A principled framework for the design and analysis of token algorithms

Hadrien Hendrikx
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, Grenoble, France

Abstract

We consider a decentralized optimization problem, in which n nodes collaborate to optimize a global objective function using local communications only. While many decentralized algorithms focus on *gossip* communications (pairwise averaging), we consider a different scheme, in which a "token" that contains the current estimate of the model performs a random walk over the network, and updates its model using the local model of the node it is at. Indeed, token algorithms generally benefit from improved communication efficiency and privacy guarantees. We frame the token algorithm as a randomized gossip algorithm on a conceptual graph, which allows us to prove a series of convergence results for variance-reduced and accelerated token algorithms for the complete graph. We also extend these results to the case of multiple tokens by extending the conceptual graph, and to general graphs by tweaking the communication procedure. The reduction from token to well-studied gossip algorithms leads to tight rates for many token algorithms, and we illustrate their performance empirically.

1 INTRODUCTION

Modern machine learning relies on increasingly large models that train on increasingly large datasets: distributed optimization is thus crucial to scaling the training process. In the centralized paradigm, at the heart of *Federated Learning* [Kairouz et al., 2021], the system relies on a server that aggregates models and gradients, and manages the nodes. Although quite efficient, this setting has several drawbacks: (i) nodes need to trust the server enough to send it sensitive data, (ii) there is a communication bottleneck at the server, which limits scaling and (iii) training stops if the server fails.

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

In the *decentralized* setting [Boyd et al., 2006, Lopes and Sayed, 2007, Shi et al., 2015, Nedic et al., 2017], nodes are linked by a communication graph, and directly communicate with their neighbours in this graph instead of a central coordinator. This allows for better scaling, and is also more robust since the server is no longer the single point of failure. Yet, due to the lack of coordination, decentralized algorithms often require many peer-to-peer communications compared to centralized ones, and a gain in privacy is not always guaranteed.

Token, or random-walk algorithms [Bertsekas, 1997, Ram et al., 2009, Johansson et al., 2010, Shah and Avrachenkov, 2018, Mao et al., 2020], work in the following way: a token owns an estimate of the model, "walks" over the graph, and sequentially visits nodes. When the token is held by a node, it updates its model, either by computing a gradient using the node's data, or by using the local model of the node. Then, the token is transmitted (or "jumps") to a new node.

Some instantiations of these algorithms can be seen as a middle-point between centralized and decentralized algorithms. Indeed, the token plays the role of a server, since it owns the global model and receives updates from nodes. Yet, the token is no longer attached to a physical node as in the centralized case, but rather exchanged between computing nodes in a decentralized way.

Besides, unlike standard centralized algorithms, each node may maintain a local parameter and update it using local updates. In that sense, some token algorithms (such as the ones developed in this work) closely resemble local methods, that are very popular in federated learning [McMahan et al., 2017, Stich, 2018, Lin et al., 2018]. The main difference is that instead of exchanging information through periodic exact averaging or gossip steps, communication is ensured through the roaming token. This allows to easily adapt the algorithms to the features of the system, by making either more local steps or more communication steps.

1.1 Related work

Many early works study token (or random-walk) algorithms [Bertsekas, 1997, Nedic and Bertsekas, 2001, Ram et al., 2009, Johansson et al., 2010]. Yet, they focus on stochastic (sub)gradients algorithms, and thus lack linear

convergence guarantees. The recent literature on token algorithms can be divided into two main lines of work that reflect the two main strengths of token algorithms: *communication efficiency* and *privacy preservation*.

Communication efficiency. Mao et al. [2020] introduce *Walkman*, a token algorithm based on an augmented Lagrangian method. Walkman works for general graphs and is shown to be communication-efficient provided graphs are well-connected enough. Yet, it only obtains linear convergence on least squares problems. When Walkman uses gradients (instead of proximal operators), it requires a step-size inversely proportional to the square of the smoothness constant of the problem, which is impractical. Variants of Walkman guarantee communication efficiency when walking over Hamiltonian cycles [Mao et al., 2018]. Balthazar et al. [2020] consider the problem of distributed linear estimation, and use a token algorithm to aggregate the measurements of all nodes. Sun et al. [2022] and Ayache et al. [2022] consider *adaptive* token algorithms, to respectively choose the step-size or the probabilities of the random walk, but neither of them consider strongly-convex objectives.

Multiple tokens. When a single token walks the graph, there are no parallel communications. A natural fix to speed up algorithms is to allow multiple tokens to walk the graph in parallel, as recently done by Chen et al. [2022], whose approach is also based on an augmented Lagrangian method.

Privacy Preservation. The favorable privacy guarantees claimed by decentralized algorithms are actually mainly proven for token algorithms. For instance, the Walkman algorithm presented above has also been extended to guarantee privacy preservation [Ye et al., 2020]. Besides, Cyffers and Bellet [2022] show that token algorithms satisfy a relaxation of local differential privacy, and match the guarantees offered by a trusted central server. They give a simple algorithm for ring and complete topologies. Similarly, Bellet et al. [2020] study the privacy guarantees of a rumour spreading algorithm, and show that a single token spreading the rumour is optimal, while multiple tokens achieve optimal trade-offs between privacy and speed. In this work, we focus on the convergence guarantees of token algorithms, and leave the privacy preservation guarantees for future work.

Dual Decentralized algorithms. Our framework is based on applying the dual approach for decentralized algorithms [Jakovetić et al., 2014, Boyd et al., 2011] to the analysis of token algorithms. This dual approach leads to very fast algorithms, and in particular Scaman et al. [2017], Uribe et al. [2020] used it to develop optimal decentralized algorithms. Then, Hendrikx et al. [2019a] showed that it can also be used to accelerate randomized gossip, and used an augmented graph formulation to obtain decentralized variance-reduced extensions [Hendrikx et al., 2019b, 2021, 2020]. Although primal-dual approaches lead to optimal *primal* decentralized algorithms [Kovalev et al., 2020], they do not mix as well with randomization, which we need to model token algorithms.

Time-varying graphs. Token algorithms resemble gossip algorithms for time-varying graphs [Koloskova et al., 2020, Kovalev et al., 2021, Rogozin et al., 2022] in the sense that not all edges can be activated at each step. However, this work views token algorithms as performing *randomized* communications over a *fixed* graph, which leads to a rather different analysis.

1.2 Our contributions

As discussed in the previous section, token algorithms are still rare, and very few algorithms offer linear convergence guarantees, let alone integrating more advanced optimization tricks such as variance reduction or acceleration. Besides, most of the literature focuses on only one token, which is communication-efficient but very slow. In this work, we pave the way for the design and analysis of new efficient token algorithms, and in particular we:

1. Introduce a general framework for designing and analyzing token algorithms.
2. Give a simple algorithm with linear convergence guarantees on complete graphs that match those of both centralized and decentralized (gossip) optimization.
3. Speed up this simple algorithm by using multiple tokens.
4. Leverage the general framework to analyze variants of the simple token algorithm, such as stochastic gradients with variance reduction and acceleration.
5. Extend the convergence results to general graphs, by tweaking the communication protocol.

The general framework is based on the dual approach for decentralized algorithms [Jakovetić et al., 2014, Scaman et al., 2017], and in particular Hendrikx et al. [2020], and so the algorithmic core similar, namely Bregman coordinate descent (with some adaptations) for the simple and variance-reduced algorithms, and Accelerated Proximal Coordinate Descent [Lin et al., 2015] for the accelerated one.

2 CONCEPTUAL GRAPH APPROACH FOR TOKEN ALGORITHMS.

As mentioned before, this paper uses the standard dual approach for obtaining Algorithm 1. Its main steps are:

1. Transform the base unconstrained problem (1) into an extended constrained problem, with equality constraints that follow a graph structure (2).

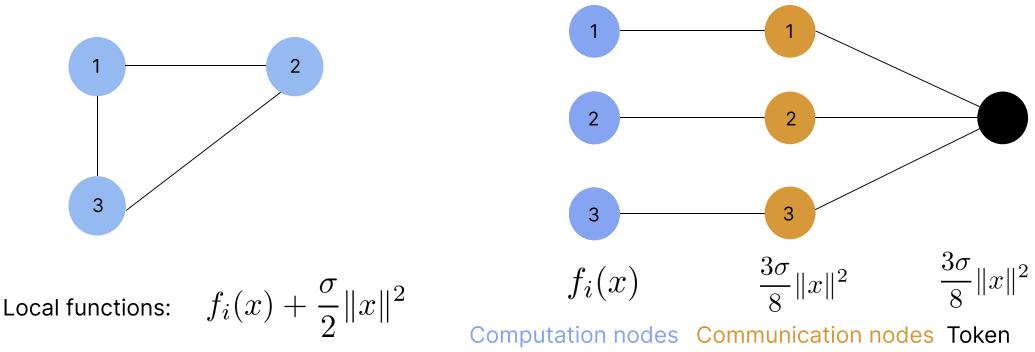


Figure 1: Left: base communication graph. Right: Conceptual graph, with modified local objectives. The sum of all local objectives remains unchanged, so the global objective remains the same. We call the right graph *conceptual* because nodes (in particular the token node) and edges have no physical meaning, and are designed for the sake of deriving Algorithm 1.

2. Use Lagrangian duality to turn the constrained problem (2) into an unconstrained dual problem.
3. Use a base algorithm (Bregman Coordinate Descent) on the dual problem to obtain a dual algorithm.
4. Rewrite the dual algorithm to unveil an equivalent decentralized (primal-friendly) formulation.

The main novelty of the paper is to use a conceptual graph (described in Figure 1) for step 1 instead of the usual communication graph. Indeed, this changes the way communications are performed, and allows for a token interpretation: nodes do not exchange information directly, but through a token that they update and pass to their neighbours. The precise derivation for steps 2 to 4 follow from adapting Hendrikx et al. [2020], and so we do not detail them here due to lack of space. However, the main derivations can be found in Appendix B.

2.1 Building the conceptual graph

We consider the following distributed problem, where each f_i is a local function at node i:

$$\min_{x\in\mathbb{R}^d} \sum_{i=1}^n \left[f_i(x) + \frac{\sigma}{2}\|x\|^2\right]. \tag{1}$$

We assume that each f_i is convex and L-smooth over $\mathbb{R}^d$, which writes if f_i is twice differentiable as $0 \preccurlyeq \nabla^2 f_i(x) \preccurlyeq L\, I_d$, where $I_d \in \mathbb{R}^{d\times d}$ is the identity matrix of dimension d. The *condition number* of this problem is $\kappa = 1 + L/\sigma$. The key idea of this paper is to reduce the analysis of token algorithms to that of standard decentralized gossip algorithms on conceptual graphs. We follow the *dual approach* for building decentralized optimization algorithms, and rewrite Problem (1) as:

$$\min_{\substack{x\in\mathbb{R}^{n\times d},\ u\in\mathbb{R}^{n\times d},\ v\in\mathbb{R}^d,\\ \forall i,\ x^{(i)}=u^{(i)},\text{ and } u^{(i)}=v}} \sum_{i=1}^n f_i(x^{(i)}) + \frac{\sigma_n}{2}\|u^{(i)}\|^2 + \frac{\sigma_n}{2}\|v\|^2, \tag{2}$$

with $\sigma_n = n\sigma/(n+1)$. To write this reformulation, we have applied the consensus constraints (equality constraints for neighbours) given by the conceptual graph represented in Figure 1 (right). To build this conceptual graph, we add a conceptual node (with its own parameter) corresponding to the token, with local objective $\sigma_n\|\cdot\|^2/2$, and we split all local nodes into a computation part (that contains f_i), and a communication part (that contains $\sigma_n\|\cdot\|^2/2$). Note that the total regularization weight is still $(n+1)\sigma_n/2 = n\sigma/2$. Then, all computation nodes are linked to their respective communication nodes, which are themselves linked to the token. Splitting each node between communication and computation parts has two benefit: (i) it allows us to use the dual-free trick from Hendrikx et al. [2020], and obtain primal updates despite the dual approach, and (ii) it allows to decouple communications and computations. In particular, nodes can perform local steps even when they don't hold the token.

