Gradient and Projection Free Distributed Online Min-Max Resource Optimization

Jingrong Wang JR.WANG@MAIL.UTORONTO.CA
Ben Liang LIANG@ECE.UTORONTO.CA
Department of Electrical and Computer Engineering, University of Toronto, Canada

Editors: R. Firoozi, N. Mehr, E. Yel, R. Antonova, J. Bohg, M. Schwager, M. Kochenderfer

Abstract

We consider distributed online min-max resource allocation with a set of parallel agents and a parameter server. Our goal is to minimize the pointwise maximum over a set of time-varying and decreasing cost functions, without a priori information about these functions. We propose a novel online algorithm, termed Distributed Online resource Re-Allocation (DORA), where non-stragglers learn to relinquish resource and share resource with stragglers. A notable feature of DORA is that it does not require gradient calculation or projection operation, unlike most existing online optimization strategies. This allows it to substantially reduce the computation overhead in large-scale and distributed networks. We analyze the worst-case performance of DORA and derive an upper bound on its dynamic regret for non-convex functions. We further consider an application to the bandwidth allocation problem in distributed online machine learning. Our numerical study demonstrates the efficacy of the proposed solution and its performance advantage over gradient- and/or projection-based resource allocation algorithms in reducing wall-clock time.

Keywords: Online min-max resource allocation, online learning.

1. Introduction

We consider an online min-max resource allocation problem with a set of agents. The objective is to minimize the accumulation of the pointwise maximum over a set of *time-varying* local cost functions of the agents, subject to a limited resource budget. The cost functions are decreasing with respect to the resource allocated to that agent. Optimization decisions are made round by round, while the cost functions vary arbitrarily over time and are revealed only *after* decision making in each round. Our objective is to decide a sequence of allocation schemes as delayed information about the cost functions becomes available over time.

Many practical applications provide strong motivation for this special type of minimization problem, e.g., minimum spanning tree (Yu (1998)), online routing (Hazan (2016)), fair resource allocation in wireless cellular networks (Boche et al. (2005)), and data/task parallelism in distributed learning/computing (Dean and Ghemawat (2008)). Take synchronous distributed learning in a parameter-server architecture as an example. The learning model is updated only after all agents send their local gradients to the parameter server. Therefore, the training time in each round, which consists of both the computation and communication times, is determined by the *straggler*, i.e., the last agent to send its local gradient. Thus, communication resource allocation for distributed computing can be formulated as a min-max optimization problem, minimizing the worst delay among all agents (Du et al. (2017)). Furthermore, both the computation and communication times are time-varying in an unpredictable fashion. This is mainly due to the wireless fading channels and the randomness of computing demands and available capacity. In this dynamic environment, an *online* min-max resource allocation algorithm is of interest.

Online min-max optimization belongs to the family of online optimization, which has been extensively studied in a wide variety of real-life applications, but most of the existing works focus on convex cost functions (Shalev-Shwartz (2012)). Typically, an online algorithm proceeds in a round-by-round manner. In each round, the agent first selects a decision. Then, the environment reveals to the agent the cost function for this round, and the agent updates its strategy for the next round accordingly. In this context, *regret* is introduced to measure the performance of an online algorithm versus a comparator. In particular, the *static* regret uses a fixed minimizer of the accumulated cost over time as the comparator. However, considering the inherent dynamics of real-world systems, the *dynamic* regret is practically more meaningful (Zinkevich (2003)). It measures the difference between the sequence generated by an online algorithm and a sequence of time-varying comparators, most commonly the instantaneous minimizers.

Online min-max optimization problems have unique characteristics. First, the pointwise maximum among the set of cost functions is generally not continuously differentiable, so that we require subgradients instead of gradients. Second, the performance guarantees of computationally efficient projection-free algorithms, such as (Hazan and Kale (2012)), require strongly convex and smooth cost functions, which do not apply to the min-max problem. The need for *distributed* implementation brings additional challenges. Unlike in the more common min-sum optimization, the max operation couples multiple local cost functions, requiring some decomposition before problem-solving in a distributed manner. This is exacerbated by the coupling constraints, e.g., due to shared communication bandwidth among the agents, which adds to the unpredictability of the time-varying global cost function in the online setting.

In response to the above challenges, we propose a new distributed online algorithm named Distributed Online resource Re-Allocation (`DORA`), to solve an online min-max optimization problem in a multi-agent system with coupling linear constraints. Our main contributions include the following: *First*, `DORA` is a distributed algorithm that leverages the unique structure of the online min-max optimization problem. We denote the agent with the highest cost as *straggler*. Under `DORA`, the agents relinquish a carefully chosen amount of resource to the straggler in the previous online round to help improve its performance. To achieve this, each agent only obtains an incomplete view of the system and does not need to keep a full copy of the others' decisions. More importantly, `DORA` does not require gradient calculation or projection operation, so it has much lower computational complexity than generic online solutions. *Second*, We analyze the dynamic regret of `DORA` for general decreasing cost functions. This compares favorably with the best existing solutions considering `DORA`'s reduced computational complexity, making it an attractive alternative. *Finally*, We apply the proposed solution to online resource allocation in distributed machine learning, which trains a convolutional neural network model for image identification with a set of edge devices in a parameter-server architecture. Our experimental results demonstrate the performance advantage of `DORA` over existing online algorithms that are based on gradient and/or projection, in terms of significantly faster convergence and reduced training time.

