x} - \dot{x}'\right\| \\
&\leq 2cL_f \left\|v - v'\right\|,
\end{aligned}$$

where L_f is the constant of the Lipschitz gradient. So the smoothness constant of the local controller g_ℓ can be expressed as $L = c\max\{2L_f, 1\}$.

Finally, to check whether the LCFL satisfies P4, let us initialize $\mathbf{v}(t) = \nabla f(\mathbf{x}(t))$, and assume that $\eta_g(t) = 0$ in (24), that is, the GCFL is inactive. Then we have:

$$\mathbf{v}(t + \tau) = \nabla f(\mathbf{x}(t + \tau)) \tag{25}$$

$$\dot{\mathbf{x}}(t + \tau) = -c\mathbf{v}(t + \tau) = -c\nabla f(\mathbf{x}(t + \tau)). \tag{26}$$

The algorithm becomes the gradient flow algorithm that satisfies P4 [24].

Finally, we verify P5. We can compute $\dot{\mathcal{E}}(t)$ as follows:

$$\begin{aligned}
\dot{\mathcal{E}}(t) &= -\left\langle \nabla f(\bar{\mathbf{x}}(t)), \frac{1}{N}\sum_{i=1}^{N} u_{\ell,x}(t) \right\rangle - \langle (I - R)\cdot\mathbf{y}(t), u_{g,y}(t) + u_{\ell,y}(t)\rangle \\
&\overset{(24)}{=} -\langle \nabla f(\bar{\mathbf{x}}(t)), c\bar{\mathbf{v}}(t)\rangle - \langle (I - R)\cdot\mathbf{y}(t), (I - W)\cdot\mathbf{y}(t)\rangle \\
&\quad - \langle (I - R)\cdot\mathbf{x}(t), c\mathbf{v}(t)\rangle + \langle (I - R)\cdot\mathbf{v}(t), \nabla f(\mathbf{x}(t)) - \nabla f(\mathbf{z}(t))\rangle.
\end{aligned} \tag{27}$$

Then we bound each term on the RHS above separately, and finally integrate. The detailed derivation is omitted due to space consideration.The final bound we can obtain is:

$$\begin{aligned}
\int_0^t \dot{\mathcal{E}} \leq &-\frac{c}{2}\int_0^t \left\|\nabla f(\bar{\mathbf{x}}(\tau))\right\|^2 \mathrm{d}\tau - \frac{c - 8L_f c^2/\beta}{2}\int_0^t \left\|\bar{\mathbf{v}}(\tau)\right\|^2 \mathrm{d}\tau \\
&- (C_g - \frac{c + 2cL_f + \beta + 16cL_f/\beta}{2}) \cdot \int_0^t \left\|(I - R)\cdot\mathbf{y}(\tau)\right\|^2 \mathrm{d}\tau.
\end{aligned}$$

By choosing $\beta < C_g/2$, $\frac{C_g^2}{64L_f} \leq c \leq \frac{C_g^2}{32L_f}$, we can verify that the dynamics of the continuous-time system (24) satisfy (6), with $\gamma_1(t) \geq \frac{C_g^2}{128L_f}$ and $\gamma_2(t) \geq \frac{C_g}{4}$. Applying Theorem 3, we know that continuous-time gradient tracking algorithm converges in $\mathcal{O}(1/T)$.

C.2.1. NEW ALGORITHM DESIGN

Now that we have verified properties P1-P5 for the continuous-time system (24), we can derive a number of related algorithms by adjusting the discretization schemes, or by changing the GCFL.

Let us first consider changing the discretization scheme from Case III to Case IV, where $\tau_g = Q\tau_\ell > 0$. In this case, there will be Q local computation steps between every two communication steps. This kind of update scheme is closely related to algorithms in FL, and we refer to the resulting algorithm the Decentralized Federated Gradient Tracking (D-FedGT) algorithm. Its steps are listed below (where $k_0 = k - (k \bmod Q)$):

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{x}(k) - \tau_\ell \mathbf{v}(k) - \tau_g(I - W)\mathbf{x}(k_0), \\ \mathbf{v}(k+1) &= \mathbf{v}(k) + \nabla f(\mathbf{x}(k+1)) - \nabla f(\mathbf{x}_k) - \tau_g(I - W)\mathbf{v}(k_0). \end{aligned} \tag{28}$$

By applying Theorem 1, we can directly obtain that this new algorithm also converges with rate $\mathcal{O}(\frac{1}{T})$ with properly chosen constant τ_ℓ, τ_g and Q.

Second, we can replace the GCFL of the DGT with an *accelerated* consensus controller [6]. This leads to the a new Accelerated Gradient Tracking (AGT) algorithm:

$$\begin{aligned} &\mathbf{x}(k+1) = \mathbf{x}(k) - \eta'_\ell \mathbf{v}(k) - \eta'_g(1+c)\mathbf{x}(k) + c\mathbf{v}_x(k), \\ &\mathbf{v}(k+1) = \mathbf{v}(k) + \nabla f(\mathbf{x}(k+1)) - \nabla f(\mathbf{x}(k)) - \eta_g(1+c)\mathbf{v}(k) + c\mathbf{v}_v(k), \\ &\mathbf{v}_x(k+1) = \mathbf{x}(k), \quad \mathbf{v}_v(k+1) = \mathbf{v}(k), \ \text{ where } c := \frac{1 - \sqrt{1-\lambda_2(W)}}{1 + \sqrt{1-\lambda_2(W)^2}}. \end{aligned} \tag{29}$$

Then by examining P1, we know that the network dependency of the new algorithm improved from C_g to $\hat{C}_g = C_g \cdot \frac{\sqrt{C_g}+\sqrt{2-C_g}}{\sqrt{C_g}+C_g\sqrt{2-C_g}} > C_g$. Then according to the derivation in the last subsection, we have $\gamma_2(t) \geq \frac{\hat{C}_g}{4}$. Finally, we can apply Theorem 1, and asserts that the new algorithm improves the convergence speed from $\mathcal{O}(\frac{1}{C_g T})$ to $\mathcal{O}(\frac{1}{\hat{C}_g T})$.

