Fast Online Node Labeling for Very Large Graphs

Baojian Zhou [1 2] **Yifan Sun** [3] **Reza Babanezhad** [4]

Abstract

This paper studies the online node classification problem under a transductive learning setting. Current methods either invert a graph kernel matrix with $\mathcal{O}(n^3)$ runtime and $\mathcal{O}(n^2)$ space complexity or sample a large volume of random spanning trees, thus are difficult to scale to large graphs. In this work, we propose an improvement based on the *online relaxation* technique introduced by a series of works (Rakhlin et al., 2012; Rakhlin & Sridharan, 2015; 2017). We first prove an effective regret $\mathcal{O}(\sqrt{n^{1+\gamma}})$ when suitable parameterized graph kernels are chosen, then propose an approximate algorithm FastONL enjoying $\mathcal{O}(k\sqrt{n^{1+\gamma}})$ regret based on this relaxation. The key of FastONL is a *generalized local push* method that effectively approximates inverse matrix columns and applies to a series of popular kernels. Furthermore, the per-prediction cost is $\mathcal{O}(\mathrm{vol}\,(\mathcal{S})\log 1/\epsilon)$ locally dependent on the graph with linear memory cost. Experiments show that our scalable method enjoys a better tradeoff between local and global consistency.

1. Introduction

This paper explores the online node labeling problem within a transductive learning framework. Specifically, we consider the scenario where multi-category node labels $\boldsymbol{y}_t$ enter online, and our goal is to predict future labels $\boldsymbol{y}_{t+1}$ under constraints imposed by an underlying graph. To illustrate this, consider the example of online product recommendations on platforms such as Amazon. At each time step t, the task is to recommend one of k products to a user (node) based on their relationships (edges) with other users. The success of these recommendations is gauged by whether the user proceeds to purchase the recommended product. In this context, leveraging local information, such as recommendations from friends, becomes a viable strategy. However, this presents a significant challenge as we need to generate per-iteration predictions within microseconds while managing large-scale graphs comprising millions or even billions of nodes. Due to their computational complexity, traditional methods that require solving linear systems of order $\mathcal{O}\left(n^3\right)$ are ill-suited for this task where n is the number of nodes. This issue is not only critical in web-spam classification (Herbster et al., 2008b), product recommendation (Ying et al., 2018), and community detection (Leung et al., 2009), but also permeates many other graph-based applications.

Previous studies in this area have fallen into two categories. The first involves sampling the spanning tree from the graph and then predicting labels via a weighted majority-like method (Herbster et al., 2008b; Cesa-Bianchi et al., 2009; 2013). Such methods require repeated spanning tree samplings and often suffer a large variance issue. The second family of methods designs a loss function and builds feature vectors from different graph kernels (Herbster et al., 2005; Herbster & Pontil, 2006; Herbster & Lever, 2009; Herbster et al., 2015; Gentile et al., 2013; Rakhlin & Sridharan, 2015; 2016b; 2017). These kernel-based methods can successfully capture the label smoothness of graphs but need to invert the graph kernel matrices, severely limiting their scalability.

Several classical and modern graph representation learning works (Zhu et al., 2003; Blum et al., 2004; Kipf & Welling, 2017) suggest that graph kernel-based methods are more effective in real-world applications. We focus on a significant line of work based on *online relaxation* (Rakhlin & Sridharan, 2015; 2016a;b; 2017), which achieves linear time per-iteration cost. However, two challenges remain. First, the choice of the used graph kernel matrix affects performance, and it is unclear how to make this choice optimally and obtain an effective regret. Second, the previous relaxation method (Rakhlin & Sridharan, 2017) assumes that the inverse of the graph kernel matrix is readily available, which is not reasonable in practice. Note that a vanilla approach to computing a matrix inverse of a graph kernel involves $\mathcal{O}(n^3)$ computational cost and $\mathcal{O}(n^2)$ space complexity. Approximate matrix inverse techniques are required, but the regret guarantee for utilizing such schemes does not yet exist. The key question, then, is whether there exists an online node

[1]School of Data Science, Fudan University, Shanghai, China [2]the Shanghai Key Laboratory of Data Science, Fudan University [3]Department of Computer Science, Stony Brook University, Stony Brook, USA [4]Samsung-SAIT AI lab, Montreal, Canada. Correspondence to: Baojian Zhou <bjzhou@fudan.edu.cn>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

labeling method which *accounts for the kernel matrix inversion*, where the per-iteration cost is independent of the whole graph and the overall method is nearly-linear time.

In this paper, we propose such a solution by extending the online relaxation method (Rakhlin & Sridharan, 2015; 2016b; 2017; 2016a) via a fast local matrix inverse approximation method. Specifically, the inversion technique is based on the Approximate PageRank (APPR) method (Andersen et al., 2006), which is particularly effective and efficient when the magnitudes of these kernel vectors follow a power-law distribution, often found in real-world graphs. Our proposed Fast Online Node Labeling algorithm FASTONL approximates the kernel matrix inverse via variants of APPR. Moreover, we compute an effective regret bound of $\mathcal{O}(k\sqrt{n^{1+\gamma}})$, which accounts for the matrix inversion steps. While we focus on static graphs, the method can naturally be extended to the dynamic graph setting.

Our contributions.

- For the first time, we show that online relaxation-based methods with suitable graph kernel parameterization enjoy an effective regret when the graph is highly structured; specifically, the regret can be bounded by $\mathcal{O}(\sqrt{n^{1+\gamma}})$ if the graph Laplacian is regularized by $\mathcal{O}(n^{\gamma})$ for some $\gamma \in (0, 1)$. This is generalized to several parameterized graph kernels.
- To overcome the $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ space complexity of the large matrix inversion barrier, we consider the APPR approach, which gives a per-iteration cost of $\mathcal{O}(\mathrm{vol}\,(\mathcal{S})\log(1/\epsilon))$. This locally linear bound is exponentially superior to the previous $\mathcal{O}(1/\epsilon)$ bound for general graphs (Andersen et al., 2006).
- On graphs between 1000 and 1M nodes, FASTONL shows a better empirical tradeoff between local and global consistency. For a case study on the English Wikipedia graph with 6.2M nodes and 178M edges, we obtain a low error rate with a per-prediction run time of less than a second.

Our code and datasets have been provided as supplementary material and are publicly available at `https://github.com/baojian/FastONL`. All proofs have been postponed to the appendix.

2. Related Work

Online node labeling. Even binary labeling of graph nodes in the online learning setting can be challenging. A series of works on online learning over graphs is considered (Herbster et al., 2005; Herbster, 2008; Herbster et al., 2008b; Herbster & Lever, 2009; Herbster & Robinson, 2019; Herbster et al., 2021). Initially, Herbster et al. (2005) considered learning graph node labels using a perceptron-based algorithm, which iteratively projected a sequence of points over a closed convex set. This initial method already requires finding the pseudoinverse of the unnormalized Laplacian matrix. Moreover, the total mistakes is bounded by $4\Phi_{\mathcal{G}}(\boldsymbol{y})D_{\mathcal{G}}\,\mathrm{bal}(\boldsymbol{y})$ where $\Phi_{\mathcal{G}}$ is the graph cut, $D_{\mathcal{G}}$ is the diameter, and $\mathrm{bal}(\boldsymbol{y})$ is the label balance ratio. This mistake bound, which is distinct from the regret bound in this paper, vanishes when the label is imbalanced. Subsequent works, such as PUNCE (Herbster, 2008) and SEMINORM (Herbster & Lever, 2009), also admitted mistake bounds. To remedy this issue, the following works (Herbster & Pontil, 2006; Herbster et al., 2008a; Herbster, 2008) proposed different methods to avoid these large bounds. However, to the best of our knowledge, their effectiveness has not been validated on large-scale graphs. Additionally, it is unclear whether these methods can be effective under multi-category label settings.

The algorithms proposed in Herbster et al. (2008a); Herbster & Lever (2009); Cesa-Bianchi et al. (2009); Vitale et al. (2011); Cesa-Bianchi et al. (2013) accelerate per-prediction by working on trees and paths of the graph; see also (Gentile et al., 2013) for evolving graphs. However, the total time complexity of the proposed method is quadratic w.r.t the graph size. Additionally, Herbster et al. (2015) considered the setting of predicting a switching sequence over multiple graphs, and Gu & Han (2014) explored an online spectral learning framework. All these works fundamentally depend on the inverse of the graph Laplacian.

More generally, the problem of transductive learning on graphs has been extensively studied over past years (Ng et al., 2001; Zhou et al., 2003; Zhu et al., 2003; Ando & Zhang, 2006; Johnson & Zhang, 2007; El Alaoui et al., 2016; Kipf & Welling, 2017). Under batch transductive learning setting, the basic assumption is that nodes with same labels are well-clustered together. In this case, the quadratic form of the graph Laplacian kernel (2) or even p-Laplacian-based (El Alaoui et al., 2016; Fu et al., 2022) should be small. However, different from batch settings, this paper considers online learning settings based on kernel computations.

Personalized PageRank and approximation. Personalized PageRank (PPR) as an important graph learning tool has been used in classic graph applications (Jeh & Widom, 2003; Andersen et al., 2008) and modern graph neural networks (Gasteiger et al., 2019; Bojchevski et al., 2020; Epasto et al., 2022) due its scalable approach to matrix inversion. The *local push* method has been proposed in a seminal work of Andersen et al. (2006) as an efficient and localized approach toward computing PPR vectors; it was later shown to be a variant of coordinate descent (Fountoulakis et al., 2019), and related to Gauss-Seidel iteration (Sun & Ye,

2021). This paper introduces a new variant of the *local push* to approximate many other graph kernel inverses.

3. Preliminaries

This section introduces notation and problem setup and presents the online relaxation method with surrogate loss.

3.1. Notations and problem formulation

Notations. We consider an undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ where $\mathcal{V} \triangleq \{1, 2, \ldots, n\}$ is the set of nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of $m = |\mathcal{E}|$ edges, and $\boldsymbol{W} \in \mathbb{R}_+^{n\times n}$ is the nonnegative weighted adjacency matrix where each edge $(u, v) \in \mathcal{E}$ has weight $W_{uv} > 0$. The unnormalized and normalized graph Laplacian is defined as $\mathcal{L} \triangleq \boldsymbol{D} - \boldsymbol{W}$ and $\boldsymbol{L} \triangleq \boldsymbol{D}^{-1/2}\mathcal{L}\boldsymbol{D}^{-1/2}$, respectively.[1] The set of neighbors of u is denoted as $\mathcal{N}(u) \triangleq \{v : (u, v) \in \mathcal{E}\}$ and the degree $d_u = |\mathcal{N}(u)|$. The weighted degree matrix is defined as a diagonal matrix $\boldsymbol{D}$ where $D_{uu} = \sum_{v\in\mathcal{N}(u)} W_{uv}$.[2] Following the work of Chung (1997), for $\mathcal{S} \subseteq \mathcal{V}$, the volume of $\mathcal{S}$ is defined as $\mathrm{vol}\,(\mathcal{S}) \triangleq \sum_{v\in\mathcal{S}} d_v$.

Given k labels, each node v has a label $y_v \in \{1, 2, \ldots, k, \}$. For convenience, we use the binary form $\boldsymbol{y}_t \in \{\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_k\} =: \mathcal{Y}$ where $\boldsymbol{e}_i$ is the one-hot encoding vector. $\boldsymbol{X}_{:,i} \in \mathbb{R}^n$ is the i-th column vector of matrix $\boldsymbol{X} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{X}_{i,:} \in \mathbb{R}^n$ is the transpose of i-th row vector of $\boldsymbol{X}$. The support of $\boldsymbol{x} \in \mathbb{R}^n$ is $\mathrm{supp}\,(\boldsymbol{x}) \triangleq \{v : x_v \neq 0, v \in \mathcal{V}\}$. The trace of a square matrix $\boldsymbol{M}$ is defined as $\mathrm{tr}\,(\boldsymbol{M}) = \sum_{i=1}^n m_{ii}$ where m_{ii} is the i-th diagonal. For a symmetric matrix $\boldsymbol{M}$, denote $\lambda(\boldsymbol{M})$ as the eigenvalue function of $\boldsymbol{M}$.

Problem formulation. This paper considers the following online learning paradigm on $\mathcal{G}$: At each time $t = 1, 2, \ldots, n$, a learner picks a node v and makes a prediction $\hat{\boldsymbol{y}}_v \in \mathcal{Y}$. The true label $\boldsymbol{y}_v$ is revealed by the adversary with a corresponding 0-1 loss $\ell(\hat{\boldsymbol{y}}_v, \boldsymbol{y}_v) = 1 - \boldsymbol{y}_v^\top \hat{\boldsymbol{y}}_v$ back to the learner. The goal is to design an algorithm, so the learner makes as few mistakes as possible. Denote a prediction of $\mathcal{V}$ as $\widehat{\boldsymbol{Y}} = [\hat{\boldsymbol{y}}_1, \hat{\boldsymbol{y}}_2, \ldots, \hat{\boldsymbol{y}}_n] \in \mathcal{F}$ and true label configuration as $\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n] \in \mathcal{F}$ from the allowed label configurations $\mathcal{F} \in \left\{\boldsymbol{F} \in \{0, 1\}^{k\times n} : \boldsymbol{F}_{:,j}^\top \cdot \boldsymbol{1} = 1, \forall j \in \mathcal{V}\right\}$. Without further restrictions, the adversary could always select a $\boldsymbol{Y}$ so that the learner makes the maximum (n) mistakes by always providing $\boldsymbol{y}_v \neq \hat{\boldsymbol{y}}_v$. Therefore, to have learnability, often the set $\mathcal{F}$ is restricted to capture *label smoothness* (Blum & Mitchell, 1998; Blum & Chawla, 2001; Zhu et al., 2003; Zhou et al., 2003; Blum et al., 2004). Formally, given an algorithm $\mathcal{A}$, the learner's goal is to minimize the regret defined as

$$\underset{\widehat{\boldsymbol{Y}}\sim\mathcal{A}}{\mathrm{Reg}} := \sum_{t=1}^n \ell(\hat{\boldsymbol{y}}_t, \boldsymbol{y}_t) - \min_{\boldsymbol{F}\in\mathcal{F}_{\lambda,\beta}} \sum_{t=1}^n \ell(\boldsymbol{F}_{:,t}, \boldsymbol{y}_t), \tag{1}$$

where

$$\mathcal{F}_{\lambda,\beta} = \left\{\boldsymbol{F} \in \mathcal{F} : \sum_{i=1}^k \boldsymbol{F}_{i,:}{}^\top \boldsymbol{K}_\beta^{-1} \boldsymbol{F}_{i,:} \leq \lambda\right\}, \tag{2}$$

and $\boldsymbol{K}_\beta$ is a positive definite kernel parameterized by β, and λ is a label smoothing parameter controlling the range of allowed label configurations. For example, assume $\boldsymbol{K}_\beta^{-1} = \boldsymbol{D} - \boldsymbol{W}$ for a unit weight graph; then if $\lambda = 1$, $\mathbf{y}_i = \mathbf{y}_j$ whenever $(i, j) \in \mathcal{E}$ for all $\mathbf{Y} \in \mathcal{F}$; clearly in this case the labeling is learnable. On the other hand, if $\lambda > n$, then $\mathbf{Y}$ can be any labeling and is not learnable. If ℓ is convex with a closed convex set $\mathcal{F}$, typical online convex optimization methods such as online gradient descent or Follow-The-Regularized-Leader could provide sublinear regret (Shalev-Shwartz et al., 2012; Hazan et al., 2016) for minimizing the regret (1). However, when ℓ is the 0-1 loss, the combinatorial nature of $\mathcal{F}$ makes directly applying these methods difficult. Inspired by Rakhlin & Sridharan (2017), we propose the following convex relaxation of $\mathcal{F}_{\lambda,\beta}$ to

$$\bar{\mathcal{F}}_{\lambda,\beta} = \left\{\boldsymbol{F} \in \mathbb{R}^{k\times n} : \sum_{i=1}^k \boldsymbol{F}_{i,:}^\top \boldsymbol{M}_{\lambda,\beta} \boldsymbol{F}_{i,:} \leq 1\right\},$$

where $\mathcal{F}_{\lambda,\beta} \subseteq \bar{\mathcal{F}}_{\lambda,\beta}$ and the regularized kernel matrix is

$$\boldsymbol{M}_{\lambda,\beta} = \left(\frac{\boldsymbol{K}_\beta^{-1}}{2\lambda} + \frac{\boldsymbol{I}_n}{2n}\right)^{-1}. \tag{3}$$

3.2. Online relaxation and surrogate loss

Algorithm 1 RELAXATION($\mathcal{G}, \lambda, D$)(Rakhlin & Sridharan)

Compute $\boldsymbol{M} = \boldsymbol{M}_{\lambda,\beta}$
$T_1 = \mathrm{tr}\,(\boldsymbol{M})\,, A_1 = 0, \boldsymbol{G} = [\boldsymbol{0}, \ldots, \boldsymbol{0}] \in \mathbb{R}^{k\times n}$
for $t = 1, \ldots, n$ **do**
  $\boldsymbol{\psi}_t = -\boldsymbol{G}\boldsymbol{M}_{:,t}/\sqrt{A_t + D^2 \cdot T_t}$
  Predict $\hat{\boldsymbol{y}}_t \sim \boldsymbol{q}_t(\boldsymbol{\psi}_t)$, $\boldsymbol{\nabla}_t = \boldsymbol{\nabla}\phi_{\boldsymbol{\psi}_t}(\cdot, \boldsymbol{y})$
  Update $\boldsymbol{G}_{:,t} = \boldsymbol{\nabla}_t$
  $A_{t+1} = A_t + 2\boldsymbol{\nabla}_t^\top \boldsymbol{G}\boldsymbol{M}_{:,t} + m_{tt} \cdot \|\boldsymbol{\nabla}_t\|_2^2$
  $T_{t+1} = T_t - m_{tt}$
end for

In the *online relaxation framework* (Alg.1), a key step of prediction node t is to choose a suitable $\boldsymbol{\psi}_t$ strategy so that the regret defined in (1) can be bounded. Specifically, the prediction $\hat{\boldsymbol{y}}_t$ is randomly generated according to distribution $\boldsymbol{q}_t(\boldsymbol{\psi}_t)$ where the score $\boldsymbol{\psi}_t \in \mathbb{R}^k$ is a scaling of $-\sum_{i<t} \boldsymbol{\nabla}_i M_{i,t}$, computed in an online fashion. The distribution, $q_i = \max(\psi_i - \tau, 0)$ for the choice of τ such that

[1] When $\mathcal{G}$ contains singletons, $\boldsymbol{D}^{-1/2} = \left(\boldsymbol{D}^+\right)^{1/2}$ where $\boldsymbol{D}^+$ is the Moore-Penrose inverse of $\boldsymbol{D}$.

[2] Note that $d_u = D_u$ only when $\mathcal{G}$ is unweighted but $d_u \neq D_{uu}$ for a weighted graph in general.

Table 1: The parameterized graph kernel matrices with their basic kernel presentation

ID	$\boldsymbol{K}_\beta^{-1}$	α	Basic Kernel Presentation	Paper
1	$\mathcal{L}$	$\frac{\lambda}{n}$	$\boldsymbol{M}_{\lambda,\beta} = 2\lambda \boldsymbol{X}_{\mathcal{L}}$	(Rakhlin & Sridharan, 2017)
2	$\boldsymbol{L}$	$\frac{\lambda}{n+\lambda}$	$\boldsymbol{M}_{\lambda,\beta} = 2n\boldsymbol{D}^{-1/2}\boldsymbol{X}_{\boldsymbol{L}}\boldsymbol{D}^{1/2}$	(Rakhlin & Sridharan, 2017)
3	$\boldsymbol{I} - \beta\boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$	$\frac{n+\lambda-\beta n}{n+\lambda}$	$\boldsymbol{M}_{\lambda,\beta} = \frac{2\lambda n}{n+\lambda-\beta n}\boldsymbol{D}^{-1/2}\boldsymbol{X}_{\boldsymbol{L}}\boldsymbol{D}^{1/2}$	(Zhou et al., 2003)
4	$\beta\boldsymbol{I} + \boldsymbol{S}^{-1/2}\mathcal{L}\boldsymbol{S}^{-1/2}$	$\frac{n\beta+\lambda}{n}$	$\boldsymbol{M}_{\lambda,\beta} = 2\lambda\boldsymbol{S}^{-1/2}\boldsymbol{X}_{\mathcal{L}}\boldsymbol{S}^{1/2}$	(Johnson & Zhang, 2008)
5	$\boldsymbol{S}^{-1/2}(\beta\boldsymbol{I} + \mathcal{L})\boldsymbol{S}^{-1/2}$	2λ	$\boldsymbol{M}_{\lambda,\beta} = \left(\frac{\boldsymbol{S}^{1/2}}{4n\lambda} + \frac{\beta\boldsymbol{S}^{-1/2}}{4\lambda^2}\right)^{-1}\boldsymbol{X}_{\mathcal{L}}\boldsymbol{S}^{1/2}$	(Johnson & Zhang, 2007)
6	$\mathcal{L} + b\cdot\mathbf{1}\mathbf{1}^\top + \beta\boldsymbol{I}$	$\beta + \frac{\lambda}{n}$	$\boldsymbol{M}_{\lambda,\beta} = 2\lambda\boldsymbol{X}_{\mathcal{L}}\left(\boldsymbol{I} - \frac{b\mathbf{1}\mathbf{1}^\top}{\alpha+nb}\right)$	(Herbster et al., 2005)

$\sum_{i=1}^k q_i = 1$. This technique corresponds to minimizing the surrogate convex loss [3]

$$\phi_{\boldsymbol{\psi}}(\boldsymbol{g},\boldsymbol{y}) = \begin{cases} \frac{1+\max_{r:\boldsymbol{e}_r\neq\boldsymbol{y}}\{\boldsymbol{g}^\top\boldsymbol{e}_r - \boldsymbol{g}^\top\boldsymbol{y}\}}{1+1/|S(\boldsymbol{\psi})|} & \boldsymbol{y}\notin S(\boldsymbol{\psi}) \\ 1 - \boldsymbol{g}^\top\boldsymbol{y} + \frac{\boldsymbol{g}^\top\mathbf{1}_{S(\boldsymbol{\psi})} - 1}{|S(\boldsymbol{\psi})|} & \boldsymbol{y}\in S(\boldsymbol{\psi}) \end{cases} \quad (4)$$

where $S(\boldsymbol{\psi})$ is the support of $\boldsymbol{\psi}$, and $\boldsymbol{\nabla}_t$ is the gradient of $\phi(\cdot,\boldsymbol{y})$ of the first variable. Specifically, $y_t \notin S(\boldsymbol{\psi})$ means the learner receives loss $\phi_{\boldsymbol{\psi}}(\boldsymbol{g},\boldsymbol{y}) \geq 1$. Note that the per-iteration cost of Alg.1 is $\mathcal{O}(kn)$ once the $\boldsymbol{\psi}_t$ is computed.

We now define an *admissible relaxation function*.

Definition 3.1 (Admissible function (Rakhlin et al., 2012))**.** Let $\boldsymbol{\nabla}_i \in \mathbb{R}^k$, $\|\boldsymbol{\nabla}_i\|_2 \leq D$ for some $D > 0$. A real-valued function $\mathrm{Rel}(\boldsymbol{\nabla}_{1:t})$ is said to be admissible if, for all $t \in \mathcal{V}$, it satisfies recursive inequalities

$$\inf_{\boldsymbol{\psi}_t\in\mathbb{R}^k}\sup_{\|\boldsymbol{\nabla}_t\|_2\leq D}\left\{\boldsymbol{\nabla}_t^\top\boldsymbol{\psi}_t + \mathrm{Rel}(\boldsymbol{\nabla}_{1:t})\right\} \leq \mathrm{Rel}(\boldsymbol{\nabla}_{1:t-1}),$$

with $\mathrm{Rel}(\boldsymbol{\nabla}_{1:n}) \geq -\inf_{\boldsymbol{F}\in\mathcal{F}_{\lambda,\beta}}\sum_{t=1}^n \boldsymbol{\nabla}_t^\top\boldsymbol{F}_{:,t}$.