2. Background and Related Work

2.1. Dynamic Regret of Gradient-based and Projection-based Online Optimization

In online optimization, an agent aims to make a sequence of decisions to minimize the accumulation of a sequence of cost functions, without knowing the cost functions ahead of time. In each round t, the agent selects a decision $\mathbf{x}_t$ from a feasible set $\mathcal{F}$. Then, the environment reveals cost function f_t to the agent, and the agent suffers an instantaneous loss $f_t(\mathbf{x}_t)$ in this round.

Regret is introduced to measure the performance of an online algorithm. We focus on the more practically meaningful dynamic regret. In recent literature, the dynamic regret is commonly defined as $\text{Reg}_T^d = \sum_{t \in \mathcal{T}} f_t(\mathbf{x}_t) - \sum_{t \in \mathcal{T}} f_t(\mathbf{x}_t^*)$, where $\mathbf{x}_t^* \in \arg\min_{\mathbf{x} \in \mathcal{F}} f_t(\mathbf{x})$. Since the cost functions fluctuate arbitrarily, the dynamic regret in the worst case scales linearly. Typically, it is related to some *regularity measures*, which indicate how dynamic the environment is. For example, the path-length of the dynamic minimizers is defined as $P_T = \sum_{t=2}^{T} ||\mathbf{x}_{t-1}^* - \mathbf{x}_t^*||_2$. For centralized online convex optimization (OCO), Zinkevich (2003) proved that with online (projected) gradient descent (OGD), the dynamic regret is upper bounded by $O(\sqrt{T}(1 + P_T))$. Zhang et al. (2018) improved the bound to $O(\sqrt{T(1 + P_T)})$ by running multiple OGD operations in parallel. For centralized online non-convex optimization (ONCO), Héliou et al. (2020) proved that the dual averaging algorithm enjoys a dynamic bound of $O(T^{\frac{2}{3}} V_T^{1/3})$, where V_T is the time-accumulated variation of the cost functions. If the cost functions are pseudo-convex, Gao et al. (2018) improved the bound to $O(\sqrt{T(1 + P_T)})$. Additional dynamic regret bounds have also been derived for centralized OCO algorithms, e.g., Mokhtari et al. (2016); Zhang et al. (2017a); Besbes et al. (2015).

In distributed implementation, most of the recent works have proposed methods that provide dynamic regret guarantee under various assumptions on the convexity and smoothness of the objective functions (Shahrampour and Jadbabaie (2018); Zhang et al. (2019); Dixit et al. (2019); Lu et al. (2020); Sharma et al. (2021); Eshraghi and Liang (2020); Yi et al. (2020); Li et al. (2021)). To the best of our knowledge, for gradient/projection-based distributed OCO, the tightest known dynamic regret bound for general convex cost functions is $O(\sqrt{T(1 + P_T)})$ (Shahrampour and Jadbabaie (2018)).There is no dynamic regret bound developed for distributed ONCO.

In all of the above works, the local solution of each agent in each round is updated with the calculated gradient and needs to be projected back to the domain of interest to restore feasibility. However, even the projection onto a simplex requires non-trivial computation time (Liu and Ye (2009)). In contrast, our work does not require gradient calculation or projection operation.

2.2. Gradient-free and Projection-free Online Optimization

Existing gradient-free algorithms replace exact gradient calculation with efficient gradient approximation (Liu et al. (2020a)). The FKM algorithm, named after its authors, uses spherical gradient estimators and then conducts gradient descent with projection (Flaxman et al. (2005); Héliou et al. (2020)). For OCO, with carefully designed diminishing step sizes, FKM and its variants can achieve dynamic regret that is upper bounded by $O(T^{\frac{4}{5}} V_T^{\frac{1}{5}})$ (Besbes et al. (2015)). For ONCO, Héliou et al. (2021) showed that the dual averaging algorithm with bandit feedback achieves a dynamic regret bound of $O(T^{\frac{n+2}{n+3}} V_T^{\frac{1}{n+3}})$, where $n \geq 1$ is the dimension of the decision variables. This bound is worse than that of gradient-based algorithms due to the biased gradient estimation.

Among projection-free solutions, the Frank-Wolfe method, also known as the conditional gradient method, replaces projections with linear optimization over the feasible set (Frank and Wolfe

(1956)). In the centralized setup, Hazan and Kale (2012) proposed an online conditional gradient (OCG) algorithm, achieving $O(T^{\frac{3}{4}})$ static regret for convex functions. If the functions are convex and smooth, with the Follow-the-Perturbed-Leader method, the static regret and dynamic regret of OCG were improved to $O(T^{\frac{2}{3}})$ (Hazan and Minasyan (2020)) and $O(T^{\frac{2}{3}}V_T^{1/3})$ (Wan et al. (2021)), respectively. There is no dynamic regret bound developed for projection-free ONCO.