C.3. Numerical Results

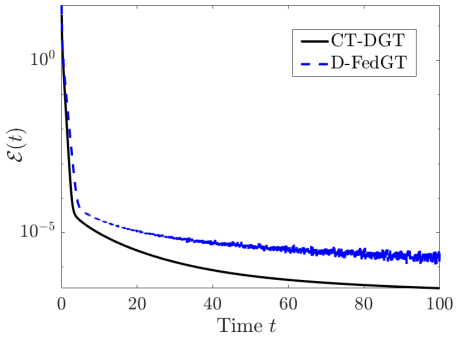
We provide numerical results for implementations of Continuous-time (CT) DGT, the D-FedGT and D-AGT algorithms discussed in the previous subsection. We first verify an observation from Theorem 1, that discretization slows down the convergence speed of the system. Towards this end, we conduct numerical experiments with different discretization patterns and compare the convergence speed in terms of the stationarity gap. Then we compare the convergence speed of CT-DGT and CT-AGT, to demonstrate the benefit of changing the controller in the GCFL from the standard consensus controller to the accelerated one.

In the experiments, we consider the non-convex regularized logistic regression problem:

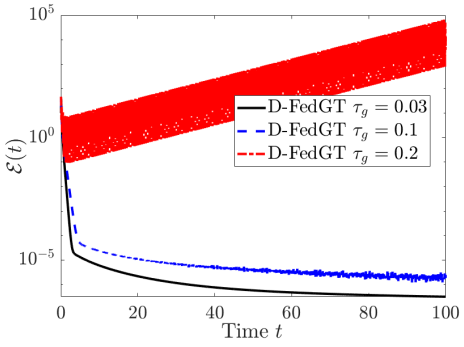
$$f_i(\mathbf{x}; (\mathbf{a}_i, b_i)) = \log(1 + \exp(-b_i \mathbf{x}^T \mathbf{a}_i)) + \sum_{d=1}^{d_x} \frac{\beta\alpha(\mathbf{x}[d])^2}{1 + \alpha(\mathbf{x}[d])^2},$$

where $\mathbf{a}_i$ denotes the features and b_i denotes the labels of the dataset on the i^{th} agent. We set the number of agent $N = 20$ and each agent has local dataset of size 500. We use an Erdős–Rényi random graph with density 0.5 for the network and optimize the weight matrix W to achieve the optimal C_g. We set $c = 1$ for gradient tracking algorithm.

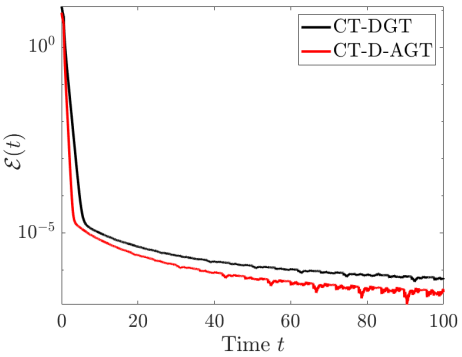
We first compare CT-DGT ($\tau_g = \tau_\ell = 0$) and D-FedGT ($\tau_g = 0.1, \tau_\ell = 0.005, Q = 20$), the result of CT-DGT and D-FedGT is showed in Figure 4(*a*)subfigure. We can see that by discretizing each loop, the system converges slower as compared with the continuous time system. Figure 4(*b*)subfigure shows the convergence behavior of the D-FedGT algorithm with different τ_g. We observe that by increasing the sampling interval for GCFL, the convergence of the system slows down and it eventually diverges. Figure 4(*c*)subfigure and Figure 4(*d*)subfigure show the convergence results of D-AGT compared with DGT in both continuous time and in Case III. We observe that by changing the GCFL, D-AGT converges faster than DGT.



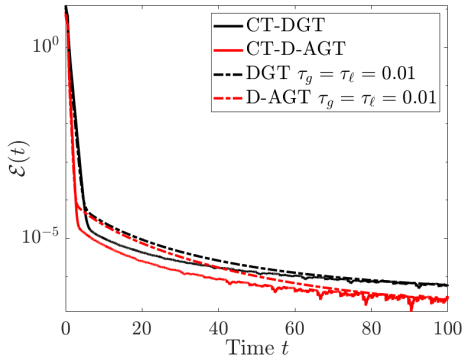
(*a*) The evoluation of the Energy function $\mathcal{E}(t)$ of CT-CGT, D-FedGT.



(*b*) Energy function $\mathcal{E}(t)$ of D-FedGT with different intervals τ_g.



(*c*) The evolution of the Energy function $\mathcal{E}(t)$ of CT-DGT and CT-D-AGT.



(*d*) The evolution of the Energy function $\mathcal{E}(t)$ of DGT and D-AGT.

Figure 4: The performance of Continuous-GT, D-FedGT, D-MGT and AGT.