Is Consensus Acceleration Possible in Decentralized Optimization over Slowly Time-Varying Networks?

Dmitry Metelev [1] **Alexander Rogozin** [1 2] **Dmitry Kovalev** [3] **Alexander Gasnikov** [4 5 1]

Abstract

We consider decentralized optimization problems where one aims to minimize a sum of strongly convex smooth objective functions distributed between nodes in the network. The links in the network can change from time to time. For the setting when the amount of changes is arbitrary, lower complexity bounds and corresponding optimal algorithms are known, and the consensus acceleration is not possible. However, in practice the magnitude of network changes may be limited. We derive lower communication complexity bounds for several regimes of velocity of networks changes. Moreover, we show how to obtain accelerated communication rates for a certain class of time-varying graphs using a specific consensus algorithm.

1. Introduction

In this paper we consider a decentralized optimization problem

$$\min_{x\in\mathbb{R}^m} f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x), \tag{1}$$

where each function f_i is convex, has a Lipschitz gradient and is stored at a separate computational node. Nodes are connected by a communication network (that may change over time). Each node is an independent computational agent that can perform local computations based only on the information in its local memory. At each communication step, nodes can only exchange information with their neighbours.

Sum-type problems of type (1) have applications in practical scenarios where centralized coordination is not possible. Communication constraints may appear due to large amounts of data or due to privacy constraints (Konečný et al., 2016) and are determined by the structure of the network. Decentralized optimization is widely used in distributed machine learning (Rabbat & Nowak, 2004; Forero et al., 2010; Nedić, 2020; Gorbunov et al., 2022), distributed control (Ram et al., 2009; Gan et al., 2012) and distributed sensing (Bazerque & Giannakis, 2009).

1.1. Time-Varying Networks

We study the setting when the network is *time-varying*. That means that the links between the nodes may appear and disappear from time to time. In practice, the changes in the links may occur due to loss of wireless connection between the agents or other technical malfunctions. Note that while the set of edges may change, the set of vertices stays the same.

1.2. Related work

In this paper, we assume the objective $f(x)$ in (1) to be L-smooth and μ-strongly convex. Complexity bounds for decentralized optimization include two quantities: objective condition number $\kappa_g = L/\mu$ and network condition number χ. In case of the time-varying network, χ denotes the worst-case condition number over time steps.

Lower complexity bounds for optimization over static graphs were proposed in (Scaman et al., 2017). The lower communication complexity bound is $\Omega(\kappa_g^{1/2}\chi^{1/2}\log(1/\varepsilon))$. The corresponding optimal algorithms are MSDA (Scaman et al., 2017) (using dual oracle) and OPAPC (Kovalev et al., 2020) (using primal oracle).

For time-varying networks, the lower communication

The work was supported by a grant for research centers in the field of artificial intelligence, provided by the Analytical Center for the Government of the Russian Federation in accordance with the subsidy agreement (agreement identifier 000000D730321P5Q0002) and the agreement with the Ivannikov Institute for System Programming of the Russian Academy of Sciences dated November 2, 2021 No. 70-2021-00142. [1]Moscow Institute of Physics and Technology, Moscow, Russia [2]HSE University, Moscow, Russia [3]Université Catholique de Louvain, Ottignies-Louvain-la-Neuve, Belgium [4]ISP RAS Research Center for Trusted Artificial Intelligence [5]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Alexander Rogozin <aleksandr.rogozin@phystech.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

complexity bound is $\Omega(\kappa_g^{1/2}\chi\log(1/\varepsilon))$ (Kovalev et al., 2021a). The corresponding optimal algorithms with primal oracle are ADOM+ (Kovalev et al., 2021a) and Acc-GT (Li & Lin, 2021) with multi-step communication. An optimal dual algorithm is ADOM (Kovalev et al., 2021b). Prior to optimal algorithms, several non-accelerated schemes like DIGing (Nedic et al., 2017) and sub-optimal methods with additional logarithmic factor, i.e. APM-C (Li et al., 2020; Dvinskikh & Gasnikov, 2021; Rogozin et al., 2021) and Mudag (Ye et al., 2020) were proposed.

Optimal algorithms both for static and time-varying scenarios use a multi-step consensus scheme. In the time-static case, the communication matrix is replaced by a Chebyshev polynomial of it (Scaman et al., 2017). The degree of polynomial is $\lceil\chi^{1/2}\rceil$ and its condition number is $O(1)$. In the time-varying case, after each oracle call multiplication is performed by χ matrices in a row instead of only one matrix (Kovalev et al., 2021a).

1.3. Contributions

The case when arbitrarily many edges can change at each time step is well-studied. However, we think that such a setting is not realistic and in practice the magnitude of graph changes may be limited. We investigate several types of such restrictions and derive lower complexity bounds for each case. This constitutes the first part of our work (see Table 1). We show that it is sufficient to change a polynomial number of vertices (i.e. $O(n^\alpha)$ for some $\alpha > 0$) at each iteration in order to slow consensus speed down to factor χ. Moreover, if a logarithmic number of edges is changed (i.e. $O(\log n)$), the consensus is slowed down to $\chi/\log\chi$. Finally, our results suggest that a partial consensus acceleration (i.e. dependency on χ in power between $1/2$ and 1) is possible if the number of changes is bounded by a constant.

Table 1. Known lower communication complexity bounds for decentralized optimization and our results. Here $\alpha > 0$ is a scalar and $d \in \mathbb{N}$ is a constant. The complexity depends on the maximum number of changes in links allowed at each iteration.

Number of changes	Lower bound	Reference
no changes	$\Omega\left(\chi^{1/2}\kappa_g^{1/2}\log\frac{1}{\varepsilon}\right)$	(Scaman et al., 2017)
$O(n)$	$\Omega\left(\chi\kappa_g^{1/2}\log\frac{1}{\varepsilon}\right)$	(Kovalev et al., 2021a)
$O(n^\alpha)$	$\Omega\left(\chi\kappa_g^{1/2}\log\frac{1}{\varepsilon}\right)$	This paper, Th. 3.1
$O(\log n)$	$\Omega\left(\frac{\chi}{\log\chi}\kappa_g^{1/2}\log\frac{1}{\varepsilon}\right)$	This paper, Th. 3.3
$12(d-1)$	$\Omega\left(\chi^{d/(d+1)}\kappa_g^{1/2}\log\frac{1}{\varepsilon}\right)$	This paper, Th. 3.5

In the setting where a constant number of edges changes at each iteration, our results allow to establish the known lower bounds for static graphs and time-varying graphs with arbitrary changes. The corresponding results are presented in the last line of Table 1. Putting $d = 1$ leads to the static case and the lower bound coincides with the one in (Scaman et al., 2017). In the opposite case, taking $d \to \infty$ leads to the scenario with arbitrary changes, and the corresponding lower bound approaches the results in (Kovalev et al., 2021a). In other words, our results suggest an interpolation between two edge cases: static graphs and time-varying graphs with arbitrary changes.

In the second part of our paper, we address a multi-step consensus technique for time-varying graphs. More precisely, we apply Nesterov acceleration technique to time-varying consensus. The acceleration is attained under an additional assumption: we assume that all graphs have a common connected subgraph that we call a *skeleton*. On the one hand, this assumption is more strict then requiring the network to stay connected all the time. On the other hand, we think that such an assumption may be realistic in practical scenarios.

The consensus procedure for graphs with connected skeleton is slightly modified: the two nodes stop communicating to each other if the connection between them has been lost at least once. In other words, the active links in the communication graph are not restored after they have failed at least once, and therefore the network is "monotonically decreasing".

Summing up, this paper makes a step in the direction of optimization over special classes of time-varying networks. Our lower bounds show that acceleration communication protocol is hard to be designed even over slowly time-varying graphs. On the other hand, we show a specific class of networks over which accelerated consensus is reachable.

The paper is organized as follows. In Section 2 we introduce notation, definitions and assumptions. In Section 3, we present our main results on lower bounds. After that, in Section 4 we describe the accelerated gossip protocol over time-varying networks with connected skeleton.

2. Definitions and Assumptions

We denote Kronecker product by $\otimes$. The nullspace of matrix $\mathbf{A}$ is denoted $\ker\mathbf{A}$ and the range of $\mathbf{A}$ is denoted $\operatorname{range}\mathbf{A}$. Moreover, if $\mathbf{A}$ is symmetric and positive semi-definite, we denote its largest eigenvalue $\lambda_{\max}(\mathbf{A})$, its minimal nonzero eigenvalue $\lambda_{\min}^{+}(\mathbf{A})$ and its condition number $\chi(\mathbf{A}) = \lambda_{\max}(\mathbf{A})/\lambda_{\min}^{+}(\mathbf{A})$. For vectors $x_1,\ldots,x_n \in \mathbb{R}^d$, we introduce a column stacked vector $\mathbf{x} = \operatorname{col}[x_1,\ldots,x_n] = (x_1^\top\ldots x_n^\top)^\top \in \mathbb{R}^{nd}$. We also denote $\mathbb{N} = \{1,2,\ldots\}$ to be the set of positive integers.

2.1. Objective Functions

Let H be an arbitrary Hilbert space, let $\|\cdot\|$ be the norm on H and let $\|\cdot\|_*$ denote the conjugate norm.

Definition 2.1. Function $h(x) : H \to \mathbb{R}$ is called L-smooth if for any $x, y \in H$ it holds

$$\|\nabla h(y) - \nabla h(x)\|_* \leq L \|y - x\| .$$

Definition 2.2. Function $h(x) : H \to \mathbb{R}$ is called μ-strongly convex if for any $x, y \in H$ it holds

$$h(y) \geq h(x) + \langle \nabla h(x), y - x \rangle + \frac{\mu}{2} \|y - x\|^2 .$$

Throughout the paper we only work with $H = l_2$. Although the lower bounds are met for optimization in a finite dimension, they are derived for l_2, as it is typically done in optimization of strongly convex smooth functions (Nesterov, 2004).

2.2. Decentralized Communication

We assume that distributed communication is performed via a series of communication rounds. In each of the rounds, the nodes interact through a network represented by an undirected communication graph $\mathcal{G}_k = (\mathcal{V}, \mathcal{E}_k)$, where $k \in \mathbb{N}$ is the current iteration number. The nodes can only communicate to their immediate neighbors in the corresponding network.

Note that the set of nodes $\mathcal{V}$ does not change over time. Throughout the paper we only consider the case when all graphs $\mathcal{G}_k$ are connected.

In the literature, analysis of optimization algorithms in the decentralized setting as well as the lower bounds are usually based on the condition number of gossip matrices.

Assumption 2.3. Matrix $W_k \in \mathbb{R}^{n \times n}$ is called a gossip matrix of undirected graph $\mathcal{G}_k = (\mathcal{V}, \mathcal{E}_k)$ if the following properties are satisfied:

1. W_k is symmetric positive semi-definite,
2. $[W_k]_{i,j} = 0$ if $i \neq j$ and $(i, j) \notin \mathcal{E}_k$,
3. $\ker W_k = \{(x_1, \ldots, x_n) \in \mathbb{R}^n : x_1 = \cdots = x_n\}$.

Given a gossip matrix W_k, we define $\mathbf{W}_k = W_k \otimes \mathbf{I}_d$. Then $\mathbf{W}_k$ is also symmetric and positive semi-definite, multiplication by $\mathbf{W}_k$ represents one communication round and $\ker \mathbf{W}_k = \{\mathbf{x} = \text{col}[x_1, \ldots, x_n] \in \mathbb{R}^{nd} : x_1 + \ldots + x_n = 0\}$.

For a given gossip matrix W, introduce its condition number $\chi(W) = \lambda_{\max}(W)/\lambda^+_{\min}(W)$.