In distributed implementation, Zhang et al. (2017b) proposed a distributed variant of OCG, achieving the same static regret bound as the centralized one. It is worth noting that the regret bounds of OCG are also worse than those of projection-based algorithms, suggesting performance penalty in exchange for reducing computation complexity. Since the above works require strongly convex and smooth cost functions, their performance guarantees do not apply to the min-max resource allocation problem in our work. Furthermore, no dynamic regret bound is known for distributed projection-free OCG.

Unlike existing gradient/projection-free algorithms, we propose to update the decisions online by directly calculating the amount of resource that the agents can relinquish and share with the stragglers. We will show in Section 6 that this can substantially reduce the computation time in comparison with FKM and OCG.

2.3. Min-Max Optimization

The offline min-max optimization problem, i.e., with fixed cost functions known ahead of time, has been studied in Liuzzi et al. (2006); Srivastava et al. (2011, 2013); Notarnicola et al. (2019); Liu et al. (2020b); Wu et al. (2021). These methods cannot be applied to our online problem. For online min-max convex optimization, Bampis et al. (2020) directly applied OGD to the min-max vertex cover problem, using a subgradient whenever the gradient does not exist. However, they overlooked the special structure of the min-max problem. As a result, there was no improvement in the static regret over general online algorithms. There is no existing work on distributed online algorithms specifically designed for the min-max problem.

3. Problem Formulation

Consider a distributed system with a set of parallel agents $\mathcal{N} = \{1, ..., N\}$ and a parameter server. The time is slotted by rounds $\mathcal{T} = \{1, ..., T\}$.

In each round t, let $x_{i,t}$ denote the amount of resource allocated to agent i. Corresponding to $x_{i,t}$, the local cost function at agent i is $f_{i,t}(x_{i,t})$. We note that $f_{i,t}$ may depend on other random factors, so it varies over time. We assume that $f_{i,t}$ at any time t is decreasing, but not necessarily strictly decreasing, in $x_{i,t}$. As shown in Section 6, an important example of decreasing cost function is where $x_{i,t}$ represents the allocated channel bandwidth and $f_{i,t}$ represents the communication delay plus any quantity independent of $x_{i,t}$.

Let $\mathbf{x}_t = (x_{1,t}, ..., x_{N,t})$. Then, the decision variables $\{\mathbf{x}_t\}_{t\in\mathcal{T}}$ is a sequence of resource allocation schemes. The global cost function in round t is the pointwise maximum over the set of time-varying local cost functions $\{f_{i,t},\ i \in \mathcal{N}\}$, i.e.,

$$f_t(\mathbf{x}_t) = \max_{i\in\mathcal{N}} f_{i,t}(x_{i,t}). \tag{1}$$

As an example, in edge learning, since each round is completed only when the last agent's task is finished, $f_t(\mathbf{x}_t)$ represents the duration of round t.

We assume that the agents can observe $f_{i,t}$ but only at the end of round t after each agent has been allocated $x_{i,t}$ and completed its task. In other words, we do not assume prior knowledge on how $x_{i,t}$ is mapped to $f_{i,t}$. This matches the standard form of online optimization and is a natural system model in many practical applications, e.g, due to the uncertain available processing power and wireless channel conditions in edge learning. Furthermore, once all agents complete their tasks at the end of round t, all local costs $f_{i,t}(x_{i,t})$, and thus the global cost in round t, are known to the parameter server.

Our objective is to minimize the accumulation of the sequence of global cost functions over time, subject to a limited resource budget. For notation simplicity, the resource budget is normalized to 1. Then, the problem is formulated as follows:

$$\min_{\{\mathbf{x}_t\}_{t\in\mathcal{T}}} \quad \sum_{t\in\mathcal{T}} f_t(\mathbf{x}_t), \tag{2}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{N}} x_{i,t} \le 1, \forall t \in \mathcal{T}, \tag{3}$$

$$x_{i,t} \ge 0, \forall i \in \mathcal{N}, \forall t \in \mathcal{T}. \tag{4}$$

As explained in Section 1, there are broad applications to this optimization formulation. If the information of local cost functions were known a priori, this problem could be solved as a traditional offline optimization problem. However, since in many applications the cost functions become known only at the end of a round, we seek an *online* solution to this problem. Moreover, in large-scale systems, a centralized solution, e.g., one completely computed by the parameter server, would require intense information exchange that incurs high communication overhead and high computation requirement at the parameter server. Instead, we are interested in developing a *distributed* algorithm. As explained in Section 2, existing distributed online solutions are inefficient for this problem. Therefore, we will next present a new gradient-free and projection-free algorithm that substantially reduces the required computation.

4. Distributed Online Resource Re-Allocation

In this section, we present the design of DORA to solve the online min-max optimization problem in (2)-(4). In DORA, the agents simultaneously learn to track the unknown local cost functions over time and solve the problem with help from the parameter server.