Now that we have defined the framework, it is important to make sure that this corresponds to a token algorithm. Updating the edge between the token and node i at time t in the conceptual graph means that the token jumped to node i at time t. Thus, to ensure the token aspect, we must enforce that if the edge between the token and node i is updated at time t, and the edge between the token and node j is updated at time $t+1$, then node j has to be a neighbour of node i (since the token jumped from i to j at time $t+1$). We apply the dual approach to Problem (2), which is inherited from the conceptual graph, but *the sampling of the edges is ruled by the actual communication graph*. In a complete graph, this does not impose any additional constraints, and this is why our convergence results are initially derived in this setting. In arbitrary graphs, this means that the sampling of the edges (and so the coordinate descent algorithm applied to the dual formulation) must follow a Markov Chain, which leads to considerably harder analyses. In Section 3.4, we present a trick to circumvent this difficulty, which consists in not performing the update step every time the token jumps to a new node.

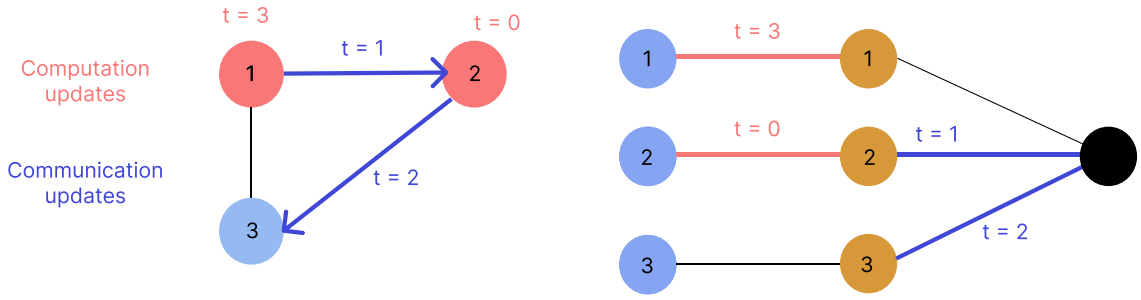


Figure 2: An example execution of the token algorithm on the base graph (left), and the corresponding edges updated in the conceptual graph (right). The sequence is: $t = 0$: local computation at node 2, $t = 1$: the token jumps to node 2, $t = 2$: the token jumps to node 3, $t = 4$: local computation at node 0. Note that the updates at $t = 2$ and $t = 3$ can actually be performed in parallel since they affect different nodes, and that the token updates its estimate with the node it arrives at after each jump.

We will now show that this conceptual graph view allows to efficiently design fast algorithms, by making clear links with dual approaches for decentralized optimization. This will prove especially useful in the next section, when we introduce multiple tokens, variance-reduction and acceleration. In the basic case (one token, full gradients), Equation (2) resembles the consensus formulation of Walkman [Mao et al., 2020], which is also obtained through a (primal-)dual formulation. Yet, we split each node into two subnodes to allow for local steps, and we can then harness the power of the dual approach for decentralized optimization to significantly improve the base algorithm, as done in Section 3.

2.2 Deriving the single token algorithm

Following Hendrikx et al. [2020], we take a dual formulation of Problem (2), and apply Bregman block coordinate descent to obtain the simple token algorithm, which corresponds to Algorithm 1 with $K = 1$. This leads to dual-free updates [Lan and Zhou, 2017], which are simple to implement. Yet, in Hendrikx et al. [2020], all communication edges are sampled at once, which would mean that the token receives updates from all nodes at the same time. This is not possible in our case, so we adapted the Bregman block coordinate descent algorithm to better fit the structure of Problem (2), as detailed in Appendix A. Before we state Theorem 1, a few remarks on the form of Algorith 1 are in order.

Parameter sequences. In the dual formulation of problem (2), minimization is performed over parameters $\mathbb{R}^{2nd}$ associated with the constraints of the primal problem, *i.e.*, the edges of the conceptual graph. However, it turns out that applying gradient (or block coordinate mirror) descent to this problem can be implemented by relying only on variables $(\theta, \theta^{\text{token}}, z) \in \mathbb{R}^{(2n+1)d}$ associated with nodes of the conceptual graph. This is achieved through multiplication by a well-chosen matrix $A \in \mathbb{R}^{(2n+1)d \times 2nd}$, which corresponds to a canonical root of the Laplacian matrix of the conceptual graph. This results in the $\theta^{(i)} \in \mathbb{R}^d$, which correspond to parameters associated with conceptual communication nodes, and in $\theta^{\text{token}} \in \mathbb{R}^d$ for the token node. The $z^{(i)} \in \mathbb{R}^d$ correspond to variables associated with the conceptual computation nodes, as introduced in Figure 1, and are required to write the dual-free formulation and perform local computation steps. Note that all sequences θ and z converge to the minimizer of the global problem (1).

Hyperparameters. There are three 'free parameters' when writing down Algorithm 1. **(i)** The first one is $\eta > 0$, which corresponds to the step-size of the *dual* algorithm. The two constraints on η that we give correspond to the directional smoothness of dual variables associated with communication and computation edges respectively. **(ii)** The second one is the weight of the edges between communication and computation nodes, which is equal to αL_i for node i, and so parameterized by $\alpha > 0$ (communication edges have unitary weights). **(iii)** The third one is p_{comm}, which corresponds to how frequently communication steps are performed. We choose these 3 parameters to obtain tight complexity guarantees. In particular, p_{comm} is chosen such there are the same number of communications and computations, α such that the two conditions on the step-size η match, and η the largest possible. Yet, other choices are possible, and we see for instance in Theorem 3 that it might be desirable to communicate less often when using variance reduction. These parameters are not explicit in Algorithm 1 (and only appear through ρ_{comp} and ρ_{comm}) because of the primal reformulation of the dual algorithm upon which Algorithm 1 is based.

More details about how Algorithm 1 is derived (and in particular the base dual algorithm and its primal reformulation) can be found in Appendix B. We now state the main result of this section.

Theorem 1 (Token algorithm)**.** *For $\varepsilon > 0$, the number of steps required by Algorithm 1 with a single token ($K = 1$)*

and $p_{\text{comp}} = p_{\text{comm}} = \frac{1}{2}$ *to reach error* $\|\theta_t - \theta_\star\|^2 \leq \varepsilon$ *is of order:*

$$T_{\text{comp}} = O\left(\kappa \log \varepsilon^{-1}\right) \text{ and } T_{\text{comm}} = O\left(n\kappa \log \varepsilon^{-1}\right),$$

where T_{comp} *is the expected number of gradient steps performed by each node, and* T_{comm} *is the expected number of communication updates (jumps) performed by the token.*

Proof sketch. This result follows from the guarantees of Bregman Coordinate Descent applied to a dual reformulation of Problem (2). We need to evaluate the relative strong convexity and directional smoothness constants of the problem, and link them with spectral properties of the conceptual graph, as well as the regularity of the local functions. Details can be found in Appendix B. □

Complete communication graph. In Algorithm 1, line 5, the token chooses the next node to jump to uniformly at random. This requires a complete communication graph in general, or multi-hop communication schemes such as the ones discussed in Section 3.4.

Implementation. Algorithm 1 requires sampling updates uniformly at random over the whole system. This can be implemented in a decentralized fashion by sharing a random seed between all nodes. An alternative is that all nodes wake up and perform updates following a Poisson point process.

Communication complexity. The total number of communications is of order $O(n\kappa)$. This matches the complexity of centralized algorithms, in which the server communicates once with each node at each round, and there are $O(\kappa)$ rounds in total. Yet, in terms of time, the $O(n)$ centralized communications can take place in parallel provided there is enough bandwidth, whereas when $K = 1$, the $O(n)$ communications from Algorithm 1 must be sequential.

Computation complexity. In average, each node performs $O(\kappa)$ local computations, which is the "right" complexity for non-accelerated algorithms. Algorithm 1 is as computationally efficient as standard centralized or decentralized algorithms in this sense. Besides, unlike token exchanges that need to be sequential, nodes can compute local gradient updates in parallel. Therefore, the computation time of Algorithm 1 matches the centralized time. Instead, when using gradients, Walkman [Mao et al., 2020, Theorem 1] requires a step-size proportional to L^{-2} just for convergence, which would lead to a significantly worse computation complexity of at least $O(L\kappa)$.

Sampling variants. In Algorithm 1, one computation update corresponds to one node performing a gradient update, without any communication involved. Note that by changing the sampling of the dual coordinates, it is possible to design other algorithms. For instance, we can choose to have nodes perform a local computation only when they receive the token, similarly to Walkman. This would yield a similar rate as the one in Theorem 1, but does not allow for local steps. It would thus not be possible for instance to perform a lot of fast communications, while slow computations take place in parallel. Another possibility is that all nodes perform their computation steps in parallel at the same time. Again, this algorithm would have similar guarantees, but introduces global synchronization steps which are not desirable and we believe go against the spirit of token algorithms. Thus, our framework is flexible and can handle many variants, including with the tricks developed in the next section.

3 EXTENSIONS OF THE SINGLE TOKEN ALGORITHM

In the previous section, we have introduced the conceptual graph, and showed how it allows to leverage the existing tools from (dual) decentralized optimization to analyze a simple yet already efficient token algorithm. We now demonstrate the flexibility and generality of this framework by introducing, analyzing and combining three important variants of the token algorithm: multiple tokens, variance reduction, and acceleration.

3.1 The case of several tokens

When there is a single token walking on the graph, resources are used in an efficient way, and privacy guarantees are strong, but mixing is very slow (up to n times slower in a complete graph for instance). This is due to the fact that there are no parallel communications. One natural solution is to use multiple tokens that walk the graph in parallel. Yet, this is generally harder to analyze and, to the best of our knowledge, there only limited theory on multi-token algorithms, with convergence rates that show actual improvement over single tokens. Our conceptual graph framework allows us to directly extend Theorem 1 to the case of multiple tokens.

To do so, we build a different conceptual graph. Namely, we add one new node for each token, and link all "token nodes" to the actual nodes of the network, as shown in the right part of Figure 3. Then, we apply the dual approach to this new conceptual graph, which has a different topology but which we know how to handle. This is how we obtain Algorithm 1 for the general case of $K \geq 1$.