A common example of a communication matrix is the graph Laplacian $\mathbf{L}(\mathcal{G}_k) = D(\mathcal{G}_k) - A(\mathcal{G}_k)$, where $A(\mathcal{G}_k)$ denotes the adjacency matrix of $\mathcal{G}_k$ and $D(\mathcal{G}_k) = \text{diag}(\sum_i A_{ij})$ is a diagonal matrix with degrees of the nodes at diagonal. Laplacian matrix $\mathbf{L}(\mathcal{G}_k)$ satisfies Assumption 2.3.

Further in the paper we will use only Laplacian matrices, therefore by slight abuse of notation we denote $\chi(G) = \chi(\mathbf{L}(G))$.

2.3. Decentralized Problem and Slowly-Changing Setup

The main results of our work are lower bounds for decentralized optimization problems. We formalize the definition of decentralized problem and its characteristics.

Definition 2.4. Let us define $\mathcal{DP}$ (decentralized time-varying problem) as a pair $(\{\mathcal{G}_k\}_{k=1}^{\infty}, \{f_i\}_{i=1}^n)$. Firstly, $\mathcal{DP}$ includes a sequence of undirected connected graphs $\{\mathcal{G}_k\}_{k=1}^{\infty}$ with a common set of vertices $\mathcal{V} = \{1, 2, \ldots, n\}$ and edge sets $\{\mathcal{E}_k\}_{k=1}^{\infty}$. Let $n(\mathcal{DP}) = n$, $\chi(\mathcal{DP}) = \sup_{k \in \mathbb{N}} \chi(\mathcal{G}_k)$. Secondly, $\mathcal{DP}$ includes a set of objective functions $\{f_i\}_{i=1}^n$. We refer to $f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x)$ as a global function.

Definition 2.5. Decentralized time-varying problem $\mathcal{DP}$ is called L-smooth if global function f is L-smooth.

Definition 2.6. Decentralized time-varying problem $\mathcal{DP}$ is called convex (μ-strongly convex) if global function f is convex (μ-strongly convex).

Definition 2.7. Let $\Delta(\mathcal{DP})$ denote the maximum amount of edges that change between consequent communication rounds. Particularly, $\Delta(\mathcal{DP}) = \max_{k \in \mathbb{N}} \left\{ \sum_{i<j} \mathbb{I}_{ij}(\mathcal{E}_k, \mathcal{E}_{k+1}) \right\}$, where

$$\mathbb{I}_{ij}(\mathcal{E}, \mathcal{E}') = \begin{cases} 1, & \text{if } (i,j) \in (\mathcal{E} \setminus \mathcal{E}') \cup (\mathcal{E}' \setminus \mathcal{E}), \\ 0, & \text{otherwise.} \end{cases}$$

The introduced quantity $\Delta(\mathcal{DP})$ expresses the maximum change in edges between two consequent time steps. Later in the paper we show that the value of $\Delta(\mathcal{DP})$ regulates the magnitude of network changes and determines the dependence of the lower bounds on condition number χ.

Example 1. Consider a static graph and denote its corresponding problem $\mathcal{DP}_{static}$. All of the graphs in $\mathcal{DP}_{static}$ are the same, therefore, we have $\Delta(\mathcal{DP}_{static}) = 0$.

Example 2. Consider the example used in lower bounds in (Kovalev et al., 2021a). The authors proposed a star graph which center changes at each iteration (denote the corresponding problem $\mathcal{DP}_{star}$). In such a setting, at every iteration every edge in the graph changes. We have $\Delta(\mathcal{DP}_{star}) = 2(n-1)$.

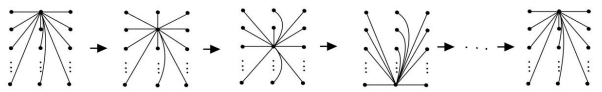

Figure 1. Example of a time-varying network with all edges changing at each iteration (Kovalev et al., 2021a)

3. Lower Bounds

This section presents three results corresponding to different constraints on the rate of change of the communication graph: a polynomial constraint on the change of edges per iteration, a logarithmic constraint, and a constant constraint. We show that each of these regimes leads to a different complexity dependency on the condition number of the gossip matrix.

3.1. First-order Decentralized Algorithms

Let us first formalize the procedure for which we derive the lower bounds. Following the definitions of (Kovalev et al., 2021a) and (Scaman et al., 2017), we consider time steps $k \in \mathbb{N}$ and introduce local memory $\mathcal{H}_i(k)$ for each of the agents at time step k. At each time step, the agents can either communicate or perform local computations. For each time step k, denote the last preceding communication time as $q(k)$.

1. If nodes perform a local computation at step k, local information is updated as

$$\mathcal{H}_i(k+1) \subseteq \operatorname{span}\left(\{x, \nabla f_i(x), \nabla f_i^*(x) : \ x \in \mathcal{H}_i(k)\}\right)$$

for all $i = 1, \ldots, n$.

2. If the nodes perform a communication round at time step k, local information is updated as

$$\mathcal{H}_i(k+1) \subseteq \operatorname{span}\left(\bigcup_{j \in \mathcal{N}_i^{q(k)} \cup \{i\}} \mathcal{H}_j(k)\right)$$

for all $i = 1, \ldots, n$. Here $\mathcal{N}_i^{q(k)}$ is a set of neighbors of agent i at time step $q(k)$, i.e. at the time of last communication.

3.2. Main Results

The following theorem discusses the polynomial constraint on the change of edges per iteration; it turns out that such a constraint leads to the same lower bound as in the unconstrained mode studied in (Kovalev et al., 2021a).

Theorem 3.1. *For any $L > 24\mu > 0$, $\alpha > 0$, $c > 0, M > 0$ there exists a constant $K(\alpha, c) > 0$ and L-smooth μ-strongly convex decentralized problem $\mathcal{DP}$ with $n(\mathcal{DP}) = n > M$, $\chi(\mathcal{DP}) = \chi > M$, $\Delta(\mathcal{DP}) \leq cn^{\alpha}$, such that for any first-order decentralized algorithm for all $p \in \mathbb{N}$ we have*

$$\|x_p - x_*\|^2 \geq \left(1 - 2\sqrt{6}\sqrt{\frac{\mu}{L}}\right)^{\frac{K(\alpha,c)p}{\chi}+2} \|x_0 - x_*\|^2.$$

Corollary 3.2. *For any $L > 24\mu > 0$, $\alpha > 0, c > 0$ there exists L-smooth μ-strongly convex decentralized problem $\mathcal{DP}$ with sufficiently large $\chi(\mathcal{DP}) = \chi$, $n(\mathcal{DP}) = n$, such that $\Delta(\mathcal{DP}) \leq cn^{\alpha}$, and for any first-order decentralized algorithm the number of communication rounds to find an ε-accurate solution of the problem 1 is lower bounded by*

$$\Omega\left(\chi\sqrt{L/\mu}\log\frac{1}{\varepsilon}\right).$$

The following theorem corresponds to the case where the number of edges that can change per iteration is at most logarithmic in the number of nodes.

Theorem 3.3. *For any $L > 10\mu > 0$, $M > 0$ there exists L-smooth μ-strongly convex decentralized problem $\mathcal{DP}$ with $n(\mathcal{DP}) = n > M$, $\chi(\mathcal{DP}) = \chi > M$, such that $\Delta(\mathcal{DP}) \leq 12\log_2(n)$ and for any first-order decentralized algorithm for all $p \in \mathbb{N}$ we have*

$$\|x_p - x_*\|^2 \geq \left(1 - \sqrt{10}\sqrt{\frac{\mu}{L}}\right)^{\frac{12\log_2(\chi/2)p}{\chi}+2} \|x_0 - x_*\|^2.$$

Corollary 3.4. *For any $L > 10\mu > 0$ there exists L smooth and μ-strongly convex decentralized problem $\mathcal{DP}$ with sufficiently large $\chi(\mathcal{DP}) = \chi$, $n(\mathcal{DP}) = n$, such that $\Delta(\mathcal{DP}) \leq 12\log_2 n$, and for any first-order decentralized algorithm the number of communication rounds to find an ε-accurate solution of the problem 1 is lower bounded by*

$$\Omega\left(\frac{\chi}{\log\chi}\sqrt{L/\mu}\log\frac{1}{\varepsilon}\right).$$

As we can see, although the logarithmic constraints are tighter than the polynomial ones, the problem cannot be solved much faster. The benefit we get from logarithmic constraints is only the logarithmic factor $\log\chi$, which is typically small compared to the main term χ.

The following theorem describes the case when the constraints on changes for sequential iteration are constant.

Theorem 3.5. *For any $L > 24\mu > 0$, $M > 0$, $d \in \mathbb{N}$ there exists a constant $K(d) > 0$ and L-smooth μ-strongly convex decentralized problem $\mathcal{DP}$ with $n(\mathcal{DP}) = n > M$, $\chi(\mathcal{DP}) = \chi > M$, $\Delta(\mathcal{DP}) \leq 12(d-1)$, such that for any first-order decentralized algorithm for all $p \in \mathbb{N}$ we have*

$$\|x_p - x_*\|^2 \geq \left(1 - 2\sqrt{6}\sqrt{\frac{\mu}{L}}\right)^{K(d)p\chi^{-\frac{d}{d+1}}+2} \|x_0 - x_*\|^2.$$

Corollary 3.6. *For any $L > 24\mu > 0$, $d \in \mathbb{N}$ there exists L-smooth μ-strongly convex decentralized problem $\mathcal{DP}$ with sufficiently large $\chi(\mathcal{DP}) = \chi$ and $n(\mathcal{DP}) = n$, such that $\Delta(\mathcal{DP}) \leq 12(d-1)$, and for any first-order decentralized algorithm the number of communication rounds to find an ε-accurate solution of the problem 1 is lower bounded by*

$$\Omega\left(\chi^{\frac{d}{d+1}}\sqrt{L/\mu}\log\frac{1}{\varepsilon}\right).$$

This result shows that even with constant constraints, the lower estimates approach the estimates without constraints. This indicates that the criterion based on the Laplacian matrix condition number is very sensitive to the degree of graph variability.

3.3. Discussion

As a result, we obtained lower bounds on the number of communications for decentralized optimization problems with smooth and strongly convex functions with different constraints on the rate of network change. In particular, three modes differing in the rate of change of the communication graph were considered.

The first mode assumes a polynomial change in the number of edges in the graph; as it turned out, the lower bounds in this case coincide with the lower bounds when no additional conditions are imposed on the graph change.

The second mode considers a logarithmic change in the number of edges in the graph. In this case the lower bounds are close to the lower bounds in the case of time-varying networks without restrictions.

The third mode considers a constant change in the number of edges; here, for each value of the constant d, an individual estimate is obtained. When $d = 1$, we restore the lower bound for static graphs (Scaman et al., 2017) and when $d \to \infty$, we approach a lower bound for time-varying networks with no restrictions on the speed of changes (Kovalev et al., 2021a). In other words, the lower bounds for the last mode can be seen as an interpolation between static and time-varying networks without restrictions.

Visualizing communication complexity. One can also consider a graphical interpretation of the first and second mode. In the proof of Theorem 3.1, three quantities are presented: number of nodes n, number of changed edges per iteration Δ and information flow T. For large graphs (when n is large enough) we have $\chi \sim n$ (multiplicative constants omitted). Quantity T/n is the coefficient under Ω-notation in the lower bound $\Omega(\chi \log 1/\varepsilon)$. Lower bounds are obtained for networks that have a structure of a Bethe tree $B_{d,k}$. A Bethe tree $B_{d,k}$ has a single root, k levels, and each non-leaf node has d children (see Figure 3). Theorem 3.1 covers the case $k = const, d \to \infty$ and Theorem 3.3 describes the case $d = const, k \to \infty$.

However, the relation between quantities n, Δ, T is wider then only asymptotic. In fact, we have $n = \frac{d^k-1}{d-1}$, $\Delta = (k-1)(d+1)$ and $T = \frac{n}{k-1}$. We can plot the mutual dependence between T, Δ and n. It is convenient to visualize quantity T/n in coordinates $(\log n, \Delta)$. We plot the corresponding heatmap in Figure 2.