It was shown in Rakhlin et al. (2012) (and later (Rakhlin & Sridharan, 2015; 2016b; 2017)) that if there exists an admissible function Rel for some $\boldsymbol{\psi}_t$, then the regret of Alg.1 is upper bounded by $\mathrm{Rel}(\emptyset) = \sqrt{D\cdot\mathrm{tr}(\boldsymbol{M})}$, providing an upper bound of the regret. Here $\boldsymbol{M}$ is either $(\frac{\mathcal{L}}{2\lambda} + \frac{\boldsymbol{I}}{2n})^{-1}$ or $(\frac{\boldsymbol{L}}{2\lambda} + \frac{\boldsymbol{I}}{2n})^{-1}$ for the binary case. Note that in both cases, since $\lambda_{\max}(\boldsymbol{M}) \leq n$, then in the worst case, the regret could be $\mathcal{O}(n)$ (vanishing in general). Thus, two questions remain.

1. Does there exist $\boldsymbol{K}_\beta^{-1}$ that not only captures label smoothness but also has regret *smaller* than $\mathcal{O}(n)$?
2. How do we reconcile the kernel computation overhead $\mathcal{O}(n^3)$ but still provide an effective regret bound?

These two main problems motivate us to study this online relaxation framework further. Sec. 4 answers the second question by showing that solving many popular kernel matrices is equivalent to solving two basic kernel matrices, and we explore local approximate methods for both. We then answer the first question in Sec. 5 by proving effective bounds when the parameterized kernel matrix is computed exactly or approximated.

[3]The method could naturally apply to other types of losses (See more candidate losses in Johnson & Zhang (2007)).

4. Local approximation of kernel $\boldsymbol{M}_{\lambda,\beta}$

Section 4.1 presents how popular kernels can be evaluated from simple transformations of the inverse approximations computed via FIFOPUSH, whose convergence is described in Section 4.2.

4.1. Basic kernel presentations of $\boldsymbol{M}_{\lambda,\beta}$

The regularized kernel matrix is defined in (3) for various instances of $\boldsymbol{K}_\beta^{-1}$ as listed in Tab. 1. As shown in the table, a key observation is that several existing online labeling methods involve the inverse of two basic kernel forms. We present this in Thm. 4.1.

Theorem 4.1. *Let $\boldsymbol{K}_\beta^{-1}$ be the inverse of the symmetric positive definite kernel matrix defined in Tab. 1. Then $\boldsymbol{M}_{\lambda,\beta}$ can be decomposed into $\boldsymbol{M}_{\lambda,\beta} = a\boldsymbol{A}^{-1}\boldsymbol{X}\boldsymbol{B}$, which is easily computed once $\boldsymbol{X}$ available. $\boldsymbol{X}$ represents two basic kernels*

$$\boldsymbol{X}_{\mathcal{L}} = (\alpha\boldsymbol{I} + \mathcal{L})^{-1}, \quad \boldsymbol{X}_{\boldsymbol{L}} = \alpha\left(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1}$$

corresponding to the inverse of variant matrices of $\mathcal{L}$ and $\boldsymbol{L}$, respectively.[4]

In Tab. 1, the column "Basic Kernel Presentation" shows how $\boldsymbol{M}_{\lambda,\beta}$ can then be efficiently computed from either $\boldsymbol{X}_{\boldsymbol{L}}$ or $\boldsymbol{X}_{\mathcal{L}}$, using minimal post-processing overhead. As in the online relaxation framework, for any time t of node v_t, it requires to access the v_t-th column of $\boldsymbol{M}_{\lambda,\beta}$. Therefore, we need to solve the following two basic equations

$$\text{Type-}\mathcal{L}: \quad \boldsymbol{x}_{v_t} = \boldsymbol{X}_{\mathcal{L}}\boldsymbol{e}_{v_t}, \quad (5)$$

$$\text{Type-}\boldsymbol{L}: \quad \boldsymbol{x}_{v_t} = \boldsymbol{X}_{\boldsymbol{L}}\boldsymbol{e}_{v_t}. \quad (6)$$

[4]Note $\boldsymbol{L} = \alpha\boldsymbol{D}^{1/2}(\boldsymbol{I} - (1-\alpha)\boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2})^{-1}\boldsymbol{D}^{-1/2}$.

For the second case, note $\boldsymbol{x}_{v_t} = \boldsymbol{X_L}\boldsymbol{e}_{v_t}$ gives the Personalized PageRank Vector (PPV) (Page et al., 1999; Jeh & Widom, 2003). For example, using $\alpha = \frac{n}{n+\kappa}$, we compute $\boldsymbol{M}_{\lambda,\beta} = 2n\boldsymbol{D}^{-1/2}\boldsymbol{X_L}\boldsymbol{D}^{1/2}$ where $\boldsymbol{X_L}$ is the Personalized PageRank matrix (See the second row of Tab. 1). We now discuss the inversion for computing $\boldsymbol{X_L}$ and $\boldsymbol{X}_{\mathcal{L}}$.

Algorithm 2 FIFOPUSH$(\mathcal{G}, \epsilon, \alpha, s)$

$\boldsymbol{r} = \begin{cases} \text{Type-}\mathcal{L}: \frac{\boldsymbol{e}_s}{\alpha} \\ \text{Type-}\boldsymbol{L}: \boldsymbol{e}_s \end{cases}$
$\boldsymbol{x} = \boldsymbol{0}, \mathcal{Q} = [s]$
while $\mathcal{Q} \neq \emptyset$ **do**
  $u = \mathcal{Q}$.pop()
  if $r_u < \epsilon \cdot d_u$ **then**
    continue
  end if
  $x_u = \begin{cases} \text{Type-}\mathcal{L}: x_u + \frac{\alpha r_u}{\alpha + D_u} \\ \text{Type-}\boldsymbol{L}: x_u + \alpha r_u \end{cases}$
  for $v \in \mathcal{N}(u)$ **do**
    $r_v = \begin{cases} \text{Type-}\mathcal{L}: r_v + \frac{r_u}{\alpha + D_u} \cdot w_{uv} \\ \text{Type-}\boldsymbol{L}: r_v + \frac{(1-\alpha) r_u}{D_u} \cdot w_{uv} \end{cases}$
    if $v \notin \mathcal{Q}$ **then**
      $\mathcal{Q}$.push(v)
    end if
  end for
  $r_u = 0$
end while
Return $\boldsymbol{x}, \boldsymbol{r}$

Before introducing the FIFOPUSH inversion method, let us consider the more commonly used power iteration for matrix inversion. $\boldsymbol{M}_{\lambda,\beta}$ can be approximated by a series of matrix multiplications. Take $\boldsymbol{K}_\beta^{-1} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$ as an example. Then a truncated power iteration gives

$$\boldsymbol{M}_{\lambda,\beta} \approx \frac{2n\lambda}{n+\lambda} \sum_{i=0}^{p} \left(\frac{n}{n+\lambda} \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2} \right)^i \tag{7}$$

assuming that $\|\frac{n}{n+\lambda}\boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}\|_2 < 1$ (See Lemma 2.3.3 of Golub & Van Loan (2013)). When p is small, this method often produces a reasonable and efficient approximation of $\boldsymbol{M}_{\lambda,\beta}$, and is especially efficient if $\boldsymbol{W}$ is sparse. However, as p gets large, the intermediate iterates quickly become dense matrices, creating challenges for online learning algorithms where the per-kernel vector operator is preferred (See also Fig. 15 on real-world graphs). Thus, the situation is even worse when we only need to access one column at each time t under the online learning setting.

For these reasons, we introduce FIFOPUSH (Alg. 2), which reduces to the well-known APPR algorithm (Andersen et al., 2006) when the goal is to approximate $\boldsymbol{X_L}$. Specifically, it

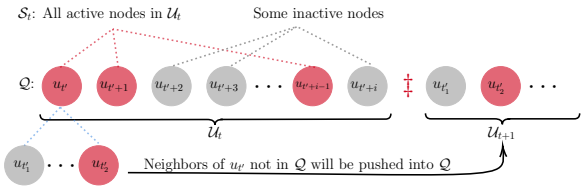


Figure 1: The illustration of t-th epoch of FIFOPUSH. At the initial of t-th epoch, $\mathcal{Q} = [u_{t'}, \ldots, u_{t'+i}]$ contains all active nodes (red) $\mathcal{S}_t$ and part of inactive nodes (gray). After the first active node $u_{t'}$ has been processed, the neighbors of $u_{t'}$ that are not in $\mathcal{Q}$ will be pushed into $\mathcal{Q}$ for next $(t+1)$-th epoch. ‡ is a dummy node to mark the end of an epoch.

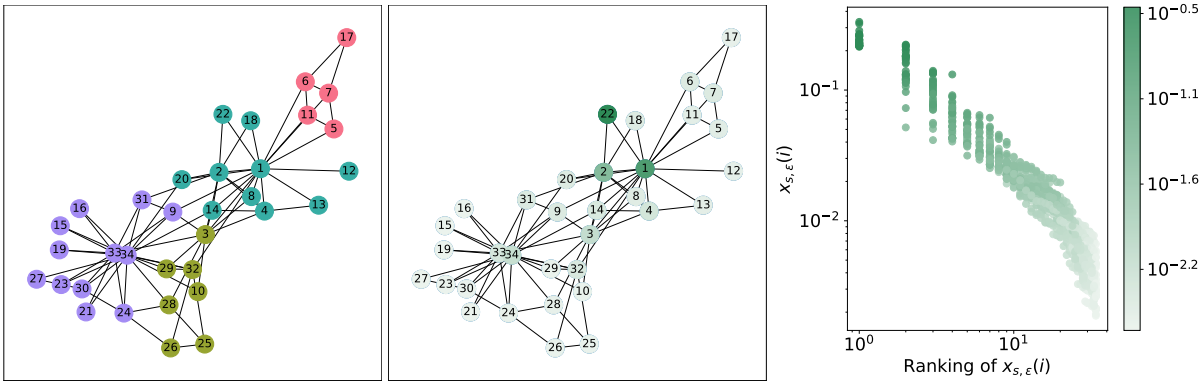
is a *local push* method for solving either (5) or (6) based on First-In-First-Out (FIFO) queue. Each node $u_t \in \mathcal{S}_t$ is either *active*, i.e., $r_{u_t} \geq \epsilon d_{u_t}$, or *inactive* otherwise. As illustrated in Fig. 1, at a higher level, it maintains a set of nonzero residual nodes $\mathcal{U}_t$ and active nodes $\mathcal{S}_t \subseteq \mathcal{U}_t$ in each epoch t. FIFOPUSH updates the solution column $\boldsymbol{x}$ and residual $\boldsymbol{r}$ (corresponding to the "gradient") by shifting mass from a high residual node (an active node) to its neighboring nodes in $\boldsymbol{x}$ and $\boldsymbol{r}$. This continues until all residual nodes are smaller than a tolerance ϵ. This method essentially retains the linear invariant property introduced in Appendix A.2.

4.2. Local linear convergence of FIFOPUSH

For calculating $\boldsymbol{X_L}$, Andersen et al. showed that FIFOPUSH gives a time complexity $\mathcal{O}(\frac{1}{\alpha\epsilon})$ for precision level $\epsilon > 0$. [5] This bound is *local sublinear*, meaning the bound is locally dependent of $\mathcal{G}$ and sublinear to the precision ϵ. Moreover, the rate's independence on $\mathcal{G}$ is a key advantage of FIFOPUSH over other numerical methods such as Power Iteration, which is similar to FIFOPUSH (Wu et al., 2021) when $\epsilon < \mathcal{O}(m^{-1})$ (recall $m = |\mathcal{E}|$). Specifically, the Power Iteration typically needs $\mathcal{O}(\frac{m}{\alpha}\log\frac{1}{\epsilon})$ operations. However, when m is large, and ϵ is very small, the advantage of the local sublinear bound is lost, and the time complexity bound is not optimal.

It is natural to ask whether any method achieves a locally dependent and logarithmically related bound to ϵ. We answer this question positively. Specifically, we notice that in most real-world sparse graphs, the columns of $\boldsymbol{X}$ have magnitudes following a power law distribution (See Karate graph in Fig. 2, real-world graphs shown in Fig. 17 and 18 for $\boldsymbol{X_L}$ and $\boldsymbol{X}_{\mathcal{L}}$, respectively.), suggesting that local approximations are sufficient in computing high fidelity approximate inverses. Note that this greatly improves computational complexity and preserves memory locality, reducing graph access time for large-scale graphs.

[5] This algorithm was also proposed in Berkhin (2006), namely Book Coloring algorithm.



(a) Labeled Karate (b) $\boldsymbol{x}_{22,\epsilon}$ on $\mathcal{G}$ (c) Power law of $\boldsymbol{X}_\epsilon$

Figure 2: Power law distribution of $|(\boldsymbol{X_L}\boldsymbol{e}_s)_i|$ on Karate graph (Girvan & Newman, 2002). (a) The Karate contains 34 nodes and 78 edges forming four communities (w_{uv} is the 2-dimensional Euclidean distance.); (b) We run FIFOPUSH$(\mathcal{G}, \epsilon, \alpha, s)$ on $s = 22$ with $\alpha = 0.2$ and $\epsilon = 10^{-12}$. Neighbors of $s = 22$ have large magnitudes of $\boldsymbol{x}_s$; (c) The power law distribution of entries of $\boldsymbol{x}_s$ for all 34 nodes.

We now provide our *local linear* and *graph independent* complexity bound. Denote the estimation matrix $\boldsymbol{X}_\epsilon = [\boldsymbol{x}_{1,\epsilon}, \ldots, \boldsymbol{x}_{n,\epsilon}]$ and the residual matrix $\boldsymbol{R}_\epsilon = [\boldsymbol{r}_{1,\epsilon}, \ldots, \boldsymbol{r}_{n,\epsilon}]$ where $(\boldsymbol{x}_{s,\epsilon}, \boldsymbol{r}_{s,\epsilon}) = \text{FIFOPUSH}(\mathcal{G}, \epsilon, \alpha, s)$ for all $s \in \mathcal{V}$. Denote $\mathcal{I}_t = \text{supp}(\boldsymbol{r}_t)$ as the support of residual after t-th epoch, and $\mathcal{S}_t$ as the set of *active* nodes.

Theorem 4.2 (Local linear convergence of FIFOPUSH for $\boldsymbol{X_L}$)**.** *Let $\boldsymbol{x}_s = \boldsymbol{X_L}\boldsymbol{e}_s$. Denote T as the total epochs executed by* FIFOPUSH*, and $\mathcal{S}_t := \{v : r_t(v) \geq \epsilon \cdot d_v, v \in \mathcal{I}_t\}$ as the set of active nodes in t-th epoch. Then, the total operations of* FIFOPUSH$(\mathcal{G}, \epsilon, \alpha, s)$ *is dominated by*

$$R_T := \sum_{t=1}^{T} \sum_{u_t \in \mathcal{S}_t} d_{u_t} \leq \frac{\text{vol}(\mathcal{S}_{1:T})}{\alpha \cdot \eta_{1:T}} \log\left(\frac{C_{\alpha,T}}{\epsilon}\right), \quad (8)$$

where $\text{vol}(\mathcal{S}_{1:T}) = \sum_{t=1}^{T} \text{vol}(\mathcal{S}_t)/T$ is the average volume of $\mathcal{S}_t$. Additionally, $\eta_{1:T} = \sum_{t=1}^{T} \eta_t/T$ is the average of local convergence factors $\eta_t \triangleq \sum_{u \in \mathcal{S}_t} d_u / \sum_{v \in \mathcal{I}_t} d_v$, and $C_{\alpha,T} = 1/(\sum_{v \in \mathcal{I}_T}(1-\alpha) d_u w_{uv}/D_u)$. For $s, i \in \mathcal{V}$, we have $\boldsymbol{x}_s = \boldsymbol{x}_{s,\epsilon} + \boldsymbol{X_L}\boldsymbol{r}_{s,\epsilon}, r_{s,\epsilon}(i) \leq [0, \epsilon d_i)$.

Thm. 4.2 provides a *local linear convergence* of FIFOPUSH where both $\text{vol}(\mathcal{S}_{1:T})$ and $\eta_{1:T}$ are locally dependent on $\mathcal{G}$, α, and ϵ. For unweighted $\mathcal{G}$, the bound in (8) can be simplified as $\frac{\text{vol}(\mathcal{S}_{1:T})}{\alpha \cdot \eta_{1:T}} \log \frac{1}{\epsilon(1-\alpha)|\mathcal{I}_T|}$. The key of Thm. 4.2 is to evaluate $\text{vol}(\mathcal{S}_{1:T})$ and $\eta_{1:T}$. To estimate $\text{vol}(\mathcal{S}_{1:T})$, since each active node appears at most T times and at least once in all T epochs, after FIFOPUSH terminates, we have

$$\frac{\text{vol}(\text{supp}(\boldsymbol{x}_{s,\epsilon}))}{T} \leq \text{vol}(\mathcal{S}_{1:T}) \leq \text{vol}(\text{supp}(\boldsymbol{x}_{s,\epsilon})),$$

where for α and ϵ such that $\mathcal{O}(|\text{supp}(\boldsymbol{x}_{s,\epsilon})|) \ll n$ means $\mathcal{O}(\text{vol}(\text{supp}(\boldsymbol{x}_{s,\epsilon}))) \ll m$ in expectation. More importantly, compared with $\mathcal{O}(1/\alpha\epsilon)$ of Andersen et al. (2006), Thm. 4.2 provides a better bound when $\epsilon \leq \mathcal{O}(m^{-1})$. The work of Fountoulakis et al. (2019) shows APPR is equivalent to the coordinate descent method, and the total time complexity is comparable to $\tilde{\mathcal{O}}(\frac{1}{\alpha\epsilon})$.

The average linear convergence factor $\eta_{1:T}$ is always > 0 by noticing that at least one active node is processed in each epoch. We empirically found that $\eta_{1:T} \geq 0.3$ in all testing cases as shown in Appendix A.2. The above theorem is a refinement of (Wu et al., 2021) where $\mathcal{O}(m \log \frac{1}{\epsilon})$ is obtained (only effective when $\epsilon < 1/2m$). However, our proof shows that obtaining m independent bound is possible by showing that local magnitudes are reduced by a constant factor. The above theorem gives a way to approximate $\boldsymbol{M}_{\lambda,\beta}$, and we will build an approximate online algorithm based on FIFOPUSH. We close this section by introducing our *local linear convergence* for $\boldsymbol{X}_\mathcal{L}\boldsymbol{e}_s$ as the following.

Theorem 4.3 (Local convergence of FIFOPUSH for $\boldsymbol{X}_\mathcal{L}$)**.** *Let $\boldsymbol{x}_s = \boldsymbol{X}_\mathcal{L}\boldsymbol{e}_s$ and run Alg. 2 for $\boldsymbol{X}_\mathcal{L}$. For $s, i \in \mathcal{V}$, we have $\boldsymbol{x}_s = \boldsymbol{x}_{s,\epsilon} + \alpha\boldsymbol{X}_\mathcal{L}\boldsymbol{r}_{s,\epsilon}$, with $r_{s,\epsilon}(i) \leq [0, \epsilon d_i), \forall i \in \mathcal{V}$. The main operations of* FIFOPUSH *for $\boldsymbol{X}_\mathcal{L}$ is bounded as*

$$R_T \leq \frac{\text{vol}(\mathcal{S}_{1:T})(\alpha + D_{\max})}{\alpha \cdot \eta_{1:T}} \log\left(\frac{C_{\alpha,T}}{\epsilon}\right), \quad (9)$$

where $\text{vol}(\mathcal{S}_{1:T})$ and $\eta_{1:T}$ are the same as in Thm. 4.2, $\eta_t \triangleq \frac{\sum_{u \in \mathcal{S}_t} d_u/(\alpha + D_u)}{\sum_{v \in \mathcal{I}_t} d_v/(\alpha + D_v)}$, $C_{\alpha,T} = 1/\sum_{v \in \mathcal{I}_T} \frac{d_u w_{uv}}{\alpha + D_u}$, and $D_{\max} = \max_{v \in \text{supp}(\boldsymbol{x}_{s,\epsilon})} D_v$.

Remark 4.4. We obtain a similar *local linear convergence* for solving $\boldsymbol{X}_\mathcal{L}$ by FIFOPUSH. The additional factor $(\alpha + D_{\max})$ appears in Equ. (9) due to the upper bound of the maximal eigenvalue of $\mathcal{L}$.

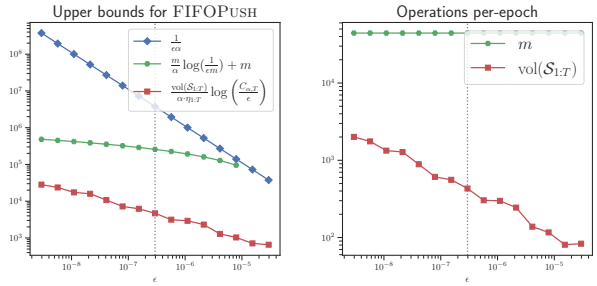


Figure 3: The bounds comparison of R_T. To see if there is a true advantage of our bound, we compare two bounds of FIFOPUSH, $\mathcal{O}(\frac{1}{\alpha\epsilon})$ (Andersen et al., 2006) and $\mathcal{O}(m\frac{m}{\alpha}\log\frac{1}{\epsilon m}+m)$ (Wu et al., 2021) with Ours. Each vertical line with its left part is when ϵ satisfies Cor. 5.4

5. Fast Online Node Labelling: FASTONL

This section shows how we can obtain a meaningful regret using parameterized graph kernels for the original online relaxation method. We then design approximated methods based on FIFOPUSH.

5.1. Regret analysis for online relaxation method

As previously discussed, simply applying $\boldsymbol{M}_{\lambda,\beta}^{-1}=\mathcal{L}$ or $\boldsymbol{L}$ will not yield a meaningful regret as the maximal eigenvalue of $\boldsymbol{M}_{\lambda,\beta}$ depends on the minimal eigenvalue of $\boldsymbol{M}_{\lambda,\beta}^{-1}$. In particular, for a connected graph $\mathcal{G}$, the second smallest eigenvalue of $\boldsymbol{L}$ is lower bounded by $\mathrm{diam}(\mathcal{G})\geq\frac{4}{\lambda_2 n}$ (Mohar, 1991), and is tight for a chain graph; this yields a $\mathcal{O}(n^2)$ bound which is non-optimal. Instead, our key idea of producing a method with improved bounds is to "normalize" the kernel matrix $\boldsymbol{K}_\beta^{-1}$ so that $\mathrm{tr}\left(\boldsymbol{M}_{\lambda,\beta}\right)\ll\mathcal{O}(n^2)$, yielding a more meaningful bound. We state the regret bound as in the following theorem.

Theorem 5.1 (Regret of RELAXATION with parameterized $\boldsymbol{M}_\beta^{-1}$)**.** *Let $\hat{\boldsymbol{Y}}$ be the prediction matrix returned by* RELAXATION*, if the true label sequences $\boldsymbol{Y}\in\mathcal{F}_{\lambda,\beta}$ with parameter $\lambda=n^\gamma$ and $\gamma\in(0,1)$. Then choosing $\beta=n^{\gamma-1}$ for kernel $\boldsymbol{K}_\beta^{-1}=\boldsymbol{I}-\beta\boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$ and $\beta=1-\frac{\lambda}{n}$ for $\boldsymbol{K}_\beta^{-1}=\beta\boldsymbol{I}+\boldsymbol{S}^{-1/2}\mathcal{L}\boldsymbol{S}^{-1/2}$, we have the following regret*

$$\mathrm{Reg}=\mathop{\mathbb{E}}_{\hat{\boldsymbol{Y}}\sim\mathcal{A}}\sum_{t=1}^{n}\ell(\hat{\boldsymbol{y}}_t,\boldsymbol{y}_t)-\frac{2k-1}{k}\min_{\boldsymbol{F}\in\mathcal{F}_{\lambda,\beta}}\sum_{t=1}^{n}\ell(\boldsymbol{f}_t,\boldsymbol{y}_t),$$

which is bounded i.e., $\mathrm{Reg}\leq D\sqrt{2n^{1+\gamma}}$. [6]

Remark 5.2. The constant D involved in the bound is the assumption of the bounded gradient of $\boldsymbol{\nabla}_t$, which is always ≤ 2 for the loss chosen in (4). The above Thm. 5.1 is an improvement upon the regret given in Rakhlin & Sridharan (2017) of $\mathcal{O}(n)$. Note that this rate does not take into account the run time $\mathcal{O}(n^3)$ required to invert $\boldsymbol{M}_{\lambda,\beta}$ in RELAXATION. In the rest, we give the regret of FASTONL, which implements RELAXATION using FIFOPUSH, and show that the regret is still small.