Theorem 2 (Multiple tokens)**.** *For* $\varepsilon > 0$*, the number of steps required by Algorithm 1 with* $1 \leq K \leq n$ *and* $p_{\text{comp}} = p_{\text{comm}} = \frac{1}{2}$ *to reach error* $\|\theta_t^{\text{token},k} - \theta_\star\|^2 \leq \varepsilon$ *is of order:*

$$T_{\text{comp}} = O\left(\kappa \log \varepsilon^{-1}\right) \text{ and } T_{\text{comm}} = O\left(\frac{n}{K}\kappa \log \varepsilon^{-1}\right), \tag{3}$$

where T_{comp} *is the expected number of gradient steps performed by each node, and* T_{comm} *is the number of jumps performed* per token*. In particular, the total communication*

Algorithm 1 Token Gradient Descent(z_0)

$\tilde{\sigma} = \frac{n}{n+K}\sigma$, $\alpha = \frac{2K}{L}$, $\eta = \min\left(\frac{\tilde{\sigma} p_{\rm comm}}{2nK}, \frac{p_{\rm comp}}{n\alpha(1+L/\tilde{\sigma})}\right)$, $\rho_{\rm comm} = \frac{nK\eta}{p_{\rm comm}\tilde{\sigma}}$, $\rho_{\rm comp} = \frac{n\alpha\eta}{p_{\rm comp}}$. *// Init*

$\forall i \in [n], \theta_0^{(i)} = -\nabla f_i(z_0^{(i)})/\tilde{\sigma}$; $\forall k \in [K], \theta_0^{{\rm token},k} = 0$. *// z_0 is arbitrary but not θ_0.*

for $t = 0$ to $T-1$ **do** *// Run for T iterations*

  if communication step (with probability $p_{\rm comm}$) **then**

    Pick $i \sim \mathcal{U}([n])$, $k \sim \mathcal{U}([K])$ *// Choose next node and token uniformly at random*

    $\theta_{t+1}^{{\rm token},k} = \theta_t^{{\rm token},k} - \rho_{\rm comm}(\theta_t^{{\rm token},k} - \theta_t^{(i)})$ *// Token update*

    $\theta_{t+1}^{(i)} = \theta_t^{(i)} + \rho_{\rm comm}(\theta_t^{{\rm token},k} - \theta_t^{(i)})$ *// Local node update*

  else

    Pick $i \sim \mathcal{U}([n])$ *// Choose one node at random*

    $z_{t+1}^{(i)} = (1-\rho_{\rm comp})\, z_t^{(i)} + \rho_{\rm comp}\theta_t^{(i)}$ *// Virtual node update*

    $\theta_{t+1}^{(i)} = \theta_t^{(i)} - \frac{1}{\tilde{\sigma}}\left(\nabla f_i(z_{t+1}^{(i)}) - \nabla f_i(z_t^{(i)})\right)$ *// Local update using f_i*

return θ_K

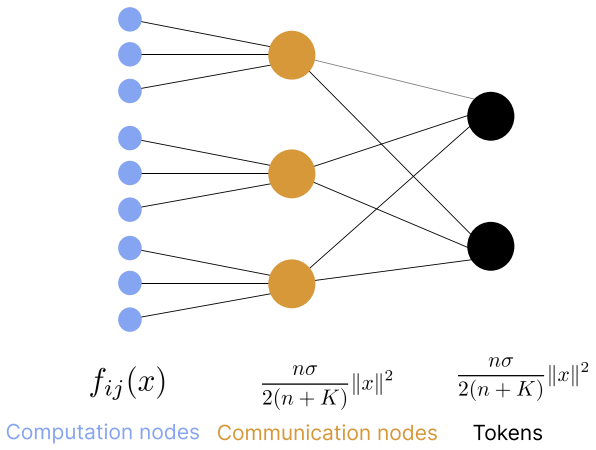


Figure 3: Conceptual graph of size $n = 3$ with finite-sum local objectives ($m = 3$) and multiple tokens ($K = 2$).

complexity is the same as in Algorithm 1, but now the burden is shared by K tokens that walk the graph in parallel.

Token interactions. In the formulation inherited from Figure 3, tokens interact with nodes, but not between themselves. We can change the formulation to add interactions between tokens by adding edges between them in the conceptual graph. This would mean that the tokens would mix information when they meet. Yet, this would only marginally increase the connectivity of the conceptual graph, and would thus not speedup the algorithm by more than constant factors.

3.2 Variance reduction in the finite sum case

We have seen that changing the conceptual graph on which the dual formulation is applied changes the resulting token algorithm. In the previous section, we used this to speed-up communications by having several tokens walk the graph in parallel. We now leverage it to speed-up computations by avoiding local full gradient computations at each node. We now assume that each local objective writes $f_i(x) = \sum_{j=1}^m f_{ij}(x)$. In this case, each full gradient requires m stochastic gradient ∇f_{ij} computations, so Algorithm 1 requires $O(m\kappa)$ stochastic gradient in total. Instead, *variance reduction* techniques [Schmidt et al., 2017, Johnson and Zhang, 2013, Defazio et al., 2014, Shalev-Shwartz, 2016] only require $m + \kappa_s$ stochastic gradients, where $\kappa_s = \sum_{j=1}^m (1 + L_{ij}/\sigma)$, where L_{ij} is the smoothness of function f_{ij}. Although $m\kappa = \kappa_s$ in the worst case (where the ∇f_{ij} are all orthogonal), κ_s is generally smaller than $m\kappa$, leading to the practical superiority of finite-sum methods.

In our case, we introduce the finite-sum aspect by combining the conceptual graph with an augmented graph formulation [Hendrikx et al., 2019b, 2020]. Instead of splitting each node into 2 parts, containing respectively f_i and the regularization part, as in Figure 1, we split it into a star sub-network, with each f_{ij} linked to the regularization part, as shown if the left part of Figure 3. This new conceptual graph leads to a new algorithm, **T**oken **V**ariance **R**eduction (TVR), that has the following convergence guarantees:

Theorem 3 (Variance Reduction)**.** *For $\varepsilon > 0$, the number of steps required by TVR with $1 \leq K \leq n$ and $p_{\rm comp} = \frac{\kappa_s}{m-1+\kappa_s}$ to reach error $\|\theta_t - \theta_\star\|^2 \leq \varepsilon$ is of order:*

$$T_{\rm comp} = O\left((m+\kappa_s)\log \varepsilon^{-1}\right) \quad \textit{and}$$
$$T_{\rm comm} = O\left(\frac{n}{K}\kappa_s \log \varepsilon^{-1}\right),$$

where $T_{\rm comp}$ is the expected number of stochastic gradient steps performed by each node, and $T_{\rm comm}$ the number of jumps performed per token. *Compared to Theorem 2, the computation complexity goes from $m\kappa$ stochastic gradients (κ full gradients) to $m+\kappa_s$, which is generally much smaller.*

TVR performs the same communication steps as Algorithm 1, with slightly different computation steps, adapted

for the stochastic case: there are now m functions f_{ij} (and so parameters $z^{(ij)}$), instead of just one. The full algorithm and the proof of Theorem 3 are detailed in Appendix B.2.

3.3 Acceleration

We have seen that the conceptual graph view of token algorithms allows to naturally extend the simple single-token batch algorithm to a multi-token variance-reduced algorithm. We now show that by applying a different optimization algorithm to the same dual formulation, we obtain an accelerated algorithm from the same framework. We refer the reader to Appendix C for details and derivations.

Theorem 4 (Token Accelerated Variance-Reduced)**.** *For $\varepsilon > 0$, the number of steps required by Accelerated TVR with $1 \leq K \leq n$ to reach error $\|\theta_t^{\text{token},k} - \theta_\star\|^2 \leq \varepsilon$ is of order:*

$$T_{\text{comp}} = O\left((m + \sqrt{m\kappa_s}) \log \varepsilon^{-1}\right) \text{ and}$$
$$T_{\text{comm}} = O\left(\frac{n}{K}\sqrt{\kappa_s} \log \varepsilon^{-1}\right)$$

In particular, the dependences on the objective regularity are replaced by their accelerated versions. This matches the optimal complexities from Hendrikx et al. [2021].

Proximal oracles. This algorithm is based on Accelerated Proximal Coordinate Gradient [Lin et al., 2015, Hendrikx et al., 2019b], and thus uses proximal operators of the functions f_{ij}, which can be much more expensive than computing gradients. Yet, this is also the case of Walkman, and it is quite cheap in case the f_{ij} are generalized linear model of the form $f_{ij}(\theta) = \ell(x_{ij}^\top \theta)$, where $\ell : \mathbb{R} \mapsto \mathbb{R}$.

Continuized framework. TAVR requires each node to perform local convex combinations at each step. This introduces global synchronization constraints, that we can get rid of using a continuized version of the algorithm [Even et al., 2021].

3.4 General graphs

All the results presented so far are for the complete communication graph, meaning that the token can directly jump from any node to any other, allowing to prove strong convergence rates. We now show how to extend these results to general graphs. To do so, we analyze a slightly different communication procedure: instead of just one jump, each token performs N_{jumps} jumps before averaging with the node it lands at according to Algorithm 1 (lines 6-7). If enough steps are taken, and the underlying Markov Chain is irreducible and aperiodic, then the probability that the token lands at node j from node i after N_{jumps} steps is approximately equal to $\pi_\star(j)$, where $\pi_\star$ is the stationary distribution of a random walk over the communication graph. In particular, by allowing multiple steps before performing the actual communication update, all nodes can be reached from all nodes, as in the complete graph case. The actual sampling probabilities $\tilde{p}_{i,t}$ depend on the node at which the token is at time t. Yet, if $\tilde{p}_{i,t}$ is close enough to $\pi_\star(i)$, we can just adapt the step-sizes in coordinate descent, and obtain convergence regardless. Details can be found in Appendix 3.4

Theorem 5. *Assume that matrix $W \in \mathbb{R}^{n\times n}$ defining the token transitions is such that for any π_0, $\|W^t\pi_0 - \pi_\star\|_\infty \leq C(1-\gamma)^t$. Then, if the token jumps $O(\gamma^{-1}\log(C/\eta\mu))$ times before performing an averaging step, the communication complexity of Algorithm 1 (ignoring log factors) is $O(\frac{n\kappa}{\gamma}\log\varepsilon^{-1})$. Theorem 3 can be adapated in the same way.*

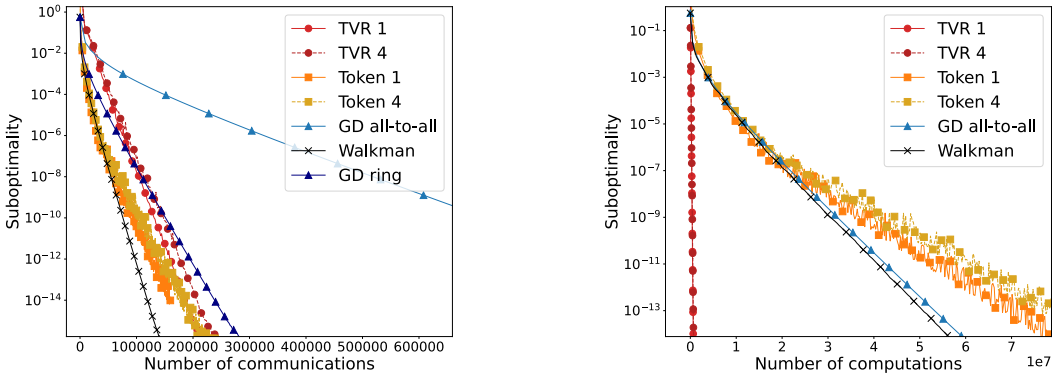
Note that these rates are for uniform stationary distribution $\pi_\star(i) = n^{-1}$ for all $i \in [n]$. Rates for non-uniform stationary distributions can also be obtained, and would depend on $\max_i \pi_i/L_i$, so that the algorithm is still fast if nodes that are visited less frequently have better smoothness.