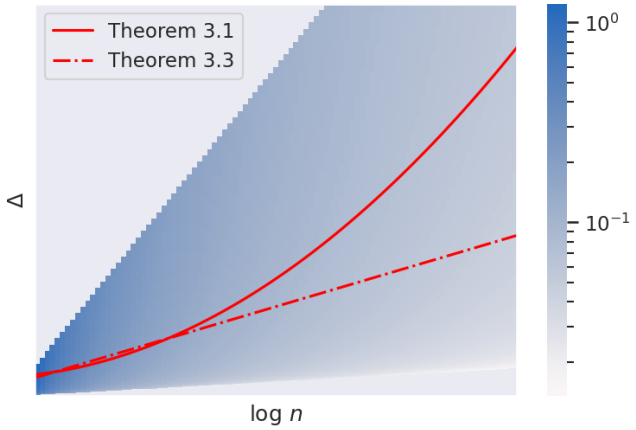

Figure 2. Heatmap of $c = T/n$ in coordinates $(\log n, \Delta)$. A more saturated color corresponds to $c = T/n$ close to 1, i.e. to lower bound $\Omega(\chi \log 1/\varepsilon)$. We see that the results of Theorem 3.1 and Theorem 3.3 are obtained when moving along appropriate trajectories in this heatmap.

The results of Theorems 3.1 and 3.3 are asymptotic. The asymptotics is taken when moving along an appropriate direction in Figure 2.

For polynomial changes (Theorem 3.1), we have $k = const$, $d \to \infty$. In this case $\log n \sim (k-1)\log d$ and $\Delta \sim (d+1)(k-1)$ (see Appendix B). Since k is fixed, Δ has an exponential dependence on $\log n$.

For logarithmic changes (Theorem 3.3), we have $d = 2 = const$, $k \to \infty$. Therefore, $\log n \sim k \log 2$ and $\Delta \sim k-1$. We have that Δ has a linear dependence on $\log n$.

Meaning of lower bounds. It is worth noting that the interpretation of our lower complexity bounds is different from previous results in (Kovalev et al., 2021a) and (Scaman et al., 2017). The mentioned papers build an example of a optimization problem for any $\chi > 0$, while our results suggest that a counterexample exists only if χ is sufficiently large. Let us illustrate how to interpret the lower bounds in the regime of polynomial change. Theorem 3.1 means that

*A decentralized optimization method that solves **any** decentralized problem with polynomial bound on changes in $O(\chi^p\sqrt{L/\mu}\log(1/\varepsilon))$, $p < 1$ communication rounds **does not exist**.*

In other words, our results are asymptotic, but they are sufficient to restrict the area for future research.

4. Accelerated Gossip for Time-Varying Graphs with Connected Skeleton

4.1. Time-Varying Graphs with Connected Skeleton

It is known that the number of communications cannot be enhanced on the class of time-varying graphs that are allowed to change arbitrarily but stay connected at each iteration. The corresponding lower complexity bounds have been proposed in (Kovalev et al., 2021a). However, the lower bounds in (Kovalev et al., 2021b) are built using a graph where $O(n)$ edges change at each time step. Namely, a "bad" graph is a star graph where the center of a star changes at each iteration (see Figure 1).

In practice, a situation where $O(n)$ edges change at every iteration may not always occur. The amplitude of network malfunctions may be not so large. We let all of the graphs in the sequence have a common subgraph (a *skeleton*) that remains connected through time.

Assumption 4.1. Graph sequence $\{\mathcal{G}_k = (\mathcal{V}, \mathcal{E}_k)\}_{k=0}^{\infty}$ has a connected skeleton: there exists a connected graph $\hat{\mathcal{G}} = (\mathcal{V}, \hat{\mathcal{E}})$ such that for all $k = 0, 1, \ldots$ we have $\hat{\mathcal{E}} \subseteq \mathcal{E}_k$.

Assumption 4.2. For each $k = 0, 1, \ldots$ we have $\lambda_{\max}(\mathbf{L}(\mathcal{G}_k)) \leq \lambda_{\max}$. Moreover, we have $\lambda_{\min}^{+} \leq \lambda_{\min}^{+}(\hat{\mathcal{G}})$.

Assumption 4.1 is more strict then the assumption on the graph staying connected at each iteration. However, under Assumption 4.1 we propose an accelerated consensus procedure over time-varying graphs.

4.2. Accelerated Gossip with Non-Recoverable Links

A common approach to accelerated consensus over static graphs is Chebyshev acceleration proposed in (Scaman et al., 2017). A gossip matrix $\mathbf{W}$ with condition number $\chi(\mathbf{W})$ can be replaced by a matrix polynomial $P_K(\mathbf{W})$ of degree $K = \lceil (\chi(\mathbf{W}))^{1/2} \rceil$ with condition number $\chi(P_K(\mathbf{W})) = O(1)$. The construction of $P_K(\mathbf{W})$ is based on Chebyshev polynomials of first type. Then, the condition number of communication matrix is reduced from $\chi(\mathbf{W})$ to $O(1)$ at the cost of performing $\lceil (\chi(\mathbf{W}))^{1/2} \rceil$ communication rounds instead of one.

However, Chebyshev method is only known to be applied to consensus over static networks. In our work, we propose an accelerated gossip scheme over time varying graphs based on Nesterov acceleration. The acceleration is possible because of assumption on connected skeleton (Assumption 4.1) and due to a specific consensus strategy.

We use the following approach to tackle with time-varying graphs that have a connected skeleton. Let agent i in the network stop exchanging information to agent j once the connection between i and j has been lost at any communication round. In other words, if a link fails once, the communication through it is not recovered afterwards. This procedure is referred to as *accelerated gossip with non-recoverable links*.

Algorithm 1 Accelerated Gossip with Non-Recoverable Links

Require: Initial guess $\mathbf{x} \in \mathbb{R}^{nd}$, stepsizes $\eta, \beta > 0$. Set $\mathbf{y}^0 = \mathbf{x}^0 = \mathbf{x}$.
1: Every node $i = 1, \ldots, n$ initializes set of neighbors $\mathcal{N}_i = \mathcal{N}_i^0$.
2: **for** $t = 0, 1, \ldots, T-1$ **do**
3: Every node i does
4: Update the set of nodes to which the node communicates: $\mathcal{N}_i = \mathcal{N}_i \cap \mathcal{N}_i^k$
5: $y_i^{k+1} = x_i^k - \eta(|\mathcal{N}_i| x_i^k - \sum_{j \in \mathcal{N}_i} x_j^k)$
6: $x_i^{k+1} = (1+\beta)\, y_i^{k+1} - \beta y_i^k$
7: **end for**
8: **return** $C_T(\mathbf{x}) = \mathbf{x} - \mathbf{x}^T$

Note that the output of Algorithm 1 is $C_T(\mathbf{x})$. We claim that $C_T(\mathbf{x})$ is a linear operator that is a time-varying analogue of $P_K(\mathbf{W})\mathbf{x}$.

Theorem 4.3. *Let Assumptions 4.1 and 4.2 hold. Denote $\chi = \lambda_{\max}/\lambda_{\min}^{+}$ and set the parameters of Algorithm 1 to $\eta = 1/\lambda_{\max}$, $\beta = (\sqrt{\chi}-1)/(\sqrt{\chi}+1)$. Then operator $C_T(\mathbf{x})$ defined in Algorithm 1 has the following properties.*

1. $C_T(\mathbf{x})$ is linear.

2. $\operatorname{range} C_T(\mathbf{x}) = \mathcal{L}^{\top} = \{\mathbf{x} \in \mathbb{R}^{md} : x_1 + \ldots + x_m = 0\}$.

3. For $T = \sqrt{\chi}\log(4\chi)$ we have that for any $\mathbf{x} \in \mathcal{L}^{\top}$ it holds $(1 - 1/\sqrt{2})\,\|\mathbf{x}\|_2 \leq \|C_T(\mathbf{x})\|_2 \leq (1 + 1/\sqrt{2})\,\|\mathbf{x}\|_2$.

The meaning of Theorem 4.3 is the following: for "monotone" graphs we can replace $\mathbf{W}^k$ with condition number χ by $C_T(\cdot)$ with condition number $O(1)$. The payment for reduction of condition number is $\sqrt{\chi}\log(4\chi)$ communication rounds.

4.3. Accelerated Gossip as Accelerated Method over Time-Varying Function

Algorithm 1 can be viewed as a gossip algorithm over a "monotonic" network where the edges only vanish and do not appear. Namely, introduce a sequence of graphs $\{\hat{\mathcal{G}}_k = (\mathcal{V}, \cap_{j=0}^{k} \mathcal{E}_j)\}_{k=0}^{\infty}$, corresponding Laplacians $\{\hat{W}^k = \mathbf{L}(\hat{\mathcal{G}}_k)\}_{k=0}^{\infty}$ and denote $\hat{\mathbf{W}}^k = \hat{W}^k \otimes \mathbf{I}_d$. Then Al-

gorithm 1 writes as

$$\begin{cases} \mathbf{y}^{t+1} = \mathbf{x}^k - \eta \hat{\mathbf{W}}^k \mathbf{x}^k, \\ \mathbf{x}^{k+1} = (1+\beta)\mathbf{y}^{k+1} - \beta \mathbf{y}^k. \end{cases} \tag{2}$$

As can be seen from (2), on each time step a multiplication by $\hat{\mathbf{W}}^k$ is performed, which corresponds to one communication over $\hat{\mathcal{G}}_k$. The edges in sequence $\{\hat{\mathcal{G}}_k\}_{k=0}^{\infty}$ only vanish and do not appear.

Algorithm 1 can also be interpreted as minimization of a time-varying functional with an accelerated gradient method. Consider problem

$$\min_{\mathbf{x} \in \mathbb{R}^{nm}} \; h_k(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \hat{\mathbf{W}}_k \mathbf{x}. \tag{3}$$

Algorithm 1 is accelerated Nesterov method with step-size η and momentum term β applied a time-varying problem (3).

Remark 4.4. Returning to the case of static networks, it is worth mentioning that Chebyshev polynomials and accelerated gradient methods for quadratic minimization have a strong connection, see i.e. Chapter 2 of (d'Aspremont et al., 2021). In fact, Polyak's momentum can be derived through application of Chebyshev polynomials to a quadratic minimization problem.

The analysis of accelerated method over a uniformly non-increasing time-varying function is based on the Lyapunov function technique.

Lemma 4.5. *Let Assumptions 4.1 and 4.2 hold. Denote* $\tau = 1/(\sqrt{\chi}+1)$, $\mathbf{z}_k = 1/\tau\ \mathbf{x}_k - (1-\tau)/\tau\ \mathbf{y}_k$, $\gamma = 1/(\sqrt{\chi}-1)$ *and introduce potential*

$$\Psi_k = (1+\gamma)^k \left(h_k(\mathbf{y}^k) + \frac{\lambda^+_{\min}}{2} \left\| \mathbf{z}^k - \mathbf{x}^* \right\|_2^2 \right),$$

where $\mathbf{x}^*$ *is a solution of* (3)*. Then* $\Psi_{k+1} - \Psi_k \leq 0$.

The proof of Lemma 4.5 is based on the standard analysis proposed in (Bansal & Gupta, 2019) and on the observation that the objective function in (3) is uniformly non-increasing, i.e. for any $\mathbf{x} \in \mathbb{R}^{nd}$ and for any $k = 0, 1, \ldots$ we have $h_{k+1}(\mathbf{x}) \leq h_k(\mathbf{x})$. Indeed, we have $\hat{W}^k = \sum_{(i,j) \in \hat{\mathcal{E}}_k} (e_i - e_j)(e_i - e_j)^\top$, where e_i denotes the i-th coordinate vector of $\mathbb{R}^m$. Therefore, it holds

$$W^k - W^{k+1} = \sum_{(i,j) \in \hat{\mathcal{E}}_k \setminus \hat{\mathcal{E}}_{k+1}} (e_i - e_j)(e_i - e_j)^\top \succeq 0.$$

In other words, for any $\mathbf{x} \in \mathbb{R}^{nm}$ it holds

$$h_{k+1}(\mathbf{x}) - h_k(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \hat{\mathbf{W}}^{k+1} \mathbf{x} - \frac{1}{2} \mathbf{x}^\top \hat{\mathbf{W}}^k \mathbf{x} \leq 0.$$

The analysis of accelerated gradient method over time-varying uniformly non-increasing functions is presented in Appendix E.