[6]Note that, for binary case, $k=2$, Reg exactly recover $\mathbb{E}_{\hat{\boldsymbol{Y}}\sim\mathcal{A}}$ Reg for binary case defined in Equ. (1).

5.2. Fast approximation algorithm FASTONL

We describe the approximated method, FASTONL in Alg. 3 as follows, and recall that $m_{tt}=(M_\epsilon)_{t:t}$.

Algorithm 3 FASTONL($\mathcal{G},\epsilon,\boldsymbol{K}_\beta^{-1},\lambda$)

$\boldsymbol{G}=[\boldsymbol{0},\boldsymbol{0},\ldots,\boldsymbol{0}]\in\mathbb{R}^{k\times n}$
$A_1=0$
$\boldsymbol{M}_\epsilon$ is obtained via FIFOPUSH($\mathcal{G},\epsilon,\alpha,s$) $\forall s\in\mathcal{V}$
$T_1=\sum_{t=1}^n m_{t,t}$
for $t=1,2,\ldots,n$ **do**
  $\boldsymbol{v}=\boldsymbol{G}(\boldsymbol{M}_\epsilon)_{:,t}+\boldsymbol{G}(\boldsymbol{M}_\epsilon)_{t,:}$
  $\boldsymbol{\psi}_t=-\boldsymbol{v}/\sqrt{A_t+k\cdot T_t}$
  Update gradient $\boldsymbol{G}_{:,t}=\boldsymbol{\nabla}_t=\boldsymbol{\nabla}\phi(\cdot,\boldsymbol{y})$
  $A_{t+1}=A_t+\boldsymbol{\nabla}_t^\top\boldsymbol{v}+m_{tt}\cdot\|\boldsymbol{\nabla}_t\|_2^2$
  $T_{t+1}=T_t-m_{tt}$
end for

Theorem 5.3 (Regret analysis of FASTONL with approximated parameterized kernel)**.** *Consider the similar residual matrix $\tilde{\boldsymbol{R}}_\epsilon=\boldsymbol{D}^{-1/2}\boldsymbol{R}_\epsilon\boldsymbol{D}^{1/2}$. Given $\lambda=n^\gamma$ for $\gamma\in(0,1)$, picking ϵ so that $\|\tilde{\boldsymbol{R}}_\epsilon\|_2\leq\frac{1}{\alpha}$ yields*

$$\mathrm{Reg}\leq D\sqrt{(1+k^2)n^{1+\gamma}},$$

where the restriction on ϵ is due to maintaining the positive semidefiniteness of $\left(\boldsymbol{M}_\epsilon+\boldsymbol{M}_\epsilon^\top\right)/2$.

Based on Thm. 5.3, we have the following runtime requirement for FASTONL.

Corollary 5.4 (Per-iteration complexity of FIFOPUSH)**.** *Based on the conditions of Thm. 5.3, the number of operations required in one iteration of* FASTONL *is bounded by*

$$\mathcal{O}\left(\frac{\mathcal{S}_{1:T}}{\alpha\cdot\eta_{1:T}}\log^{3/2}(n)\right). \tag{10}$$

Fig. 3 illustrates the advantage of our local bound by plotting all constants for the PubMed graph (Similar trends are observed in other graphs in Appendix A.2). In practice, we observe that $\epsilon\sim\mathcal{O}(n^{-1})\gg\mathcal{O}(n^{-3/2})$, given from a pessimistic estimation of $\|\boldsymbol{D}^{-1/2}\boldsymbol{R}_\epsilon\boldsymbol{D}^{1/2}\|_2$. In particular, we notice a significant improvement of our bound over the previous ones when α is large.

Practical implementation. A caveat of approximate inversion in FASTONL is that $\boldsymbol{M}_\epsilon$ is not in general symmetric; therefore, for analysis, we require $\boldsymbol{\psi}_t$ to be computed using the symmetrized $\boldsymbol{M}_\epsilon[:,t]+\boldsymbol{M}_\epsilon[t,:]$, which requires row

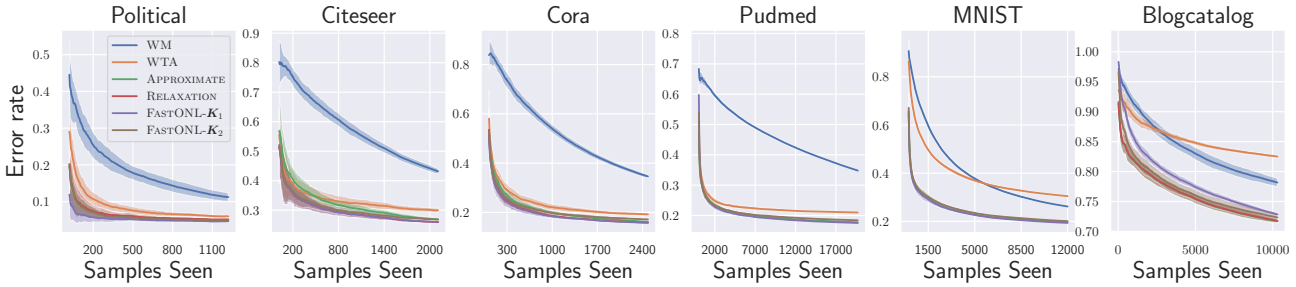


Figure 4: Error rate as the function of samples seen on six small-scale graphs.

and column access at time t; effectively, this requires that $\boldsymbol{M}_\epsilon$ is fully pre-computed. While this does not affect our overall bounds, the memory requirements may be burdensome. However, when $\boldsymbol{M}_\epsilon \approx \boldsymbol{M}$ (which *is* symmetric), then $(\boldsymbol{M}_\epsilon)_{:,t} \approx (\boldsymbol{M}_\epsilon)_{t,:}$ and in practice, we use the column to represent the row; our experiments show that this does not incur noticeable performance drop. To avoid pre-computing diagonal elements of $\boldsymbol{M}_\epsilon$, we estimate $\sum_{t=1}^{n} m_{tt} \approx kn^2$; experiments show this works well in practice.

Dynamic setting. An extension of our current setting is the dynamic setting, in which newly labeled nodes and their edges are dynamically added or deleted. As is, FASTONL is well-suited to this extension; the key idea is to use an efficient method to keep updating FIFOPUSH, which can quickly keep track of these kernel vectors (Zhang et al., 2016, e.g.). The regret analysis of the dynamic setting is more challenging, and we will consider it as future work.

6. Experiments

In this section, we conduct extensive experiments on the online node classification for different-sized graphs and compare FASTONL with baselines. We address the following questions: *1) Do these parameterized kernels work and capture label smoothness?; 2) How does* FASTONL *compare in terms of classification accuracy and run time with baselines?*

Experimental setup. We collect ten graph datasets where nodes have true labels (Tab. 4) and create one large-scale Wikipedia graph where chronologically-order node labels are from ten categories of English Wikipedia articles. We consider four baselines, including 1) Weighted Majority (WM), where we predict each node u by its previously labeled neighbors (a purely local but strong baseline described in Appendix D); 2) RELAXATION (Rakhlin & Sridharan, 2017), a globally guaranteed method; 3) Weighted Tree Algorithm (WTA) (Cesa-Bianchi et al., 2013), a representative method based on sampling random spanning trees;[7] and 4) APPROXIMATE, the power iteration-based method defined by Equ. (7). We implemented these baselines using Python. For FASTONL, we chose the first two kernels defined in Tab. 1 and named them as FASTONL-$\boldsymbol{K}_1$ and FASTONL-$\boldsymbol{K}_2$, respectively. All experimental setups, including parameter tuning, are further discussed in Appendix D.

[7] We note that the performance of WTA is competitive to, sometimes outperforms PERCEPTRON-based methods (Herbster et al., 2005).

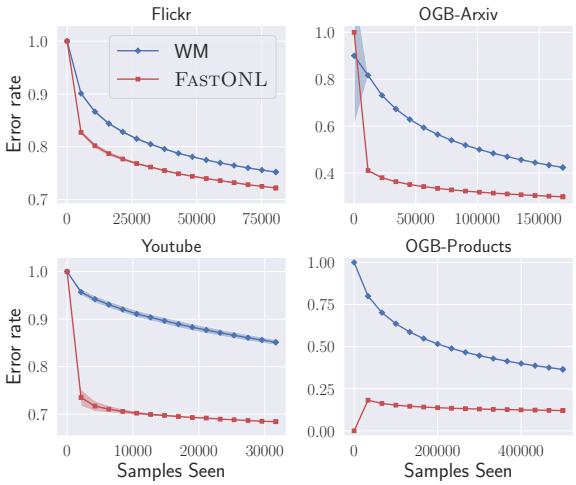


Figure 5: The comparison of error rate of FASTONL and WM on middle-scale graphs.

Online node labeling performance. The online labeling error rates over 10 trials on small-scale graphs are presented in Fig. 4 where we pick $\epsilon = 10^{-5}$. As we can see from Fig. 4, the approximated performance is almost the same as of RELAXATION but with great improvements on runtime as shown in Tab. 2. The WM can be treated as a strong baseline. For middle and large-scale graphs, matrix inversion is infeasible, and these baselines are unavailable. We compare FASTONL with the local method WM. Fig. 5 and 7 present the error rate as a function of the number of seen nodes. FASTONL outperforms the local WM by a large margin. This indicates that FASTONL has a better tradeoff between local and global consistency. The average run time of these methods is presented in Tab. 2. The local method WM has the lowest run time per iteration. Our method is between RELAXATION and WTA.

Table 2: Run time of online node labeling methods over six graphs (seconds) averaged over 10 trials.

	Political	Citeseer	Cora	Pubmed	MNIST	Blogcatalog
WM	0.01	0.01	0.01	0.08	0.09	0.05
WTA	66.61	146.97	213.00	2177.49	10726.67	5108.45
APPROXIMATE	1.47	0.66	0.97	159.48	43.83	68.52
RELAXATION	0.78	1.66	2.94	122.45	976.69	154.32
FASTONL-$\boldsymbol{K}_1$	1.12	1.10	1.73	4.86	22.42	22.14
FASTONL-$\boldsymbol{K}_2$	1.21	1.12	2.57	7.27	33.00	12.03

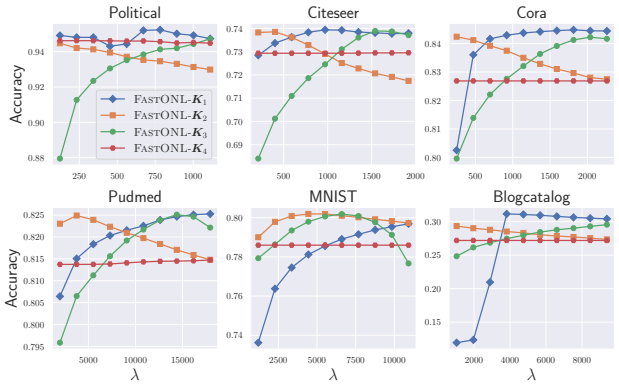


Figure 6: The accuracy of applying the first four kernel matrices for FASTONL on six small graphs.

Performance of parameterized kernels. In our theory, we showed what the effect of parameters (λ, β) is on the regret (see Thm.5.1). The parameter λ is a label smoothing parameter controlling the range of allowed label configurations while β is the kernel parameter. We tested the first four kernels where kernels $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$ solely depend on λ, while kernels $\boldsymbol{K}_3$ and $\boldsymbol{K}_4$ involve both λ and β. However, for $\boldsymbol{K}_3, \beta$ is defined as λ/n, and for $\boldsymbol{K}_4$, it is defined as $\beta = 1 - \lambda/n$, as established in Thm.5.1. By defining β this way, our theorem ensures an effective regret. We experimented with various values of λ, selecting from $0.1 \cdot n, 0.2 \cdot n, \ldots, 0.9 \cdot n, n$. Fig. 6 shows how different kernels perform over different graphs. All of the kernels successfully captured label smoothing but exhibited differing performances with varying λ. We consider the first four kernels as listed in Tab. 1, sweeping λ. To answer our first question, we find that *all kernels can capture the label smoothing well but perform differently with different* λ. Overall, the normalized kernel of $\boldsymbol{K}_2$ enjoys a large range of λ, while $\boldsymbol{K}_1$ and $\boldsymbol{K}_3$ tend to prefer big λ.

Case study of labeling Wikipedia articles. We apply our method to a real-world Wikipedia graph, which contains 6,216,199 nodes where corresponding labels appear chronically and unweighted 177,862,656 edges (edges are hyperlinks among these Wikipedia articles). Each node may have a label (downloaded from DBpedia (2021), about 50% percentage of nodes have labels, we use the first 150,000 labeled nodes) belonging to ten categories describing the Wikipedia articles, such as people, places, etc. Fig. 7 presents our results on this large-scale graph. Compared with the strong baseline WM, our FASTONL truly outperforms it by a large margin with only about 0.3 seconds for each article.

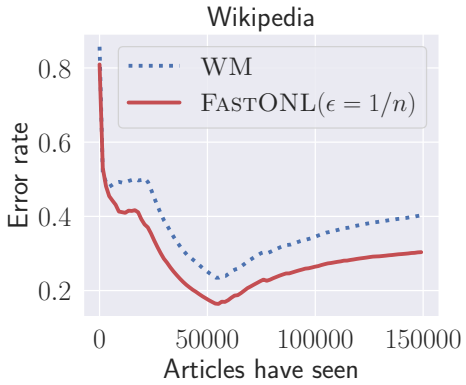


Figure 7: The comparison of error rate of FASTONL and WM on one large-scale graph.

7. Conclusion

We study the online relaxation framework for the node labeling problem. We propose, for the first time, a fast approximate method that yields effective online regret bounds, filling a significant gap in the theoretical analysis of this online learning problem. We then design a general FIFOPUSH algorithm to quickly compute an approximate column of the kernel matrix in an online fashion that does not require large local memory storage. Therefore, the actual computational complexity per-iteration is truly local and competitive to other baseline methods. The local analysis of FIFOPUSH is challenging when the acceleration is added to the algorithm. It is interesting to see if there is any local analysis for accelerated algorithms (See the open question (Fountoulakis & Yang, 2022)). It is also interesting to see whether our work can be extended to directed or dynamic graph settings.

8. Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments. The work of Baojian Zhou is sponsored by Shanghai Pujiang Program (#22PJ1401300).

References

Adamic, L. A. and Glance, N. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pp. 36–43, 2005.

Andersen, R., Chung, F., and Lang, K. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pp. 475–486. IEEE, 2006.

Andersen, R., Borgs, C., Chayes, J., Hopcroft, J., Jain, K., Mirrokni, V., and Teng, S. Robust pagerank and locally computable spam detection features. In *Proceedings of the 4th international workshop on Adversarial information retrieval on the web*, pp. 69–76, 2008.

Anderson Jr, W. N. and Morley, T. D. Eigenvalues of the laplacian of a graph. *Linear and multilinear algebra*, 18 (2):141–145, 1985.

Ando, R. and Zhang, T. Learning on graph with laplacian regularization. *Advances in neural information processing systems*, 19, 2006.

Berkhin, P. Bookmark-coloring algorithm for personalized pagerank computing. *Internet Mathematics*, 3(1):41–62, 2006.

Blum, A. and Chawla, S. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pp. 19–26, 2001.

Blum, A. and Mitchell, T. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 92–100, 1998.

Blum, A., Lafferty, J., Rwebangira, M. R., and Reddy, R. Semi-supervised learning using randomized mincuts. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 13, 2004.

Bojchevski, A., Klicpera, J., Perozzi, B., Kapoor, A., Blais, M., Rózemberczki, B., Lukasik, M., and Günnemann, S. Scaling graph neural networks with approximate pagerank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2464–2473, 2020.

Cesa-Bianchi, N., Gentile, C., and Vitale, F. Fast and optimal prediction on a labeled tree. In *Annual Conference on Learning Theory*, pp. 145–156. Omnipress, 2009.

Cesa-Bianchi, N., Gentile, C., Vitale, F., and Zappella, G. Random spanning trees and the prediction of weighted graphs. *Journal of Machine Learning Research*, 14(1): 1251–1284, 2013.

Chung, F. R. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

DBpedia. The DBpedia ontology. https://www.dbpedia.org/resources/ontology/, 2021. [Online; accessed 25-Jan-2023].

El Alaoui, A., Cheng, X., Ramdas, A., Wainwright, M. J., and Jordan, M. I. Asymptotic behavior of\ell_p-based laplacian regularization in semi-supervised learning. In *Conference on Learning Theory*, pp. 879–906. PMLR, 2016.

Epasto, A., Mirrokni, V., Perozzi, B., Tsitsulin, A., and Zhong, P. Differentially private graph learning via sensitivity-bounded personalized pagerank. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*, 2022.

Fountoulakis, K. and Yang, S. Open problem: Running time complexity of accelerated ℓ_1-regularized pagerank. In *Conference on Learning Theory*, pp. 5630–5632. PMLR, 2022.

Fountoulakis, K., Roosta-Khorasani, F., Shun, J., Cheng, X., and Mahoney, M. W. Variational perspective on local graph clustering. *Mathematical Programming*, 174(1): 553–573, 2019.

Fu, G., Zhao, P., and Bian, Y. p-laplacian based graph neural networks. In *International Conference on Machine Learning*, pp. 6878–6917. PMLR, 2022.

Gasteiger, J., Bojchevski, A., and Günnemann, S. Combining neural networks with personalized pagerank for classification on graphs. In *International Conference on Learning Representations*, 2019.

Gentile, C., Herbster, M., and Pasteris, S. Online similarity prediction of networked data from known and unknown graphs. In *Conference on Learning Theory*, pp. 662–695. PMLR, 2013.

Girvan, M. and Newman, M. E. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

Golub, G. H. and Van Loan, C. F. *Matrix computations*. JHU press, 2013.

Gu, Q. and Han, J. Online spectral learning on a graph with bandit feedback. In *2014 IEEE International Conference on Data Mining*, pp. 833–838. IEEE, 2014.

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

Herbster, M. Exploiting cluster-structure to predict the labeling of a graph. In *International Conference on Algorithmic Learning Theory*, pp. 54–69. Springer, 2008.

Herbster, M. and Lever, G. Predicting the labelling of a graph via minimum p-seminorm interpolation. In *NIPS Workshop 2010: Networks Across Disciplines: Theory and Applications*, 2009.

Herbster, M. and Pontil, M. Prediction on a graph with a perceptron. *Advances in neural information processing systems*, 19, 2006.

Herbster, M. and Robinson, J. Online prediction of switching graph labelings with cluster specialists. *Advances in Neural Information Processing Systems*, 32, 2019.

Herbster, M., Pontil, M., and Wainer, L. Online learning over graphs. In *Proceedings of the 22nd international conference on Machine learning*, pp. 305–312, 2005.

Herbster, M., Lever, G., and Pontil, M. Online prediction on large diameter graphs. *Advances in Neural Information Processing Systems*, 21, 2008a.

Herbster, M., Pontil, M., and Galeano, S. Fast prediction on a tree. *Advances in Neural Information Processing Systems*, 21, 2008b.

Herbster, M., Pasteris, S., and Pontil, M. Predicting a switching sequence of graph labelings. *J. Mach. Learn. Res.*, 16:2003–2022, 2015.

Herbster, M., Pasteris, S., Vitale, F., and Pontil, M. A gang of adversarial bandits. *Advances in Neural Information Processing Systems*, 34, 2021.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Jeh, G. and Widom, J. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*, pp. 271–279, 2003.

Johnson, R. and Zhang, T. On the effectiveness of laplacian normalization for graph semi-supervised learning. *Journal of Machine Learning Research*, 8(7), 2007.

Johnson, R. and Zhang, T. Graph-based semi-supervised learning and spectral kernel design. *IEEE Transactions on Information Theory*, 54(1):275–288, 2008.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Leung, I. X., Hui, P., Lio, P., and Crowcroft, J. Towards real-time community detection in large networks. *Physical Review E*, 79(6):066107, 2009.

Mohar, B. Eigenvalues, diameter, and mean distance in graphs. *Graphs and combinatorics*, 7(1):53–64, 1991.

Namata, G., London, B., Getoor, L., Huang, B., and Edu, U. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs*, volume 8, pp. 1, 2012.

Ng, A., Jordan, M., and Weiss, Y. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14, 2001.

Page, L., Brin, S., Motwani, R., and Winograd, T. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

Rakhlin, A. and Sridharan, K. Hierarchies of relaxations for online prediction problems with evolving constraints. In *Conference on Learning Theory*, pp. 1457–1479. PMLR, 2015.

Rakhlin, A. and Sridharan, K. Bistro: An efficient relaxation-based method for contextual bandits. In *International Conference on Machine Learning*, pp. 1977–1985. PMLR, 2016a.

Rakhlin, A. and Sridharan, K. A tutorial on online supervised learning with applications to node classification in social networks. *arXiv preprint arXiv:1608.09014*, 2016b.

Rakhlin, A. and Sridharan, K. Efficient online multiclass prediction on graphs via surrogate losses. In *Artificial Intelligence and Statistics*, pp. 1403–1411. PMLR, 2017.

Rakhlin, S., Shamir, O., and Sridharan, K. Relax and randomize: From value to algorithms. *Advances in Neural Information Processing Systems*, 25, 2012.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

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

Sun, R. and Ye, Y. Worst-case complexity of cyclic coordinate descent: $o(n^2)$ gap with randomized version. *Mathematical Programming*, 185(1):487–520, 2021.

Vitale, F., Cesa-Bianchi, N., Gentile, C., and Zappella, G. See the tree through the lines: The shazoo algorithm. *Advances in Neural Information Processing Systems*, 24, 2011.

Wilson, D. B. Generating random spanning trees more quickly than the cover time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 296–303, 1996.

Wu, H., Gan, J., Wei, Z., and Zhang, R. Unifying the global and local approaches: an efficient power iteration with forward push. In *Proceedings of the 2021 International Conference on Management of Data*, pp. 1996–2008, 2021.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 974–983, 2018.

Zhang, H., Lofgren, P., and Goel, A. Approximate personalized pagerank on dynamic graphs. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1315–1324, 2016.

Zhou, D., Bousquet, O., Lal, T., Weston, J., and Schölkopf, B. Learning with local and global consistency. *Advances in neural information processing systems*, 16, 2003.

Zhu, X., Ghahramani, Z., and Lafferty, J. D. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pp. 912–919, 2003.

The appendix is organized as follows: Section A presents all missing proofs for graph kernel matrices computation and approximation. Section B proves the bounds of eigenvalues of $\boldsymbol{M}_\epsilon$ and $\boldsymbol{R}_\epsilon$. Section C provides the regret analysis. Section D presents more experimental details and results.

A. Proofs

A.1. Graph kernel matrices and their equivalence: The proof of Thm. 4.1

Recall that our goal is to compute the following augmented kernel matrix

$$\boldsymbol{M}_{\lambda,\beta} = \left(\frac{\boldsymbol{K}_\beta^{-1}}{2\lambda} + \frac{\boldsymbol{I}_n}{2n}\right)^{-1}$$

for various instances of $\boldsymbol{K}_\beta^{-1}$ as listed in Table 1. The unnormalized Laplacian is defined as $\mathcal{L} \triangleq \boldsymbol{D} - \boldsymbol{W}$ where $\boldsymbol{W}$ is the nonnegative symmetric weighted adjacency matrix, and $\boldsymbol{D}$ is the corresponding weighted degree matrix defined as $\mathbf{diag}(\boldsymbol{W}\mathbf{1})$. The normalized graph Laplacian is $\boldsymbol{L} \triangleq \boldsymbol{D}^{-1/2}\mathcal{L}\boldsymbol{D}^{-1/2}$. Using FIFOPUSH, we can approximate the following two basic matrices

$$\text{Type-}\boldsymbol{L}: \quad \boldsymbol{X}_{\boldsymbol{L}} = \alpha\left(\boldsymbol{I}_n - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1}, \qquad \text{Type-}\mathcal{L}: \quad \boldsymbol{X}_{\mathcal{L}} = \left(\alpha\boldsymbol{I}_n + \boldsymbol{D} - \boldsymbol{W}\right)^{-1}. \tag{11}$$

We repeat our theorem in the following

Theorem 4.1. *Let $\boldsymbol{K}_\beta^{-1}$ be the inverse of the symmetric positive definite kernel matrix defined in Table 1. Then $\boldsymbol{M}_{\lambda,\beta}$ can be decomposed into $\boldsymbol{M}_{\lambda,\beta} = a\boldsymbol{A}^{-1}\boldsymbol{X}\boldsymbol{B}$, which is easily computed once $\boldsymbol{X}$ available. $\boldsymbol{X}$ represents two basic kernel inverse*

$$\boldsymbol{X}_{\mathcal{L}} = (\alpha\boldsymbol{I} + \mathcal{L})^{-1}, \quad \boldsymbol{X}_{\boldsymbol{L}} = \alpha\left(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1}$$

corresponding to inverse of regularized $\boldsymbol{L}$ and $\mathcal{L}$, respectively.