Comparison with existing decentralized algorithms. Note that decentralized algorithms such as EXTRA [Shi et al., 2015, Li and Lin, 2020] require a total of $O(E(\kappa + \gamma^{-1})\log\varepsilon^{-1})$ communications, where E is the number of edges in the graph. In particular, our token algorithm is more communication efficient even in general graphs with this reduction as long as either κ or γ^{-1} is small compared to E/n, the average node degree, meaning that the graph is dense. This is remarkable since EXTRA is as fast as accelerated algorithms when $\kappa \approx \gamma^{-1}$, and so in particular it might be fairer to compare it to the accelerated algorithm from Theorem 4 instead. Note that this complexity is also better than that of Walkman [Mao et al., 2020], which depends on γ^2 instead of γ, besides proving linear convergence only in limited settings.

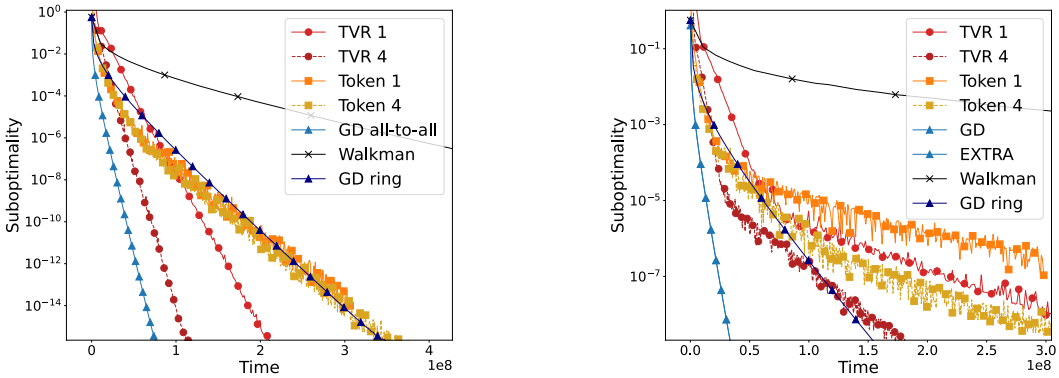
Directed and time-varying graphs. With the communication-skipping variant, all that matters is that the probability of being at node j after taking $O(\gamma^{-1}\log(\eta\mu))$ steps from node i is close to some $\pi_\star(j) > 0$. In particular, this does not imply the reversibility of the Markov Chain used for communications (and so can be used for directed graphs), or even its stationarity.

4 EXPERIMENTS

In the previous section, we leveraged the conceptual graph framework to design and analyze several (multi-)token algorithms. We now illustrate their differences when solving a logistic regression task on the RCV1 dataset [Lewis et al., 2004] ($d = 47236$), with $n = 20$ nodes, and $m = 9841$ samples per node. Results are presented in in Figure 4. We use a step-size $\beta = 1/L$ for Walkman, which is much higher than the one from Mao et al. [2020]. Suboptimality is computed as $f(x_t) - \min_x f(x)$, where the minimum is approximated by using the lowest value found across all algorithms. The communication complexity is the to-



a) Communication (left) and computation (right) complexities of various algorithms for the complete graph.



b) Time complexities for complete (left) and ring (right) graphs.

Figure 4: Time complexities for straggler (left) and (right) graphs.

tal number of token jumps (regardless of the number of tokens), and the computation complexity is the total number of gradients (on single f_{ij}) computed. Time is obtained by setting $\tau_{\text{comp}} = 1$ for computing one individual gradient and $\tau_{\text{comm}} = 10^3$ for one communication, so that one communication is faster than computing one full local gradient (more details in Appendix D). For gradient descent (GD), we present 2 variants of the allreduce protocol: *all-to-all* , which has a high number of communications $n(n-1)$ but small time (1) per step, and *ring* (sequentially averaging over a directed ring), which has a small number of communication ($2n$) but high time ($2n$) per step. Note that a fully centralized implementation would get both small communication complexity ($2n$) and time (2). *Token* and *TVR* respectively refer to the algorithms analyzed in Theorems 2 and 3. EXTRA is a standard decentralized algorithm [Shi et al., 2015]. Additional details on the setting can be found in Appendix D.

Computation and communication complexities. We see that for the complete graph, all token algorithms have similar communication complexity. This is consistent with our theory, and confirms that using multiple token does not hurt efficiency. We also confirm that the communication complexity of token algorithms is lower than that of all-to-all gradient descent (GD), and comparable to that of the efficient ring GD. Similarly, all batch algorithms have comparable computation complexities, with GD and Walkman performing slightly better. TVR is more computationally efficient thanks to the stochastic variance-reduced updates.

Time complexity. In terms of time, the fastest algorithm is all-to-all gradient descent, since we assume no limits on the bandwidth, so that it performs all communications in parallel. Yet, Algorithm 1 is as fast as ring GD, since both algorithms require $O(n)$ sequential communications, but perform computations in parallel. Walkman is the slowest algorithm in this setting, because it needs to perform both communications and computations sequentially, since it does not use local updates.

Note that with the values we have chosen for delays, *computation* dominates the execution time of Algorithm 1 (Token 1), which is why TVR is faster, because variance reduction reduces the computational burden. This is also why using more tokens (Token 4) does not speed up Token 1 in this setting, but speeds up TVR. Indeed, thanks to variance reduction, *communication* now dominates the execution time of TVR, so that it benefits from using several tokens.

For the ring graph, we find that the same token algorithms as for the complete graph are stable (consistently across a wide range of m and n), and so we do not use the skip variant presented in Section 3.4. We observe similar results to the complete graph case, although using multiple tokens now accelerates Algorithm 1 since they allow to compensate for the worse graph connectivity. GD and EXTRA have the

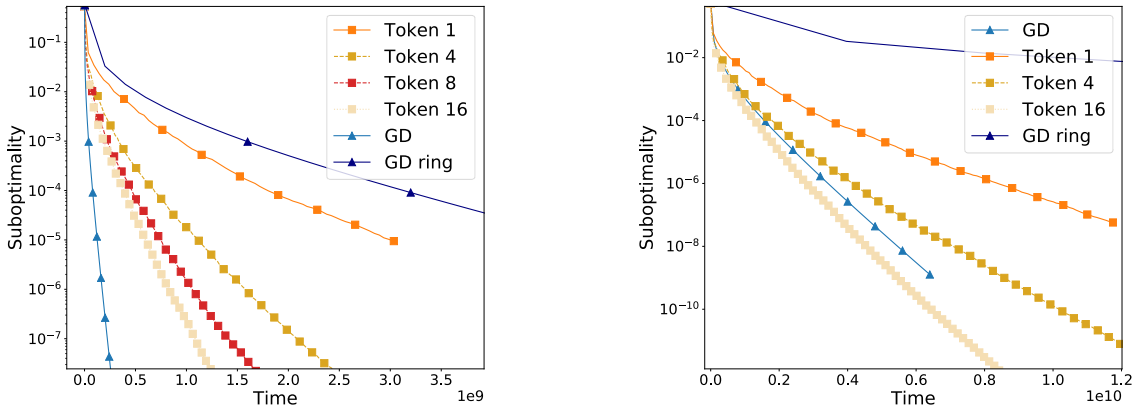


Figure 5: Evaluation of the impact on convergence time of (left): the number of tokens, (right): a straggler.

same rate in this case (since $\kappa > \gamma^{-1}$), and their curves are thus almost indistinguishable. We present additional experiments on different datasets (Phishing) and task (Ridge Regression) in Appendix D.

We have shown in Figure 4 that the empirical behaviour of token algorithms match our theory. Yet, computation and communication times are of the same order, and so the impact of some phenomena might not be straightforward. Thus, we now show in Figure 5 how using multiple tokens or introducing straggler nodes affects the execution time of token algorithms. In particular, we set $\tau_{\text{comp}} = 0.1$ and $\tau_{\text{comm}} = 10^5$, so that the computation time is small compared to the communication time. Note that this is the worst possible setting for token algorithms, which are roughly as fast as centralized algorithms (up to constants) when the computation time dominates (since they perform computations in parallel, but communications sequentially).

Impact of the number of tokens. In this case, we clearly see that the communication time decreases substantially as the number of tokens K increases, but that the marginal gains of using more tokens decrease as K approaches n. This is due to the fact that if we want to strictly enforce an i.i.d. sampling of communications, a token might have to wait that a node receives another token before being able to perform its own update, to respect the order in which transitions were sampled. This is negligible for $K << n$, but starts slowing the algorithm down for higher values of K. Removing this synchronization (for instance by picking token transitions according to a Markov Chain) would allow for faster algorithms in the limit $K \to n$.

Impact of a straggler node. When a node in the network is significantly slower than the others, synchronous algorithms such as all-to-all GD significantly slow down. Token algorithms, while also impacted, do not suffer from such a large slowdown. In Figure 5 (right), we report results for the same setting as the left figure, but with node 0 being 20 times slower than the others. In this case, we observe that token algorithms might outperform all-to-all GD in the presence of straggler nodes. Note that the same result would hold with one random slow node at each step, or with constrained bandwidth.

5 CONCLUSION

We have presented a general framework for analyzing token algorithms, and derived several variants such as variance-reduction and acceleration from it. All these token algorithms are competitive with their centralized counterparts in terms of computation and communication complexities. Multiple tokens can be used to increase the level of parallelism, and reduce the communication time.

We have also discussed a reduction from the general case to the complete communication graph case, in which our results are proven. We claim this reduction leads to efficient algorithms for general graphs, although these algorithms seem to waste communications. An important research direction would be to formalize these claims, and directly analyze versions of these algorithms in which tokens exchange information with all the nodes they visit. This would involve tight analyses of coordinate descent with Markov Chain sampling.

ACKNOWLEDGEMENTS

I would like to thank Martin Jaggi, Thijs Vogels, and the MLO group at EPFL (where this paper was written) for their support, and for reading early versions of the manuscript.

References

Ghadir Ayache, Venkat Dassari, and Salim El Rouayheb. Walk for learning: A random walk approach for federated learning from heterogeneous data. *arXiv preprint arXiv:2206.00737*, 2022.

Lucas Balthazar, João Xavier, and Bruno Sinopoli. Distributed linear estimation via a roaming token. *IEEE Transactions on Signal Processing*, 68:780–792, 2020.

Aurélien Bellet, Rachid Guerraoui, and Hadrien Hendrikx. Who started this rumor? quantifying the natural differential privacy of gossip protocols. In *34th International Symposium on Distributed Computing (DISC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

Dimitri P Bertsekas. A new class of incremental gradient methods for least squares problems. *SIAM Journal on Optimization*, 7(4):913–926, 1997.

Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine learning*, 3(1):1–122, 2011.

Andries E Brouwer and Willem H Haemers. *Spectra of graphs*. Springer Science & Business Media, 2011.

Hao Chen, Yu Ye, Ming Xiao, and Mikael Skoglund. Asynchronous parallel incremental block-coordinate descent for decentralized machine learning. *arXiv preprint arXiv:2202.03263*, 2022.