4.1. Reformulation of the Cost Function in Round *t*

To transform the global cost function toward distributed computation, we first consider the *instantaneous* optimization problem in each round t, i.e., where the objective is $\min_{\mathbf{x}_t} \max_{i\in\mathcal{N}} f_{i,t}(x_{i,t})$. The equivalent epigraph representation of this problem is

$$\min_{\eta_t,\mathbf{x}_t} \quad \eta_t, \tag{5}$$

$$\text{s.t.} \quad f_{i,t}(x_{i,t}) \le \eta_t, \forall i \in \mathcal{N}, \tag{6}$$

$$\sum_{i\in\mathcal{N}} x_{i,t} \le 1, \tag{7}$$

$$x_{i,t} \ge 0, \forall i \in \mathcal{N}. \tag{8}$$

We can further penalize the constraints (6) with some multiplier $r_{i,t} > 1$. An equivalent form of problem (5) can be obtained as follows (Srivastava et al. (2013)):

$$\min_{\eta_t,\mathbf{x}_t} \quad \sum_{i\in\mathcal{N}} \frac{\eta_t}{N} + r_{i,t}[f_{i,t}(x_{i,t}) - \eta_t]^+, \tag{9}$$

$$\text{s.t.} \quad (7) - (8).$$

where $[\cdot]^+ = \max\{0, \cdot\}$.An important property of problem (9) is that $r_{i,t}$ can be *arbitrarily* chosen by each agent independently of the others, which allows distributed implementation.

However, we note that (9) is not continuously differentiable. If we used it over time to construct a conventional online optimization, we would face the same challenges as solving our original online min-max problem. Therefore, we seek a new solution. Next, we will present how problem (9) can be used as the starting point to build the proposed DORA algorithm.

4.2. Distributed Online Optimization

A naive solution to problem (9) is to use Lagrange decomposition. By penalizing the coupling constaint (7) with some partial Lagrange multiplier $\pi_t \geq 0$, the partial Lagrangian of (9) is

$$L_t(\eta_t, \mathbf{x}_t, \pi_t) = \sum_{i \in \mathcal{N}} \left(\frac{\eta_t}{N} + r_{i,t}[f_{i,t}(x_{i,t}) - \eta_t]^+ \right) + \pi_t \left(\sum_{i \in \mathcal{N}} x_{i,t} - 1 \right). \tag{10}$$

For convex problems, strong duality holds under Slater's condition. However, even for convex problems, it is unclear how to find the optimal π_t^*.

To overcome this obstacle, we first observe in the following lemma that a *local* version of the Lagrangian admits a closed-form minimizer for any choice of η_t in the range of $f_t(\mathbf{x})$ for $\mathbf{x}$ satisfying (7) and (8), even though the optimal π_t^* is unknown. Let $x'_{i,t}$ be an arbitrary solution to $f_{i,t}(x'_{i,t}) = \eta_t$, i.e.,

$$x'_{i,t} \in f_{i,t}^{-1}(\eta_t). \tag{11}$$

Since $f_{i,t}$ is a monotonic function, root $x'_{i,t}$ can be found efficiently by bisection (Hansen and Patrick (1976)). As we will see later, this is an important quantity to our algorithm design. Further details are given in Remark 1. For now, in the following lemma, we show that $x'_{i,t}$ is a minimizer of the local Lagrangian. Its proof can be found in Appendix A in Wang and Liang (2021).

Lemma 1 *Suppose $f_{i,t}$ is convex for all i and t. Let $(\mathbf{x}_t^*, \eta_t^*, \pi_t^*)$ denote a minimizer of (10). Then, for any choice of η_t in the range of $f_t(\mathbf{x})$ for $\mathbf{x}$ satisfying (7) and (8), $x'_{i,t}$ is a minimizer of the local Lagrangian*

$$L_{i,t}(x_{i,t}) = r_{i,t}[f_{i,t}(x_{i,t}) - \eta_t]^+ + \pi_t^* x_{i,t}. \tag{12}$$

Unfortunately, if each agent updates its $x_{i,t}$ based on (11), the resultant solution is still not optimal to problem (9). This is because the global cost can be further reduced by simultaneously allocating the remaining resource budget to each agent. Interestingly, a similar phenomenon has been observed in general unconstrained sum minimization problems (Li et al. (2020)). This inspires the proposed DORA algorithm, which will be shown in Section 5 to provide bounded dynamic regret for our online min-max problem.

The intuition of DORA is to make fast agents relinquish an appropriate amount of resource and give it to the agent who was the straggler in the *previous* online round. Thus, in each round, the agents who are non-stragglers in the previous online round move towards the minimizer of their local Lagrangian in (12), while the stragglers will be allocated the remaining resource budget.

Algorithm 1: Distributed Online Resource Re-Allocation

```
Input: Number of rounds $T$, and step size $\alpha$.
Initialization: Arbitrary initial allocation $\mathbf{x}_0 \in \mathcal{F}$.
Agent $i = 1, 2, \dots, N$ runs in parallel:
for round $t = 1, 2, ..., T$ do
    Observe $f_{i,t-1}(\cdot)$;
    Receive $f_{t-1}(\mathbf{x}_{t-1})$ from PS;
    Compute $x_{i,t}$ using (13);
    Send $x_{i,t}$ to the parameter server;
end
Parameter server runs:
for round $t = 1, 2, ..., T$ do
    Receive $x_{i,t}$ from agent $i$;
    Update $x_{s_t,t}$ using (14);
    Allocate resource based on $\mathbf{x}_t$;
    Observe and send $f_t(\mathbf{x}_t)$ to all agents;
end
```