5. Conclusion

In this paper, we study new classes of time-varying networks that may be more practical than the scenarios previously studied in the literature. We propose to look into a new direction of research – slowly time-varying graphs. In this work, we formalize several regimes covering the velocity of graph changes and provide the corresponding lower bounds for each case. Our results outline the limits of what communication rates can be achieved over slowly time-varying graphs. Moreover, we propose a slightly modified consensus technique that leads to acceleration over time-varying networks with connected skeleton. Our technique may be seen as an analogue of Chebyshev acceleration that is used for time-static graphs.

Moreover, by the time of the publication of this paper, we had obtained the following result. There exists a sequence of graphs such that at each iteration only two edges are changed and the resulting lower complexity bound is $\Omega(\chi \sqrt{\kappa_g} \log(1/\varepsilon))$. We will describe the corresponding result in a separate work.

References

Bansal, N. and Gupta, A. Potential-function proofs for gradient methods. *Theory of Computing*, 15(1):1–32, 2019.

Bazerque, J. A. and Giannakis, G. B. Distributed spectrum sensing for cognitive radio networks by exploiting sparsity. *IEEE Transactions on Signal Processing*, 58 (3):1847–1862, 2009.

Das, K. The laplacian spectrum of a graph. *Computers & Mathematics with Applications*, 48 (5):715–724, 2004. ISSN 0898-1221. doi: https://doi.org/10.1016/j.camwa.2004.05.005. URL https://www.sciencedirect.com/science/article/pii/S0898122104003074.

d'Aspremont, A., Scieur, D., Taylor, A., et al. Acceleration methods. *Foundations and Trends® in Optimization*, 5 (1-2):1–245, 2021.

Dvinskikh, D. and Gasnikov, A. Decentralized and parallel primal and dual accelerated methods for stochastic convex programming problems. *Journal of Inverse and Ill-posed Problems*, 29(3):385–405, 2021.

Forero, P. A., Cano, A., and Giannakis, G. B. Consensus-based distributed support vector machines. *Journal of Machine Learning Research*, 11(5), 2010.

Gan, L., Topcu, U., and Low, S. H. Optimal decentralized protocol for electric vehicle charging. *IEEE Transactions on Power Systems*, 28(2):940–951, 2012.

Gorbunov, E., Rogozin, A., Beznosikov, A., Dvinskikh, D., and Gasnikov, A. Recent theoretical advances in decentralized distributed convex optimization. In *High-Dimensional Optimization and Probability*, pp. 253–325. Springer, 2022.

Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Kovalev, D., Salim, A., and Richtárik, P. Optimal and practical algorithms for smooth and strongly convex decentralized optimization. *Advances in Neural Information Processing Systems*, 33, 2020.

Kovalev, D., Gasanov, E., Gasnikov, A., and Richtarik, P. Lower bounds and optimal algorithms for smooth and strongly convex decentralized optimization over time-varying networks. *Advances in Neural Information Processing Systems*, 34, 2021a.

Kovalev, D., Shulgin, E., Richtárik, P., Rogozin, A. V., and Gasnikov, A. Adom: Accelerated decentralized optimization method for time-varying networks. In *International Conference on Machine Learning*, pp. 5784–5793. PMLR, 2021b.

Li, H. and Lin, Z. Accelerated gradient tracking over time-varying graphs for decentralized optimization. *arXiv preprint arXiv:2104.02596*, 2021.

Li, H., Fang, C., Yin, W., and Lin, Z. Decentralized accelerated gradient methods with increasing penalty parameters. *IEEE Transactions on Signal Processing*, 68: 4855–4870, 2020.

Molitierno, J., Neumann, M., and Shader, B. Tight bounds on the algebraic connectivity of a balanced binary tree. *The Electronic Journal of Linear Algebra*, 6:62–71, 1999.

Nedić, A. Distributed gradient methods for convex machine learning problems in networks: Distributed optimization. *IEEE Signal Processing Magazine*, 37(3):92–101, 2020.

Nedic, A., Olshevsky, A., and Shi, W. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4): 2597–2633, 2017.

Nesterov, Y. *Introductory Lectures on Convex Optimization: a basic course*. Kluwer Academic Publishers, Massachusetts, 2004.

Rabbat, M. and Nowak, R. Distributed optimization in sensor networks. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pp. 20–27, 2004.

Ram, S. S., Veeravalli, V. V., and Nedic, A. Distributed non-autonomous power control through distributed convex optimization. In *IEEE INFOCOM 2009*, pp. 3001–3005. IEEE, 2009.

Rogozin, A., Uribe, C. A., Gasnikov, A. V., Malkovsky, N., and Nedić, A. Optimal distributed convex optimization on slowly time-varying graphs. *IEEE Transactions on Control of Network Systems*, 7(2):829–841, 2019.

Rogozin, A., Lukoshkin, V., Gasnikov, A., Kovalev, D., and Shulgin, E. Towards accelerated rates for distributed optimization over time-varying networks. In *International Conference on Optimization and Applications*, pp. 258–272. Springer, 2021.

Rojo, O. and Medina, L. Tight bounds on the algebraic connectivity of bethe trees. *Linear algebra and its applications*, 418(2-3):840–853, 2006.

Scaman, K., Bach, F., Bubeck, S., Lee, Y. T., and Massoulié, L. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3027–3036. JMLR. org, 2017.

Stevanović, D. Bounding the largest eigenvalue of trees in terms of the largest vertex degree. *Linear algebra and its applications*, 360:35–42, 2003.

Ye, H., Luo, L., Zhou, Z., and Zhang, T. Multi-consensus decentralized accelerated gradient descent. *arXiv preprint arXiv:2005.00797*, 2020.

Supplementary material

A. Outline of Proofs of Theorems 3.1, 3.3, 3.5

Proofs of Theorems 3.1, 3.3 and 3.5 are based on similar ideas. Our aim is to build a decentralized problem that is bad enough, i.e. the complexity of any decentralized first-order algorithm is estimated from below by the corresponding bound.

Each counterexample is built the following way. We introduce two disjoint groups of vertices $\mathcal{V}_1$ and $\mathcal{V}_2$ and change the graph so that communication between the two groups is interfered. At each time step, information traverses one edge, and the time of information spread from $\mathcal{V}_1$ to $\mathcal{V}_2$ is made as much as possible. Also, all the nodes hold functions of specific type.

We use the following terms in our analysis.

- *"Bad" vertices*: the vertices that hold the information. At initialization, the set of bad vertices equals $\mathcal{V}_1$. Every vertex adjacent to a *bad* vertex at time step t becomes *bad* itself on time step $t+1$.
- *"Good" vertices*: vertices that are not *bad*. At initialization, the set of good vertices equals $\mathcal{V}\backslash\mathcal{V}_1$.
- *Information flow*: first moment when one of vertices in $\mathcal{V}_2$ becomes *bad*. In other words, time of information spread from $\mathcal{V}_1$ to $\mathcal{V}_2$.

We now present an overview of the proof sketch that will be used in Theorem 3.1, Theorem 3.3, and Theorem 3.5:

1. We begin with the introduction of a counterexample graph and an examination of its Laplacian spectral properties. By construction, all the graphs in the sequence are isomorphic, so we only need to analyze the spectral properties of one of them.
2. Subsequently, we construct local functions on the nodes of the graph and establish a relationship between the local and global condition numbers.
3. We then develop an edge modification algorithm. The edges are changed in such a way that all the graphs in the sequence are equal up to a renumerating of vertices. Moreover, we compute the *information flow*.
4. Finally, we summarize the results of items 1-3 and derive a lower bound on the number of communications.

B. Proof of the Theorem 3.1

B.1. Graph Condition Number

Denote as $B_{d,k}$ a Bethe tree of degree d and depth k, where the root has a degree of d, vertices at levels from 2 to $k-1$ have a degree of $d+1$, and vertices at the k'th level have a degree of 1 (see Figure 3). Let $n = n(B_{d,k})$ be a number of vertices of $B_{d,k}$. By simple calculations we get $n = \frac{d^k-1}{d-1}$. Suppose $k \geq 2$, $d \geq 3$, thus using Theorem 2 and Theorem 3 from (Rojo & Medina, 2006) and considering the asymptotic behavior, we obtain that $\exists d_0 : \forall d \geq d_0$

$$\frac{(d-1)^2}{d^k-1} \leq \lambda_{n-1}(L(B_{d,k})) \leq 2\frac{(d-1)^2}{d^k-1}.$$

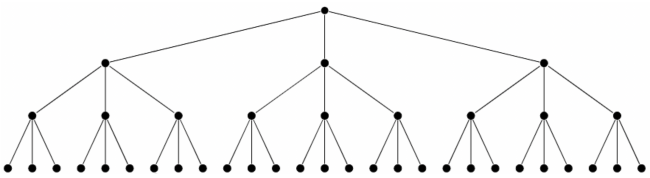

Figure 3. Example of $B_{3,4}$

Using results from (Stevanović, 2003) we get $d+2 \leq \lambda_1(L(B_{d,k})) \leq (\sqrt{d}+1)^2$, thus we conclude that $\exists d_1 : \forall d \geq d_1, k \geq 2$

$$\frac{n(B_{d,k})}{2} \leq \chi(B_{d,k}) \leq 2n(B_{d,k}). \tag{4}$$

B.2. Local Functions

Denote $\mathcal{V}_1$ the set of vertices of type 1, $\mathcal{V}_2$ the set of vertices of type 2 ($\mathcal{V}_1 \cap \mathcal{V}_2 = \varnothing$), and let $\mathcal{W}$ the set of remaining vertices. Let $d \geq t > 2$. Let $\mathcal{V}_1$ and $\mathcal{V}_2$ consist of $[\frac{d}{t}]$ subtrees with roots adjacent to the root of $B_{d,k}$. Therefore $|\mathcal{V}_1| = |\mathcal{V}_2| = [\frac{d}{t}]\frac{d^{k-1}-1}{d-1}$.

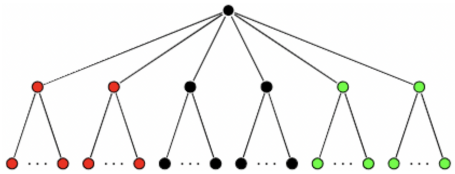

Figure 4. Example of splitting a graph into three sets. The vertices of $\mathcal{V}_1$ ("Bad" vertices) are indicated in red. The vertices of $\mathcal{V}_2$ are indicated in green.