Edwige Cyffers and Aurélien Bellet. Privacy Amplification by Decentralization. In *AISTATS*, 2022.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014.

Mathieu Even, Raphaël Berthier, Francis Bach, Nicolas Flammarion, Hadrien Hendrikx, Pierre Gaillard, Laurent Massoulié, and Adrien Taylor. Continuized accelerations of deterministic and stochastic gradient descents, and of gossip algorithms. *Advances in Neural Information Processing Systems*, 34, 2021.

Hadrien Hendrikx, Francis Bach, and Laurent Massoulié. Accelerated decentralized optimization with local updates for smooth and strongly convex objectives. In *Artificial Intelligence and Statistics*, 2019a.

Hadrien Hendrikx, Francis Bach, and Laurent Massoulié. An accelerated decentralized stochastic proximal algorithm for finite sums. In *Advances in Neural Information Processing Systems*, 2019b.

Hadrien Hendrikx, Francis Bach, and Laurent Massoulié. Dual-free stochastic decentralized optimization with variance reduction. In *Advances in Neural Information Processing Systems*, 2020.

Hadrien Hendrikx, Francis Bach, and Laurent Massoulie. An optimal algorithm for decentralized finite sum optimization. *SIAM Journal on Optimization*, 2021.

Dušan Jakovetić, José MF Moura, and Joao Xavier. Linear convergence rate of a class of distributed augmented lagrangian algorithms. *IEEE Transactions on Automatic Control*, 60(4):922–936, 2014.

Björn Johansson, Maben Rabi, and Mikael Johansson. A randomized incremental subgradient method for distributed optimization in networked systems. *SIAM Journal on Optimization*, 20(3):1157–1170, 2010.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. A unified theory of decentralized SGD with changing topology and local updates. In *International Conference on Machine Learning*, pages 5381–5393. PMLR, 2020.

Dmitry Kovalev, Adil Salim, and Peter Richtárik. Optimal and practical algorithms for smooth and strongly convex decentralized optimization. *Advances in Neural Information Processing Systems*, 2020.

Dmitry Kovalev, Elnur Gasanov, Peter Richtárik, and Alexander Gasnikov. Lower bounds and optimal algorithms for smooth and strongly convex decentralized optimization over time-varying networks. *arXiv preprint arXiv:2106.04469*, 2021.

Guanghui Lan and Yi Zhou. An optimal randomized incremental gradient method. *Mathematical programming*, pages 1–49, 2017.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5 (Apr):361–397, 2004.

Huan Li and Zhouchen Lin. Revisiting extra for smooth distributed optimization. *SIAM Journal on Optimization*, 30(3):1795–1821, 2020.

Qihang Lin, Zhaosong Lu, and Lin Xiao. An accelerated randomized proximal coordinate gradient method and its

application to regularized empirical risk minimization. *SIAM Journal on Optimization*, 25(4):2244–2273, 2015.

Tao Lin, Sebastian U. Stich, and Martin Jaggi. Don't use large mini-batches, use local SGD. *arXiv preprint arXiv:1808.07217*, 2018.

Cassio G Lopes and Ali H Sayed. Incremental adaptive strategies over distributed networks. *IEEE Transactions on Signal Processing*, 55(8):4064–4077, 2007.

Xianghui Mao, Yuantao Gu, and Wotao Yin. Walk proximal gradient: An energy-efficient algorithm for consensus optimization. *IEEE Internet of Things Journal*, 6(2): 2048–2060, 2018.

Xianghui Mao, Kun Yuan, Yubin Hu, Yuantao Gu, Ali H Sayed, and Wotao Yin. Walkman: A communication-efficient random-walk algorithm for decentralized optimization. *IEEE Transactions on Signal Processing*, 68: 2513–2528, 2020.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

Angelia Nedic and Dimitri P Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, 12(1):109–138, 2001.

Angelia Nedic, Alex Olshevsky, and Wei Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27 (4), 2017.

S Sundhar Ram, A Nedić, and Venugopal V Veeravalli. Incremental stochastic subgradient algorithms for convex optimization. *SIAM Journal on Optimization*, 20(2):691–717, 2009.

Alexander Rogozin, Alexander Gasnikov, Aleksander Beznosikov, and Dmitry Kovalev. Decentralized optimization over time-varying graphs: a survey. *arXiv preprint arXiv:2210.09719*, 2022.

Kevin Scaman, Francis Bach, Sébastien Bubeck, Yin Tat Lee, and Laurent Massoulié. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In *International Conference on Machine Learning*, 2017.

Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

Suhail M Shah and Konstantin E Avrachenkov. Linearly convergent asynchronous distributed admm via markov sampling. *arXiv preprint arXiv:1810.05067*, 2018.

Shai Shalev-Shwartz. Sdca without duality, regularization, and individual convexity. In *International Conference on Machine Learning*, pages 747–754. PMLR, 2016.

Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

Sebastian U Stich. Local SGD converges fast and communicates little. *International Conference on Learning Representations*, 2018.

Tao Sun, Dongsheng Li, and Bao Wang. Adaptive random walk gradient descent for decentralized optimization. In *International Conference on Machine Learning*, pages 20790–20809. PMLR, 2022.

César A Uribe, Soomin Lee, Alexander Gasnikov, and Angelia Nedić. A dual approach for optimal algorithms in distributed optimization over networks. *Optimization Methods and Software*, 2020.

Yu Ye, Hao Chen, Ming Xiao, Mikael Skoglund, and H Vincent Poor. Incremental admm with privacy-preservation for decentralized consensus optimization. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 209–214. IEEE, 2020.

A Revisiting Bregman Coordinate Descent

The algorithmic core for the non-accelerated methods is based on Bregman Coordinate descent. Yet, the existing theory from Hendrikx et al. [2020] was not satisfactory, as it did not allow to consider stochastic communications: the communication block was sampled all at once, which is not possible in the token approach.

Similarly, in general graphs, we do not know the exact sampling probability $\tilde{p}_{i,t}$, but an upper bound p_i of it. We thus changed the coordinate descent algorithm to also take that into account.

We adapt the result from Hendrikx et al. [2020] to tackle these two problems in this section. First of all, for two vectors $x, y \in \mathbb{R}^d$, define the Bregman divergence:

$$D_h(x,y) = h(x) - h(y) - \nabla h(y)^\top (x-y). \tag{4}$$

In order not to worry about further regularity assumptions, we assume throughout this paper that all the functions we consider are twice continuously differentiable and strictly convex on $\operatorname{dom} h$, and that $\nabla h(x) = \min_y h(y) - x^\top y$ is uniquely defined. This is not very restrictive for typical machine learning objectives. The Bregman divergence has some interesting properties. In particular, $D_h(x,y) \geq 0$ for all $x, y \in \mathbb{R}^d$ as soon as h is convex. For $i \in [d]$, we denote $e_i \in \mathbb{R}^d$ the unit vector corresponding to coordinate i. Consider the following Bregman coordinate descent algorithm, in which the iterates are given by:

$$x_{t+1} = \arg\min_x \left\{ V_{i,t}(x) \hat{=} \frac{\eta_t}{p_i} \nabla_i f(x_t)^\top A^\dagger A x + D_h(x, x_t) \right\}, \tag{5}$$

where $\nabla_i f(x_t) = e_i e_i^\top \nabla f(x_t)$, and $A^\dagger A$ is some projection matrix, which is such that $A^\dagger A \nabla f(x) = \nabla f(x)$ for all $x \in \mathbb{R}^d$. An equivalent way to write these iterations is:

$$\nabla h(x_{t+1}) = \nabla h(x_t) - \frac{\eta_t}{p_i} A^\dagger A \nabla_i f(x_t) \tag{6}$$

Although we use some p_i for the learning rate, we assume that coordinates are sampled according to another distribution, namely, $\tilde{p}_{i,t}$. In order to make the training stable, we assume that for all i, t, there exist $\delta_{i,t} > 0$ which are such that for all i,

$$\tilde{p}_{i,t}(1 + \delta_{i,t}) = p_i. \tag{7}$$

In particular, this means that the p_i are not a proper probability distribution, since they don't sum to one. We denote $\Delta = \sum_{i=1}^d \tilde{p}_{i,t}\delta_{i,t}$, which is such that $1 + \Delta$ is normalizing factor of the p_i. Similarly, we denote $\delta = \min_{i,t} \delta_{i,t}$. We denote $R_i = (A^\dagger A)_{ii}$ and $R_p = \min_i R_i^{-1} \tilde{p}_{i,t}$. We make the following assumptions on f and h.

Assumption 1 (Regularity assumptions)**.** *Function f is μ-relatively strongly convex and L_i-relatively smooth in the direction i with respect to h, meaning that for all $x, y \in \mathbb{R}^d$, and some v_i supported by e_i:*

$$\mu D_h(x,y) \leq D_f(x,y), \quad \text{and} \quad D_f(x + v_i, x) \leq L_i D_h(x + v_i, x). \tag{8}$$

We also make some technical assumptions, that are verified for our problem. In particular, we assume that h and $A^\dagger A$ are such that:

$$\nabla_i f(x_t)^\top A^\dagger A (x_t - x_{t+1}^{(i)}) = R_i \nabla f(x_t)^\top (x_t - x_{t+1}^{(i)}). \tag{9}$$

Using this assumption, we first prove a technical lemma, which ensures that each step reduces the function value:

Lemma 1 (Monotonicity)**.** *Under Assumption 1, if $\eta_t \leq \frac{p_i}{L_i R_i}$ the iterates of Equation (5) verify for all $i \in [d]$:*

$$f(x_{t+1}^{(i)}) \leq f(x_t). \tag{10}$$

Proof. Using Assumption 1 , we have that:

$$\begin{aligned}
D_h(x_{t+1}^{(i)}, x_t) + D_h(x_t, x_{t+1}^{(i)}) &= \left[\nabla h(x_{t+1}^{(i)}) - \nabla h(x_t)\right]^\top (x_{t+1}^{(i)} - x_t) \\
&= \frac{\eta_t}{p_i} \nabla_i f(x_t)^\top A^\dagger A (x_t - x_{t+1}^{(i)}) \\
&= \frac{\eta_t R_i}{p_i} \nabla f(x_t)^\top (x_t - x_{t+1}^{(i)}) \\
&= \frac{\eta_t R_i}{p_i} \left[D_f(x_{t+1}^{(i)}, x_t) - f(x_{t+1}^{(i)}) + f(x_t)\right] \\
&\leq \frac{\eta_t}{p_i} R_i \left[L_i D_h(x_{t+1}^{(i)}, x_t) - f(x_{t+1}^{(i)}) + f(x_t)\right] \\
&\leq D_h(x_{t+1}^{(i)}, x_t) - \frac{\eta_t R_i}{p_i} \left[f(x_t) - f(x_{t+1}^{(i)})\right],
\end{aligned}$$

where the last line uses that $\eta_t \leq \frac{p_i}{L_i R_i}$. In particular:

$$f(x_{t+1}^{(i)}) \leq f(x_t) - D_h(x_t, x_{t+1}^{(i)}) \leq f(x_t).$$

□

Theorem 6. *Consider two functions f and h that verify Assumption 1. If the iterates are given by Equation (5), and denoting for any $x \in \operatorname{dom} h$, $\mathcal{L}_t = (1+\delta) D_\phi(x, x_t) + \frac{\eta_t}{R_p}[f(x_t) - f(x)]$, we obtain for $\eta_t \leq \frac{p_i}{L_i R_i}$:*

$$\mathcal{L}_{t+1} \leq \max\left(\frac{1 + \Delta - \eta_t \mu}{1+\delta}, 1 - R_p\right) \mathcal{L}_t$$

Note that $\frac{1+\Delta-\eta\mu}{1+\delta} \geq 1 - \eta\mu \geq 1 - \frac{p_i}{R_i}\frac{\mu}{L_i} \geq 1 - \frac{(1+\delta_{i,t})\mu}{L_i} R_p$, so the first term generally dominates since we generally have $(1+\delta_{i,t})\mu \leq L_i$ unless the condition number is very small, or $\delta_{i,t}$ very large.