Proof. We now show how each of these kernels $\boldsymbol{K}$ can be efficiently computed given either $\boldsymbol{X}_{\boldsymbol{L}}$ or $\boldsymbol{X}_{\mathcal{L}}$ as follows:

Instance 1. For the kernel $\boldsymbol{K}_\beta^{-1} = \mathcal{L}$, we have

$$\begin{aligned}
\left(\frac{\boldsymbol{K}_\beta^{-1}}{2\lambda} + \frac{\boldsymbol{I}_n}{2n}\right)^{-1} &= \left(\frac{\mathcal{L}}{2\lambda} + \frac{\boldsymbol{I}_n}{2n}\right)^{-1} \\
&= \left(\frac{\boldsymbol{D} - \boldsymbol{W}}{2\lambda} + \frac{\boldsymbol{I}_n}{2n}\right)^{-1} \\
&= 2\lambda\left(\alpha\boldsymbol{I}_n + \boldsymbol{D} - \boldsymbol{W}\right)^{-1} \\
&= 2\lambda\boldsymbol{X}_{\mathcal{L}},
\end{aligned}$$

where we let $\alpha = \frac{\lambda}{n}$.

Instance 2. For the kernel $\boldsymbol{K}_\beta^{-1} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$, we have

$$\begin{aligned}
\left(\frac{\boldsymbol{K}_\beta^{-1}}{2\lambda} + \frac{\boldsymbol{I}_n}{2n}\right)^{-1} &= \left(\frac{\boldsymbol{I}_n - \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}}{2\lambda} + \frac{\boldsymbol{I}_n}{2n}\right)^{-1} \\
&= \left(\frac{1}{2\lambda} + \frac{1}{2n}\right)^{-1} \cdot \left(\boldsymbol{I}_n - \frac{\left(\frac{1}{2\lambda} + \frac{1}{2n}\right)^{-1}}{2\lambda}\boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}\right)^{-1} \\
&= \frac{2n\lambda}{n+\lambda}\boldsymbol{D}^{-1/2}\left(\boldsymbol{I}_n - \frac{n}{n+\lambda}\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1}\boldsymbol{D}^{1/2} \\
&= 2n\alpha\boldsymbol{D}^{-1/2}\left(\boldsymbol{I}_n - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1}\boldsymbol{D}^{1/2} \\
&= 2n\boldsymbol{D}^{-1/2}\boldsymbol{X}_{\boldsymbol{L}}\boldsymbol{D}^{1/2},
\end{aligned}$$

where $\alpha = \frac{\lambda}{n+\lambda}$.

Instance 3. The kernel $\boldsymbol{K}_\beta^{-1} = \boldsymbol{I} - \beta \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$. Different from **Instance 2**, the kernel is now parameterized. Specifically, we have

$$\begin{aligned}
\left(\frac{\boldsymbol{K}_\beta^{-1}}{2\lambda} + \frac{\boldsymbol{I}_n}{2n}\right)^{-1} &= \left(\frac{\boldsymbol{I}_n - \beta \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}}{2\lambda} + \frac{\boldsymbol{I}_n}{2n}\right)^{-1} \\
&= \left(\frac{1}{2\lambda} + \frac{1}{2n}\right)^{-1} \cdot \left(\boldsymbol{I}_n - \frac{\beta\left(\frac{1}{2\lambda} + \frac{1}{2n}\right)^{-1}}{2\lambda}\boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}\right)^{-1} \\
&= \frac{2n\lambda}{n+\lambda}\boldsymbol{D}^{-1/2}\left(\boldsymbol{I}_n - \frac{\beta n}{n+\lambda}\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1}\boldsymbol{D}^{1/2} \\
&= \frac{2n\lambda}{(n+\lambda)\alpha}\boldsymbol{D}^{-1/2}\alpha\left(\boldsymbol{I}_n - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1}\boldsymbol{D}^{1/2} = \frac{2\lambda n}{n+\lambda-n\beta}\boldsymbol{D}^{-1/2}\boldsymbol{X}_{\boldsymbol{L}}\boldsymbol{D}^{1/2},
\end{aligned}$$

where $1-\alpha = \frac{n\beta}{n+\lambda}$, and the parameter $\beta \in (0, \frac{n+\lambda}{n})$.

Instance 4. The kernel $\boldsymbol{K}_\beta^{-1} = \beta\boldsymbol{I} + \boldsymbol{S}^{-1/2}\mathcal{L}\boldsymbol{S}^{-1/2}$ was initially considered in Johnson & Zhang (2008) where $\boldsymbol{S}$ is a positive diagonal matrix. Typical examples of $\boldsymbol{S}$ could be $\boldsymbol{D}$, $\boldsymbol{I}$, etc. Note that

$$\begin{aligned}
\left(\frac{\boldsymbol{K}_\beta^{-1}}{2\lambda} + \frac{\boldsymbol{I}_n}{2n}\right)^{-1} &= \left(\frac{\beta\boldsymbol{I} + \boldsymbol{S}^{-1/2}\mathcal{L}\boldsymbol{S}^{-1/2}}{2\lambda} + \frac{\boldsymbol{I}_n}{2n}\right)^{-1} \\
&= \left(\frac{1}{b}\left(ab\boldsymbol{I} + \boldsymbol{S}^{-1/2}(\boldsymbol{D}-\boldsymbol{W})\boldsymbol{S}^{-1/2}\right)\right)^{-1} \qquad \text{// let } a = \left(\frac{\beta}{2\lambda} + \frac{1}{2n}\right), b = 2\lambda \\
&= b\boldsymbol{S}^{1/2}\left(ab\boldsymbol{S} + \boldsymbol{D} - \boldsymbol{W}\right)^{-1}\boldsymbol{S}^{1/2} \\
&= b\boldsymbol{S}^{-1/2}\left(ab\boldsymbol{I} + (\boldsymbol{D}-\boldsymbol{W})\boldsymbol{S}^{-1}\right)^{-1}\boldsymbol{S}^{1/2} \\
&= b\boldsymbol{S}^{-1/2}\left(ab\boldsymbol{I} + (\boldsymbol{D}-\boldsymbol{W})\boldsymbol{S}^{-1}\right)^{-1}\boldsymbol{S}^{1/2} \\
&= 2\lambda\boldsymbol{S}^{-1/2}\boldsymbol{X}_{\mathcal{L}}\boldsymbol{S}^{1/2},
\end{aligned}$$

where $\alpha = \frac{\beta n+\lambda}{n}$. Note $(\boldsymbol{D}-\boldsymbol{W})\boldsymbol{S}^{-1}$ is a positive semidefinite matrix as $\boldsymbol{D}-\boldsymbol{W}$ is positive semidefinite and $\boldsymbol{S}^{-1}$ is positive definite, then applying $\boldsymbol{D}' = \boldsymbol{D}\boldsymbol{S}^{-1}$ and $\boldsymbol{W}' = \boldsymbol{W}\boldsymbol{S}^{-1}$. Therefore, it is essential to solve $(\alpha\boldsymbol{I} + \boldsymbol{D}' - \boldsymbol{W}')^{-1}$ because $\boldsymbol{D}' - \boldsymbol{W}'$ has nonnegative eigenvalues and $D'_u = \sum_{v\in\mathcal{N}(u)} W'_{uv}$ for all $u \in \mathcal{V}$. Therefore, it belongs to Type-II. Here, we abuse notations where we let $\boldsymbol{D} = \boldsymbol{D}'$ and $\boldsymbol{W} = \boldsymbol{W}'$.

Instance 5. The normalized kernel matrix $\boldsymbol{K}_\beta^{-1} = \boldsymbol{S}^{-1/2}(\beta\boldsymbol{I} + \mathcal{L})\boldsymbol{S}^{-1/2}$, the $\boldsymbol{S}$ is the normalization matrix (Ando & Zhang, 2006; Johnson & Zhang, 2007). Just like **Instance 4**, we could have different choices for $\boldsymbol{S}$: $\boldsymbol{I}$, $\boldsymbol{D}$, or $(\beta\boldsymbol{I} + \mathcal{L})^{-1}$ (for the last case, the kernel is then normalized with unit diagonal), etc. Note

$$\begin{aligned}
\left(\frac{\boldsymbol{K}_\beta^{-1}}{2\lambda} + \frac{\boldsymbol{I}_n}{2n}\right)^{-1} &= \left(\frac{\boldsymbol{S}^{-1/2}(\beta\boldsymbol{I} + \mathcal{L})\boldsymbol{S}^{-1/2}}{2\lambda} + \frac{\boldsymbol{I}_n}{2n}\right)^{-1} \\
&= \boldsymbol{S}^{1/2}\left(\tilde{\boldsymbol{S}} + \frac{\mathcal{L}}{2\lambda}\right)^{-1}\boldsymbol{S}^{1/2} \qquad \text{// let } \tilde{\boldsymbol{S}} = \frac{\beta\boldsymbol{I}}{2\lambda} + \frac{\boldsymbol{S}}{2n} \\
&= 2\lambda\boldsymbol{S}^{1/2}\left(2\lambda\tilde{\boldsymbol{S}} + \mathcal{L}\right)^{-1}\boldsymbol{S}^{1/2} \\
&= 2\lambda\boldsymbol{S}^{1/2}\tilde{\boldsymbol{S}}^{-1}\left(2\lambda\boldsymbol{I} + (\boldsymbol{D}-\boldsymbol{W})\tilde{\boldsymbol{S}}^{-1}\right)^{-1}\boldsymbol{S}^{1/2},
\end{aligned}$$

where $(\boldsymbol{D}-\boldsymbol{W})\tilde{\boldsymbol{S}}^{-1}$ is a transformed version of $\mathcal{L}$. We continue to have

$$\begin{aligned}\left(\frac{\boldsymbol{K}_\beta^{-1}}{2\lambda}+\frac{\boldsymbol{I}_n}{2n}\right)^{-1} &= 2\lambda\boldsymbol{S}^{1/2}\tilde{\boldsymbol{S}}^{-1}\left(2\lambda\boldsymbol{I}+(\boldsymbol{D}-\boldsymbol{W})\tilde{\boldsymbol{S}}^{-1}\right)^{-1}\boldsymbol{S}^{1/2}\\ &= 2\lambda\boldsymbol{S}^{1/2}\left(\frac{\beta\boldsymbol{I}}{2\lambda}+\frac{\boldsymbol{S}}{2n}\right)^{-1}\left(2\lambda\boldsymbol{I}+(\boldsymbol{D}-\boldsymbol{W})\tilde{\boldsymbol{S}}^{-1}\right)^{-1}\boldsymbol{S}^{1/2}\\ &= \left(\frac{\boldsymbol{S}^{1/2}}{4n\lambda}+\frac{\beta\boldsymbol{S}^{-1/2}}{4\lambda^2}\right)^{-1}\boldsymbol{X}_{\mathcal{L}}\boldsymbol{S}^{1/2},\end{aligned}$$

where $\alpha=2\lambda$. Similar to **Instance 4**, $(\boldsymbol{D}-\boldsymbol{W})\tilde{\boldsymbol{S}}^{-1}$ is transformed Laplacian matrix. Let $\boldsymbol{D}'=\boldsymbol{D}\tilde{\boldsymbol{S}}^{-1}$ and $\boldsymbol{W}'=\boldsymbol{W}\tilde{\boldsymbol{S}}^{-1}$. We abuse of notations of $\boldsymbol{D},\boldsymbol{W}$ and let $\boldsymbol{D}=\boldsymbol{D}'$ and $\boldsymbol{W}=\boldsymbol{W}'$.

Instance 6. The augmented kernel $\boldsymbol{K}_\lambda=\mathcal{L}+b\cdot\mathbf{1}\mathbf{1}^\top+\beta\boldsymbol{I}$ (Herbster et al., 2005) can be reformulated as

$$\begin{aligned}\left(\frac{\boldsymbol{K}_\beta^{-1}}{2\lambda}+\frac{\boldsymbol{I}_n}{2n}\right)^{-1} &= \left(\frac{\mathcal{L}+b\cdot\mathbf{1}\mathbf{1}^\top+\beta\boldsymbol{I}}{2\lambda}+\frac{\boldsymbol{I}}{2n}\right)^{-1}\\ &= \left(\frac{\mathcal{L}+\beta\boldsymbol{I}}{2\lambda}+\frac{\boldsymbol{I}}{2n}+\frac{b\cdot\mathbf{1}\mathbf{1}^\top}{2\lambda}\right)^{-1}\\ &= 2\lambda\boldsymbol{X}_{\mathcal{L}}\left(\boldsymbol{I}-\frac{b\mathbf{1}\mathbf{1}^\top\boldsymbol{X}_{\mathcal{L}}}{1+b\mathbf{1}^\top\boldsymbol{X}_{\mathcal{L}}\mathbf{1}}\right),\end{aligned}$$

where $\alpha=\beta+\frac{\lambda}{n}$ and we use the similar method as in **Instance 4** but plus a rank one matrix $\mathbf{1}\mathbf{1}^\top$. The last equality is from the fact that, for any invertible matrix $\boldsymbol{X}$, by the Sherman–Morrison formula, we have

$$(\boldsymbol{X}+\mathbf{1}\mathbf{1}^\top)^{-1}=\boldsymbol{X}^{-1}-\frac{\boldsymbol{X}^{-1}\mathbf{1}\mathbf{1}^\top\boldsymbol{X}^{-1}}{1+\mathbf{1}^\top\boldsymbol{X}^{-1}\mathbf{1}}.$$

Furthermore, note the summation of each column of $(\alpha\boldsymbol{I}+\mathcal{L})^{-1}$ is a constant $1/\alpha$, then $\mathbf{1}^\top\boldsymbol{X}_{\mathcal{L}}\mathbf{1}=n/\alpha$. Then, we continue to have

$$\begin{aligned}\left(\frac{\boldsymbol{K}_\beta^{-1}}{2\lambda}+\frac{\boldsymbol{I}_n}{2n}\right)^{-1} &= 2\lambda\boldsymbol{X}_{\mathcal{L}}\left(\boldsymbol{I}-\frac{b\mathbf{1}\mathbf{1}^\top\boldsymbol{X}_{\mathcal{L}}}{1+b\mathbf{1}^\top\boldsymbol{X}_{\mathcal{L}}\mathbf{1}}\right)\\ &= 2\lambda\boldsymbol{X}_{\mathcal{L}}\left(\boldsymbol{I}-\frac{\alpha b\mathbf{1}\mathbf{1}^\top\boldsymbol{X}_{\mathcal{L}}}{\alpha+nb}\right)\\ &= 2\lambda\boldsymbol{X}_{\mathcal{L}}\left(\boldsymbol{I}-\frac{b\mathbf{1}\mathbf{1}^\top}{\alpha+nb}\right),\end{aligned}$$

where note that $\mathbf{1}\mathbf{1}^\top\boldsymbol{X}_{\mathcal{L}}=\mathbf{1}\mathbf{1}^\top/\alpha$. □

A.2. Local linear convergence of FIFOPUSH for $\alpha\left(\boldsymbol{I}-(1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1}\boldsymbol{e}_s$: The proof of Thm. 4.2

Given any $\alpha\in(0,1),\epsilon>0$, Algorithm 4 is to approximate $\alpha\left(\boldsymbol{I}-(1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1}\boldsymbol{e}_s$. Before the proof of Theorem 4.2, we provide an equivalent version of FIFOPUSH as presented in Algorithm 4 where time index t of $\boldsymbol{r},\boldsymbol{x}$ and time index t' of processed nodes u are added. Our proof is based on this equivalent version. Compared with Algorithm 2, the only difference is that we added a dummy node ‡. Still, Algorithm 4 is essentially the same as Algorithm 2. The chronological order of processed nodes $u_{t'}$ by FIFOPUSH can then be represented as the following order

$$\ddagger\ \underbrace{u_1}_{\mathcal{U}_1}\ \ddagger\ \underbrace{u_2,u_3,\dots}_{\mathcal{U}_2}\ \ddagger\ \cdots\ \ddagger\ \underbrace{u_{t'},u_{t'+1},u_{t'+2},\dots,u_{t'+i}}_{\mathcal{U}_t}\ \ddagger\ \cdots, \tag{12}$$

where $\mathcal{U}_t=\{u_{t'},u_{t'+1},\dots,u_{t'+i}\}$ is the set of nodes processed in t-th epoch. Hence, (12) defines super epochs indexed by t where $\mathcal{U}_t$ will be processed. From epoch t, new nodes will be added into $\mathcal{Q}$ for the next epoch as illustrated in

Algorithm 4 FIFOPUSH$(\mathcal{G}, \epsilon, \alpha, s)$(Andersen et al., 2006) with a dummy node ‡

Initialize: $\boldsymbol{r}_t = \boldsymbol{e}_s$, $\boldsymbol{x}_t = \boldsymbol{0}$, $t = 1$, $t' = 1$
$\mathcal{Q} = [s, ‡]$ // At the initial stage, $\mathcal{Q}$ contains s and a dummy node ‡.
while $\mathcal{Q}$.size() $\neq 1$ **do**
  $u_{t'} = \mathcal{Q}$.pop()
  if $u_{t'} ==$ ‡ **then**
    $t = t + 1$ // Nodes in $\mathcal{U}_t$ has been processed. Go to the next epoch.
    $\mathcal{Q}$.push$(u_{t'})$
    continue
  end if
  if $r_{u_{t'}} < \epsilon \cdot d_{u_{t'}}$ **then**
    $t' = t' + 1$ // $u_{t'}$ is an "*inactive*" node
    continue
  end if
  $x_{u_{t'}} = x_{u_{t'}} + \alpha r_{u_{t'}}$ // $u_{t'}$ is an "*active*" node
  for $v \in \mathcal{N}(u_{t'})$ **do**
    $r_v = r_v + \frac{(1-\alpha) r_{u_{t'}}}{D_{u_{t'}}} \cdot w_{u_{t'} v}$
    if $v \notin \mathcal{Q}$ **then**
      $\mathcal{Q}$.push(v)
    end if
  end for
  $r_{u_{t'}} = 0$
  $t' = t' + 1$
end while
Return $(\boldsymbol{x}_t, \boldsymbol{r}_t)$

Fig. 1. $\mathcal{U}_t$ contains: 1) a set of *active* nodes $\mathcal{S}_t \triangleq \left\{u_{t'} : r_{u_{t'}} \geq \epsilon \cdot d_{u_{t'}}, u_{t'} \in \mathcal{U}_t\right\}$; and 2) a set of *inactive* nodes $\left\{u_{t'} : 0 < r_{u_{t'}} < \epsilon \cdot d_{u_{t'}}, u_{t'} \in \mathcal{U}_t\right\} = \mathcal{U}_t \backslash \mathcal{S}_t$.[8] FIFOPUSH terminates only when $\mathcal{Q}$ contains the dummy node ‡.

Define $\boldsymbol{X} = \alpha\left(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1}$. Denote the estimation matrix $\boldsymbol{X}_\epsilon = [\boldsymbol{x}_{1,\epsilon}, \ldots, \boldsymbol{x}_{n,\epsilon}]$ and the residual matrix $\boldsymbol{R}_\epsilon = [\boldsymbol{r}_{1,\epsilon}, \ldots, \boldsymbol{r}_{n,\epsilon}]$ where $(\boldsymbol{x}_{s,\epsilon}, \boldsymbol{r}_{s,\epsilon}) =$ FIFOPUSH$(\mathcal{G}, \epsilon, \alpha, s)$ for all $s \in \mathcal{V}$. The next lemma shows that $\boldsymbol{X}_\epsilon$ is a good approximation of $\boldsymbol{X}$ from the bottom when ϵ is small.

Lemma A.1. *Let $\boldsymbol{X} = \alpha\left(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1}$ and denote s-th column of $\boldsymbol{X}$ as $\boldsymbol{x}_s = \alpha\left(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1}\boldsymbol{e}_s$. Let $(\boldsymbol{x}_{s,\epsilon}, \boldsymbol{r}_{s,\epsilon}) =$ FIFOPUSH$(\mathcal{G}, \epsilon, \alpha, s)$ be the pair of vectors returned by Alg. 4 where $\boldsymbol{x}_{s,\epsilon}$ is an estimate of $\boldsymbol{x}_s$ and $\boldsymbol{r}_{s,\epsilon}$ is the corresponding residual vector. Denote the estimation matrix $\boldsymbol{X}_\epsilon = [\boldsymbol{x}_{1,\epsilon}, \ldots, \boldsymbol{x}_{n,\epsilon}]$ and the residual matrix $\boldsymbol{R}_\epsilon = [\boldsymbol{r}_{1,\epsilon}, \ldots, \boldsymbol{r}_{n,\epsilon}]$ by calling* FIFOPUSH *for all $s \in \mathcal{V}$. For any $\epsilon > 0$, we have*

$$\boldsymbol{X} = \boldsymbol{X}_\epsilon + \boldsymbol{X}\boldsymbol{R}_\epsilon,$$

where $\boldsymbol{R}_\epsilon$ satisfies $\boldsymbol{0}_{n\times n} \leq \boldsymbol{R}_\epsilon < \epsilon \cdot \mathbf{diag}(d_1, \ldots, d_n) \cdot \boldsymbol{1}\boldsymbol{1}^\top$.

Proof. Let us assume $t = t'$ at the beginning of t-th epoch. During the t-th epoch, FIFOPUSH updates t to $t+1$ and updates t' from $t' = t'_0, t'_1, t'_2, \ldots$ to $t'_{|\mathcal{S}_t|} = t + 1$ where t'_i is the time after the update of node $u_{t'_i}$. Recall $\mathcal{S}_t$ is the set of processed active nodes (at the beginning of t-th epoch, we do not know how many nodes in $\mathcal{S}_t$ since some inactive nodes could be active after some push operations). For each active node $u_{t'_i}$ $(i = 1, 2, \ldots, |\mathcal{S}_t|)$ at t-th epoch, we denote $\boldsymbol{x}_{u_{t'_i}}$ and $\boldsymbol{r}_{u_{t'_i}}$ as the updated vectors of $\boldsymbol{x}_t$ and $\boldsymbol{r}_t$, respectively. After all active nodes $u_{t'_i} \in \mathcal{S}_t$ have been processed, $\boldsymbol{x}$ is updated from $\boldsymbol{x}_t$ to

[8] We say a node $u_{t'}$ is *active* if $r_{u_{t'}} \geq \epsilon d_{u_{t'}}$ and *inactive* if $0 \leq r_{u_{t'}} < \epsilon d_{u_{t'}}$.