Assign each vertex a function $f_v : \ell_2 \to \mathbb{R}$ that depends on the vertex type:

$$f_v(x) = \begin{cases} \frac{\mu}{2n}\|x\|^2 + \frac{L-\mu}{4|\mathcal{V}_1|}\left[(x_1-1)^2 + \sum_{k=1}^{\infty}(x_{2k}-x_{2k+1})^2\right], & v \in \mathcal{V}_1 \\ \frac{\mu}{2n}\|x\|^2 + \frac{L-\mu}{4|\mathcal{V}_2|}\sum_{k=1}^{\infty}(x_{2k-1}-x_{2k})^2, & v \in \mathcal{V}_2 \\ \frac{\mu}{2n}\|x\|^2, & v \in \mathcal{W} \end{cases}. \tag{5}$$

Using that $d \geq t$, $k \geq 2, d \geq 3$, we estimate $|\mathcal{V}_1|$ and $|\mathcal{V}_2|$:

$$|\mathcal{V}_1| = |\mathcal{V}_2| \geq \frac{d}{2t}\frac{d^{k-1}-1}{d-1} \geq \frac{n}{4t}. \tag{6}$$

Estimate the network's global characteristic number using the local one

$$\kappa_l = \frac{\frac{L-\mu}{|\mathcal{V}_1|} + \frac{\mu}{n}}{\frac{\mu}{n}} \leq \frac{\frac{4t(L-\mu)}{n} + \frac{\mu}{n}}{\frac{\mu}{n}} = \frac{4t(L-\mu)+\mu}{\mu} = 4(\kappa_g - 1)t + 1,$$

thus we have

$$\kappa_g \geq \frac{\kappa_l - 1}{4t} + 1. \tag{7}$$

B.3. Edge Modification and Information Flow

Let us now start describing the sequence of graphs $\{\mathcal{G}_i\}_{i=1}^{\infty}$. Next, introduce the following scheme: A graph consists of "good" and "bad" vertices. At each iteration, each "good" vertex adjacent to at least one "bad" vertex becomes a bad vertex. After that, we somehow change the edges in the graph, and the process continues. At initialization all vertices of $\mathcal{V}_1$ are "bad" and all vertices of $\mathcal{V} \setminus \mathcal{V}_1$ are "good". Our goal is to make the $\mathcal{V}_2$ vertices remain "good" as long as possible. The algorithm returns a graph sequence $\{\mathcal{G}_i\}_{i=1}^{\infty}$.

The aim of the edge modification algorithm is to dampen the spread of "bad" vertices. At each step of the algorithm, we determine the vertices that will become bad at the forthcoming iteration, and move this vertices to lower levels of the tree. Let us describe the required auxiliary procedures.

Firstly, let us introduce a *local numbering* of the vertices. Namely, let each non-leaf node locally assign numbers $1, \ldots, d$ to its children in random order. We assume that the children of root vertex are numbered in such a way that roots of subtrees in $\mathcal{V}_1$ have the smallest numbers and roots of subtrees in $\mathcal{V}_2$ have the largest numbers.

Let us describe the auxiliary functions used in Algorithm 2.

1. $\mathrm{PotentialBadVertices}(\mathcal{G}, \mathcal{B})$ computes the set of "good" vertices which would become "bad" ones at the next iteration. These are the vertices adjacent to at least one vertex at $\mathcal{B}$, i.e. the vertex boundary of $\mathcal{B}$.

2. $\mathrm{FindCandidateToSwap}(\mathcal{G}, u)$ finds a vertex that will be swapped with the vertex u. Let us start at the root and move to the lowest numbered child of the vertex until we run into a "bad" vertex or to a leaf node. $\mathrm{FindCandidateToSwap}(\mathcal{G}, u)$ returns the vertex where we stopped.

3. $\mathrm{Swap}(\mathcal{G}, u, v)$ operation swaps the edges of vertices u and v, but not the vertices themselves. It returns a graph $\mathcal{G}'$ such that the neighbors of u in $\mathcal{G}'$ are the neighbors of v in $\mathcal{G}$ and the neighbors of v in $\mathcal{G}'$ are the neighbors of u in $\mathcal{G}$. Note that $\mathrm{Swap}(\mathcal{G}, u, v)$ operation changes no more than $2(\deg(u) + \deg(v))$ edges ($\Delta \leq 2(\deg(v_1) + \deg(v_2))$).

Remark B.1. Consider a graph $B_{d,k}$ that initially consists of only "good" vertices and is endowed with a local numeration of the vertices. Let r denote the root of the graph. Consider the following procedure: find $v = \mathrm{FindCandidateToSwap}(\mathcal{G}, r)$ and mark v as a "bad" vertex. Repeating this procedure for a sufficient number of times will mark all the nodes of $|\mathcal{V}_1|$ and only them as "bad".

Now we present the edge modification algorithm.

Algorithm 2 InnerLoopPoly

Input: $\mathcal{G} = B_{d,k}$, $\mathcal{V}_1$, $\mathcal{V}_2$, i
Initialize set of "bad" vertices $\mathcal{B} = \mathcal{V}_1$.
while $\mathcal{V}_2 \cap \mathcal{B} = \varnothing$ **do**
  $U = \mathrm{PotentialBadVertices}(\mathcal{G}, \mathcal{B})$
  for u in U **do**
    $v = \mathrm{FindCandidateToSwap}(\mathcal{G}, u)$
    $\mathcal{G} = \mathrm{Swap}(\mathcal{G}, u, v)$
    $\mathcal{B} = \mathcal{B} \cup \{v\}$
  end for
  $\mathcal{G}_i = \mathcal{G}$
  $i = i + 1$
end while

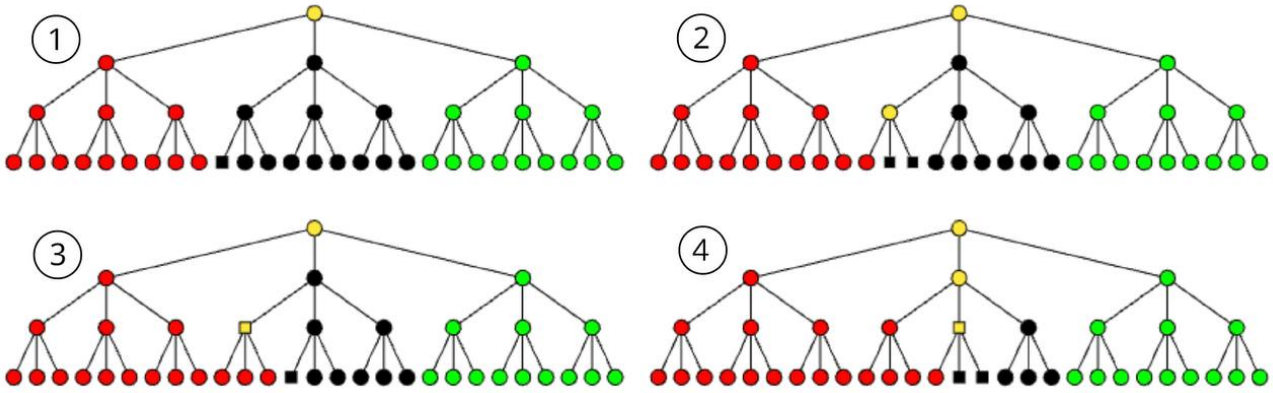

Figure 5. Example of a graph change scheme in the case with poly constraints. Potentially "bad" vertices, i.e. those that will become "bad" in one move, are indicated in yellow. A square indicates where "bad" vertices will be moved after a Swap operations. The circled numbers denote the iterations of Algorithm 2.

Lemma B.2. *Consider an Algorithm 2, applied to $B_{d,k}$. At any moment of this algorithm, the number of new bad vertices is bounded as follows:*

$$|U| \leq k - 1. \tag{8}$$

Proof. Let us define a hierarchy of levels $\{L_i\}_{i=1}^{k}$ in the rooted tree, where L_1 is the root level and L_k is the deepest level.

The "bad" vertex invariant property can be defined as follows: if a vertex p is "bad", then all vertices in its subtree are also "bad".

Firstly, note that $U \cap L_k = \varnothing$. If there were any vertices in $U \cap L_k$, this would contradict the invariant property.

Secondly, for each level L_i, $i < k$, we have $|U \cap L_i| \leq 1$. To show that, we assume the contrary: let there exist two vertices $v, u \in U \cap L_i$. These vertices are adjacent to vertices $v', u' \in B$ at the level L_{i+1}. The common ancestor of v, u at level L_j ($j < i$), denoted as p, is a "good" vertex with two "good" successors, v and u, which have "bad" vertices v' and u' in their subtrees. According to Algorithm 2, the entire subtree of p with the least vertex number should be filled with "bad" vertices. This contradicts our assumption that both v and u are "good" vertices. An example of how this theorem works can be seen in 5. In Figure 5, vertices in U are marked yellow.

Hence, at any iteration of Algorithm 2 we have $|U| \leq k - 1$. □

Let Δ_i denote the number of edges changed at iteration i. Using Lemma B.2, we get an upper bound on Δ_i. Keeping in mind that each node has degree at most $(d + 1)$, we obtain

$$\Delta_i \leq 4|U|(d+1) \leq 4(k-1)(d+1). \tag{9}$$

Using that $n = \frac{d^k-1}{d-1}$ and assuming $d \geq k$ we get

$$\Delta_i \leq 4(k-1)(d+1) \leq 4kd \leq 4kn^{1/(k-1)}. \tag{10}$$

Let T be the number of iterations needed for one of the vertices in $\mathcal{V}_2$ to become "bad" (it equals the number of iterations of Algorithm 2). As discussed in Appendix A, quantity T equals the information flow. At each iteration, there are at most $k - 1$ new "bad" vertices. Therefore, we get:

$$T \geq \left\lfloor \frac{|\mathcal{W}|}{k-1} \right\rfloor + 1 \geq \frac{|\mathcal{W}|}{k-1}. \tag{11}$$

We estimate the size of the neutral vertex set $\mathcal{W}$ using $n = \frac{d^k-1}{d-1}$ and the definitions of $\mathcal{V}_1, \mathcal{V}_2$:

$$|\mathcal{W}| = n - |\mathcal{V}_1| - |\mathcal{V}_2| \geq \frac{1}{d-1}\left(d^k - 1 - 2\frac{d^k - d}{t}\right) \geq \left(1 - \frac{2}{t}\right) n.$$

Therefore using 4 we get lower bound on the information flow:

$$T \geq \frac{1 - 2/t}{2(k-1)}\chi. \tag{12}$$

B.4. Lower Bound on the Number of Communications

Now, we are going to define the whole sequence of graphs in $\{\mathcal{G}_i\}$ for every $i \in \mathbb{N}$. Algorithm 2 only defines the graphs for $i = i_0, i_0 + 1, \ldots, i_0 + T - 1$, where i_0 is the input state number for the Inner algorithm. We run Algorithm 2 iteratively. After each iteration, at least one vertex of $\mathcal{V}_2$ becomes "bad", then we rearrange the sets $\mathcal{V}_1$ and $\mathcal{V}_2$, reverse the order of children for each vertex in $\mathcal{G}$, and run the algorithm again.

The auxiliary procedures used in the algorithm are the following.

1. $\mathrm{InnerLoopPoly}(\mathcal{G}, \mathcal{V}_1, \mathcal{V}_2, i)$ is Algorithm 2.
2. $\mathrm{Rearrange}(\mathcal{G}, \mathcal{V}_1, \mathcal{V}_2)$ changes the pointers for variables $\mathcal{V}_1$ and $\mathcal{V}_2$.
3. $\mathrm{ReverseOrders}(\mathcal{G})$ reverses the order of children for each vertex.

Algorithm 3 OuterLoopPoly

Input: $\mathcal{G}$, $\mathcal{V}_1$, $\mathcal{V}_2$
$i = 1$
while True **do**
  $\text{InnerLoopPoly}(\mathcal{G}, \mathcal{V}_1, \mathcal{V}_2, i)$
  $\text{Rearrange}(\mathcal{G}, \mathcal{V}_1, \mathcal{V}_2)$
  $\text{ReverseOrders}(\mathcal{G})$
end while

We have found a sequence of graphs in which information flows slowly. Specifically, to get from $\mathcal{V}_1$ to $\mathcal{V}_2$, it takes T iterations. To get back (from $\mathcal{V}_2$ to $\mathcal{V}_1$) it takes T iterations as well and so on.

Let $x_0 = 0$ be the initial point for the first-order decentralized algorithm. Recall the definition of $\mathcal{H}_v(p)$ from Section 3.1. For every $m \geq 1$, we define $l_m = \min\{p \geq 1 | \exists v : \exists x \in \mathcal{H}_v(p) : x_m \neq 0\}$ as the first moment when we can get a non-zero element at the m-th place at any node.