Proof of Theorem 6. For any $x \in \operatorname{dom} h$ We start by writing the 3 points inequality:

$$\begin{aligned}
D_h(x, x_{t+1}) + D_h(x_{t+1}, x_t) - D_h(x, x_t) &= [\nabla h(x_t) - \nabla h(x_{t+1})]^\top (x - x_{t+1}) \\
&= \frac{\eta_t}{p_i} \nabla_i f(x_t)^\top A^\dagger A (x - x_{t+1}) \\
&= \frac{\eta_t}{p_i} \nabla_i f(x_t)^\top (x - x_t) + \frac{\eta_t}{p_i} \nabla_i f(x_t)^\top A^\dagger A (x_t - x_{t+1}).
\end{aligned}$$

We now multiply everything by $1 + \delta_{i,t}$, leading to:

$$\begin{aligned}
(1+\delta_{i,t}) \left[D_h(x, x_{t+1}) + D_h(x_{t+1}, x_t) - D_h(x, x_t)\right] &= \frac{\eta_t (1+\delta_{i,t})}{p_i} \nabla_i f(x_t)^\top A^\dagger A (x - x_t) \\
&\quad + \frac{\eta_t (1+\delta_{i,t})}{p_i} \nabla_i f(x_t)^\top A^\dagger A (x_t - x_{t+1}),
\end{aligned}$$

which we rewrite as:

$$\begin{aligned}
(1+\delta) D_h(x, x_{t+1}) &\leq (1+\delta_{i,t}) D_h(x, x_t) + \frac{\eta_t}{\tilde{p}_{i,t}} \nabla_i f(x_t)^\top A^\dagger A (x - x_t) \\
&\quad + \frac{\eta_t}{\tilde{p}_{i,t}} \nabla_i f(x_t)^\top A^\dagger A (x_t - x_{t+1}) - (1+\delta_{i,t}) D_h(x_{t+1}^{(i)}, x_t),
\end{aligned}$$

In particular, when taking an expectation with respect to the sampling distribution, we obtain:

$$\begin{aligned}
(1+\delta) \mathbb{E}\left[D_h(x, x_{t+1})\right] &\leq \sum_{i=1}^d p_i D_h(x, x_t) - \sum_{i=1}^d p_i D_h(x_{t+1}^{(i)}, x_t) \\
&\quad + \eta_t \nabla f(x_t)^\top (x - x_t) + \sum_{i=1}^d \eta_t \nabla_i f(x_t)^\top A^\dagger A (x_t - x_{t+1}^{(i)}).
\end{aligned}$$

Using Assumption 1 leads to:

$$\begin{aligned}\eta_t \nabla_i f(x_t)^\top A^\dagger A(x_t - x_{t+1}^{(i)}) &= \eta_t R_i \nabla f(x_t)^\top (x_t - x_{t+1}^{(i)}) \\ &= \eta_t R_i \left[D_f(x_{t+1}^{(i)}, x_t) - f(x_{t+1}^{(i)}) + f(x_t)\right] \\ &\leq \eta_t R_i \left[L_i D_h(x_{t+1}^{(i)}, x_t) - f(x_{t+1}^{(i)}) + f(x_t)\right]\end{aligned}$$

Using that $\eta_t \leq \frac{p_i}{L_i R_i}$, we obtain:

$$(1+\delta)\mathbb{E}\left[D_h(x, x_{t+1})\right] \leq (1+\Delta) D_h(x, x_t) + \eta_t \nabla f(x_t)^\top (x - x_t) + \sum_{i=1}^d \eta_t R_i \left[-f(x_{t+1}^{(i)}) + f(x_t)\right].$$

Note that by monotonicity of the iterations (Lemma 1), $f(x_{t+1}^{(i)}) \leq f(x_t)$ for all t, i, and so, with $R_p = \min_i R_i^{-1} \tilde{p}_{i,t}$:

$$\sum_{i=1}^d \frac{R_i}{\tilde{p}_{i,t}} \tilde{p}_{i,t} \left[-f(x_{t+1}^{(i)}) + f(x_t)\right] \leq R_p^{-1} \left[-\sum_{i=1}^d \tilde{p}_{i,t} f(x_{t+1}^{(i)}) + f(x_t)\right] \leq -R_p^{-1} \left(\mathbb{E}\left[f(x_{t+1})\right] - f(x_t)\right) \tag{11}$$

Finally, the μ-relative strong convexity of f gives:

$$\nabla f(x_t)^\top (x - x_t) = -\left[f(x_t) - f(x)\right] - D_f(x, x_t) \leq -\mu D_h(x, x_t) - f(x_t) + f(x) \tag{12}$$

Combining everything leads to:

$$\begin{aligned}(1+\delta)\mathbb{E}\left[D_h(x, x_{t+1})\right] + \frac{\eta_t}{R_p}\left[f(x_{t+1} - f(x)\right] \\ \leq \frac{1 + \Delta - \eta\mu}{1+\delta}(1+\delta) D_h(x, x_t) + \frac{\eta_t}{R_p}(1 - R_p)\left[f(x_t) - f(x)\right]\end{aligned}$$

□

B Convergence results

B.1 Introducing the problem

In this section, we introduce the consensus problem derived from the conceptual graph, and take its dual formulation. This section follows the same framework as Hendrikx et al. [2020]. Denoting $\tilde{\sigma} = \frac{n\sigma}{n+K}$, this problem writes:

$$\min_{\substack{\theta \in \mathbb{R}^{n\times d},\ u\in\mathbb{R}^{n\times d},\ v\in\mathbb{R}^d \\ \forall i,\ \forall j,\ \theta^{(ij)} = u^{(i)},\ \forall k,\ u^{(i)} = v^{(j)}}} \sum_{i=1}^n \sum_{j=1}^m f_{ij}(\theta^{(ij)}) + \frac{\tilde{\sigma}}{2}\|u^{(i)}\|^2 + \sum_{k=1}^K \frac{\tilde{\sigma}}{2}\|v^{(k)}\|^2. \tag{13}$$

Introducing Lagrangian multipliers y for each consensus constraint in the virtual part of the graph (between center nodes $u^{(i)}$ and computation nodes $\theta^{(ij)}$), and multipliers x for the communication part of the graph, (between center nodes $u^{(i)}$ and tokens $v^{(k)}$), the dual problem writes:

$$\min_{x\in\mathbb{R}^{nKd},\ y\in\mathbb{R}^{nmd}} q_A(x, y) + \sum_{i=1}^n \sum_{j=1}^m f_{ij}^*((Ay)^{(ij)}), \text{ with } q_A(x, y) \triangleq \frac{1}{2\tilde{\sigma}}(x, y)^\top A^\top A(x, y), \tag{14}$$

where $A \in \mathbb{R}^{(n(m+1)+K)d \times n(K+m)d}$ is the (weighted) incidence matrix of the augmented conceptual graph, which is such that $A e_{\ell_1\ell_2} = \mu_{\ell_1\ell_2}(e_{\ell_1} - e_{\ell_2})$, for any two nodes ℓ_1 and ℓ_2 (and an arbitrary orientation), where $e_{\ell_1\ell_2}$ is the unit vector corresponding to edge (ℓ_1, ℓ_2), and e_{ℓ_1}, e_{ℓ_2} are the unit vectors corresponding to nodes ℓ_1 and ℓ_2. Note that we abused notations and wrote $(Ay)^{(ij)}$ instead of $(A(x, y))^{(ij)}$, since $(A(x, y))^{(ij)}$ does not actually depend on x.

Matrix A has a very special structure, since it is the incidence matrix of a tripartite graph between computation nodes, communication nodes, and token nodes. We now have to choose the weights of matrix A. For communication edges

(between communication nodes and tokens), we make the simple choice $\mu_{ik} = 1$. For computation edges (between communication nodes and computation nodes), we choose:

$$\mu_{ij}^2 = \alpha L_{ij}, \text{ with } \alpha = \frac{2\tilde{\sigma}K}{\kappa_s}, \tag{15}$$

where L_{ij} is the smoothness of function f_{ij} and $\kappa_s = \max_i 1 + \sum_{j=1}^m L_{ij}/\tilde{\sigma}$. These choices follow from Hendrikx et al. [2020], where we used that the "communication part" of the conceptual graph used to derive the token algorithm is quite specific (complete bipartite graph), and so $\lambda_{\min}(A_{\text{comm}}^\top A_{\text{comm}}) = K$, the number of tokens, as long as $K \leq n$ [Brouwer and Haemers, 2011].

B.2 The TVR algorithm

We now proceed to the derivation of the DVR algorithm, and to proving its theoretical guarantees. To achieve this, we use the Bregman coordinate descent iterations defined in the previous section, which are of the form:

$$(x,y)_{t+1} = \arg\min_{x,y} \frac{\eta_t}{p_i}\nabla_i \left[q_A((x,y)_t) + F^*(Ay_t)\right]^\top A^\dagger Ax + D_h((x,y),(x,y)_t), \tag{16}$$

where $F^*(\lambda) = \sum_{i,j} f_{ij}^*(\lambda^{(ij)})$. We now need to define the reference function h, which we define, following Hendrikx et al. [2020], as $h(x,y) = h_x(x) + h_y(y)$, with:

$$\forall i,j,\ h_y^{(ij)}(y^{(ij)}) = \frac{L_{ij}}{\mu_{ij}^2} f_{ij}^*(\mu_{ij} y^{(ij)}), \text{ and } h_x(x) = \frac{1}{2}\|x\|^2_{A_{\text{comm}}^\dagger A_{\text{comm}}}, \tag{17}$$

where $A_{\text{comm}} \in \mathbb{R}^{(n+K)d\times(n(m+K)}$ is the restriction of A to communication nodes (and tokens). In order to avoid notations clutter, we slightly abuse notations and use $A^\dagger A$ instead in the remainder of this section.