The pseudocode of DORA is shown in Algorithm 1. After observing the local cost function $f_{i,t-1}(\cdot)$ in the previous round and receiving the feedback $f_{t-1}(\mathbf{x}_{t-1})$ from the parameter server, each agent adjusts its decision by moving towards the minimizer of its local Lagrangian with any step size $0 < \alpha < 1$, i.e.,

$$x_{i,t} \leftarrow x_{i,t-1} - \alpha(x_{i,t-1} - x'_{i,t-1}), \forall i \in \mathcal{N}, \tag{13}$$

where $x'_{i,t-1}$ is a minimizer of (12) in round $t-1$ and can be computed as in (11). Afterward, the local decisions are synchronized with the parameter server. After receiving $x_{i,t}$ from each agent, the resource relinquished by non-stragglers is re-allocated to the agent who is the straggler in the previous round:

$$x_{s_{t-1},t} \leftarrow 1 - \sum_{i \neq s_{t-1}} x_{i,t} = x_{s_{t-1},t-1} - \alpha \sum_{i \neq s_{t-1}} (x'_{i,t-1} - x_{i,t-1}), \tag{14}$$

where $s_{t-1} = \arg\max_{i \in \mathcal{N}} f_{i,t-1}(x_{i,t-1})$ denotes the straggler in round $t-1$. Then, the parameter server observes the value of the global cost incurred in this round and sends it to all agents. It can be easily verified that constraint (3) is always satisfied in re-allocation since the relinquished resource is shared among stragglers.

Remark 1 *$x'_{i,t-1}$ can be interpreted as the minimum resource agent i needs to maintain such that $f_{i,t-1}(x'_{i,t-1}) \leq f_{t-1}(\mathbf{x}_{t-1})$. Thus, $(x_{i,t-1} - x'_{i,t-1})$ represents the maximum resource the agent can relinquish without making itself the straggler based on the historical loss function in round $t-1$. Furthermore, $\alpha(x_{i,t-1} - x'_{i,t-1}), \forall \alpha \in [0, 1]$ can be viewed as the amount of resource that agent i chooses to relinquish and share with the straggler. It is easy to verify that the straggler does not relinquish its resource at this step since $x_{s_{t-1},t-1} = x'_{s_{t-1},t-1}$.*

Remark 2 *The rationale behind the step size $\alpha < 1$ in (13) is that, for instantaneous optimization, if each agent maintains the minimum resource $x'_{i,t}$, the total resource relinquished to the straggler s_t could exceed its optimal allocation $x^*_{s_t,t}$. Then the total resource shared by the non-stragglers would exceed the straggler's needs. The detailed proof is presented in the next section. Rather than directly minimizing (12), the agents who are non-stragglers in the previous round take a step toward the minimum with step size α.*

From Algorithm 1, we see that DORA has several advantages in terms of its ease of implementation: 1) it avoids gradient calculation and projection onto the feasible set, 2) each agent only needs to keep its local variable rather than a full copy of all decision variables, and 3) communication only needs to be maintained between the agents and the server.

We emphasize here that the updates in (13) and (14) are different from gradient descent. In fact, since each agent only needs to compute $x'_{i,t-1}$ in each round, independently of the other agents, the overall computation complexity of DORA is $O(N)$ per round. In comparison, applying projected gradient-based online algorithms would lead to $O(N^2)$ computation complexity per round for gradient calculation alone (Boyd and Vandenberghe (2004)). Furthermore, since the computation complexity of projection is $O(N \log N)$ in Euclidean space (Liu and Ye (2009)), the overall computation complexity would be at least $O(N^2 \log N)$ per round. Therefore, the $O(N)$ complexity of DORA is a substantial improvement.

5. Dynamic Regret Analysis

Besides ease of implementation, another main advantage of `DORA` is that it provides strong performance guarantees in the form of bounded dynamic regret. For dynamic regret analysis, we make the following assumptions:

Assumption 1 (Monotonicity) *If $x \leq y$, then $f_{i,t}(x) \geq f_{i,t}(y), \forall i$ and t.*

Assumption 2 (Lipschitz) $|f_{i,t}(x) - f_{i,t}(y)| \leq L||x - y||, \forall i$ *and* t.

Theorem 2 *Consider the online min-max problem defined in (2) with Assumptions 1 and 2. With a fixed step size α, the dynamic regret Reg_T^d for the sequence of decisions $\mathbf{x}_t$ generated by `DORA` is upper bounded by $\mathrm{Reg}_T^d \leq \sqrt{TL^2(\frac{3}{2\alpha} + \frac{P_T}{\alpha} + \frac{T(4+\alpha)}{2})}$, where P_T is the path-length of the dynamic minimizers.*

Proof Please refer to Appendix B in Wang and Liang (2021). ■

6. Application to Distributed Learning in Mobile Edge Computing

To complement the theoretical findings in the previous section, we numerically study the performance of `DORA`, for an application to online resource allocation in distributed edge learning (Amiri and Gündüz (2020); Niknam et al. (2020)). We consider standard synchronous distributed learning, where multiple agents $i \in \mathcal{N}$ train the same learning model in parallel with their local training datasets and the parameter server updates the models only after aggregating all local gradients. The parameter server also functions as the edge server, which is responsible for resource allocation and necessary coordination. Therefore, the latency of each round depends on the agent with the highest latency. We are interested in minimizing the maximum latency per round over time. Therefore, we define the cost function $f_t(\mathbf{x}_t)$ as in (1). Then we can formulate resource allocation in distributed edge learning as an online min-max optimization problem as in (2).