Considering the types of functions on vertices of the graph 5, we can conclude that functions on vertices from $\mathcal{V}_1$ can "transfer" (by calculating the gradient) information (non-zero element) from the even positions $(2, 4, 6, \ldots)$ to the next ones, and functions on vertices from $\mathcal{V}_2$ can transfer information from the odd positions $(1, 3, 5, \ldots)$ to the next ones. Therefore, for the network to get a new non-zero element at the next position, a complete iteration of Algorithm 2 is required, that is T communication iterations. In other words, the "information" cannot spread faster than "bad" vertices.

To reach the m-th non-zero element, we need to make at least m local steps and $(m-1)T$ communication steps to transfer information from gradients between $\mathcal{V}_1$ and $\mathcal{V}_2$ sets. Therefore, we can estimate l_m:

$$l_m \geq (m-1)T + m. \tag{13}$$

The solution of the global optimization problem is $(x_*)_p = \left(\frac{\sqrt{\kappa_g}-1}{\sqrt{\kappa_g}+1}\right)^p$.

For any m, p such that $l_m > p$ we have

$$\|x_p - x_*\|^2 \geq (x_*)_m^2 + (x_*)_{m+1}^2 + \ldots = \left(\frac{\sqrt{\kappa_g}-1}{\sqrt{\kappa_g}+1}\right)^m \|x_0 - x_*\|^2.$$

Using (13) we can take $m = \lceil \frac{p}{T+1} \rceil + 1$. From (7) we conclude that $\frac{\sqrt{\kappa_g}-1}{\sqrt{\kappa_g}+1} \geq 1 - \frac{4\sqrt{3}}{\sqrt{\kappa_l}}$

Therefore using (7), (12) and assign $t = 3$ we get

$$\|x_p - x_*\|^2 \geq \left(\frac{\sqrt{\kappa_g}-1}{\sqrt{\kappa_g}+1}\right)^{\left\lceil p/\left(\frac{1}{6(k-1)}\chi+1\right)\right\rceil+1} \|x_0 - x_*\|^2.$$

Rearranging it, we get

$$\|x_p - x_*\|^2 \geq \left(\max\left\{0, 1 - 4\sqrt{3}\sqrt{\frac{\mu}{L}}\right\}\right)^{\frac{6(k-1)p}{\chi}+2} \|x_0 - x_*\|^2. \tag{14}$$

C. Proof of the Theorem 3.3

C.1. Graph condition number

The proof is very similar to the proof of the Theorem 3.1, but here we fix $d = 2$ and $k \to \infty$.

Let B_k be a binary tree $B_{2,k}$. Using the lower and upper bounds on algebraic connectivity (λ_{n-1}) of such trees from (Molitierno et al., 1999) and following the same logic as in (4), we can conclude that there exists a k_0 such that for all $k > k_0$ the following holds:

$$2n(B_k) \leq \chi(B_k) \leq 6n(B_k). \tag{15}$$

C.2. Local Functions

Denote the set of vertices of type 1 as $\mathcal{V}_1$, the set of vertices of type 2 as $\mathcal{V}_2$ ($\mathcal{V}_1 \cap \mathcal{V}_2 = \varnothing$), and the set of remaining vertices as $\mathcal{W}$. Suppose that every non-leaf vertex has a "left" and "right" child. Let $\mathcal{V}_1$ be a subtree with a "left-left" root vertex (it can be reached from the graph's root by going to the left child and then back to the left child). $\mathcal{V}_2$ is defined in the same way. Therefore $|\mathcal{V}_1| = |\mathcal{V}_2| = 2^{k-2} - 1$.

Denote the vertex functions $f_v : \ell_2 \to \mathbb{R}$ similarly as in 5.

Estimate $\mathcal{V}_1$ and $\mathcal{V}_2$ through n, using that $k \geq 4$ we get

$$\frac{n}{5} \leq |\mathcal{V}_1| = |\mathcal{V}_2| \leq \frac{n}{4}. \tag{16}$$

Estimate global characteristic number of the network through local one

$$\kappa_l = \frac{\frac{L-\mu}{|\mathcal{V}_1|} + \frac{\mu}{n}}{\frac{\mu}{n}} \leq \frac{\frac{5(L-\mu)}{n} + \frac{\mu}{n}}{\frac{\mu}{n}} = \frac{5(L-\mu)+\mu}{\mu} = 5(\kappa_g - 1) + 1,$$

thus we have

$$\kappa_g \geq \frac{1}{5}(\kappa_l - 1) + 1. \tag{17}$$

C.3. Edge Modification Algorithm and Information Flow

Next, we will use exactly the same technique to construct a sequence of communication graphs as in the proof of the Theorem 3.1 (Algorithm 2 and Algorithm 3). As a result, we get something resembling 11 inequality on "information flow" defined in previous proof

$$T \geq \frac{|\mathcal{W}|}{k-1}. \tag{18}$$

Using $n = 2^k - 1$ and the definitions of $\mathcal{V}_1, \mathcal{V}_2$, we can estimate the size of the neutral vertex set $\mathcal{W}$.

$$|\mathcal{W}| = n - |\mathcal{V}_1| - |\mathcal{V}_2| = 2^k - 1 - 2(2^{k-2} - 1) \geq \frac{n}{2}.$$

By using (15) and inequality $k - 1 \leq \log_2 n$ we derive a lower bound on T:

$$T \geq \frac{\chi}{12(k-1)} \geq \frac{\chi}{12\log_2(\chi/2)}. \tag{19}$$

Also we similarly take an upper bound on edge change Δ_i in graph sequence

$$\Delta_i \leq 4|U|(2+1) = 12(k-1) \leq 12\log_2 n. \tag{20}$$

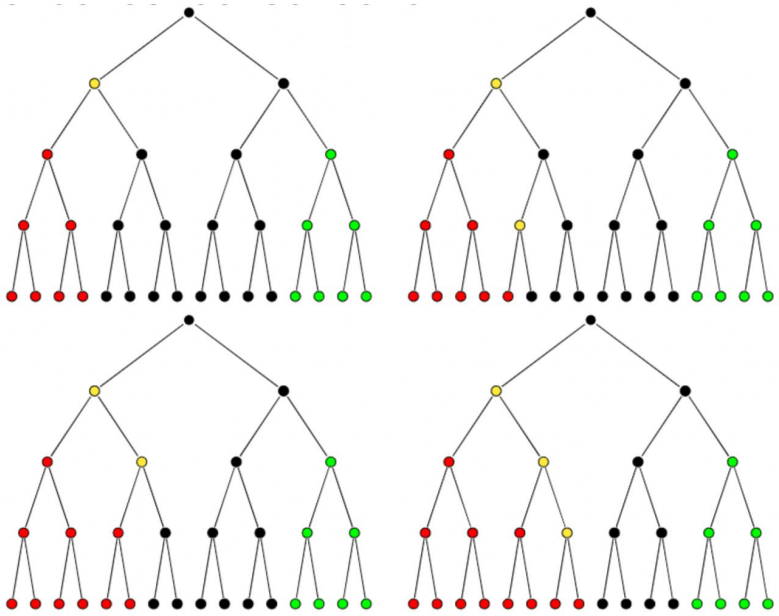

Figure 6. Example of a graph change scheme in the case with log constraints. Potentially "bad" vertices, i.e. those that will become "bad" in one move, are indicated in yellow.

C.4. Lower Bound on the Number of Communications

Do the same reasoning with l_m (defined in the last section).

In order to reach the m-th non-zero element, at least m local steps and $(m-1)T$ communication steps are required to transfer information from gradients between $\mathcal{V}_1$ and $\mathcal{V}_2$ sets. Based on this, we can estimate l_m:

$$l_m \geq (m-1)T + m. \tag{21}$$

The solution of the global optimization problem is $(x_*)_p = \left(\frac{\sqrt{\kappa_g}-1}{\sqrt{\kappa_g}+1}\right)^p$. (In this case, x^* is the optimal solution of the optimization problem, and x_p^* is its coordinates in l_2.)

For any m, p such that $l_m > p$

$$\|x_p - x_*\|^2 \geq (x_*)_m^2 + (x_*)_{m+1}^2 + \ldots = \left(\frac{\sqrt{\kappa_g}-1}{\sqrt{\kappa_g}+1}\right)^m \|x_0 - x_*\|^2.$$

Using (21) we can take $m = \lceil \frac{p}{T+1} \rceil + 1$. From (17) we conclude that $\frac{\sqrt{\kappa_g}-1}{\sqrt{\kappa_g}+1} \geq 1 - \frac{2\sqrt{5}}{\sqrt{\kappa_l}}$

Therefore using (17), (19) we get

$$\|x_p - x_*\|^2 \geq \left(\max\left\{0, 1 - 2\sqrt{5}\sqrt{\frac{\mu}{L}}\right\}\right)^{\left\lceil p/\left(\frac{\chi}{12\log_2(\chi/2)}+1\right)\right\rceil+1} \|x_0 - x_*\|^2.$$

Rearranging it, we get

$$\|x_p - x_*\|^2 \geq \left(\max\left\{0, 1 - 2\sqrt{5}\sqrt{\frac{\mu}{L}}\right\}\right)^{\frac{12\log_2(\chi/2)p}{\chi}+2} \|x_0 - x_*\|^2. \tag{22}$$

D. Proof of the Theorem 3.5

D.1. Graph Condition Number

Firstly, let us define the structure of the graph that will serve as a counter-example in the case under consideration and study its properties.

We will define the graph $H_{d,k}$ through induction. Let $H_{1,k}$ be a path on the k vertices and call any of its leaf vertices the root. Then, assuming that we have defined $H_{d,k}$ for all k, we define $H_{d+1,k}$ as follows: take a path on k vertices (and call the leaf vertex of the path the root of the graph) and attach a copy of $H_{d,k}$ to the root of each vertex in the path.

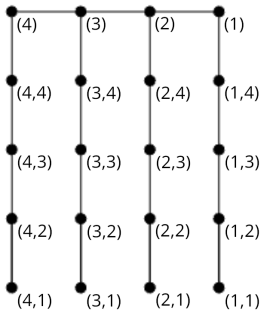

Figure 7. Example of graph $H_{2,4}$ with vertex numbering.

It can be seen that each tree $H_{d,k}$ consists of paths on k vertices with fixed leafs, which we will refer to as roots. Consider one such path, with a "start" vertex and an "end" vertex, where the "end" vertex is a root, and the "start" vertex is the other leaf vertex. We will also assign a number from 1 to k to each vertex in the path, corresponding to the distance from the "start" vertex, increased by 1. That is, the "start" and root vertices have the numbers 1 and k respectively.

We will divide the tree $H_{d,k}$ into levels. As the graph was constructed through induction, the first level will consist of the vertices added in the first iteration of the induction, the second level will consist of the vertices added in the second iteration of induction, and so on, up to level d.

To move forward, let us assign coordinates to the vertices of $H_{d,k}$ as follows. At the top level, let the vertices be numbered from 1 to k in increasing order. Consider the vertex v at level g, then its coordinates will be a tuple $x = (x_1, \ldots, x_g)$, where x_i is the number corresponding to the vertex closest to v at level i. That is, the entire graph consists of paths on k vertices, for which leaf vertices "start" and "end" are fixed, where the "end" is the root of the tree. The numbering for $H_{2,4}$ is presented in Figure 7. Then, the vertex $(x_1, \ldots, x_g)$ will be adjacent to the vertices $(x_1, \ldots, x_g - 1)$, $(x_1, \ldots, x_g + 1)$, $(x_1, \ldots, x_g, k)$ if $x_g \neq k$ and to the vertex $(x_1, \ldots, x_{g-1})$ if $x_g = k$.

Let us introduce a linear order relation on these vertices: if the coordinate of the first vertex is the prefix of the second one, then the second one is smaller, otherwise the one with the smallest element, which has the first difference from left to right, is smaller.