$\boldsymbol{x}_{t+1}$ and $\boldsymbol{r}$ from $\boldsymbol{r}_t$ to $\boldsymbol{r}_{t+1}$ as the following

$$\begin{aligned}
\boldsymbol{x}_t = \boldsymbol{x}_{u_{t'_0}} &\xrightarrow{u_{t'_1}} \boldsymbol{x}_{t'_1} \xrightarrow{u_{t'_2}} \boldsymbol{x}_{t'_2} \quad \cdots \quad \xrightarrow{u_{t'_{|\mathcal{S}_t|}}} \boldsymbol{x}_{t'_{|\mathcal{S}_t|}} = \boldsymbol{x}_{t+1} \\
\boldsymbol{r}_t = \boldsymbol{r}_{u_{t'_0}} &\xrightarrow{u_{t'_1}} \boldsymbol{r}_{u_{t'_1}} \xrightarrow{u_{t'_2}} \boldsymbol{r}_{u_{t'_2}} \quad \cdots \quad \xrightarrow{u_{t'_{|\mathcal{S}_t|}}} \boldsymbol{r}_{u'_{|\mathcal{S}_t|}} = \boldsymbol{r}_{t+1}.
\end{aligned}$$

For each i-th active node $u_{t'_i}$, the updates are from Line 14 to Line 21 of Alg. 4 give us the following iterations

$$\boldsymbol{x}_{t'_i} = \boldsymbol{x}_{t'_{i-1}} \underbrace{+\alpha r_{u_{t'_{i-1}}} \cdot \boldsymbol{e}_{u_{t'_{i-1}}}}_{\text{Line 14}} \tag{13}$$

$$\boldsymbol{r}_{t'_i} = \boldsymbol{r}_{t'_{i-1}} \underbrace{+(1-\alpha) r_{u_{t'_{i-1}}} \cdot \boldsymbol{W}\boldsymbol{D}^{-1}\boldsymbol{e}_{u_{t'_{i-1}}}}_{\text{Line 15,16}} \underbrace{-r_{u_{t'_{i-1}}} \cdot \boldsymbol{e}_{u_{t'_{i-1}}}}_{\text{Line 21}}. \tag{14}$$

These two iterations (13) and (14) essentially moves residual $r_{u_{t'_{i-1}}}$ out of node $u_{t'_{i-1}}$ to its estimate vector $\boldsymbol{x}$ and residual entries of its neighbors. Specifically, the first iteration (13) moves α times magnitude of $r_{u_{t'_{i-1}}}$ to $\boldsymbol{x}$ and the second iteration (14) moves $(1-\alpha)$ times of $r_{u_{t'_{i-1}}}$ to neighbors spread the magnitude by the distribution vector $\boldsymbol{W}\boldsymbol{D}^{-1}\boldsymbol{e}_{u_{t'_{i-1}}}$. The last term $-r_{u_{t'_{i-1}}} \cdot \boldsymbol{e}_{u_{t'_{i-1}}}$ is to remove $r_{u_{t'_{i-1}}}$ from node $u_{t'_{i-1}}$. Rearrange (14), we have

$$\begin{aligned}
\boldsymbol{r}_{t'_i} &= \boldsymbol{r}_{t'_{i-1}} + (1-\alpha) r_{u_{t'_{i-1}}} \cdot \boldsymbol{W}\boldsymbol{D}^{-1}\boldsymbol{e}_{u_{t'_{i-1}}} - r_{u_{t'_{i-1}}} \cdot \boldsymbol{e}_{u_{t'_{i-1}}} \\
\boldsymbol{r}_{t'_i} &= \boldsymbol{r}_{t'_{i-1}} - \left(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right) r_{u_{t'_{i-1}}} \cdot \boldsymbol{e}_{u_{t'_{i-1}}} \\
\alpha r_{u_{t'_{i-1}}} \cdot \boldsymbol{e}_{u_{t'_{i-1}}} &= \alpha \left(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1} \left(\boldsymbol{r}_{t'_{i-1}} - \boldsymbol{r}_{t'_i}\right).
\end{aligned} \tag{15}$$

Use (15) and (13), we have

$$\begin{aligned}
\boldsymbol{x}_{t'_i} &= \boldsymbol{x}_{t'_{i-1}} + \alpha r_{u_{t'_{i-1}}} \cdot \boldsymbol{e}_{u_{t'_{i-1}}} \\
&= \boldsymbol{x}_{t'_{i-1}} + \alpha \left(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1} \left(\boldsymbol{r}_{t'_{i-1}} - \boldsymbol{r}_{t'_i}\right)
\end{aligned}$$

Since $i = 1, 2, \ldots, |\mathcal{S}_t|$, we sum above equation over all active nodes $\mathcal{S}_t$, we have

$$\begin{aligned}
\boldsymbol{x}_{t+1} &= \boldsymbol{x}_{t_{|\mathcal{S}_t|'}} \\
&= \boldsymbol{x}_{t_{|\mathcal{S}_t|-1'}} + \alpha \left(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1} \left(\boldsymbol{r}_{t'_{|\mathcal{S}_t|-1}} - \boldsymbol{r}_{t'_{|\mathcal{S}_t|}}\right) \\
&\vdots \\
&= \boldsymbol{x}_{t_{0'}} + \alpha \left(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1} \sum_{i=1}^{|\mathcal{S}_t|} \left(\boldsymbol{r}_{t'_{i-1}} - \boldsymbol{r}_{t'_i}\right) \\
&= \boldsymbol{x}_{t_{0'}} + \alpha \left(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1} \left(\boldsymbol{r}_{t'_0} - \boldsymbol{r}_{t'_{|\mathcal{S}_t|}}\right) \\
&= \boldsymbol{x}_t + \alpha \left(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1} (\boldsymbol{r}_t - \boldsymbol{r}_{t+1})
\end{aligned}$$

On the other hand, for all epochs, we continue to use the last equation to have

$$\begin{aligned}
\boldsymbol{x}_{t+1} &= \boldsymbol{x}_t + \alpha \left(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1} (\boldsymbol{r}_t - \boldsymbol{r}_{t+1}) \\
&= \alpha \left(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1} \sum_{i=1}^{t} (\boldsymbol{r}_i - \boldsymbol{r}_{i+1}) \\
&= \alpha \left(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1} (\boldsymbol{r}_1 - \boldsymbol{r}_{t+1}) \\
&= \boldsymbol{X}(\boldsymbol{e}_s - \boldsymbol{r}_{t+1}).
\end{aligned} \tag{16}$$

Since $\boldsymbol{x}_{s,\epsilon}$ is an estimate of $\boldsymbol{x}_s$ returned by FIFOPUSH, for any node $s \in \mathcal{V}$ and by (16), we have

$$\boldsymbol{X}\boldsymbol{e}_s = \boldsymbol{x}_{s,\epsilon} + \boldsymbol{X}\boldsymbol{r}_{s,\epsilon}, \quad \forall s \in \mathcal{V}.$$

Write the above equation as a matrix form; we obtain $\boldsymbol{X} = \boldsymbol{X}_\epsilon + \boldsymbol{X}\boldsymbol{R}_\epsilon$. Notice that each u-th element of $\boldsymbol{r}_{s,\epsilon}$ satisfies $0 \le r_{s,\epsilon}(u) < \epsilon d_u$. Hence we have $\mathbf{0}_{n\times n} \le \boldsymbol{R}_\epsilon < \epsilon \cdot \mathbf{diag}(d_1, \ldots, d_n) \cdot \mathbf{1}\mathbf{1}^\top$. □

Remark A.2. The above lemma is essentially the linear invariant property (Andersen et al., 2006). Here, we show a relation between the estimation and residual vectors. Recall that for any subset of nodes $\mathcal{S} \subseteq \mathcal{V}$, the volume of $\mathcal{S}$ is defined $\mathrm{vol}\,(\mathcal{S}) = \sum_{v\in\mathcal{S}} d_v$. We are ready to prove Thm. 4.2.

Theorem 4.2 (Local linear convergence of FIFOPUSH for $\boldsymbol{X_L}$)**.** *Let $\boldsymbol{x}_s = \boldsymbol{X_L}\boldsymbol{e}_s$. Denote T as the total epochs executed by* FIFOPUSH*, and $\mathcal{S}_t := \{v : r_t(v) \ge \epsilon \cdot d_v, v \in \mathcal{I}_t\}$ as the set of active nodes in t-th epoch. Then, the total operations of* FIFOPUSH$(\mathcal{G}, \epsilon, \alpha, s)$ *is dominated by*

$$R_T := \sum_{t=1}^{T} \sum_{u_t \in \mathcal{S}_t} d_{u_t} \le \frac{\mathrm{vol}\,(\mathcal{S}_{1:T})}{\alpha \cdot \eta_{1:T}} \log\left(\frac{C_{\alpha,T}}{\epsilon}\right), \tag{17}$$

where $\mathrm{vol}\,(\mathcal{S}_{1:T}) = \sum_{t=1}^{T} \mathrm{vol}\,(\mathcal{S}_t)\,/T$ is the average volume of $\mathcal{S}_t$. Additionally, $\eta_{1:T} = \sum_{t=1}^{T} \eta_t/T$ is the average of local convergence factors $\eta_t \triangleq \sum_{u\in\mathcal{S}_t} d_u / \sum_{v\in\mathcal{I}_t} d_v$, and $C_{\alpha,T} = 1/(\sum_{v\in\mathcal{I}_T}(1-\alpha)d_u w_{uv}/D_u)$. For $s, i \in \mathcal{V}$, we have $\boldsymbol{x}_s = \boldsymbol{x}_{s,\epsilon} + \boldsymbol{X_L}\boldsymbol{r}_{s,\epsilon}, r_{s,\epsilon}(i) \le [0, \epsilon d_i)$.

Proof of Theorem 4.2. From Lemma A.1, we know that for $s, i \in \mathcal{V}$, we have $\boldsymbol{x}_s = \boldsymbol{x}_{s,\epsilon} + \boldsymbol{X_L}\boldsymbol{r}_{s,\epsilon}, r_{s,\epsilon}(i) \le [0, \epsilon d_i)$. At epoch $t \ge 1$, recall $\mathcal{Q}$ contains a set of *active* nodes $\mathcal{S}_t$ and a set of *inactive* nodes $\mathcal{U}_t \backslash \mathcal{S}_t$. After FIFOPUSH processed the last node $u_{t'+i}$ in $\mathcal{U}_t$, it is easy to see that the total operations of t-th epoch are dominated by the volume of $\mathcal{S}_t$, i.e., $\mathrm{vol}\,(\mathcal{S}_t)$ (from Line 13 to Line 16). Hence, the total time complexity is dominated by $R_T = \sum_{t=1}^{T} \mathrm{vol}\,(\mathcal{S}_t)$. In the rest, we shall provide two upper bounds of R_T.

1. The first is to prove an upper bound $1/\alpha\epsilon$, which is directly followed from Andersen et al. (2006). We repeat the main idea here. For each active iteration of Algo. 4, we have $r_{u_{t'}} \ge \epsilon \cdot d_{u_{t'}}$, which indicates $r_{u_{t'}}$ was at least $\epsilon \cdot d_{u_{t'}}$; hence $\|\boldsymbol{r}_t\|_1$ decreased by at least $\alpha\epsilon \cdot d_{u_{t'}}$ with total $d_{u_{t'}}$ operations. Hence, overall $u_{t'} \in \mathcal{S}_t$, we have

$$\alpha\epsilon \sum_{u_{t'}\in\mathcal{S}_t} d_{u_{t'}} \le \alpha \sum_{u_{t'}\in\mathcal{S}_t} r_{u_{t'}} = \|\boldsymbol{r}_t\|_1 - \|\boldsymbol{r}_{t+1}\|_1.$$

   Summing the above inequality over t, we have the total operations of FIFOPUSH bounded by

$$R_T = \sum_{t=1}^{T} \mathrm{vol}\,(\mathcal{S}_t) = \sum_{t=1}^{T} \sum_{u\in\mathcal{S}_t} d_u \le \frac{1}{\alpha\epsilon} \sum_{t=1}^{T} (\|\boldsymbol{r}_t\|_1 - \|\boldsymbol{r}_{t+1}\|_1) = \frac{1 - \|\boldsymbol{r}_{T+1}\|_1}{\alpha\epsilon} \le \frac{1}{\alpha\epsilon}, \tag{18}$$

   where note that $\|\boldsymbol{r}_1\|_1 = 1$.

2. The sublinear bound $\mathcal{O}(\frac{1}{\alpha\epsilon})$ in (18) is independent of the graph size which is the key advantage of FIFOPUSH over other numerical methods such as Power Iteration where $\mathcal{O}(\frac{m}{\alpha}\log(\frac{1}{\epsilon}))$ operations needed, where m is the number of edges in the graph. However, when ϵ becomes small, $\mathcal{O}(\frac{1}{\alpha\epsilon})$ is too pessimistic. It is natural to ask whether there exists any bound that takes advantage of both FIFOPUSH and POWERITER. We answer this question positively by providing a local linear convergence. There are two key components in our proof: 1) residuals left in $\boldsymbol{r}_T$ are relatively significant so that total epochs T can be bound by $\mathcal{O}(\log\frac{1}{\epsilon})$; 2) the average operations of t epochs is equal to $\mathrm{vol}\,(\mathcal{S}_{1:T})$, which is independent of m.

   After the T-th epoch finished, the set of all inactive nodes is exactly $\mathcal{I}_{T+1}$, i.e., $\mathcal{I}_{T+1} = \{v : 0 < r_{T+1}(v) < \epsilon \cdot d_v, v \in \mathcal{V}\}$. For each $v \in \mathcal{I}_{T+1}$, note that there exists at least one of its neighbor $u_{t'} \in \mathcal{N}(v)$ such that $r_{u_{t'}} \ge \epsilon \cdot d_{u_{t'}}$ had happened in a previous active iteration. Combine with Line 12 of Algorithm 4, we have

$$\forall v \in \mathcal{I}_{T+1}, \quad \frac{(1-\alpha)r_u}{D_u} w_{uv} \ge \frac{(1-\alpha)\epsilon d_u}{D_u} w_{uv} \triangleq \tilde{r}_v,^{9} \tag{19}$$

[9] We ignore the time index t', which is unrelated with our analysis.

where $\tilde{r}_v$ is the residual pushed into v but never popped out. From (19), note that $\sum_{v\in\mathcal{I}_{T+1}} \tilde{r}_v$ is an estimate of $\|\boldsymbol{r}_{T+1}\|_1$ from bottom. That is

$$\sum_{v\in\mathcal{I}_{T+1}} \tilde{r}_v = \sum_{v\in\mathcal{I}_{T+1}} \frac{(1-\alpha)\epsilon d_u}{D_u} w_{uv} \leq \|\boldsymbol{r}_{T+1}\|_1. \tag{20}$$

Next, we show a significant amount of residual that has been pushed out from $\boldsymbol{r}_t$ to $\boldsymbol{r}_{t+1}$. For t-th epoch, the total amount of residual that had been pushed out is $\alpha \sum_{u_{t'}\in\mathcal{S}_t} r_{u_{t'}}$ (Line 10). That is,

$$\|\boldsymbol{r}_t\|_1 - \|\boldsymbol{r}_{t+1}\|_1 \geq \alpha \sum_{u_{t'}\in\mathcal{S}_t} r_{u_{t'}}. \tag{21}$$

On the other hand, by the activation condition, we have

$$\forall u_{t'} \in \mathcal{S}_t, r_{u_{t'}} \geq \epsilon \cdot d_{u_{t'}}, \qquad \forall v \in \mathcal{I}_t \backslash \mathcal{S}_t, 0 < r_t(v) < \epsilon \cdot d_v.$$

Summation above inequalities over all active nodes $u_{t'}$ and inactive nodes v, we have

$$\frac{\sum_{u_{t'}\in\mathcal{S}_t} r_{u_{t'}}}{\sum_{u_{t'}\in\mathcal{S}_t} d_{u_{t'}}} \geq \epsilon > \frac{\sum_{v\in\mathcal{I}_t\backslash\mathcal{S}_t} r_v}{\sum_{v\in\mathcal{I}_t\backslash\mathcal{S}_t} d_v},$$

which indicates

$$\frac{\sum_{u_{t'}\in\mathcal{S}_t} r_{u_{t'}}}{\sum_{u_{t'}\in\mathcal{S}_t} d_{u_{t'}}} > \frac{\sum_{u_{t'}\in\mathcal{S}_t} r_{u_{t'}} + \sum_{v\in\mathcal{I}_t\backslash\mathcal{S}_t} r_v}{\sum_{u_{t'}\in\mathcal{S}_t} d_{u_{t'}} + \sum_{v\in\mathcal{I}_t\backslash\mathcal{S}_t} d_v} = \frac{\sum_{v\in\mathcal{I}_t} r_v}{\sum_{v\in\mathcal{I}_t} d_v} = \frac{\|\boldsymbol{r}_t\|_1}{\sum_{v\in\mathcal{I}_t} d_v}, \tag{22}$$

where the last equality is due to the fact that $\mathcal{I}_t$ indexes all nonzero entries of $\boldsymbol{r}_t$, i.e., $\|\boldsymbol{r}_t\|_1 = \sum_{v\in\mathcal{I}_t} r_t(v)$. Combine (21) and (22), for $t = 1, 2, \ldots, T$, we have

$$\|\boldsymbol{r}_{t+1}\|_1 < \left(1 - \frac{\alpha \sum_{u\in\mathcal{S}_t} d_u}{\sum_{v\in\mathcal{I}_t} d_v}\right) \|\boldsymbol{r}_t\|_1. \tag{23}$$

Notice that $\|\boldsymbol{r}_{T+1}\|_1$ is lower bounded by (20). Use (23) from $t = 1$ to $t = T$, we obtain

$$\epsilon(1-\alpha)\tilde{C}_T \leq \|\boldsymbol{r}_{T+1}\|_1 \leq \prod_{t=1}^{T} \left(1 - \frac{\alpha \sum_{u\in\mathcal{S}_t} d_u}{\sum_{v\in\mathcal{I}_t} d_v}\right), \tag{24}$$

where $\tilde{C}_T = \sum_{v\in\mathcal{I}_{T+1}} \frac{d_u w_{uv}}{D_u}$. Take the logarithm on both sides of (24) and use the fact that $\log(1-x) \leq -x, \forall x \geq 0$. We reach

$$\alpha \sum_{t=1}^{T} \frac{\sum_{u\in\mathcal{S}_t} d_u}{\sum_{v\in\mathcal{I}_t} d_v} \leq \log\left(\frac{1}{\epsilon(1-\alpha)\tilde{C}_T}\right). \tag{25}$$

Let $\eta_t = \frac{\sum_{u\in\mathcal{S}_t} d_u}{\sum_{v\in\mathcal{I}_t} d_v}$ be the *active* ratio at t-th epoch. The average of all η_t is then defined as $\eta_{1:T} = \frac{1}{T}\sum_{t=1}^{T} \eta_t$. Note both η_t and $\eta_{1:T}$ are in $(0, 1]$. Then by (25), the total number of epochs can be bounded by

$$T \leq \frac{1}{\alpha \cdot \eta_{1:T}} \log\left(\frac{1}{\epsilon(1-\alpha)\tilde{C}_T}\right).$$

The total operations for processing *active* nodes is R_T, which can be represented as

$$R_T = \sum_{t=1}^{T} \sum_{u\in\mathcal{S}_t} d_u = \sum_{t=1}^{T} \mathrm{vol}\left(\mathcal{S}_t\right) = T \cdot \mathrm{vol}\left(\mathcal{S}_{1:T}\right) \leq \frac{\mathrm{vol}\left(\mathcal{S}_{1:T}\right)}{\alpha \cdot \eta_{1:T}} \log\left(\frac{1}{\epsilon(1-\alpha)\tilde{C}_T}\right) \tag{26}$$

Combine two bounds in (18) and (26), we have

$$R_T \leq \min\left\{\frac{1}{\alpha\epsilon}, \frac{\mathrm{vol}\left(\mathcal{S}_{1:T}\right)}{\alpha \cdot \eta_{1:T}} \log\left(\frac{1}{\epsilon(1-\alpha)\tilde{C}_T}\right)\right\}.$$

Let $C_{\alpha,T} = 1/((1-\alpha)\tilde{C}_T)$, we finish the proof. □

Remark A.3. One of the key components in our proof is the local convergence factor η_t in (23), which is inspired by a critical observation in Lemma 4.4 of Wu et al. (2021). The authors show that FIFOPUSH is similar to a variant of Power Iteration when $\epsilon < \frac{1}{2m}$ with admitted time complexity $\mathcal{O}(\frac{m}{\alpha}\log(\frac{1}{\epsilon m}))$. There is no bound for $\epsilon > \frac{1}{2m}$. However, our provided bound works for all $\epsilon > 0$. We first show that there is a relatively significant amount of residual left in $\boldsymbol{r}_t$, which makes us bound the total epochs T by $\mathcal{O}(\frac{1}{\alpha\cdot\eta_{1:T}}\log(\frac{1}{\epsilon(1-\alpha)})$. The other critical component is that we show the number of operations of each epoch mainly depends on $\mathcal{O}(\mathrm{vol}\left(\mathcal{S}_t\right))$ instead of $\mathcal{O}(m)$.

To see the effectiveness of the local linear convergence bound, we apply FIFOPUSH with $\alpha = 0.1, 0.5, 0.9$ where the results as illustrated in Fig. 8, 9, and 10 of Cora dataset. We also include the results of *Citeseer* dataset as shown in Fig. 11, 12, and 13. Our bound is much better, especially when α is large. We find similar patterns on other graph datasets.

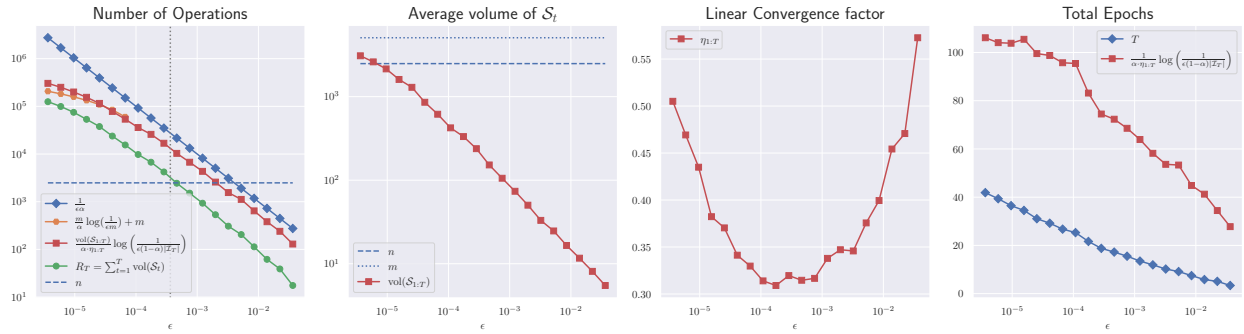


Figure 8: The illustration of operations on the *Cora* dataset. We run FIFOPUSH$(\mathcal{S}, \epsilon, \alpha, s)$ for $\alpha = 0.1$ and $\epsilon \in \left[10^{-2}, 10^{2}\right] * \sqrt{\frac{1-\alpha}{1+\alpha}}/n$. Compared with the linear bound $1/\alpha\epsilon$ in Andersen et al. (2006) and power-iteration bound $\frac{m}{\alpha}\log\left(\frac{1}{\epsilon m}\right)+m$ provided in Wu et al. (2021), our bound is better and shows the "locality" property of FIFOPUSH. Note that the bound $\frac{m}{\alpha}\log\left(\frac{1}{\epsilon m}\right)+m$ only works when $\epsilon < 1/2m$.

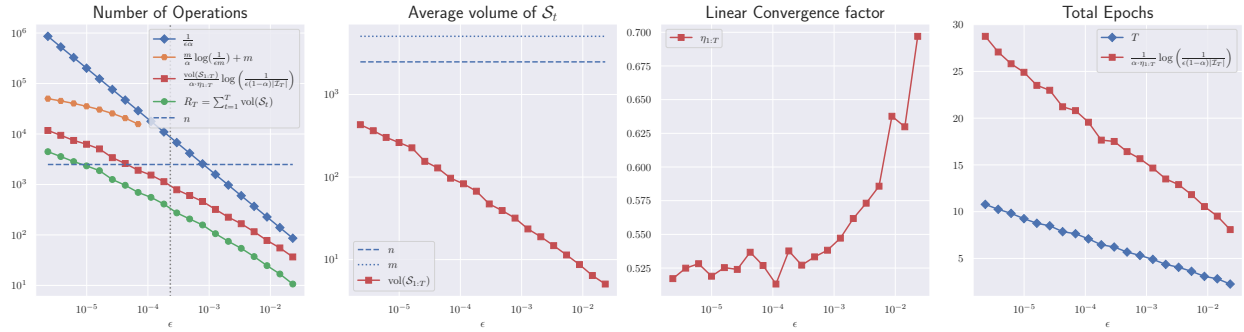


Figure 9: The illustration of our bound and parameters on the *Cora* dataset. We run FIFOPUSH$(\mathcal{S}, \epsilon, \alpha, s)$ with $\alpha = 0.5$.

A.3. Local linear convergence of FIFOPUSH for $(\alpha\boldsymbol{I} + \boldsymbol{D} - \boldsymbol{W})^{-1}$: The proof of Thm. 4.3

Given any $\alpha > 0, \epsilon > 0$, Algo. 5 is to approximate $(\alpha\boldsymbol{I} + \boldsymbol{D} - \boldsymbol{W})^{-1}$

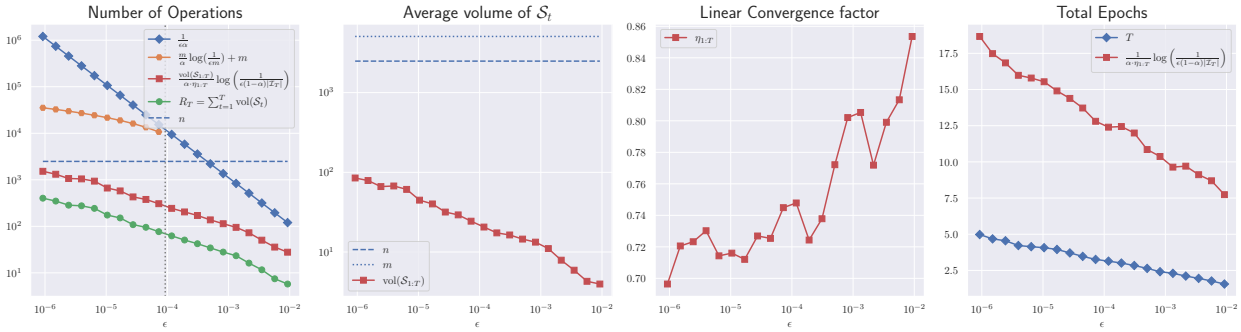


Figure 10: The illustration of our bound and parameters on the *Cora* dataset. We run FIFOPUSH$(\mathcal{S}, \epsilon, \alpha, s)$ with $\alpha = 0.9$.