The computation task associated with agent i in round t has a data size of $d_{i,t}$. For spectrum sharing among agents, we assume orthogonal frequency division. Therefore, the wireless communication delay for agent i is given by $f_{i,t}^{\mathrm{C}} = d_{i,t}/(x_{i,t} B \log(1 + \frac{h_{i,t}^2 p_{i,t}}{\sigma_n^2}))$, where we have used the Shannon bound for the data rate, $x_{i,t}$ is the fraction of bandwidth allocated to agent i, B is the total available bandwidth, $h_{i,t}^2$ is the channel power gain, $p_{i,t}$ is the transmit power, and σ_n^2 is the white noise power. The total delay due to agent i consists of both the communication delay and the processing delay, i.e., $f_{i,t}(x_{i,t}) = f_{i,t}^{\mathrm{C}} + f_{i,t}^{\mathrm{P}}$, where $f_{i,t}^{\mathrm{P}}$ is the time required for agent i to process its task in round t. Note that $f_{i,t}^{\mathrm{P}}$ depends on the computation intensity of the task and the available processing capacity of the agent, both of which can be time-varying and unpredictable. We do not require knowledge of $f_{i,t}^{\mathrm{P}}$ at the beginning of round t.

We assume five agents train the LeNet model in parallel, using the MNIST database (LeCun et al. (1998)). Our learning system is implemented with the distributed package (i.e., torch.distributed) in PyTorch. The hardware used for the experiments features a 2.9 GHz Intel Core i5 processor and 8 GB of memory. LeNet is a CNN composed of two convolutional layers, followed by two fully-connected layers and a softmax classifier. In total, we have 61,706 trainable parameters with a size of $d_{i,t} = 0.35$ MB, $\forall i, t$. We train LeNet with the cross-entropy loss and the Adam optimizer. The learning rate of LeNet model training is set to 0.001. The MNIST database has a training set of 60,000 handwritten digits images, each of which contains 28 x 28 greyscale pixels. It is equally partitioned into five local training datasets, each assigned to an agent with a batch size of 256. Therefore, we have $\lceil \frac{60{,}000}{5\times 256} \rceil = 47$ rounds in each training epoch. We set the number of training epochs to 10, i.e., 470 rounds. We measure and use the actual processing time $f_{i,t}^{\mathrm{P}}$ for training at each agent in each round on the processor.

The parameter server is placed at the center of a 500 m $\times$ 500 m area. The bandwidth resource budget B is 20 MHz. The channel power gain $h_{i,t}^2$ is generated as $h_0(\frac{D_0}{D_i})^n$, where $h_0 = -40$ dB is the path-loss constant, $D_0 = 1$m is the reference distance, D_i is the distance between agent i and the parameter server, and $n = 4$ is the path-loss component (Mao et al. (2016)). The transmission power $p_{i,t} = 1$ W, $\forall i, t$. The noise density is -174 dBm/Hz. We compare the performance of `DORA` with that of the following gradient- and/or projection-based benchmarks: **EQUAL**, where resource is equally shared among all edge devices; **OGD-OMM** (Bampis et al. (2020)), where projection onto the simplex is implemented using the method in (Blondel et al. (2014)); **OMD** (Shahrampour and Jadbabaie (2018)), where we use the Kullback-Leibler (KL) divergence; **FKM** (Flaxman et al. (2005)); **OCG** (Zhang et al. (2017b)); and **Dynamic OPT**, where we ideally assume a priori knowledge of all system variables, and we optimally solve the instantaneous problem in each round. All algorithms are initialized with equal bandwidth allocation. We use a fixed step size $\alpha = 0.02$ for resource allocation schemes `OGD-OMM`, `OMD`, and `DORA`.

Figs. 1 compares the performance of various algorithms over the number of rounds T in terms of per-round latency. Note that, besides serving as the optimum baseline, `Dynamic OPT` also provides an indication on how fast the environment fluctuates. We observe that `DORA` tracks closely to `Dynamic OPT` after the first few rounds. `EQUAL` incurs the worst latency since it ignores the heterogeneity among agents as well as the time-varying cost. `FKM`, `OMD`, `OGD-OMM`, and `OCG` gradually track close to `Dynamic OPT` as time goes on, but they require many more rounds than `DORA`. `OGD-OMM` and `OMD` adjust their decisions only using parameters relevant to the historical communication delay since the gradient of function $f_{i,t}$ only depends on the communication parameters. `FKM` converges more slowly than `OGD-OMM` since it depends on the accuracy of gradient estimation. For `OCG`, the decision is updated through a trade-off between the current decision and the output of the linear optimization, which allocates all resource to the straggler given the resource constraints. Therefore, non-stragglers in the current round could become stragglers in the next round, thus leading to frequent latency spikes. In contrast, in `DORA`, the non-stragglers relinquish some resource to the agent who is the straggler in the previous round, so it updates its decisions by jointly considering the historical computation and communication latency.