Consider a graph $H_{d,k}$. The number of nodes in this graph is $n = k + k^2 + \ldots + k^d$. Let D be the diameter of this graph, it can be easily seen that $D = (2d-1)k - 1$. According to Theorem 4.1.1 in (Das, 2004), we can obtain an estimation of $\lambda_{n-1}(L(H_{d,k}))$

$$\lambda_{n-1}(L(H_{d,k})) \geq \frac{1}{nD} \geq \frac{1}{d(2d-1)k^{d+1}}. \tag{23}$$

Using results from (Stevanović, 2003) we get

$$4 \leq \lambda_1(L(H_{d,k})) \leq 3 + 2\sqrt{2} \leq 6. \tag{24}$$

As a result , using 23 and 24 we can obtain an upper bound for $\chi(H_{d,k})$.

$$\chi(H_{d,k}) \leq 6d(2d-1)k^{d+1} \leq 6d(2d-1)n^{\frac{d+1}{d}}. \tag{25}$$

D.2. Local Functions

Let $\mathcal{V}_1$ and $\mathcal{V}_2$ be disjoint sets of vertices of type 1 and type 2, respectively, and let $\mathcal{W}$ be the set of remaining vertices. $\mathcal{V}_1$ consists of vertices that have a coordinate of at most $(\left[\frac{k}{3}\right])$, and $\mathcal{V}_2$ consists of vertices that have a coordinate of at least $(k - \left[\frac{k}{3}\right])$. Therefore, $|\mathcal{V}_1| = |\mathcal{V}_2| = \left[\frac{k}{3}\right](k + k^2 + \ldots + k^{d-1})$.

Denote the vertex functions $f_v : \ell_2 \to \mathbb{R}$ depending on vertex type:

$$f_v(x) = \begin{cases} \frac{\mu}{2n}\|x\|^2 + \frac{L-\mu}{4|\mathcal{V}_1|}\left[(x_1-1)^2 + \sum_{k=1}^{\infty}(x_{2k} - x_{2k+1})^2\right], & v \in \mathcal{V}_1 \\ \frac{\mu}{2n}\|x\|^2 + \frac{L-\mu}{4|\mathcal{V}_2|}\sum_{k=1}^{\infty}(x_{2k-1} - x_{2k})^2, & v \in \mathcal{V}_2 \\ \frac{\mu}{2n}\|x\|^2, & v \in \mathcal{W} \end{cases}. \tag{26}$$

Let $k \geq 3$ and estimate $\mathcal{V}_1$ and $\mathcal{V}_2$ through n

$$|\mathcal{V}_1| = |\mathcal{V}_2| \geq \frac{k}{6}(k + k^2 + \ldots + k^{d-1}) = \frac{n}{12}. \tag{27}$$

Estimate the network's global characteristic number using the local one

$$\kappa_l = \frac{\frac{L-\mu}{|\mathcal{V}_1|} + \frac{\mu}{n}}{\frac{\mu}{n}} \leq \frac{\frac{6(L-\mu)}{n} + \frac{\mu}{n}}{\frac{\mu}{n}} = \frac{12(L-\mu)+\mu}{\mu} = 12(\kappa_g - 1) + 1,$$

thus we have

$$\kappa_g \geq \frac{\kappa_l - 1}{12} + 1. \tag{28}$$

D.3. Edge Modification Algorithm and Information Flow

Let us now describe the sequence of graphs $\{\mathcal{G}_i\}_{i=1}^{\infty}$. Similar to the proof of Theorem 3.1, we will construct an algorithm that generates a sequence of graphs that works under the same conditions as Algorithm 2 and Algorithm 3. In this scheme, we will refer to vertices in $\mathcal{V}_1$ as "bad" and the remaining vertices as "good". After each iteration, a "good" vertex that is adjacent to a "bad" vertex becomes "bad", and the graph is modified in some way. The goal is to keep the vertices in $\mathcal{V}_2$ "good" for as long as possible.

We will maintain the *invariant* that after each graph change, a "good" vertex cannot be less than a "bad" vertex.

Auxiliary procedures in the edge changing algorithm work as follows.

1. $\mathrm{FindCandidateToSwap}$ finds a vertex that would be swapped with the input vertex. It finds the smallest "good" vertex in the graph. It is simple to check that the invariant is preserved.

2. $\mathrm{AtLastLevel}$ checks if there is a vertex at the last level, makes it "bad", and removes it from U.

3. Swap works the same as in Algorithm 2.

4. $\mathrm{PotentialBadVertices}$ works the same as in Algorithm 2.

Algorithm 4 InnerLoopConstant

Input: $\mathcal{G}, \mathcal{V}_1, \mathcal{V}_2, i$
$\mathcal{B} = \mathcal{V}_1$
while $\mathrm{NoBadVertices}(\mathcal{V}_2, \mathcal{B})$ **do**
  $U = \mathrm{PotentialBadVertices}(\mathcal{B})$
  $\mathrm{AtLastLevel}(U, \mathcal{B})$
  for u in U **do**
    $v = \mathrm{FindCandidateToSwap}(u, \mathcal{B})$
    $\mathcal{G} = \mathrm{Swap}(\mathcal{G}, u, v)$
    $\mathcal{B} = \mathcal{B} \cup \{v\}$
  end for
  $\mathcal{G}_i = \mathcal{G}$
  $i = i + 1$
end while

Then we apply Algorithm 3, but using Algorithm 4 as the inner algorithm, thus obtaining a sequence of graphs.

Lemma D.1. *Consider an Algorithm 4, applied to $H_{d,k}$. At any moment of this algorithm, the number of new bad vertices is bounded as follows:*

$$|U| \leq d. \tag{29}$$

Proof. We consider the set U and show that it can contain at most one element from each level. Suppose the converse, let the vertices $v, u \in U$ and belong to level g. Let vertex v have coordinates $(x_1, \ldots, x_g)$ and vertex u have coordinates $(y_1, \ldots, y_g)$ and $v < u$. The vertex u is adjacent to the "bad" vertex b, and $b < u$, so it can be $(y_1, \ldots, y_g - 1)$ or $(y_1, \ldots, y_g, d)$. The second case is impossible because the "bad" vertex $(y_1, \ldots, y_g, d)$ is greater than v, and this contradicts the invariant. Consider the first case when b has coordinates $(y_1, \ldots, y_g - 1)$. $v < u$, so $v \leq b$, but they cannot be equal, since at the given iteration, v is only potentially "bad" (i.e. "good") so far, so we are led to the same contradiction when the "bad" vertex is greater than the "good" one. □

Note that either U contains no vertices on the last level, in which case $|U| \leq d - 1$ (as can be proved similarly to Lemma D.1), or there is a vertex on the last level, but it need not be swapped. Therefore, we can obtain an upper bound on the number of edges changed at iteration i, that we denote Δ_i.

$$\Delta_i \leq 12(d - 1). \tag{30}$$

Let T be the number of iterations, needed for one of the vertices in V_2 to become bad (i.e. the "information flow"). It equals to the number of iterations of Algorithm 4. According to Lemma D.1 at each While iteration there are not more than d new bad vertices, therefore we get

$$T \geq \left\lfloor \frac{|\mathcal{W}|}{d} \right\rfloor + 1 \geq \frac{|\mathcal{W}|}{d}. \tag{31}$$

Using $|\mathcal{V}_1| = |\mathcal{V}_2| \leq \frac{n}{3}$ to estimate the size of the neutral vertex set $\mathcal{W}$, we get

$$|\mathcal{W}| = n - |\mathcal{V}_1| - |\mathcal{V}_2| \geq \frac{n}{3}.$$

Therefore using (25) we get lower bound on the information flow:

$$T \geq \frac{n}{3d} \geq \frac{\chi^{\frac{d}{d+1}}}{3d(6d(2d-1))^{\frac{d}{d+1}}}. \tag{32}$$

D.4. Lower Bound on the Number of Communications

To proceed further, we apply a similar approach to determine l_m (defined in the the proof of the Theorem 3.1).

To reach the m-th non-zero element, we need to make at least m local steps and $(m-1)T$ communication steps to transfer information from gradients between $\mathcal{V}_1$ and $\mathcal{V}_2$ sets. Therefore, we can estimate l_m:

$$l_m \geq (m-1)T + m. \tag{33}$$

The solution of the global optimization pro blem is $(x_*)_p = \left(\frac{\sqrt{\kappa_g}-1}{\sqrt{\kappa_g}+1}\right)^p$.

For any m, p such that $l_m > p$

$$\|x_p - x_*\|^2 \geq (x_*)_m^2 + (x_*)_{m+1}^2 + \ldots = \left(\frac{\sqrt{\kappa_g}-1}{\sqrt{\kappa_g}+1}\right)^m \|x_0 - x_*\|^2.$$

Using 33 we can take $m = \lceil \frac{p}{T+1} \rceil + 1$. From 28 we conclude that $\frac{\sqrt{\kappa_g}-1}{\sqrt{\kappa_g}+1} \geq 1 - \frac{4\sqrt{3}}{\sqrt{\kappa_l}}$

Let $C(d) = 3d(6d(2d-1))^{\frac{d}{d+1}}$. Therefore using 28, 32 we get

$$\|x_p - x_*\|^2 \geq \left(\max\left\{0, 1 - 4\sqrt{3}\sqrt{\frac{\mu}{L}}\right\}\right)^{\left\lceil p/\left(C(d)^{-1}\chi^{\frac{d}{d+1}}+1\right)\right\rceil+1} \|x_0 - x_*\|^2.$$

Rearranging it, we get

$$\|x_p - x_*\|^2 \geq \left(\max\left\{0, 1 - 4\sqrt{3}\sqrt{\frac{\mu}{L}}\right\}\right)^{C(d)p\chi^{-\frac{d}{d+1}}+2} \|x_0 - x_*\|^2. \tag{34}$$

E. Accelerated Method over Time-Varying Function

In this section, we show the convergence of accelerated Nesterov method over a uniformly non-increasing time-varying function. The proof is based on potential analysis in (Bansal & Gupta, 2019), and a similar proof technique was used in (Rogozin et al., 2019).

Consider a sequence of functions $\{f_k(x)\}_{k=0}^{\infty}$ such that the following assumptions hold.

Assumption E.1. For every $k = 0, 1, 2, \ldots$, function $f_k(x)$ is L-smooth and μ-strongly convex, that is, for any $x, y \in \mathbb{R}^d$ we have

$$\frac{\mu}{2}\|y - x\|_2^2 \leq f(y) - f(x) - \langle \nabla f(x), y - x\rangle \leq \frac{L}{2}\|y - x\|_2^2.$$

Assumption E.2. Sequence $\{f_k(x)\}_{k=0}^{\infty}$ is uniformly non-increasing: for each $x \in \mathbb{R}^d$ and for any $k = 0, 1, 2, \ldots$ we have

$$f_{k+1}(x) \leq f_k(x).$$

Assumption E.3. Functions of sequence $\{f_k(x)\}_{k=0}^{\infty}$ have a common minimizer x^*.

Let an accelerated method be run over $\{f_k(x)\}_{k=0}^{\infty}$.

$$y_{k+1} = x_k - \frac{1}{L}\nabla f_k(x_k), \tag{35a}$$

$$x_{k+1} = \left(1 + \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right) y_{k+1} - \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} y_k, \tag{35b}$$

where $\kappa = L/\mu$. Then we have the following convergence result.

Theorem E.4. *Let accelerated Nesterov method be run over sequence $\{f_k\}_{k=0}^{\infty}$ and let Assumptions E.1 and E.2 hold. We have*

$$f_N(y_N) - f^* \leq \frac{(L+\mu)R^2}{2}(1-1/\sqrt{\kappa})^N,$$
$$\|y_N - y^*\|_2^2 \leq \frac{(L+\mu)R^2}{\mu}(1-1/\sqrt{\kappa})^N.$$

Before passing to proof of Theorem E.4, we need the following auxiliary lemma.