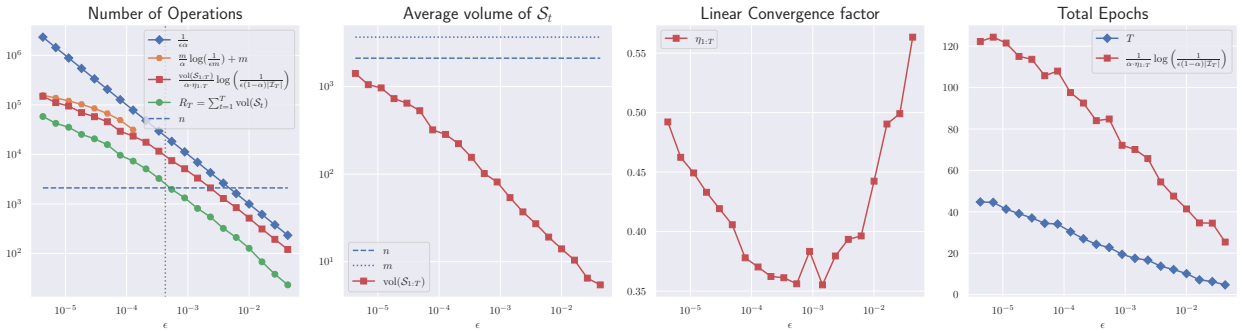


Figure 11: The illustration of operations on the *Citeseer* dataset. We run FIFOPUSH$(\mathcal{S}, \epsilon, \alpha, s)$ for $\alpha = 0.1$ and $\epsilon \in \left[10^{-2}, 10^{2}\right] * \sqrt{\frac{1-\alpha}{1+\alpha}}/n$. Compared with the linear bound $1/\alpha\epsilon$ in Andersen et al. (2006) and power-iteration bound $\frac{m}{\alpha}\log\left(\frac{1}{\epsilon m}\right) + m$ provided in Wu et al. (2021), our bound is better and shows the "locality" property of FIFOPUSH. Note that the bound $\frac{m}{\alpha}\log\left(\frac{1}{\epsilon m}\right) + m$ only works when $\epsilon < 1/2m$.

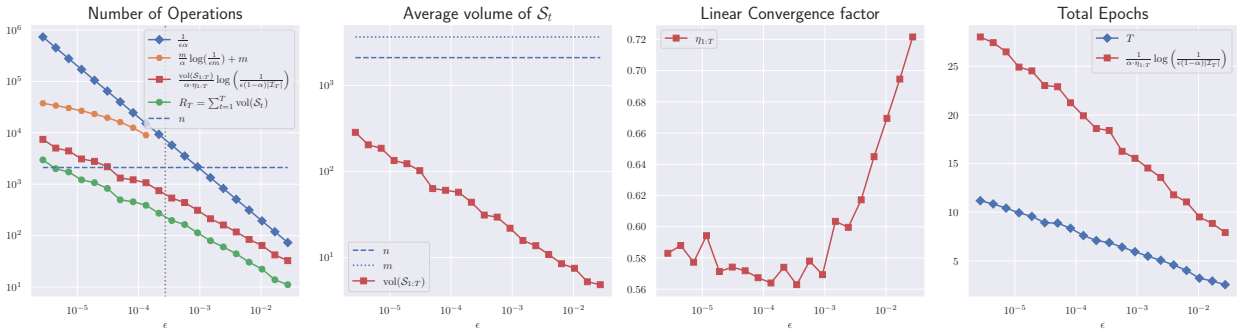


Figure 12: The illustration of our bound on the *Citeseer* dataset. We run FIFOPUSH$(\mathcal{S}, \epsilon, \alpha, s)$ with $\alpha = 0.5$.

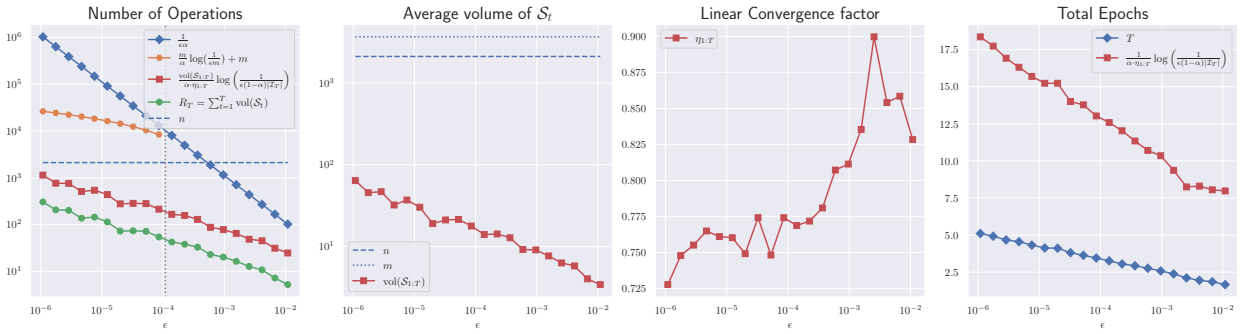


Figure 13: The illustration of our bound on the *Citeseer* dataset. We run FIFOPUSH$(\mathcal{S}, \epsilon, \alpha, s)$ with $\alpha = 0.9$.

Algorithm 5 FIFOPUSH$(\mathcal{G}, \epsilon, \alpha, s)$ with a dummy node ‡

Initialize: $\boldsymbol{r}_t = \frac{\boldsymbol{e}_s}{\alpha}$, $\boldsymbol{x}_t = \boldsymbol{0}$, $t = 1$, $t' = 1$
$\mathcal{Q} = [s, ‡]$ // At the initial stage, $\mathcal{Q}$ contains s and a dummy node ‡.
while $\mathcal{Q}$.size() $\neq 1$ **do**
  $u_{t'} = \mathcal{Q}$.pop()
  if $u_{t'} ==$ ‡ **then**
    $t = t + 1$ // Nodes in $\mathcal{U}_t$ has been processed. Go to the next epoch.
    $\mathcal{Q}$.push$(u_{t'})$
    continue
  end if
  if $r_{u_{t'}} < \epsilon \cdot d_{u_{t'}}$ **then**
    $t' = t' + 1$ // $u_{t'}$ is an "*inactive*" node
    continue
  end if
  $x_{u_{t'}} = x_{u_{t'}} + \frac{\alpha r_{u_{t'}}}{\alpha + D_{u_{t'}}}$ // $u_{t'}$ is an "*active*" node
  for $v \in \mathcal{N}(u_{t'})$ **do**
    $r_v = r_v + \frac{r_{u_{t'}} w_{v u_{t'}}}{\alpha + D_{u_{t'}}}$
    if $v \notin \mathcal{Q}$ **then**
      $\mathcal{Q}$.push(v)
    end if
  end for
  $r_{u_{t'}} = 0$
  $t' = t' + 1$
end while
Return $(\boldsymbol{x}_t, \boldsymbol{r}_t)$

Theorem 4.3 (Local convergence of FIFOPUSH for $\boldsymbol{X}_{\mathcal{L}}$)**.** *Let $\boldsymbol{x}_s = \boldsymbol{X}_{\mathcal{L}} \boldsymbol{e}_s$ and run Aglo. 2 for $\boldsymbol{X}_{\mathcal{L}}$. For $s, i \in \mathcal{V}$, we have $\boldsymbol{x}_s = \boldsymbol{x}_{s,\epsilon} + \alpha \boldsymbol{X}_{\mathcal{L}} \boldsymbol{r}_{s,\epsilon}$, with $r_{s,\epsilon}(i) \leq [0, \epsilon d_i), \forall i \in \mathcal{V}$. The main operations of* FIFOPUSH *for $\boldsymbol{X}_{\mathcal{L}}$ is bounded as*

$$R_T \leq \frac{\mathrm{vol}\left(\mathcal{S}_{1:T}\right)(\alpha + D_{\max})}{\alpha \cdot \eta_{1:T}} \log\left(\frac{C_{\alpha,T}}{\epsilon}\right), \tag{27}$$

where $\mathrm{vol}\left(\mathcal{S}_{1:T}\right)$ *and $\eta_{1:T}$ are the same as in Thm. 4.2, $\eta_t \triangleq \frac{\sum_{u \in \mathcal{S}_t} d_u/(\alpha + D_u)}{\sum_{v \in \mathcal{I}_t} d_v/(\alpha + D_v)}$, $C_{\alpha,T} = 1/\sum_{v \in \mathcal{I}_T} \frac{d_u w_{uv}}{\alpha + D_u}$, and $D_{\max} = \max_{v \in \mathrm{supp}(\boldsymbol{x}_{s,\epsilon})} D_v$.*

Proof. The key of Alg. 5 is to maintain $\boldsymbol{x}_t$ and $\boldsymbol{r}_t$ so that $1/\alpha$ magnitudes will move from $\boldsymbol{r}$ to $\boldsymbol{x}$. For each active node u_t, $\boldsymbol{x}_t$ updates to $\tilde{\boldsymbol{x}}_{t+1}$ and $\boldsymbol{r}_t$ updates to $\tilde{\boldsymbol{r}}_{t+1}$ as the following

$$\begin{aligned}
\tilde{\boldsymbol{x}}_{t+1} &= \boldsymbol{x}_t + \frac{\alpha r_u}{\alpha + D_u} \boldsymbol{e}_u \\
\tilde{\boldsymbol{r}}_{t+1} &= \boldsymbol{r}_t + \frac{r_u}{\alpha + D_u} \boldsymbol{W} \boldsymbol{e}_u - r_u \boldsymbol{e}_u = \boldsymbol{r}_t - \left(\boldsymbol{I} + \frac{\boldsymbol{D} - \boldsymbol{W}}{\alpha}\right) \frac{\alpha r_u}{\alpha + D_u} \boldsymbol{e}_u \\
\frac{\alpha r_u}{\alpha + D_u} \boldsymbol{e}_u &= \left(\boldsymbol{I} + \frac{\boldsymbol{D} - \boldsymbol{W}}{\alpha}\right)^{-1} (\boldsymbol{r}_t - \tilde{\boldsymbol{r}}_{t+1})
\end{aligned}$$

Bring back t' for u. After all active nodes in t-th epoch have been updated, we have

$$\begin{aligned}\boldsymbol{x}_{t+1} &= \boldsymbol{x}_t + \sum_{u_{t'}\in\mathcal{S}_t} \frac{\alpha r_{u_{t'}}}{\alpha + D_{u_{t'}}} \boldsymbol{e}_{u_{t'}} \\ &= \boldsymbol{x}_t + \left(\boldsymbol{I} + \frac{\boldsymbol{D}-\boldsymbol{W}}{\alpha}\right)^{-1} (\boldsymbol{r}_t - \boldsymbol{r}_{t+1}) = \boldsymbol{x}_1 + \left(\boldsymbol{I} + \frac{\boldsymbol{D}-\boldsymbol{W}}{\alpha}\right)^{-1} \sum_{i=1}^{t} (\boldsymbol{r}_i - \boldsymbol{r}_{i+1}) \\ &= \left(\boldsymbol{I} + \frac{\boldsymbol{D}-\boldsymbol{W}}{\alpha}\right)^{-1} \left(\frac{\boldsymbol{e}_s}{\alpha} - \boldsymbol{r}_{t+1}\right),\end{aligned}$$

where $\boldsymbol{r}_t = \tilde{\boldsymbol{r}}_t \to \tilde{\boldsymbol{r}}_{t'+1} \to \tilde{\boldsymbol{r}}_{t'+2} \to \cdots \tilde{\boldsymbol{r}}_{t'+|\mathcal{S}_t|} = \boldsymbol{r}_{t+1}$. Denote $\boldsymbol{x}_{t+1}$ as $\boldsymbol{x}_{s,\epsilon}$ and $\boldsymbol{r}_{t+1}$ as $\boldsymbol{r}_{s,\epsilon}$, then we have

$$\boldsymbol{x}_s = \boldsymbol{x}_{s,\epsilon} + \alpha \boldsymbol{X} \boldsymbol{r}_{s,\epsilon}. \tag{28}$$

After the T-th epoch finished, $\mathcal{I}_{T+1} = \operatorname{supp}(\boldsymbol{r}_{s,\epsilon})$. For each $v \in \mathcal{I}_{T+1}$, note that there exists at least one of its neighbor $u \in \mathcal{N}(v)$ such that $r_u \geq \epsilon \cdot d_u$ had happened in a previous active iteration.

$$\forall v \in \mathcal{I}_{T+1}, \quad \frac{r_u w_{uv}}{\alpha + D_u} \geq \frac{\epsilon d_u w_{uv}}{\alpha + D_u} \triangleq \tilde{r}_v,$$

where $\tilde{r}_v$ is the residual pushed into v but never popped out; hence $\sum_{v\in\mathcal{I}_{T+1}} \tilde{r}_v$ is an estimate of $\|\boldsymbol{r}_{T+1}\|_1$ from bottom. That is

$$\sum_{v\in\mathcal{I}_{T+1}} \tilde{r}_v = \sum_{v\in\mathcal{I}_{T+1}} \frac{\epsilon d_u w_{uv}}{\alpha + D_u} \leq \|\boldsymbol{r}_{T+1}\|_1. \tag{29}$$

The operation bounds are similar to that of 4.2. Note that for each epoch, the updates of $\boldsymbol{r}$ satisfies

$$\|\boldsymbol{r}_t\|_1 - \|\boldsymbol{r}_{t+1}\|_1 \geq \sum_{u_{t'}\in\mathcal{S}_t} \frac{\alpha r_{u_{t'}}}{\alpha + D_{u_{t'}}}. \tag{30}$$

By the condition, we have $\forall u_{t'} \in \mathcal{S}_t, r_{u_{t'}} \geq \epsilon \cdot d_{u_{t'}}, \forall v \in \mathcal{I}_t \backslash \mathcal{S}_t, 0 < r_t(v) < \epsilon \cdot d_v$. Summation above inequalities over all active $u_{t'}$ and inactive v, multiply $\frac{\alpha}{\alpha+D_u}$ on both sides, we have

$$\frac{\sum_{u_{t'}\in\mathcal{S}_t} \frac{\alpha r_{u_{t'}}}{\alpha+D_{u_{t'}}}}{\sum_{u_{t'}\in\mathcal{S}_t} \frac{\alpha d_{u_{t'}}}{\alpha+D_{u_{t'}}}} \geq \epsilon > \frac{\sum_{v\in\mathcal{I}_t\backslash\mathcal{S}_t} \frac{\alpha r_v}{\alpha+D_v}}{\sum_{v\in\mathcal{I}_t\backslash\mathcal{S}_t} \frac{\alpha d_v}{\alpha+D_v}},$$

which indicates

$$\frac{\sum_{u_{t'}\in\mathcal{S}_t} \frac{\alpha r_{u_{t'}}}{\alpha+D_{u_{t'}}}}{\sum_{u_{t'}\in\mathcal{S}_t} \frac{\alpha d_{u_{t'}}}{\alpha+D_{u_{t'}}}} \geq \frac{\sum_{u_{t'}\in\mathcal{S}_t} \frac{\alpha r_{u_{t'}}}{\alpha+D_{u_{t'}}} + \sum_{v\in\mathcal{I}_t\backslash\mathcal{S}_t} \frac{\alpha r_v}{\alpha+D_v}}{\sum_{u_{t'}\in\mathcal{S}_t} \frac{\alpha d_{u_{t'}}}{\alpha+D_{u_{t'}}} + \sum_{v\in\mathcal{I}_t\backslash\mathcal{S}_t} \frac{\alpha d_v}{\alpha+D_v}} \geq \frac{\frac{\alpha}{\alpha+D_{\max}}\|\boldsymbol{r}_t\|_1}{\sum_{v\in\mathcal{I}_t} \frac{\alpha d_v}{\alpha+D_v}} \tag{31}$$

where the last equality is due to the fact that $\mathcal{I}_t$ indexes all nonzero entries of $\boldsymbol{r}_t$, i.e., $\|\boldsymbol{r}_t\|_1 = \sum_{v\in\mathcal{I}_t} r_t(v)$. Combine (30) and (31), for $t = 1, 2, \ldots, T$, we have

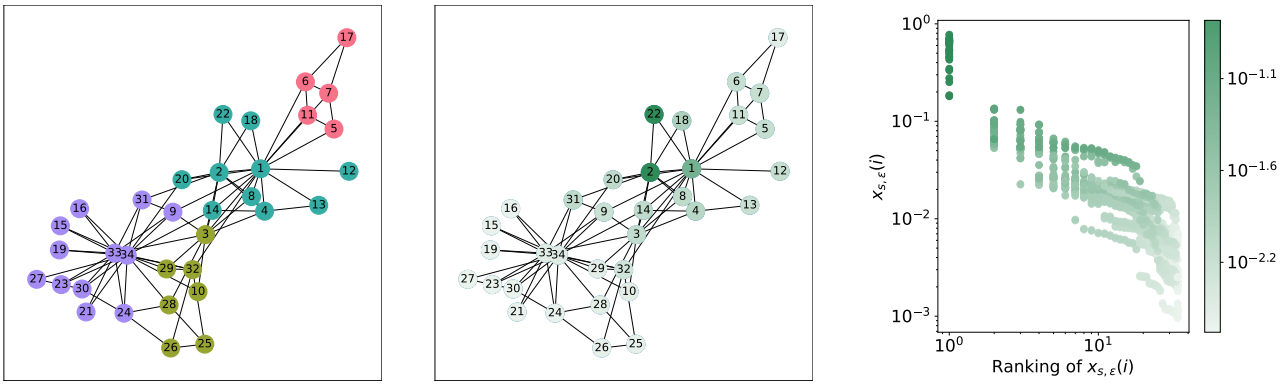
$$\|\boldsymbol{r}_{t+1}\|_1 < \left(1 - \frac{\alpha}{\alpha + D_{\max}} \frac{\sum_{u\in\mathcal{S}_t} d_u/(\alpha + D_u)}{\sum_{v\in\mathcal{I}_t} d_v/(\alpha + D_v)}\right) \|\boldsymbol{r}_t\|_1.$$

From $t = 1$ to $t = T$, we obtain

$$\sum_{v\in\mathcal{I}_{T+1}} \frac{\epsilon d_u w_{uv}}{\alpha + D_u} \leq \|\boldsymbol{r}_{T+1}\|_1 \leq \prod_{t=1}^{T} \left(1 - \frac{\alpha}{\alpha + D_{\max}} \frac{\sum_{u\in\mathcal{S}_t} d_u/(\alpha + D_u)}{\sum_{v\in\mathcal{I}_t} d_v/(\alpha + D_v)}\right).$$

Denote $C_T = 1/\sum_{v\in\mathcal{I}_{T+1}} \frac{d_u w_{uv}}{\alpha+D_u}$. Take the logarithm on both sides of the above and we reach

$$\frac{\alpha}{\alpha + D_{\max}} \sum_{t=1}^{T} \frac{\sum_{u\in\mathcal{S}_t} d_u/(\alpha + D_u)}{\sum_{v\in\mathcal{I}_t} d_v/(\alpha + D_v)} \leq \log\left(\frac{C_T}{\epsilon}\right). \tag{32}$$



(a) Labeled Karate (b) $\boldsymbol{x}_{22,\epsilon}$ on $\mathcal{G}$ (c) Power law of $\boldsymbol{X}_\epsilon$

Figure 14: Power law distribution of $\boldsymbol{X}_{\mathcal{L}}\boldsymbol{e}_s$ on Karate graph (Girvan & Newman, 2002). (a) The Karate contains 34 nodes and 78 edges forming four communities (w_{uv} is the 2-dimensional Euclidean distance.); (b) We run FIFOPUSH$(\mathcal{G}, \epsilon, \alpha, s)$ on $s = 22$ with $\alpha = 0.85$ and $\epsilon = 10^{-12}$. Neighbors of $s = 22$ have large magnitudes of $\boldsymbol{x}_s$; (We changed the magnitude of $\boldsymbol{x}_s(22)$ to the largest magnitude of rest for better visibility.) (c) The power law distribution of entries of $\boldsymbol{x}_s$ for all 34 nodes.