Fig. 2 shows the training accuracy of LeNet versus the training (wall-clock) time. The training time consists of both the computation time for LeNet's forward and backward propagation and the

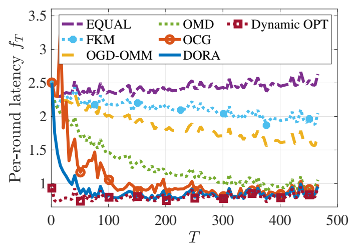

Figure 1: Per-round delay.

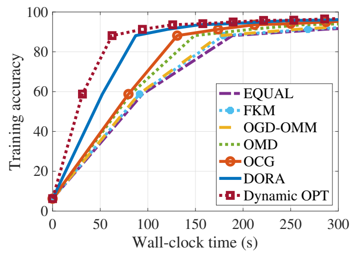

Figure 2: Training accuracy.

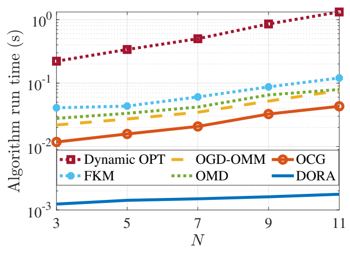

Figure 3: Algorithm run time.

communication time for parameter synchronization, which accounts for both the number of rounds and the latency per round. We observe that DORA has the fastest training time since it responds quickly towards the dynamic optimum. For 90% training accuracy, DORA speeds up the training time by 53.7%, 51.4%, 50.3%, 41.2%, and 34%, respectively, when compared with EQUAL, FKM, OGD−OMM, OMD, and OCG.

To demonstrate the computation complexity of DORA and the other algorithms, we measure their run time (in seconds) for calculating the resource allocation solutions only, i.e., it excludes the LeNet training time. Since EQUAL is deterministic, its run time is zero and is ignored in this comparison. Fig. 3 shows the total run time over the horizon of $T = 470$ rounds, for a different number of agents N. With increasing N, the run time of Dynamic OPT and OCG increases dramatically since solving an instantaneous optimization with the standard optimization package includes multiple inner loops. FKM, OGD−OMM, and OMD are also sensitive to N due to their projection operation. In contrast, the algorithm run time of DORA is negligible. DORA is robust and light-weight due to its gradient-free and projection-free properties.

We have further experimented with various bandwidth budgets and agent mobility, where we observe similar performance benefit under DORA. The details can be found in Wang and Liang (2021).

7. Conclusion

We have proposed a new distributed online algorithm termed DORA to solve an online min-max optimization problem in a multi-agent system with coupling linear constraints. DORA has high computation efficiency, since it is gradient-free and projection-free. We analyze the dynamic regret of DORA for non-convex functions. We applied the proposed solution to resource allocation in distributed online machine learning. Numerical studies show the efficacy of the proposed solution in terms of significantly faster convergence and reduced training time.

Acknowledgments

This work has been funded by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

Mohammad Mohammadi Amiri and Deniz Gündüz. Federated learning over wireless fading channels. *IEEE Transactions on Wireless Communications*, 19(5):3546–3557, 2020.

Evripidis Bampis, Dimitris Christou, Bruno Escoffier, and Nguyen Kim Thang. Online-learning for min-max discrete problems. In *Proc. PGMO*, 2020.

Omar Besbes, Yonatan Gur, and Assaf Zeevi. Non-stationary stochastic optimization. *Operations Research*, 63(5):1227–1244, 2015.

M. Blondel, A. Fujino, and N. Ueda. Large-scale multiclass support vector machine training via Euclidean projection onto the simplex. In *Proc. ICPR*, 2014.

H. Boche, M. Wiczanowski, and S. Stanczak. Unifying view on min-max fairness and utility optimization in cellular networks. In *Proc. IEEE WCNC*, 2005.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

Rishabh Dixit, Amrit Singh Bedi, Ketan Rajawat, and Alec Koppel. Distributed online learning over time-varying graphs via proximal gradient descent. In *Proc. IEEE CDC*, 2019.

Jianbo Du, Liqiang Zhao, Jie Feng, and Xiaoli Chu. Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. *IEEE Transactions on Communications*, 66(4):1594–1608, 2017.

Nima Eshraghi and Ben Liang. Distributed online optimization over a heterogeneous network with any-batch mirror descent. In *Proc. ICML*, 2020.

Abraham D. Flaxman, Adam Tauman Kalai, and H. Brendan McMahan. Online convex optimization in the bandit setting: Gradient descent without a gradient. In *Proc. ACM-SIAM SODA*, 2005.

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

Xiang Gao, Xiaobo Li, and Shuzhong Zhang. Online learning with non-convex losses and non-stationary regret. In *Proc. of AISTATS*, volume 84, pages 235–243, 2018.

Eldon Hansen and Merrell Patrick. A family of root finding methods. *Numerische mathematik*, 27 (3):257–269, 1976.

Elad Hazan. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2(3-4):157–325, 2016.

Elad Hazan and Satyen Kale. Projection-free online learning. In *Proc. ICML*, 2012.

Elad Hazan and Edgar Minasyan. Faster projection-free online learning. In *Proc. COLT*, 2020.

Amélie Héliou, Matthieu Martin, Panayotis Mertikopoulos, and Thibaud Rahier. Online non-convex optimization with imperfect feedback. In *Proc. of NeurIPS*, 2020.