Lemma E.5. *Consider updates in (35) and define*

$$\tau = \frac{1}{\sqrt{\kappa}+1}, \text{ and } z_{k+1} = \frac{1}{\tau}x_{k+1} - \frac{1-\tau}{\tau}y_{k+1}.$$

Then, $z_{k+1} = \frac{1}{1+\gamma}z_k + \frac{\gamma}{1+\gamma}x_k - \frac{\gamma}{\mu(1+\gamma)}\nabla f_k(x_k)$, where $\gamma = \frac{1}{\sqrt{\kappa}-1}$.

Proof. By the update rule for x_{k+1} given in (35) and the definition of τ, we have that

$$\begin{aligned} x_{k+1} &= \left(1 + \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)y_{k+1} - \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}y_k \\ &= (2-2\tau)y_{k+1} - (1-2\tau)y_k. \end{aligned}$$

Moreover, by the definition of z_{k+1}, it follows that

$$\begin{aligned} z_{k+1} &= \frac{1}{\tau}x_{k+1} - \frac{1-\tau}{\tau}y_{k+1} \\ &= \frac{1}{\tau}\left((2-2\tau)y_{k+1} - (1-2\tau)y_k\right) - \frac{1-\tau}{\tau}y_{k+1} \\ &= \frac{1}{\tau}\left((1-\tau)y_{k+1} - (1-2\tau)y_k\right). \end{aligned}$$

Now we use the update rule for y_{k+1} given in (35) and also note that $x_k = (1-\tau)y_k + \tau z_k$:

$$\begin{aligned} z_{k+1} &= \frac{1}{\tau}\left[(1-\tau)(x_k - \frac{1}{L}\nabla f_k(x_k)) - \frac{1-2\tau}{1-\tau}(x_k - \tau z_k)\right] \\ &= \frac{1-2\tau}{1-\tau}z_k + \frac{\tau}{1-\tau}x_k - \frac{1-\tau}{L\tau}\nabla f_k(x_k) \\ &\overset{①}{=} \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}}z_k + \frac{1}{\sqrt{\kappa}}x_k - \frac{1}{\mu\sqrt{\kappa}} \\ &\overset{②}{=} \frac{1}{1+\gamma}z_k + \frac{\gamma}{1+\gamma}x_k - \frac{\gamma}{\mu(1+\gamma)}\nabla f_k(x_k), \end{aligned}$$

where ① is obtained by using the definitions of τ and κ, and ② is obtained by using the definition of γ. □

Lemma E.6. *Let $\{f_k(x)\}_{k=0}^{\infty}$ be a sequence of functions for which Assumptions E.1 and E.2 hold. Introduce potential function*

$$\Psi_k = (1+\gamma)^k \cdot \left(f_k(y_k) - f^* + \frac{\mu}{2}\|z_k - x^*\|_2^2\right), \tag{36}$$

Then, it holds that

$$\Delta\Psi_k = \Psi_{k+1} - \Psi_k \leq 0. \tag{37}$$

Proof. The proof is analogous to the proof in Section 5.4 in (Bansal & Gupta, 2019). We use the definitions of τ, z_k given in Lemma E.5. We have

$$\begin{aligned}\Delta\Psi_k \cdot (1+\gamma)^{-k} &= (1+\gamma)\big(f(y_{k+1}) - f^* + \frac{\mu}{2}\|z_{k+1} - x^*\|_2^2\big) \\ &\quad - \big(f(y_k) - f^* + \frac{\mu}{2}\|z_k - x^*\|_2^2\big) \\ &= (1+\gamma)\big(f_{k+1}(y_{k+1}) - f_{k+1}(x^*)\big) - \big(f_k(y_k) - f_k(x^*)\big) \\ &\quad + \frac{\mu}{2}\Big[(1+\gamma)\|z_{k+1} - x^*\|_2^2 - \|z_k - x^*\|_2^2\Big]. \end{aligned} \tag{38}$$

Note that from Assumption E.2 and from basic gradient step inequality we have

$$f_k(y_k) \le f_k(y_{k+1}) \le f_k(x_k) - \frac{1}{2L}\|\nabla f_k(x_k)\|_2^2,$$

We bound th first term in (38) as follows:

$$\begin{aligned}&(1+\gamma)\big(f_{k+1}(y_{k+1}) - f_{k+1}(x^*)\big) - \big(f_k(y_k) - f_k(x^*)\big) \\ &\le (1+\gamma)\big(f_k(x_k) - \frac{1}{2L}\|\nabla f_k(x_k)\|_2^2 - f^*\big) - \big(f_k(y_k) - f^*\big) \\ &= f_k(x_k) - f_k(y_k) + \gamma(f_k(x_k) - f^*) - (1+\gamma)\frac{\|\nabla f_k(x_k)\|_2^2}{2L} \\ &\le \langle \nabla f_k(x_k), x_k - y_k\rangle + \gamma\big(\langle \nabla f_k(x_k), x_k - x^*\rangle - \frac{\mu}{2}\|x_k - x^*\|_2^2\big) \\ &\quad - \frac{1+\gamma}{2L}\|\nabla f_k(x_k)\|_2^2. \end{aligned} \tag{39}$$

Let us employ Lemma E.5 to get rid of references to y_k. We have

$$\begin{aligned} z_k &= \big(\frac{1}{\tau} - 1\big)(x_k - y_k) + x_k = \sqrt{\kappa}(x_k - y_k) + x_k \\ \gamma(z_k - x^*) &= \sqrt{\kappa}\gamma(x_k - y_k) + \gamma(x_k - x^*). \end{aligned}$$

Note that $\sqrt{\kappa}\gamma = 1 + \gamma$. We have

$$(x_k - y_k) + \gamma(x_k - x^*) = \frac{1}{1+\gamma} \cdot \Big[\gamma(z_k - x^*) + \gamma^2(x_k - x^*)\Big].$$

After that, we rewrite the expression on the right hand side of (39) as follows:

$$\begin{aligned}&\frac{1}{1+\gamma}\langle \nabla f_k(x_k), \gamma(z_k - x^*) + \gamma^2(x_k - x^*)\rangle - \\ &\frac{\mu\gamma}{2}\|x_k - x^*\|_2^2 - \frac{1+\gamma}{2L}\|\nabla f_k(x_k)\|_2^2. \end{aligned} \tag{40}$$

We bound the second term in (38) similarly to (Bansal & Gupta, 2019). By Lemma E.5:

$$
\begin{aligned}
&\frac{\mu}{2}\Big[(1+\gamma)\|z_{k+1}-x^*\|_2^2-\|z_k-x^*\|_2^2\Big]\\
&\quad=\frac{\mu}{2}(1+\gamma)\Big\|\frac{1}{1+\gamma}(z_k-x^*)+\frac{\gamma}{1+\gamma}(x_k-x^*)-\frac{\gamma}{\mu(1+\gamma)}\nabla f_k(x_k)\Big\|_2^2-\frac{\mu}{2}\|z_k-x^*\|_2^2\\
&\quad=\frac{\mu}{2}\frac{1}{1+\gamma}\Big[\|z_k-x^*\|_2^2+\gamma^2\|x_k-x^*\|_2^2+\frac{\gamma^2}{\mu^2}\|\nabla f_k(x_k)\|_2^2\\
&\qquad+2\gamma\langle z_k-x^*,x_k-x^*\rangle-\frac{2\gamma}{\mu}\langle z_k-x^*,\nabla f_k(x_k)\rangle\\
&\qquad-\frac{2\gamma^2}{\mu}\langle x_k-x^*,\nabla f_k(x_k)\rangle\Big]-\frac{\mu}{2}\|z_k-x^*\|_2^2. \qquad (41)
\end{aligned}
$$

Adding (40) yields a bound on $\Delta\Psi_k$. Moreover, note that terms involving $\langle\nabla f_k(x_k),x_k-x^*\rangle$ and $\langle\nabla f_k(x_k),z_k-x^*\rangle$ cancel out.

$$
\begin{aligned}
&\Delta\Psi_k(1+\gamma)^{-k}\\
&\quad\le\left(-\frac{1+\gamma}{2L}+\frac{\gamma^2}{2\mu(1+\gamma)}\right)\|\nabla f_k(x_k)\|_2^2\\
&\qquad+\frac{\mu\gamma}{2}\left(\frac{\gamma}{1+\gamma}-1\right)\|x_k-x^*\|_2^2+\frac{\mu}{2}\left(\frac{1}{1+\gamma}-1\right)\|z_k-x^*\|_2^2\\
&\qquad+\frac{\mu\gamma}{1+\gamma}\langle z_k-x^*,x_k-x^*\rangle\\
&\quad\le-\frac{\mu\gamma}{2(1+\gamma)}\big(\|x_k-x^*\|_2^2+\|z_k-x^*\|_2^2-2\langle z_k-x^*,x_k-x^*\rangle\big)\\
&\quad=-\frac{\mu\gamma}{2(1+\gamma)}\|(x_k-x^*)-(z_k-x^*)\|_2^2\le 0,
\end{aligned}
$$

and the proof is complete. □

Now we can prove Theorem E.4 using the potentials technique.

Proof of Theorem E.4. Following the definition of Ψ_k and using the Lemma E.6, we obtain

$$
\begin{aligned}
(1+\gamma)^N(f_N(y_N)-f^*)\le\Psi_N\le\Psi_0\le\frac{(L+\mu)R^2}{2},\\
f_N(y_N)-f^*\le\frac{(L+\mu)R^2}{2(1+\gamma)^N}=\frac{(L+\mu)R^2}{2}(1-1/\sqrt{\kappa})^N.
\end{aligned}
$$

□

F. Missing Proofs from Section 4

F.1. Proof of Theorem 4.3

Statement 1 follows directly from Algorithm 1. To see why statement 2 holds it is sufficient to note that $\operatorname{range}\mathbf{W}^k=\mathcal{L}^\top$. For statement 3, denote $\overline{\mathbf{x}}=\frac{1}{m}\mathbf{1}\mathbf{1}^\top\otimes\mathbf{I}$ let us apply Theorem E.4 to see that

$$
\left\|\mathbf{x}^T-\overline{\mathbf{x}}\right\|_2^2\le 2\chi\left\|\mathbf{x}^0-\overline{\mathbf{x}}\right\|_2^2(1-1/\sqrt{\chi})^T.
$$

Taking $T = \sqrt{\chi}\log(4\chi)$ we obtain $\left\|\mathbf{x}^T - \overline{\mathbf{x}}\right\|_2^2 \leq 1/2\left\|\mathbf{x}^0 - \overline{\mathbf{x}}\right\|_2^2$. If $\mathbf{x} \in \mathcal{L}^\top$, then $\overline{\mathbf{x}} = 0$ and following the definition $C_T(\mathbf{x}) = \mathbf{x} - \mathbf{x}^T$, we write

$$\begin{aligned}\|C_T(\mathbf{x})\|_2 = \left\|\mathbf{x} - \mathbf{x}^T\right\|_2 &\leq \|\mathbf{x} - \overline{\mathbf{x}}\|_2 + \left\|\mathbf{x}^T - \overline{\mathbf{x}}\right\|_2 \\ &\leq (1 + 1/\sqrt{2})\left\|\mathbf{x} - \overline{\mathbf{x}}\right\|_2, \\ \|C_T(\mathbf{x})\|_2 = \left\|\mathbf{x} - \mathbf{x}^T\right\|_2 &\geq \|\mathbf{x} - \overline{\mathbf{x}}\|_2 - \left\|\mathbf{x}^T - \overline{\mathbf{x}}\right\|_2 \\ &\geq (1 - 1/\sqrt{2})\left\|\mathbf{x} - \overline{\mathbf{x}}\right\|_2\end{aligned}$$

F.2. Proof of Lemma 4.5

The proof directly follows from Lemma E.6 applied to problem (3).