Then by (32), the total number of epochs can be bounded by $T \leq \frac{\alpha + D_{\max}}{\alpha \cdot \eta_{1:T}} \log\left(\frac{C_T}{\epsilon}\right)$. The total operations for processing *active* nodes is $R_T = \sum_{t=1}^{T} \mathrm{vol}\left(\mathcal{S}_t\right)$, which is bounded as

$$R_T \leq \frac{(\alpha + D_{\max})\,\mathrm{vol}\left(\mathcal{S}_{1:T}\right)}{\alpha \cdot \eta_{1:T}} \log\left(\frac{C_T}{\epsilon}\right).$$

□

A.4. Approximation kernels and their residuals

Table 3: The parameterized graph kernel matrices with their kernel approximation

ID	α	Basic Kernel Presentation	Approximation $\boldsymbol{M}_\epsilon$	Residual Matrix $\boldsymbol{E}_\epsilon$
1	$\frac{\lambda}{n}$	$\boldsymbol{M}_{\lambda,\beta} = 2\lambda \boldsymbol{X}_{\mathcal{L}}$	$2\lambda \boldsymbol{X}_\epsilon$	$2\lambda\alpha \boldsymbol{X}_{\mathcal{L}} \boldsymbol{R}_\epsilon$
2	$\frac{\lambda}{n+\lambda}$	$\boldsymbol{M}_{\lambda,\beta} = 2n\boldsymbol{D}^{-1/2}\boldsymbol{X}_{\boldsymbol{L}}\boldsymbol{D}^{1/2}$	$2n\boldsymbol{D}^{-1/2}\boldsymbol{X}_\epsilon\boldsymbol{D}^{1/2}$	$2n\boldsymbol{D}^{-1/2}\boldsymbol{X}_{\boldsymbol{L}}\boldsymbol{R}_\epsilon\boldsymbol{D}^{1/2}$
3	$\frac{n+\lambda-\beta n}{n+\lambda}$	$\boldsymbol{M}_{\lambda,\beta} = \frac{2\lambda n}{n+\lambda-\beta n}\boldsymbol{D}^{-1/2}\boldsymbol{X}_{\boldsymbol{L}}\boldsymbol{D}^{1/2}$	$\frac{2\lambda n}{n+\lambda-\beta n}\boldsymbol{D}^{-1/2}\boldsymbol{X}_\epsilon\boldsymbol{D}^{1/2}$	$\frac{2\lambda n}{n+\lambda-\beta n}\boldsymbol{D}^{-1/2}\boldsymbol{X}_{\boldsymbol{L}}\boldsymbol{R}_\epsilon\boldsymbol{D}^{1/2}$
4	$\frac{n\beta+\lambda}{n}$	$\boldsymbol{M}_{\lambda,\beta} = 2\lambda\boldsymbol{S}^{-1/2}\boldsymbol{X}_{\mathcal{L}}\boldsymbol{S}^{1/2}$	$2\lambda\boldsymbol{S}^{-1/2}\boldsymbol{X}_\epsilon\boldsymbol{S}^{1/2}$	$2\lambda\boldsymbol{S}^{-1/2}\alpha\boldsymbol{X}_{\mathcal{L}}\boldsymbol{R}_\epsilon\boldsymbol{S}^{1/2}$
5	2λ	$\boldsymbol{M}_{\lambda,\beta} = \left(\frac{\boldsymbol{S}^{1/2}}{4n\lambda} + \frac{\beta\boldsymbol{S}^{-1/2}}{4\lambda^2}\right)^{-1}\boldsymbol{X}_{\mathcal{L}}\boldsymbol{S}^{1/2}$	$\left(\frac{\boldsymbol{S}^{1/2}}{4n\lambda} + \frac{\beta\boldsymbol{S}^{-1/2}}{4\lambda^2}\right)^{-1}\boldsymbol{X}_\epsilon\boldsymbol{S}^{1/2}$	$\alpha\left(\frac{\boldsymbol{S}^{1/2}}{4n\lambda} + \frac{\beta\boldsymbol{S}^{-1/2}}{4\lambda^2}\right)^{-1}\boldsymbol{X}_{\mathcal{L}}\boldsymbol{R}_\epsilon\boldsymbol{S}^{1/2}$
6	$\beta + \frac{\lambda}{n}$	$\boldsymbol{M}_{\lambda,\beta} = 2\lambda\boldsymbol{X}_{\mathcal{L}}\left(\boldsymbol{I} - \frac{b\mathbf{1}\mathbf{1}^\top}{\alpha+nb}\right)$	$\boldsymbol{M}_{\lambda,\beta} = 2\lambda\boldsymbol{X}_\epsilon\left(\boldsymbol{I} - \frac{b\mathbf{1}\mathbf{1}^\top}{\alpha+nb}\right)$	$2\lambda\alpha\boldsymbol{X}_{\mathcal{L}}\boldsymbol{R}_\epsilon\left(\boldsymbol{I} - \frac{b\mathbf{1}\mathbf{1}^\top}{\alpha+nb}\right)$

Recall two types of matrix presentation and their approximations

$$\boldsymbol{X}_{\boldsymbol{L}} = \alpha(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1})^{-1} = \boldsymbol{X}_\epsilon + \boldsymbol{X}_{\boldsymbol{L}}\boldsymbol{R}_\epsilon,$$
$$\boldsymbol{X}_{\mathcal{L}} = (\alpha\boldsymbol{I} + \boldsymbol{D} - \boldsymbol{W})^{-1} = \boldsymbol{X}_\epsilon + \alpha\boldsymbol{X}_{\mathcal{L}}\boldsymbol{R}_\epsilon.$$

Based on the above lemmas, we list the approximation kernels in Tab. 3. $\boldsymbol{X}_\epsilon = [\boldsymbol{x}_{1,\epsilon}, \boldsymbol{x}_{2,\epsilon}, \ldots, \boldsymbol{x}_{n,\epsilon}]$ be the matrix obtained by applying FIFOPUSH as described in Algorithm 2. We only show the first two cases to see how to represent these matrices in terms of $\boldsymbol{X}_\epsilon$ and $\boldsymbol{R}_\epsilon$. For example, in the first case, we have

$$\boldsymbol{M}_{\lambda,\beta} = \left(\frac{\boldsymbol{D}-\boldsymbol{W}}{2\lambda} + \frac{\boldsymbol{I}_n}{2n}\right)^{-1} = 2\lambda\left(\alpha\boldsymbol{I}_n + \mathcal{L}\right)^{-1} = 2\lambda\boldsymbol{X}_{\mathcal{L}} = 2\lambda(\boldsymbol{X}_\epsilon + \alpha\boldsymbol{X}_{\mathcal{L}}\boldsymbol{R}_\epsilon).$$

For the second case, the kernel matrix can be rewritten as $\boldsymbol{M} = 2n\boldsymbol{D}^{-1/2}\boldsymbol{X_L}\boldsymbol{D}^{1/2}$. By Lemma A.1, we have

$$\begin{aligned}
\boldsymbol{X_L} &= \boldsymbol{X}_\epsilon + \boldsymbol{X_L}\boldsymbol{R}_\epsilon \\
2n\boldsymbol{D}^{-1/2}\boldsymbol{X_L}\boldsymbol{D}^{1/2} &= 2n\boldsymbol{D}^{-1/2}(\boldsymbol{X}_\epsilon + \boldsymbol{X_L}\boldsymbol{R}_\epsilon)\boldsymbol{D}^{1/2} \\
\boldsymbol{M} &= 2n\boldsymbol{D}^{-1/2}\boldsymbol{X}_\epsilon\boldsymbol{D}^{1/2} + 2n\boldsymbol{D}^{-1/2}\boldsymbol{X_L}\boldsymbol{R}_\epsilon\boldsymbol{D}^{1/2} \\
\boldsymbol{M} &= 2n\boldsymbol{D}^{-1/2}\boldsymbol{X}_\epsilon\boldsymbol{D}^{1/2} + \boldsymbol{M}\boldsymbol{D}^{-1/2}\boldsymbol{R}_\epsilon\boldsymbol{D}^{1/2}.
\end{aligned}$$

For all these cases, we have the relationship that $\boldsymbol{M} = \boldsymbol{M}_\epsilon + \boldsymbol{E}_\epsilon$.

B. Eigenvalues of $\boldsymbol{M}_{\lambda,\beta}$ and quadratic approximation guarantees of $\boldsymbol{M}_\epsilon$

$\boldsymbol{M}_{\lambda,\beta}$ is the matrix we want to approximate. Recall $\boldsymbol{X}_\epsilon$ be the approximated matrix by FIFOPUSH. $\boldsymbol{M}_\epsilon$ be the matrix built upon $\boldsymbol{X}_\epsilon$ (See 3rd column of Table 3). In the following, we will construct the relaxation function based on $\boldsymbol{M}_\epsilon$. The following lemma shows that if $\boldsymbol{M}_\epsilon$ is a good approximation of $\boldsymbol{M}$ then we can define a relaxation function based $\boldsymbol{M}_\epsilon$. Before we present the theorem, let us characterize the eigenvalues of $\boldsymbol{X_L}$ and $\boldsymbol{X}_\mathcal{L}$.

Lemma B.1. *The eigenvalue functions of these two basic kernel matrices $\boldsymbol{X_L}$ and $\boldsymbol{X}_\mathcal{L}$ satisfy*

$$\begin{aligned}
\lambda(\boldsymbol{X_L}) &= \lambda\left(\alpha\left(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1}\right) \in \left[\frac{\alpha}{2-\alpha}, 1\right], \\
\lambda(\boldsymbol{X}_\mathcal{L}) &= \lambda\left((\alpha\boldsymbol{I} + \boldsymbol{D} - \boldsymbol{W})^{-1}\right) \in \left[\frac{1}{\alpha + 2D_{\max}}, \frac{1}{\alpha}\right],
\end{aligned}$$

where $D_{\max}$ is the maximum weighted degree among $\mathcal{V}$. For $\boldsymbol{X_L}$, we assume $\alpha \in (0,1)$, and for $\boldsymbol{X}_\mathcal{L}$, $\alpha > 0$.

Proof. Notice that since the magnitude of the eigenvalues of the column stochastic matrix $\boldsymbol{W}\boldsymbol{D}^{-1}$ is bounded by 1, i.e., $|\lambda(\boldsymbol{W}\boldsymbol{D}^{-1})| \leq 1$, then we have

$$\begin{aligned}
\lambda(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}) &\in [\alpha, 2-\alpha] \\
\lambda\left(\left(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1}\right) &\in \left[\frac{1}{2-\alpha}, \frac{1}{\alpha}\right] \\
\lambda\left(\alpha\left(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1}\right) &\in \left[\frac{\alpha}{2-\alpha}, 1\right].
\end{aligned}$$

Hence, we have the first bounding inequality. To show the second inequality, notice that if $\boldsymbol{x}$ is an eigenvalue of $\boldsymbol{D} - \boldsymbol{W}$, then

$$\begin{aligned}
(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{x} &= \lambda\boldsymbol{x} \\
\boldsymbol{x}^\top(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{x} &= \lambda\boldsymbol{x}^\top\boldsymbol{x} \\
\sum_{(u,v)\in\mathcal{E}} w_{uv}(x_u - x_v)^2 &= \lambda\boldsymbol{x}^\top\boldsymbol{x}.
\end{aligned}$$

Hence $\lambda \geq 0$, and the lower bound is achieved when $\boldsymbol{x} = \mathbf{1}$. On the other hand, a well-known result (Anderson Jr & Morley, 1985) of the upper bound is

$$\lambda(\boldsymbol{D} - \boldsymbol{W}) \leq \max_{(u,v)\in\mathcal{E}} D_u + D_v \leq 2D_{\max}.$$

where $D_{\max}$ is the maximum weighted degree among all nodes. It follows that

$$\begin{aligned}
\lambda(\boldsymbol{D} - \boldsymbol{W}) &\in [0, 2D_{\max}] \\
\lambda(\alpha\boldsymbol{I} + \boldsymbol{D} - \boldsymbol{W}) &\in [\alpha, \alpha + 2D_{\max}] \\
\lambda\left((\alpha\boldsymbol{I} + \boldsymbol{D} - \boldsymbol{W})^{-1}\right) &\in \left[\frac{1}{\alpha + 2D_{\max}}, \frac{1}{\alpha}\right].
\end{aligned}$$

□

Lemma B.2. *Let $\boldsymbol{M}$ be a symmetric positive definite matrix and $\boldsymbol{D}$ be a positive diagonal matrix. Then, for any nonnegative real matrix $\boldsymbol{R} \in \mathbb{R}_+^{n\times n}$ and $\boldsymbol{x} \neq \boldsymbol{0}$, we have*

$$\frac{1}{2}\frac{\boldsymbol{x}^\top(\boldsymbol{M}\boldsymbol{D}^{-1/2}\boldsymbol{R}\boldsymbol{D}^{1/2}+\boldsymbol{D}^{1/2}\boldsymbol{R}^\top\boldsymbol{D}^{-1/2}\boldsymbol{M})\boldsymbol{x}}{\boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{x}} \leq \|\boldsymbol{D}^{-1/2}\boldsymbol{R}\boldsymbol{D}^{1/2}\|_2. \tag{33}$$

Proof. Since $\boldsymbol{M}$ is a symmetric positive definite matrix, one can write $\boldsymbol{M} = \boldsymbol{B}\boldsymbol{B}^\top$ and $\boldsymbol{B}$ are invertible. The decomposition of $\boldsymbol{M}$ is $\boldsymbol{M} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top = \boldsymbol{Q}\boldsymbol{\Lambda}^{1/2}\boldsymbol{\Lambda}^{1/2}\boldsymbol{Q}^\top$ where we can let $\boldsymbol{B} = \boldsymbol{Q}\boldsymbol{\Lambda}^{1/2}$ and $\boldsymbol{Q}$ is an orthonormal matrix. Let $\boldsymbol{y} = \boldsymbol{B}^\top\boldsymbol{x}$, then $\boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{x} = \boldsymbol{y}^\top\boldsymbol{y}$ and $\boldsymbol{x} = (\boldsymbol{B}^\top)^{-1}\boldsymbol{y}$. Let $\boldsymbol{Z} = \boldsymbol{D}^{-1/2}\boldsymbol{R}\boldsymbol{D}^{1/2}$. We have

$$\begin{aligned}
(*) &= \frac{1}{2}\frac{\boldsymbol{x}^\top\left(\boldsymbol{M}\boldsymbol{D}^{-1/2}\boldsymbol{R}\boldsymbol{D}^{1/2}+\boldsymbol{D}^{1/2}\boldsymbol{R}^\top\boldsymbol{D}^{-1/2}\boldsymbol{M}\right)\boldsymbol{x}}{\boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{x}} \\
&= \frac{1}{2}\frac{\boldsymbol{x}^\top\left(\boldsymbol{M}\boldsymbol{Z}+\boldsymbol{Z}^\top\boldsymbol{M}\right)\boldsymbol{x}}{\boldsymbol{y}^\top\boldsymbol{y}} \\
&= \frac{1}{2}\frac{\boldsymbol{y}^\top\boldsymbol{B}^{-1}\left(\boldsymbol{B}\boldsymbol{B}^\top\boldsymbol{Z}+\boldsymbol{Z}^\top\boldsymbol{B}\boldsymbol{B}^\top\right)(\boldsymbol{B}^\top)^{-1}\boldsymbol{y}}{\boldsymbol{y}^\top\boldsymbol{y}} \\
&= \frac{1}{2}\frac{\boldsymbol{y}^\top\left(\boldsymbol{B}^\top\boldsymbol{Z}(\boldsymbol{B}^\top)^{-1}+\boldsymbol{B}^{-1}\boldsymbol{Z}^\top\boldsymbol{B}\right)\boldsymbol{y}}{\boldsymbol{y}^\top\boldsymbol{y}} \\
&= \frac{1}{2}\frac{\boldsymbol{y}^\top\left(\boldsymbol{\Lambda}^{1/2}\boldsymbol{Q}^\top\boldsymbol{Z}(\boldsymbol{Q}^\top)^{-1}\boldsymbol{\Lambda}^{-1/2}+\boldsymbol{\Lambda}^{-1/2}\boldsymbol{Q}^{-1}\boldsymbol{Z}^\top\boldsymbol{Q}\boldsymbol{\Lambda}^{1/2}\right)\boldsymbol{y}}{\boldsymbol{y}^\top\boldsymbol{y}} \\
&\leq \frac{1}{2}\left\|\boldsymbol{\Lambda}^{1/2}\boldsymbol{Q}^\top\boldsymbol{Z}(\boldsymbol{Q}^\top)^{-1}\boldsymbol{\Lambda}^{-1/2}+\boldsymbol{\Lambda}^{-1/2}\boldsymbol{Q}^{-1}\boldsymbol{Z}^\top\boldsymbol{Q}\boldsymbol{\Lambda}^{1/2}\right\|_2
\end{aligned}$$

where the inequality follows Rayleigh's quotient property, that is, any matrix norm bounds the maximal absolute eigenvalue, and the spectral radius is less than any matrix norm. Denoting $\mathbf{A} = \mathbf{Q}^\top\mathbf{Z}(\mathbf{Q}^\top)^{-1}$, we may write

$$\begin{aligned}
2(*) &= \left\|\boldsymbol{\Lambda}^{1/2}\boldsymbol{Q}^\top\boldsymbol{Z}(\boldsymbol{Q}^\top)^{-1}\boldsymbol{\Lambda}^{-1/2}+\boldsymbol{\Lambda}^{-1/2}\boldsymbol{Q}^{-1}\boldsymbol{Z}^\top\boldsymbol{Q}\boldsymbol{\Lambda}^{1/2}\right\|_2 \\
&= \max_{\|\boldsymbol{u}\|_2=1}\left|\boldsymbol{u}^\top\boldsymbol{\Lambda}^{1/2}\boldsymbol{A}\boldsymbol{\Lambda}^{-1/2}\boldsymbol{u}+\boldsymbol{u}^\top\boldsymbol{\Lambda}^{-1/2}\boldsymbol{A}\boldsymbol{\Lambda}^{1/2}\boldsymbol{u}\right|
\end{aligned}$$

and denoting $\boldsymbol{w} = \boldsymbol{\Lambda}^{1/2}\boldsymbol{u}, \boldsymbol{v} = \boldsymbol{\Lambda}^{-1/2}\boldsymbol{u}$

$$\left|\boldsymbol{u}^\top\boldsymbol{\Lambda}^{1/2}\boldsymbol{A}\boldsymbol{\Lambda}^{-1/2}\boldsymbol{u}+\boldsymbol{u}^\top\boldsymbol{\Lambda}^{-1/2}\boldsymbol{A}\boldsymbol{\Lambda}^{1/2}\boldsymbol{u}\right| = 2|\boldsymbol{w}^\top\boldsymbol{A}\boldsymbol{v}| = 2|\operatorname{tr}\left(\boldsymbol{v}\boldsymbol{w}^\top\boldsymbol{A}\right)| \overset{(\circ)}{\leq} 2|\boldsymbol{w}^\top\boldsymbol{v}|\|\boldsymbol{A}\|_2,$$

where the inequality $(\circ)$ is due to Holder's $p = 1$ inequality applied to the matrix singular values. Since $\boldsymbol{w}^\top\boldsymbol{v} = \boldsymbol{u}^\top\boldsymbol{u} = 1$, $(*) \leq \|\boldsymbol{Q}^\top\boldsymbol{Z}(\boldsymbol{Q}^T)^{-1}\|_2 = \|\boldsymbol{Z}\|_2$. □

The next lemma shows FIFOPUSH provides good approximations. Here we only show **Instance 3** and **Instance 4**.

Lemma B.3. *Let $\boldsymbol{X}_\epsilon$ and $\boldsymbol{R}_\epsilon$ be the approximate and the residual matrix obtained by applying* FIFOPUSH$(\mathcal{G}, \epsilon, \alpha, s)$ *$\forall s \in \mathcal{V}$. Then we have the following inequalities*

$$\boldsymbol{x}^\top\left(\frac{\boldsymbol{M}_\epsilon+\boldsymbol{M}_\epsilon^\top}{2}\right)\boldsymbol{x} \geq \boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{x}\left(1-\|\boldsymbol{D}^{-1/2}\boldsymbol{R}_\epsilon\boldsymbol{D}^{1/2}\|_2\right) \textit{ for } \boldsymbol{M} = \frac{2\lambda n}{n+\lambda-\beta n}\boldsymbol{D}^{-1/2}\boldsymbol{X}_{\boldsymbol{L}}\boldsymbol{D}^{1/2},$$

$$\boldsymbol{x}^\top\left(\frac{\boldsymbol{M}_\epsilon+\boldsymbol{M}_\epsilon^\top}{2}\right)\boldsymbol{x} \geq \boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{x}\left(1-\|\boldsymbol{S}^{-1/2}\boldsymbol{R}_\epsilon\boldsymbol{S}^{1/2}\|_2\right) \textit{ for } \boldsymbol{M} = 2\lambda\boldsymbol{S}^{-1/2}\boldsymbol{X}_{\mathcal{L}}\boldsymbol{D}^{1/2}.$$

Proof. The inequality is trivially true when $\boldsymbol{x} = \boldsymbol{0}$. In the rest, we assume $\boldsymbol{x} \neq \boldsymbol{0}$. For ease of notation, we simply write $\boldsymbol{M} = \boldsymbol{M}_{\lambda,\beta}$, and

$$\boldsymbol{M} = \boldsymbol{M}_\epsilon + \boldsymbol{E}_\epsilon, \qquad \boldsymbol{X}_{\boldsymbol{L}} = \boldsymbol{X}_\epsilon + \boldsymbol{X}_{\boldsymbol{L}}\boldsymbol{R}_\epsilon.$$

We consider two parameterized kernels as follows

1. $\boldsymbol{M} = c\boldsymbol{D}^{-1/2}\boldsymbol{X_L}\boldsymbol{D}^{1/2}$ where $c = \frac{2\lambda n}{n+\lambda-\beta n}$. Here,

$$\boldsymbol{X_L} = \alpha(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1})^{-1}, \qquad \boldsymbol{M}_\epsilon = c\boldsymbol{D}^{-1/2}\boldsymbol{X}_\epsilon\boldsymbol{D}^{1/2}, \qquad \boldsymbol{E}_\epsilon = c\boldsymbol{D}^{-1/2}\boldsymbol{X_L}\boldsymbol{R}_\epsilon\boldsymbol{D}^{1/2}.$$

We have

$$\begin{aligned}
\boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{x} &= \boldsymbol{x}^\top\boldsymbol{M}_\epsilon\boldsymbol{x} + \boldsymbol{x}^\top\boldsymbol{E}_\epsilon\boldsymbol{x} \\
&= \boldsymbol{x}^\top\left(\frac{\boldsymbol{M}_\epsilon + \boldsymbol{M}_\epsilon}{2}^\top\right)\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^\top\left(\boldsymbol{M}\boldsymbol{D}^{-1/2}\boldsymbol{R}_\epsilon\boldsymbol{D}^{1/2} + \boldsymbol{D}^{1/2}\boldsymbol{R}_\epsilon^\top\boldsymbol{D}^{-1/2}\boldsymbol{M}\right)\boldsymbol{x} \\
&= \boldsymbol{x}^\top\left(\frac{\boldsymbol{M}_\epsilon + \boldsymbol{M}_\epsilon}{2}^\top\right)\boldsymbol{x} + \boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{x}\cdot\frac{\boldsymbol{x}^\top\left(\boldsymbol{M}\boldsymbol{D}^{-1/2}\boldsymbol{R}_\epsilon\boldsymbol{D}^{1/2} + \boldsymbol{D}^{1/2}\boldsymbol{R}_\epsilon^\top\boldsymbol{D}^{-1/2}\boldsymbol{M}\right)\boldsymbol{x}}{2\boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{x}} \\
&\le \boldsymbol{x}^\top\left(\frac{\boldsymbol{M}_\epsilon + \boldsymbol{M}_\epsilon}{2}^\top\right)\boldsymbol{x} + \boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{x}\cdot\|\boldsymbol{D}^{-1/2}\boldsymbol{R}\boldsymbol{D}^{1/2}\|_2,
\end{aligned}$$

where the last inequality follows from Lemma B.2 of (33). Rearrange the above; we finish the first case.

2. $\boldsymbol{M} = 2\lambda\boldsymbol{S}^{-1/2}\boldsymbol{X}_{\mathcal{L}}\boldsymbol{S}^{1/2}$. Here,

$$\boldsymbol{X}_{\mathcal{L}} = (\alpha\boldsymbol{I} + \boldsymbol{D} - \boldsymbol{W})^{-1} \qquad \boldsymbol{M}_\epsilon = 2\lambda\boldsymbol{S}^{-1/2}\boldsymbol{X}_\epsilon\boldsymbol{S}^{1/2}, \qquad \boldsymbol{E}_\epsilon = 2\lambda\boldsymbol{S}^{-1/2}\alpha\boldsymbol{X}_{\mathcal{L}}\boldsymbol{R}_\epsilon\boldsymbol{S}^{1/2}.$$

Then

$$\begin{aligned}
\boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{x} &= \boldsymbol{x}^\top(\boldsymbol{M}_\epsilon + \boldsymbol{E}_\epsilon)\boldsymbol{x} \\
&= \boldsymbol{x}^\top\left(\frac{\boldsymbol{M}_\epsilon + \boldsymbol{M}_\epsilon}{2}^\top\right)\boldsymbol{x} + \frac{\alpha}{2}\boldsymbol{x}^\top\left(\boldsymbol{M}\boldsymbol{S}^{-1/2}\boldsymbol{R}_\epsilon\boldsymbol{S}^{1/2} + \boldsymbol{S}^{1/2}\boldsymbol{R}_\epsilon^\top\boldsymbol{S}^{-1/2}\boldsymbol{M}\right)\boldsymbol{x} \\
&= \boldsymbol{x}^\top\left(\frac{\boldsymbol{M}_\epsilon + \boldsymbol{M}_\epsilon}{2}^\top\right)\boldsymbol{x} + \boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{x}\cdot\frac{\alpha\boldsymbol{x}^\top\left(\boldsymbol{M}\boldsymbol{S}^{-1/2}\boldsymbol{R}_\epsilon\boldsymbol{S}^{1/2} + \boldsymbol{S}^{1/2}\boldsymbol{R}_\epsilon^\top\boldsymbol{S}^{-1/2}\boldsymbol{M}\right)\boldsymbol{x}}{2\boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{x}} \\
&\le \boldsymbol{x}^\top\left(\frac{\boldsymbol{M}_\epsilon + \boldsymbol{M}_\epsilon}{2}^\top\right)\boldsymbol{x} + \boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{x}\cdot\alpha\|\boldsymbol{S}^{-1/2}\boldsymbol{R}\boldsymbol{S}^{1/2}\|_2,
\end{aligned}$$

by applying Lemma B.2 with $\boldsymbol{D} = \boldsymbol{S}^{1/2}$.

Rearrange the above; we finish the proof. □

Remark B.4. The above theorem allows us to control the error of $\boldsymbol{M}_\epsilon$. Next, we show that if $\frac{\boldsymbol{M}_\epsilon+\boldsymbol{M}_\epsilon^\top}{2}$ is a positive semidefinite matrix, then we can find an approximate relaxation function.