Amélie Héliou, Panayotis Mertikopoulos, and Zhengyuan Zhou. Gradient-free online learning in games with delayed rewards. In *Proc. ICML*, 2020.

Amélie Héliou, Matthieu Martin, Panayotis Mertikopoulos, and Thibaud Rahier. Zeroth-order non-convex learning via hierarchical dual averaging. In *Proc. of ICML*, 2021.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Xiuxian Li, Xinlei Yi, and Lihua Xie. Distributed online optimization for multi-agent networks with coupled inequality constraints. *IEEE Transactions on Automatic Control*, 2020.

Xiuxian Li, Xinlei Yi, and Lihua Xie. Distributed online convex optimization with an aggregative variable. *IEEE Transactions on Control of Network Systems*, 2021.

Jun Liu and Jieping Ye. Efficient Euclidean projections in linear time. In *Proc. ICML*, 2009.

Sijia Liu, Pin-Yu Chen, Bhavya Kailkhura, Gaoyuan Zhang, Alfred O. Hero III, and Pramod K. Varshney. A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications. *IEEE Signal Processing Magazine*, 37(5):43–54, 2020a.

Sijia Liu, Songtao Lu, Xiangyi Chen, Yao Feng, Kaidi Xu, Abdullah Al-Dujaili, Mingyi Hong, and Una-May OReilly. Min-max optimization without gradients: Convergence and applications to black-box evasion and poisoning attacks. In *Proc. ICML*, 2020b.

Giampaolo Liuzzi, Stefano Lucidi, and Marco Sciandrone. A derivative-free algorithm for linearly constrained finite minimax problems. *SIAM Journal on Optimization*, 16(4):1054–1075, 2006.

Kaihong Lu, Gangshan Jing, and Long Wang. Online distributed optimization with strongly pseudoconvex-sum cost functions. *IEEE Transactions on Automatic Control*, 65(1):426–433, 2020.

Yuyi Mao, Jun Zhang, and Khaled B Letaief. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE JSAC*, 34(12):3590–3605, 2016.

Aryan Mokhtari, Shahin Shahrampour, Ali Jadbabaie, and Alejandro Ribeiro. Online optimization in dynamic environments: Improved regret rates for strongly convex problems. In *Proc. IEEE CDC*, 2016.

Solmaz Niknam, Harpreet S Dhillon, and Jeffrey H Reed. Federated learning for wireless communications: Motivation, opportunities, and challenges. *IEEE Communications Magazine*, 58(6): 46–51, 2020.

Ivano Notarnicola, Mauro Franceschelli, and Giuseppe Notarstefano. A duality-based approach for distributed Min-Max optimization. *IEEE Transactions on Automatic Control*, 64(6):2559–2566, 2019.

Shahin Shahrampour and Ali Jadbabaie. Distributed online optimization in dynamic environments using mirror descent. *IEEE Transactions on Automatic Control*, 63(3):714–725, 2018.

Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2012.

Pranay Sharma, Prashant Khanduri, Lixin Shen, Donald J. Bucci, and Pramod K. Varshney. On distributed online convex optimization with sublinear dynamic regret and fit. In *Proc. of ASILOMARSSC*, pages 1013–1017, 2021.

Kunal Srivastava, Angelia Nedić, and Dušan Stipanović. Distributed min-max optimization in networks. In *Proc. ICDSP*, 2011.

Kunal Srivastava, Angelia Nedić, and Dušan Stipanović. Distributed Bregman-distance algorithms for min-max optimization. *Agent-Based Optimization*, pages 143–174, 2013.

Yuanyu Wan, Bo Xue, and Lijun Zhang. Projection-free online learning in dynamic environments. In *Proc. AAAI*, 2021.

Jingrong Wang and Ben Liang. Gradient and projection free distributed online min-max resource optimization. 2021. URL https://arxiv.org/abs/2112.03896.

Shuang Wu, Xi Peng, and Guangjian Tian. Decentralized max-min resource allocation for monotonic utility functions. In *Proc. INFOCOM WKSHPS*, 2021.

Xinlei Yi, Xiuxian Li, Lihua Xie, and Karl H Johansson. Distributed online convex optimization with time-varying coupled inequality constraints. *IEEE Transactions on Signal Processing*, 68: 731–746, 2020.

Gang Yu. Min-max optimization of several classical discrete optimization problems. *Journal of Optimization Theory and Applications*, 98(1):221–242, 1998.

Lijun Zhang, Tianbao Yang, Jinfeng Yi, Rong Jin, and Zhi-Hua Zhou. Improved dynamic regret for non-degenerate functions. In *Proc. NeurIPS*, 2017a.

Lijun Zhang, Shiyin Lu, and Zhi-Hua Zhou. Adaptive online learning in dynamic environments. In *Proc. NeurIPS*, 2018.

Wenpeng Zhang, Peilin Zhao, Wenwu Zhu, Steven CH Hoi, and Tong Zhang. Projection-free distributed online learning in networks. In *Proc. ICML*, 2017b.

Yan Zhang, Robert J Ravier, Michael M Zavlanos, and Vahid Tarokh. A distributed online convex optimization algorithm with improved dynamic regret. In *Proc. IEEE CDC*, 2019.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proc. ICML*, 2003.