Algorithm 2 Token Variance Reduced (z_0)

$\tilde{\sigma} = \sigma\frac{n}{n+K}$ $\alpha = \frac{2K\tilde{\sigma}}{\kappa_s}$, $p_{\text{comp}} = \left(1 + \frac{\kappa_s}{m-1+\kappa_s}\right)^{-1}$, $p_{\text{comm}} = 1 - p_{\text{comp}}$,
$\eta = \min\left(\frac{\tilde{\sigma}p_{\text{comm}}}{2nK}, \frac{p_{\text{comp}}}{\alpha n(m-1+\kappa_s)}\right)$, $\rho_{\text{comm}} = \frac{nK\eta}{p_{\text{comm}}\tilde{\sigma}}$, $\rho_{\text{comp}} = \frac{mn\alpha\eta}{p_{\text{comp}}}$. *// Init*
$\forall i \in [n]$, $\theta_0^{(i)} = -\sum_{j=1}^m \nabla f_{ij}(z_0^{(i,j)})/\tilde{\sigma}$; $\forall k \in [K]$, $\theta_0^{\text{token},k} = 0$. *// z_0 is arbitrary but not θ_0.*
for $t = 0$ to $T-1$ **do** *// Run for T iterations*
  if communication step (with probability p_{comm}) **then**
    Pick $i \sim \mathcal{U}([n])$, $k \sim \mathcal{U}([K])$ *// Choose next node and token uniformly at random*
    $\theta_{t+1}^{\text{token},k} = \theta_t^{\text{token},k} - \rho_{\text{comm}}(\theta_t^{\text{token},k} - \theta_t^{(i)})$ *// Token update*
    $\theta_{t+1}^{(i)} = \theta_t^{(i)} + \rho_{\text{comm}}(\theta_t^{\text{token},k} - \theta_t^{(i)})$ *// Local node update*
  else
    Pick $i \sim \mathcal{U}([n])$, $j \sim \mathcal{U}([m])$ *// Choose one node and data point at random*
    $z_{t+1}^{(i,j)} = (1-\rho_{\text{comp}})\, z_t^{(i,j)} + \rho_{\text{comp}}\theta_t^{(i)}$ *// Virtual node update*
    $\theta_{t+1}^{(i)} = \theta_t^{(i)} - \frac{1}{\tilde{\sigma}}\left(\nabla f_{ij}(z_{t+1}^{(i,j)}) - \nabla f_{ij}(z_t^{(i,j)})\right)$ *// Local update using f_{ij}*
return θ_K

For the computation part, the algorithm can be recovered by following the exact same steps as Hendrikx et al. [2020, Section 2], which are themselves inspired by the dual-free updates from Lan and Zhou [2017].

For the communication part, there is a small difference in the fact that we do not sample all coordinates at once anymore. Instead, we sample edges of the communication graph one by one. The other difference is that communication edges are not between node i and node k anymore, but between node i and *token* k. In particular, the communication step writes:

$$A^\dagger A x_{t+1} = A^\dagger A x_t - \frac{\eta}{\tilde{\sigma}p_{ik}} A^\dagger A e_{ij} e_{ik}^\top A^\top A(x_t, y_t). \tag{18}$$

Multiplying by A on the left and defining $\theta_t = A_{\text{comm}}(x_t, y_t)$ leads to:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\tilde{\sigma}p_{ik}} A_{\text{comm}} e_{ik} e_{ik}^\top A^\top \theta_t, \tag{19}$$

where we used the fact that the update only affects communication nodes. Therefore, when we sample an edge between node i and token k, this leads to:

$$\theta_{t+1} = \theta_t - \frac{\eta n K}{p_{\text{comm}} \tilde{\sigma}} W_{ik} \theta_t, \tag{20}$$

where $W_{ik} = (e_i - e_k)(e_i - e_k)^\top$, thus leading to the form obtained in Algorithm 2. From there, we can prove the following convergence theorem, from which all the non-accelerated results from the main text are derived.

Theorem 7. *The iterates of Algorithm 2 verify:*

$$\|\theta_t - \theta_\star\|^2 \leq \left(1 - \frac{\eta K \tilde{\sigma}}{\kappa_s}\right)^t C_0, \tag{21}$$

where $C_0 = 2\tilde{\sigma}\mathcal{L}_0$, with $\mathcal{L}_0$ the Lyapunov function from Theorem 6 instantiated on the dual problem, which thus depends on the initialization. In particular, ignoring log factors in C_0, if $p_{\text{comm}} = p_{\text{comp}} = \frac{1}{2}$ and for any $\varepsilon > 0$,

$$T_{\text{comp}} = O((m + \kappa_s) \log \varepsilon^{-1}) \;\; \textit{and} \;\; T_{\text{comm}} = O(n \kappa_s \log \varepsilon^{-1}) \tag{22}$$

are required in total in order to obtain $\|\theta_t - \theta_\star\|^2 \leq \varepsilon$.

The proof of this algorithm follows several steps, that we detail in the next subsections. We first show that the objective function defined in Equation (13), together with the reference function h defined in Equation (17) satisfies Assumption 1, so that we can apply Theorem 6. Then, we show how to choose the remaining parameters (and in particular p_{comm} and p_{comp}) optimally, and evaluate the rate in terms of constants of the problem (number of nodes, number of tokens, smoothness and strong convexity of the local functions...).

B.3 Verifying Assumption 1

We start by showing that Assumption 1 is verified in this case, which includes three parts: Equation (9), relative strong convexity, and directional relative smoothness.

B.3.1 Verifying Structural assumptions.

We first verify that the updates of our problem verify Equation (9) from Assumption 1.

1 - Computation coordinates. Computation coordinates corresponding to the edge between computation and communication subnodes in Figure 1. In particular, the graph becomes disconnected if they are removed, thus implying that $A^\dagger A e_i = e_i$. In particular,

$$\nabla h(x_{t+1}) = \nabla h(x_t) - \frac{\eta_t}{p_i} \nabla_i f(x_t). \tag{23}$$

Yet, for our specific choice of h (which is such that $h_{\text{comp}}(x) = \sum_i {h_{\text{comp}}}^{(i)}(x^{(i)})$), this implies that $x_{t+1}^{(i)} - x_t = v_i$ for some v_i that only has support on coordinate i, and in particular:

$$\nabla_i f(x_t)^\top A^\dagger A (x_t - x_{t+1}^{(i)}) = R_i \nabla_i f(x_t)^\top v_i = R_i \nabla f(x_t)^\top v_i = R_i \nabla f(x_t)^\top (x_t - x_{t+1}^{(i)}) \tag{24}$$

2 - Network coordinates. Network coordinates have a different structure. In this case, the reference function h_{comm} is the quadratic form induced by $A^\dagger A$, which facilitates analysis. The updates write:

$$A^\dagger A x_{t+1} = A^\dagger A x_t - A^\dagger A \frac{\eta_t}{p_i} \nabla_i f(x_t) \tag{25}$$

Although h_{comm} is not separable, we can leverage the presence of $A^\dagger A$ to write:

$$\nabla_i f(x_t)^\top A^\dagger A (x_t - x_{t+1}^{(i)}) = \frac{\eta_t}{p_i} \nabla_i f(x_t)^\top A^\dagger A \nabla_i f(x_t) = \frac{\eta_t}{p_i} R_i \nabla f(x_t)^\top \nabla_i f(x_t).$$

We then use that $\nabla f(x_t) = A^\dagger A \nabla f(x_t)$, leading to:

$$\nabla_i f(x_t)^\top A^\dagger A (x_t - x_{t+1}^{(i)}) = \frac{\eta_t}{p_i} R_i \nabla f(x_t)^\top A^\dagger A \nabla_i f(x_t) = R_i \nabla f(x_t)^\top A^\dagger A (x_t - x_{t+1}^{(i)}).$$

Now that we have proven that h and f verify the structural assumptions given by Equation (9), it remains to evaluate the relative strong convexity and directional smoothness constants μ and L_i.

B.3.2 Relative Strong Convexity.

Since the structure of the dual problem and the reference function h are the same, we directly have from Hendrikx et al. [2020, Appendix B.1] that the relative strong convexity constant is equal to

$$\mu = \frac{\alpha}{2} = \frac{K}{\tilde{\sigma} + \sum_{j=1}^{m} L_{ij}}, \tag{26}$$

since the smallest eigenvalue of the communication graph is equal to K (the number of tokens) in this case.

B.3.3 Directional Relative Smoothness.

We now evaluate the relative smoothness constants.

1 - Computation edges. In the computation case, similarly to strong convexity, we directly get from Hendrikx et al. [2020, Appendix B.1] that $\tilde{L}_{ij}$, the relative directional smoothness for virtual node i, j (or just $\tilde{L}_i$ if there is only one virtual node), can be obtained as:

$$\tilde{L}_{ij} = \alpha \left(1 + \frac{L_{ij}}{\sigma_i}\right), \tag{27}$$

Plugging in the value of α, this leads to:

$$\tilde{L}_{ij} = \frac{2K}{\tilde{\sigma}} \frac{\tilde{\sigma} + L_{ij}}{\tilde{\sigma} + \sum_{j=1}^{m} L_{ij}} \tag{28}$$

2 - Communication edges. In this case, we cannot use the results from DVR directly because the sampling of communication coordinates is different. While DVR sampled all communication edges at once, we only sample one at each step. In this case, we have that the directional relative smoothness is equal to:

$$D_f(x + \Delta_{uv}, x) = \|\Delta_{uv}\|^2_{A^\top \Sigma A} = \mu_{uv}^2(\sigma_u^{-1} + \sigma_v^{-1})\|\Delta_{uv}\|^2 = \frac{\mu_{uv}^2(\sigma_u^{-1} + \sigma_v^{-1})}{e_{uv}^\top A^\dagger A e_{uv}} \|\Delta_{uv}\|^2_{A^\dagger A}. \tag{29}$$

In particular, for communication edges, and with the choice that $\mu_{uv}^2 = 1$ and $\sigma_u = \sigma_v = \tilde{\sigma}$:

$$D_f(x + \Delta_{uv}, x) \leq \tilde{L}_{uv} D_h(x + \Delta_{uv}, x), \quad \text{with } \tilde{L}_{uv} = \frac{2}{\tilde{\sigma} R_{uv}} \tag{30}$$

B.4 Convergence guarantees

We have shown in the previous subsection that we can apply Theorem 6 to obtain convergence guarantees for our token algorithms. For the communication edges, the step-size constraint leads to:

$$\eta_t \leq \frac{p_{uv}}{R_{uv}\tilde{L}_{uv}} = \frac{p_{\text{comm}}\tilde{\sigma}}{2nK} \tag{31}$$

For the computation edges, we can set (as in the DVR article) $p_{ij} \propto 1 + L_{ij}/\tilde{\sigma}$. In particular, the normalizing factor is equal to $\sum_{i=1}^{n}\sum_{j=1}^{m} 1 + L_{ij}/\tilde{\sigma} = n(m + \sum_{j=1}^{m} L_{ij}/\tilde{\sigma}) = n(m - 1 + \kappa_s)$, where we recall that $\kappa_s = 1 + \sum_{j=1}^{m} L_{ij}/\tilde{\sigma}$. Therefore, we obtain:

$$\eta_t \leq \frac{p_{ij}}{\tilde{L}_{ij}} \leq \frac{p_{\text{comp}}\tilde{\sigma}\kappa_s}{2nK(m - 1 + \kappa_s)}. \tag{32}$$

We want to balance p_{comm} and p_{comp} such that these two constraints match, leading to:

$$p_{\text{comm}} = \frac{\kappa_s}{m - 1 + \kappa_s} p_{\text{comp}}. \tag{33}$$

Since $p_{\text{comm}} = 1 - p_{\text{comp}}$, this leads to:

$$p_{\text{comp}} = \left(1 + \frac{\kappa_s}{m - 1 + \kappa_s}\right)^{-1}. \tag{34}$$

Assuming $\delta \geq 0$ and $\Delta < \frac{\eta_t \mu}{2}$, the rate of convergence of Algorithm 2 is $\rho = \eta_t \mu - \Delta \geq \eta_t \mu / 2$. In particular, we directly have that the computation complexity is equal to:

$$\frac{p_{\text{comp}}}{\rho} = 4(m - 1 + \kappa_s). \tag{35}$$

Similarly, the communication complexity is equal to:

$$\frac{p_{\text{comm}}}{\rho} = 2n\kappa_s. \tag{36}$$

B.5 Special cases

B.5.1 Complete graphs

All the theorems in the main paper are actually direct corollaries of Theorem 7. We provide below how they can be derived in each case.

Theorem 3: We apply Theorem 7 with $\delta = 0$ (since the graph is complete, so we know the true sampling distribution).

Theorem 2: When $m = 1$, all the derivations remain the same, but we now have that $\kappa_s = 1 + L_i/\tilde{\sigma} = \kappa$, and so the computation complexity is equal to $m - 1 + \kappa_s = \kappa$.

Theorem 1: This result can be recovered by simply taking $K = 1$.

B.5.2 General graphs.

Consider that the transitions between nodes are ruled by matrix W, which is such that $\|W^t \pi_0 - \pi_\star\|_\infty \leq C(1-\gamma)^t$ for any starting distribution π_0, with $C > 0$ a constant, $\pi_\star$ the stationary distribution of the random walk, and $\gamma > 0$ a constant which can be interpreted as the inverse of the mixing time of the Markov Chain with transition matrix W. This is true as long as the underlying Markov Chain is irreducible and aperiodic. In this case, then after $O(\gamma^{-1} \log(C/(\eta\mu))$ steps, we have that for all i:

$$|\tilde{p}_{i,t} - (\pi_\star)_i| \leq \frac{\eta\mu}{4}, \tag{37}$$

so in particular by taking $p_i = (\pi_\star)_i + \frac{\eta\mu}{4}$ satisfies $\tilde{p}_{i,t}(1 + \delta_{i,t}) = p_i$, with $0 \leq \delta_{i,t} \leq \frac{\eta\mu}{2}$. Then, using Theorem 6, we recover the same result as in Theorem 7, with $\eta\mu$ replaced by $\eta\mu - \Delta \geq \frac{\eta\mu}{2}$.

Note that the value of η depends on Δ, which itself depends on η, so the above derivations technically result in a circular argument. To avoid this, one can simply use a slightly different $\tilde{\eta} = \min_i \frac{(\pi_\star)_i}{L_i R_i} \leq \eta$ to set the number of token jumps. In practice, we do not need to precisely evaluate these log factors, and taking $\tilde{C}\gamma^{-1}$ jumps with a small constant $\tilde{C}$ is enough.

C Acceleration

In the accelerated case, the theory does not follow directly from Theorem 7, since the algorithmic core is different. Indeed, we use a variant of Accelerated Proximal Coordinate Gradient [Lin et al., 2015] instead of Bregman coordinate descent on the dual formulation. Yet, we can directly reuse the convergence results for ADFS [Hendrikx et al., 2019b, Theorem 1], which we (informally) state below:

Theorem 8. *ADFS has iteration complexity $O(\rho \log \varepsilon^{-1})$, with*

$$\rho^2 \leq \min_{k\ell} \frac{\lambda_{\min}^+(A^\top \Sigma^{-1} A)}{\Sigma_{kk}^{-1} + \Sigma_{ll}^{-1}} \frac{p_{k\ell}^2}{\mu_{k\ell}^2 R_{k\ell}}. \tag{38}$$

For our problem (conceptual graph), we obtain the following values for the parameters involved in the computation of ρ

when a communication edge (k,ℓ) between a node and the token is sampled:

$$\lambda_{\min}^{+}(A^\top \Sigma^{-1} A) = \frac{K}{2\tilde{\sigma}\kappa_s}$$
$$\Sigma_{kk}^{-1} = \Sigma_{\ell\ell}^{-1} = \tilde{\sigma}^{-1}$$
$$p_{k\ell} = \frac{p_{\text{comm}}}{nK}$$
$$\mu_{k\ell} = 1$$
$$R_{k\ell} = \frac{1}{K}.$$

In the end, this leads to

$$\rho_{\text{comm}}^2 = \frac{p_{\text{comm}}^2}{4n^2\kappa_s}. \tag{39}$$

Similarly, we have (just like in the ADFS paper, since the computation part of the graph is the same):

$$\rho_{\text{comp}}^2 = \frac{p_{\text{comp}}^2}{2n^2(m+\sqrt{m\kappa_s})^2}. \tag{40}$$

We now fix p_{comm} and p_{comp} so that $\rho_{\text{comm}} = \rho_{\text{comp}}$, similarly to Section B.4. This leads to

$$p_{\text{comm}} = \frac{2\kappa_s}{m+\sqrt{m\kappa_s}} p_{\text{comp}}, \tag{41}$$

and so the communication and computation complexities are respectively:

$$\frac{p_{\text{comm}}}{\rho_{\text{comm}}} = 2n\kappa_s, \text{ and } \frac{p_{\text{comp}}}{\rho_{\text{comp}}} = \sqrt{2}n(m+\sqrt{m\kappa_s}).$$

Theorem 4 is obtained by expressing these complexities in terms of per-node and per-token quantities.

D Experiments

D.1 Experimental setting

For the experiments, we use the same setting as Hendrikx et al. [2020], meaning that we solve the following logistic regression problem:

$$\min_{\theta\in\mathbb{R}^d} \sum_{i=1}^{n} \left[\frac{\sigma}{2}\|\theta\|^2 + \sum_{j=1}^{m} \frac{1}{m} \log(1+\exp(-y_{ij} X_{ij}^\top \theta)) \right], \tag{42}$$

where the pairs $(X_{ij}, y_{ij}) \in \mathbb{R}^d \times \{-1,1\}$ are taken from the RCV1 dataset, which we downloaded from https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html. We choose the regularization parameter as $\sigma = 10^{-5}$. All experiments were run on a standard laptop, but using MPI to communicate between nodes.

Time. We choose to report ideal times: to get the execution time of an algorithm, we compute the minimum time it takes to execute its sequence of updates, given fixed communication and computation delays τ_{comm} and τ_{comp}. More specifically, we draw a sequence of actions S, and denote S_ℓ the ℓ-th action from this sequence, and $T_i(\ell)$ the time at which node i finishes executing update ℓ. All nodes start from $T_i(0) = 0$.

- If S_ℓ is a local computation at node i, then node i increases its local time by τ_{comp}, *i.e.*, $T_i(\ell) = T_i(\ell-1) + \tau_{\text{comp}}$. For $j \neq i$, $T_j(\ell) = T_j(\ell-1)$.
- If S_ℓ is the jump of token k from node j to node i, then $T_i(\ell) = \max(T_i(\ell-1), T_j(\ell-\tau_k) + \tau_{\text{comm}})$, where $T_j(\ell-\tau_k)$ is the time at which node j received token k (and so was able to start sending it to another node). For $j' \neq i, j$, $T_{j'}(\ell) = T_{j'}(\ell-1)$.

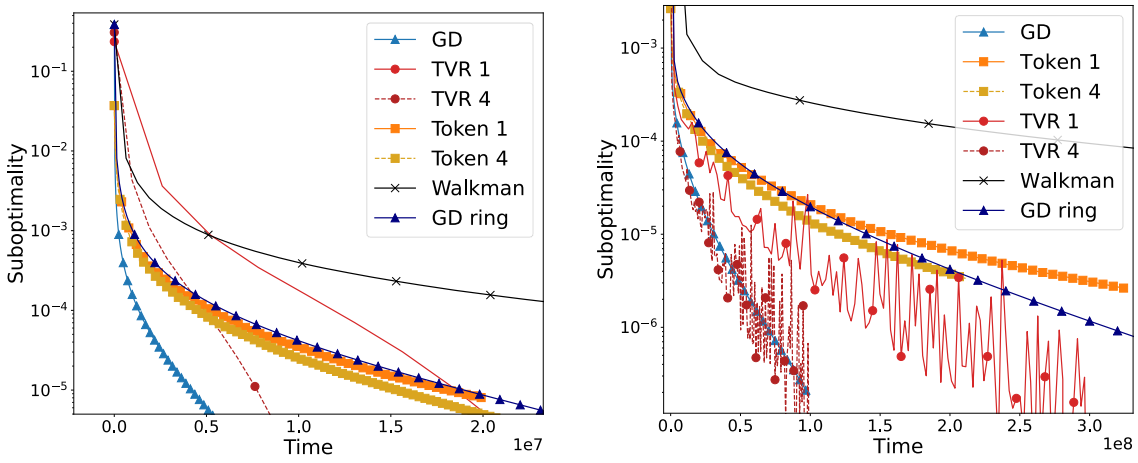


Figure 6: Additional convergence results. Left: Logistic Regression on the Phishing dataset. Right: Ridge Regression over the YearPredictionMSD dataset.

The sequence S_ℓ is implemented by simply sharing a random seed between nodes. Token algorithms could also be implemented without executing this shared schedule, but this would not strictly correspond to Algorithm 2 since the sampling of the edges would not be *i.i.d.*.

Batch smoothness. Since the smoothness of the full functions f_i is hard to compute, we approximated it by taking $L_{\text{batch}} = 0.02 \times \max_{ij} L_{ij}$ for the logistic regression task over the RCV1 dataset. Note that, following Hendrikx et al. [2020], we implemented TVR with this batch smoothness instead of $\sum_j L_{ij}$ (which corresponds to taking $\alpha = 2\tilde{\sigma}K/\kappa$ instead of $\alpha = 2\tilde{\sigma}K/\kappa_s$). In particular, the communication complexity of TVR is thus proportional to κ (similarly to that of Algorithm 1) instead of κ_s. We proved Theorem 3 with κ_s since it is simpler and less restrictive.

Code. We provide the code used to run all experiments in supplementary material. All algorithms are coded in Python, using MPI for communications. This code has not been optimized for efficiency, but rather aims at providing an actual implementation of token algorithms that can be used out of the box. Due to the similarities between algorithms, we based this code on the code in the supplementary material from Hendrikx et al. [2020].

D.2 Additional experiments

In this section, we give additional experiments to strengthen our conclusions.

Consistency across settings. Figure 6 shows the time results for Logistic Regression over the Phishing dataset (left), and Ridge Regression over the YearPredictionMSD dataset (right), both also from LibSVM. For the phishing dataset, $N = 552$, and so computation and communication times were scaled to the number of samples per node so that the timing ratios are the same as in Figure 4 ($\tau_{\text{comm}} = 55$, $\tau_{\text{comp}} = 1$). For the YearPredictionMSD dataset, we scaled the features to $[-1, 1]$ and increased the regularization to $c = 10^{-4}$. We also choose the local initial points $z_0^{(i)}$ as (very rough) approximations of $\nabla f_i^*(0)$ (the local solution), by performing 10 steps of gradient descent on the local objective starting from 0. For these datasets, we used $L_{\text{batch}} = \max_{ij} L_{ij}$ since smaller values did not lead to convergence.

We observe that, up to small differences (TVR is more noisy in the quadratic setting for instance), the behaviour and ordering of the curves is comparable to Figure 4, thus indicating consistency across tasks and datasets.