Lemma B.5. *Let $\boldsymbol{G}^t = [\boldsymbol{\nabla}_1, \ldots, \boldsymbol{\nabla}_t, \boldsymbol{0}, \ldots, \boldsymbol{0}] \in \mathbb{R}^{k\times n}$ with $\|\boldsymbol{\nabla}_t\|_2 \le D$. Let $\boldsymbol{M}_\epsilon$ be the approximation built upon $\boldsymbol{X}_\epsilon$ as defined in Table 3 such that $\left(\frac{\boldsymbol{M}_\epsilon+\boldsymbol{M}_\epsilon^\top}{2}\right)$ is positive semidefinite. If we provide the following score*

$$\boldsymbol{\psi}_t = -\frac{\boldsymbol{G}^{t-1}(\boldsymbol{M}_\epsilon)_{:,t} + \boldsymbol{G}^{t-1}(\boldsymbol{M}_\epsilon)_{t,:}^\top}{\sqrt{Q_{t-1} + D^2\sum_{j=t}^n (M_\epsilon)_{j,j}}},$$

where $Q_t \triangleq \sum_{i=1}^k (\boldsymbol{G}_i^t)^\top\left(\frac{\boldsymbol{M}_\epsilon+\boldsymbol{M}_\epsilon^\top}{2}\right)\boldsymbol{G}_i^t$, then the relaxation function defined

$$\mathrm{Rel}_t\left(\boldsymbol{\nabla}_1, \ldots, \boldsymbol{\nabla}_t; \boldsymbol{M}_\epsilon\right) \triangleq \sqrt{Q_t + D^2\sum_{j=t+1}^n (M_\epsilon)_{j,j}}$$

is admissible; that is, for all $t = 1, 2, \ldots, n$,

$$\inf_{\boldsymbol{\psi}_t\in\mathbb{R}^k}\sup_{\|\boldsymbol{\nabla}_t\|\le D}\left\{\boldsymbol{\nabla}_t^\top\boldsymbol{\psi}_t + \mathrm{Rel}_t\left(\boldsymbol{\nabla}_1\ldots, \boldsymbol{\nabla}_t; \boldsymbol{M}_\epsilon\right)\right\} \le \mathrm{Rel}_{t-1}(\boldsymbol{\nabla}_1, \ldots, \boldsymbol{\nabla}_{t-1}; \boldsymbol{M}_\epsilon)$$

where $\mathrm{Rel}_0(\emptyset; \boldsymbol{M}_\epsilon) = \sqrt{D^2\cdot\mathrm{tr}\left(\boldsymbol{M}_\epsilon\right)}$.

Proof. Define $\boldsymbol{b}_i^t = [0, \ldots, \nabla_t(i), 0, \ldots, 0]^\top \in \mathbb{R}^n$. Note that $(\boldsymbol{G}_i^t)^\top = (\boldsymbol{G}_i^{t-1} + \boldsymbol{b}_i^t)^\top$ where $i = 1, 2, \ldots, k$ indexing label id and $t = 1, 2, \ldots, n$ indexing node id. We define the following quadratic based on $\boldsymbol{M}_\epsilon$

$$
\begin{aligned}
Q_t &\triangleq \sum_{i=1}^{k} \left(\boldsymbol{G}_i^t\right)^\top \left(\frac{\boldsymbol{M}_\epsilon + \boldsymbol{M}_\epsilon^\top}{2}\right) \boldsymbol{G}_i^t \\
&= \sum_{i=1}^{k} \left(\boldsymbol{G}_i^{t-1} + \boldsymbol{b}_i^t\right)^\top \left(\frac{\boldsymbol{M}_\epsilon + \boldsymbol{M}_\epsilon^\top}{2}\right) \left(\boldsymbol{G}_i^{t-1} + \boldsymbol{b}_i^t\right) \\
&= \sum_{i=1}^{k} \left\{ \left(\boldsymbol{G}_i^{t-1}\right)^\top \left(\frac{\boldsymbol{M}_\epsilon + \boldsymbol{M}_\epsilon^\top}{2}\right) \boldsymbol{G}_i^{t-1} + \left(\boldsymbol{G}_i^{t-1}\right)^\top \left(\boldsymbol{M}_\epsilon + \boldsymbol{M}_\epsilon^\top\right) \boldsymbol{b}_i^t + (\boldsymbol{b}_t^i)^\top \left(\frac{\boldsymbol{M}_\epsilon + \boldsymbol{M}_\epsilon^\top}{2}\right) \boldsymbol{b}_t^i \right\} \\
&= Q_{t-1} + \sum_{i=1}^{k} \left\{ \left(\boldsymbol{G}_i^{t-1}\right)^\top \boldsymbol{M}_\epsilon \boldsymbol{b}_i^t + (\boldsymbol{b}_i^t)^\top \boldsymbol{M}_\epsilon \boldsymbol{G}_i^{t-1} + \left(\nabla_t(i)\right)^2 (M_\epsilon)_{t,t} \right\} \\
&= Q_{t-1} + \sum_{i=1}^{k} \left(\nabla_t(i)((\boldsymbol{G}_i^{t-1})^\top (\boldsymbol{M}_\epsilon)_{:,t} + (\boldsymbol{M}_\epsilon)_{t,:}^\top \boldsymbol{G}_i^{t-1}) + (\nabla_t(i))^2 (M_\epsilon)_{t,t} \right),
\end{aligned}
$$

where $(\boldsymbol{M}_\epsilon)_{t,:}^\top$ is the t-th row vector of $\boldsymbol{M}_\epsilon$ and $(\boldsymbol{M}_\epsilon)_{:,t}$ is t-th column vector of $\boldsymbol{M}_\epsilon$. When $t = 1$, we initialize $Q_0 = 0$ and $\boldsymbol{G}^0 = \boldsymbol{0}_{k \times n}$. Finally, the recursion of the above is,

$$
Q_t = Q_{t-1} + \boldsymbol{\nabla}_t^\top \boldsymbol{G}^{t-1} (\boldsymbol{M}_\epsilon)_{:,t} + \boldsymbol{\nabla}_t^\top \boldsymbol{G}^{t-1} (\boldsymbol{M}_\epsilon)_{t,:} + (M_\epsilon)_{t,t} \cdot \|\boldsymbol{\nabla}_t\|_2^2,
$$

where $t = 1, 2, \ldots, n$. Now we can obtain an upper bound of the relaxation function Rel_t as the following ($t = 0, 1, 2, \ldots, n-1$)

$$
\begin{aligned}
\mathrm{Rel}_t\left(\boldsymbol{\nabla}_1, \ldots, \boldsymbol{\nabla}_t; \boldsymbol{M}_\epsilon\right) &\triangleq \sqrt{Q_t + D^2 \sum_{j=t+1}^{n} (M_\epsilon)_{j,j}} \\
&= \sqrt{Q_{t-1} + \boldsymbol{\nabla}_t^\top \boldsymbol{G}^{t-1} (\boldsymbol{M}_\epsilon)_{:,t} + \boldsymbol{\nabla}_t^\top \boldsymbol{G}^{t-1} (\boldsymbol{M}_\epsilon)_{t,:}^\top + (M_\epsilon)_{t,t} \cdot \|\boldsymbol{\nabla}_t\|^2 + D^2 \sum_{j=t+1}^{n} (M_\epsilon)_{j,j}} \\
&\leq \sqrt{Q_{t-1} + \boldsymbol{\nabla}_t^\top \boldsymbol{G}^{t-1} (\boldsymbol{M}_\epsilon)_{:,t} + \boldsymbol{\nabla}_t^\top \boldsymbol{G}^{t-1} (\boldsymbol{M}_\epsilon)_{t,:}^\top + D^2 \sum_{j=t}^{n} M_\epsilon(j,j)},
\end{aligned}
$$

where the above inequality step is due to $\|\boldsymbol{\nabla}_t\|_2^2 \leq D^2$. Hence, we have

$$
\begin{aligned}
&\inf_{\boldsymbol{\psi}_t \in \mathbb{R}^k} \sup_{\|\boldsymbol{\nabla}_t\| \leq D} \left\{ \boldsymbol{\nabla}_t^\top \boldsymbol{\psi}_t + \mathrm{Rel}_n\left(\boldsymbol{\nabla}_1, \ldots, \boldsymbol{\nabla}_t; \boldsymbol{M}_\epsilon\right) \right\} \\
&\leq \inf_{\boldsymbol{\psi}_t \in \mathbb{R}^k} \sup_{\|\boldsymbol{\nabla}_t\| \leq D} \left\{ \boldsymbol{\nabla}_t^\top \boldsymbol{\psi}_t + \sqrt{Q_{t-1} + \boldsymbol{\nabla}_t^\top \boldsymbol{G}^{t-1} (\boldsymbol{M}_\epsilon)_{:,t} + \boldsymbol{\nabla}_t^\top \boldsymbol{G}^{t-1} (\boldsymbol{M}_\epsilon)_{t,:}^\top + D^2 \sum_{j=t}^{n} (M_\epsilon)_{j,j}} \right\} \\
&\leq \sup_{\|\boldsymbol{\nabla}_t\| \leq D} \left\{ -\frac{\boldsymbol{\nabla}_t^\top \boldsymbol{G}^{t-1} (\boldsymbol{M}_\epsilon)_{:,t} + \boldsymbol{\nabla}_t^\top \boldsymbol{G}^{t-1} (\boldsymbol{M}_\epsilon)_{t,:}^\top}{\sqrt{Q_{t-1} + D^2 \sum_{j=t}^{n} M_\epsilon(j,j)}} + \sqrt{Q_{t-1} + \boldsymbol{\nabla}_t^\top \boldsymbol{G}^{t-1} (\boldsymbol{M}_\epsilon)_{:,t} + \boldsymbol{\nabla}_t^\top \boldsymbol{G}^{t-1} (\boldsymbol{M}_\epsilon)_{t,:}^\top + D^2 \sum_{j=t}^{n} (M_\epsilon)_{j,j}} \right\} \\
&\leq \sqrt{Q_{t-1} + D^2 \sum_{j=t}^{n} M_\epsilon(j,j)} = \mathrm{Rel}_n(\boldsymbol{\nabla}_1, \ldots, \boldsymbol{\nabla}_{t-1}; \boldsymbol{M}_\epsilon),
\end{aligned}
$$

where the first inequality is due to the upper bound of Rel_n, and the second is that we replace $\boldsymbol{\psi}_t$ by its definition. To see the last inequality, we can prove it in the following way: Letting $\boldsymbol{v} = \boldsymbol{G}^{t-1} (\boldsymbol{M}_\epsilon)_{:,t} + \boldsymbol{G}^{t-1} (\boldsymbol{M}_\epsilon)_{t,:}^\top, a = Q_{t-1} + D^2 \sum_{j=t}^{n} M_\epsilon(j,j)$, we can simplify the above equality as

$$
\sup_{\|\boldsymbol{\nabla}_t\| \leq D} \left\{ h(\boldsymbol{\nabla}_t) := -\frac{\boldsymbol{\nabla}_t^\top \boldsymbol{v}}{\sqrt{a}} + \sqrt{a + 2\boldsymbol{\nabla}_t^\top \boldsymbol{v}} \right\}. \tag{34}
$$

The function h is concave in $\boldsymbol{\nabla}_t$, and setting its gradient to 0 gives

$$\frac{\boldsymbol{v}}{\sqrt{a}} = \frac{\boldsymbol{v}}{\sqrt{a + \boldsymbol{\nabla}_t^T \boldsymbol{v}}} \iff \boldsymbol{\nabla}_t^T \boldsymbol{v} = 0$$

which is feasible in the domain $\|\boldsymbol{\nabla}_t\| \leq D$. In other words, $h(\boldsymbol{\nabla}_t) \leq \sqrt{a}$, for all $\boldsymbol{\nabla}_t \in \mathbb{R}^k$. This upper bound is always achievable by noticing that there exists $\boldsymbol{\nabla}_t$ such that $\boldsymbol{\nabla}_t^\top \boldsymbol{v} = 0$. □

Remark B.6. The main argument of the above proof follows from Rakhlin & Sridharan (2017). However, this proof differs in a way that unlike in Rakhlin & Sridharan (2017), we assume $\frac{\boldsymbol{M}_\epsilon + \boldsymbol{M}_\epsilon^\top}{2}$ is the one such that all eigenvalues are nonnegative. Even if $\boldsymbol{M}_\epsilon$ is asymmetric, we can still find a relaxation function accordingly.

C. Regret based on the estimation of $\boldsymbol{M}_{\lambda,\beta}$

Recall we consider the following online learning paradigm on $\mathcal{G}$: At each time t, a learner picks up a node v and makes a prediction $\hat{y}_v \in \mathcal{Y}$; then the true label y_v is revealed with a cost of corresponding 0-1 loss as $\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}) = 1 - \boldsymbol{y}^\top \hat{\boldsymbol{y}}$, the goal is to design an algorithm so that the learner makes as few mistakes as possible. Denote a prediction of $\mathcal{V}$ as $\widehat{\boldsymbol{Y}} = [\hat{\boldsymbol{y}}_1, \hat{\boldsymbol{y}}_2, \ldots, \hat{\boldsymbol{y}}_n] \in \mathcal{F}$ and true label configuration as $\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n] \in \mathcal{F}$ where the set of allowed label configurations $\mathcal{F} \triangleq \left\{\boldsymbol{F} \in \{0,1\}^{k \times n} : \boldsymbol{F}_{:,j}^\top \cdot \mathbf{1} = 1, \forall j \in \mathcal{V}\right\}$. Formally, the goal is to find an algorithm $\mathcal{A}$, which minimizes the following regret

$$\operatorname*{Reg}_{\mathcal{A},\boldsymbol{Y}}(\mathcal{F}) := \mathop{\mathbb{E}}_{\widehat{\boldsymbol{Y}} \sim \mathcal{A}} \sum_{t=1}^n \ell(\hat{\boldsymbol{y}}_t, \boldsymbol{y}_t) - \min_{\boldsymbol{F} \in \mathcal{F}} \sum_{t=1}^n \ell(\boldsymbol{f}_t, \boldsymbol{y}_t),$$

where the graph Laplacian constraint set is defined as $\mathcal{F} \triangleq \mathcal{F}_{\lambda,\boldsymbol{K}} = \left\{\boldsymbol{F} \in \mathcal{F} : \sum_{i=1}^k \boldsymbol{F}_i^\top \boldsymbol{K}^{-1} \boldsymbol{F}_i \leq \lambda\right\}$.

Before we present the main theorem, we shall state the important properties of $\phi(\cdot, \boldsymbol{y})$ defined in (4). We repeat these lemmas and their proofs as the following. In the rest, we denote ξ for ϕ when $y_t \notin S(\boldsymbol{\psi}_t))$.

Lemma C.1. *(Rakhlin & Sridharan, 2017) If we use the loss ϕ defined in (4), then we have the regret upper bounded by*

$$\begin{aligned}\operatorname*{Reg}_{\mathcal{A},\boldsymbol{Y}}(\mathcal{F}) &\leq \sum_{t=1}^n \phi_{\boldsymbol{\psi}_t}(\boldsymbol{\psi}_t, \boldsymbol{y}_t) - \inf_{\boldsymbol{F} \in \mathcal{F}} \sum_{t=1}^n \phi_{\boldsymbol{\psi}_t}(\boldsymbol{f}_t, \boldsymbol{y}_t) \\ &\leq \sum_{t=1}^n \boldsymbol{\nabla}\phi_{\boldsymbol{\psi}_t}(\boldsymbol{\psi}_t, \boldsymbol{y}_t)^\top \boldsymbol{\psi}_t - \inf_{\boldsymbol{F} \in \mathcal{F}} \sum_{t=1}^n \boldsymbol{\nabla}\phi_{\boldsymbol{\psi}_t}(\boldsymbol{\psi}_t, \boldsymbol{y}_t)^\top \boldsymbol{f}_t \triangleq B_n. \end{aligned} \tag{35}$$

Proof. For any time t, we have $\phi_{\boldsymbol{\psi}_t}(\boldsymbol{\psi}_t, \boldsymbol{y}_t)$ satisfies the following inequality (Rakhlin & Sridharan, 2017)

$$\begin{aligned}&\mathbb{E}_{\widehat{\boldsymbol{y}}_t \sim \boldsymbol{q}_t(\boldsymbol{\psi}_t)}\left[1\{\widehat{\boldsymbol{y}}_t \neq \boldsymbol{y}_t\}\right] - \left[\xi(\boldsymbol{f}_t, \boldsymbol{y}_t) 1\{\boldsymbol{y}_t \notin \mathcal{S}(\boldsymbol{\psi}_t)\} + 1\{\boldsymbol{y} \neq \boldsymbol{f}_t\} 1\{\boldsymbol{y}_t \in \mathcal{S}(\boldsymbol{\psi}_t)\}\right] \\ &\quad \leq \phi_{\boldsymbol{\psi}_t}(\boldsymbol{\psi}_t, \boldsymbol{y}_t) - \phi_{\boldsymbol{\psi}_t}(\boldsymbol{f}_t, \boldsymbol{y}_t),\end{aligned}$$

where $\xi(\cdot,\cdot)$ denote ϕ when $\boldsymbol{y}_t \notin S(\boldsymbol{\psi}_t)$. Summing over t from 1 to n, we obtain the first inequality (35). Notice $\phi_{\boldsymbol{\psi}}(\boldsymbol{g}, \boldsymbol{y})$ is a convex function over $\boldsymbol{g}$. Specifically, for any $\boldsymbol{\psi}$ and $\boldsymbol{y}$ and any $\boldsymbol{g}, \boldsymbol{h} \in \mathbb{R}^k$, we have that

$$\phi_{\boldsymbol{\psi}}(\boldsymbol{g}, \boldsymbol{y}) - \phi_{\boldsymbol{\psi}}(\boldsymbol{h}, \boldsymbol{y}) \leq \boldsymbol{\nabla}_{\boldsymbol{g}} \phi_{\boldsymbol{\psi}}(\boldsymbol{g}, \boldsymbol{y})^\top (\boldsymbol{g} - \boldsymbol{h}).$$

Let $\boldsymbol{g} = \boldsymbol{\psi}_t$ and $\boldsymbol{h} = \boldsymbol{f}_t$, we obtain the following

$$\sum_{t=1}^n \phi_{\boldsymbol{\psi}_t}(\boldsymbol{\psi}_t, \boldsymbol{y}_t) - \sum_{t=1}^n \phi_{\boldsymbol{\psi}_t}(\boldsymbol{f}_t, \boldsymbol{y}_t) \leq \sum_{t=1}^n \boldsymbol{\nabla}\phi_{\boldsymbol{\psi}_t}(\boldsymbol{\psi}_t, \boldsymbol{y}_t)^\top (\boldsymbol{\psi}_t - \boldsymbol{f}_t).$$

Taking the sup over both sides, we have

$$\sup_{\boldsymbol{F}\in\mathcal{F}}\left\{\sum_{t=1}^{n}\phi_{\boldsymbol{\psi}_t}\left(\boldsymbol{\psi}_t,\boldsymbol{y}_t\right)-\sum_{t=1}^{n}\phi_{\boldsymbol{\psi}_t}\left(\boldsymbol{f}_t,\boldsymbol{y}_t\right)\right\}\leq\sup_{\boldsymbol{F}\in\mathcal{F}}\left\{\sum_{t=1}^{n}\boldsymbol{\nabla}\phi_{\boldsymbol{\psi}_t}\left(\boldsymbol{\psi}_t,\boldsymbol{y}_t\right)^{\top}\left(\boldsymbol{\psi}_t-\boldsymbol{f}_t\right)\right\}$$

$$\sum_{t=1}^{n}\phi_{\boldsymbol{\psi}_t}\left(\boldsymbol{\psi}_t,\boldsymbol{y}_t\right)-\inf_{\boldsymbol{F}\in\mathcal{F}}\sum_{t=1}^{n}\phi_{\boldsymbol{\psi}_t}\left(\boldsymbol{f}_t,\boldsymbol{y}_t\right)\leq\sup_{\boldsymbol{F}\in\mathcal{F}}\left\{\sum_{t=1}^{n}\boldsymbol{\nabla}\phi_{\boldsymbol{\psi}_t}\left(\boldsymbol{\psi}_t,\boldsymbol{y}_t\right)^{\top}\left(\boldsymbol{\psi}_t-\boldsymbol{f}_t\right)\right\}$$

$$\sum_{t=1}^{n}\phi_{\boldsymbol{\psi}_t}\left(\boldsymbol{\psi}_t,\boldsymbol{y}_t\right)-\inf_{\boldsymbol{F}\in\mathcal{F}}\sum_{t=1}^{n}\phi_{\boldsymbol{\psi}_t}\left(\boldsymbol{f}_t,\boldsymbol{y}_t\right)\leq\sum_{t=1}^{n}\boldsymbol{\nabla}\phi_{\boldsymbol{\psi}_t}\left(\boldsymbol{\psi}_t,\boldsymbol{y}_t\right)^{\top}\boldsymbol{\psi}_t-\inf_{\boldsymbol{F}\in\mathcal{F}}\sum_{t=1}^{n}\boldsymbol{\nabla}\phi_{\boldsymbol{\psi}_t}\left(\boldsymbol{\psi}_t,\boldsymbol{y}_t\right)^{\top}\boldsymbol{f}_t.$$

We finish the proof. □

The above lemma tells us that there is a way to use surrogate loss ϕ_ψ to possibly obtain a regret if $\sum_{t=1}^{n}\boldsymbol{\nabla}_t^{\top}(\boldsymbol{\psi}_t-\boldsymbol{F}_{:,t})$ can be properly bounded. The next lemma tells us that if we can properly choose a method such that $\boldsymbol{\psi}_t$ satisfies the above lemma, then we have the following lemma. Following the notation in (Rakhlin & Sridharan, 2017), we define B_n as in (35).

Lemma C.2. *(Rakhlin & Sridharan, 2017) If the loss is defined in (4), we obtain the regret bound*

$$\sum_{t=1}^{n}\mathop{\mathbb{E}}_{\widehat{\boldsymbol{y}}_t\sim\boldsymbol{q}(\boldsymbol{\psi}_t)}1\left\{\widehat{\boldsymbol{y}}_t\neq\boldsymbol{y}_t\right\}\leq\inf_{\boldsymbol{F}\in\mathcal{F}}\left\{2\left(1-\frac{1}{k}\right)\sum_{t:\boldsymbol{y}_t\notin S(\boldsymbol{\psi}_t)}1\left\{\boldsymbol{f}_t\neq\boldsymbol{y}_t\right\}+\sum_{t:\boldsymbol{y}_t\in S(\boldsymbol{\psi}_t)}1\left\{\boldsymbol{f}_t\neq\boldsymbol{y}_t\right\}\right\}+B_n.$$

Proof. From the Lemma C.1, we have

$$\begin{aligned}\sum_{t=1}^{n}\mathop{\mathbb{E}}_{\widehat{\boldsymbol{y}}_t\sim\boldsymbol{q}(\boldsymbol{\psi}_t)}1\left\{\widehat{\boldsymbol{y}}_t\neq\boldsymbol{y}_t\right\}&\leq\inf_{\boldsymbol{F}\in\mathcal{F}}\left\{\sum_{t:\boldsymbol{y}_t\notin S(\boldsymbol{\psi}_t)}\xi\left(\boldsymbol{f}_t,\boldsymbol{y}_t\right)+\sum_{t:\boldsymbol{y}_t\in S(\boldsymbol{\psi}_t)}1\left\{\boldsymbol{f}_t\neq\boldsymbol{y}_t\right\}\right\}+B_n\\&=\inf_{\boldsymbol{F}\in\mathcal{F}}\left\{\sum_{t:\boldsymbol{y}_t\notin S(\boldsymbol{\psi}_t)}\xi\left(\boldsymbol{f}_t,\boldsymbol{y}_t\right)\mathop{\mathbb{E}}_{\widehat{\boldsymbol{y}}_t\sim\boldsymbol{q}(\boldsymbol{\psi}_t)}1\left\{\widehat{\boldsymbol{y}}_t\neq\boldsymbol{y}_t\right\}+\sum_{t:\boldsymbol{y}_t\in S(\boldsymbol{\psi}_t)}1\left\{\boldsymbol{f}_t\neq\boldsymbol{y}_t\right\}\right\}+B_n,\end{aligned}$$

where the equality holds is due to the fact that $\mathop{\mathbb{E}}_{\widehat{\boldsymbol{y}}_t\sim\boldsymbol{q}(\boldsymbol{\psi}_t)}1\left\{\widehat{\boldsymbol{y}}_t\neq\boldsymbol{y}_t\right\}=1$ when $y_t\notin S(\boldsymbol{\psi}_t)$. Furthermore, when $y_t\notin S(\boldsymbol{\psi}_t)$, $\xi(\boldsymbol{f}_t,\boldsymbol{y}_t)\geq 1$ and for any $j\in\mathcal{Y}, y_t\notin S(\boldsymbol{\psi}_t)$, we have

$$\begin{aligned}\xi\left(\boldsymbol{f}_t,\boldsymbol{y}_t\right)&=\frac{1+\max_{r:\boldsymbol{e}_r\neq\boldsymbol{y}_t}\left\{\boldsymbol{f}_t^{\top}\boldsymbol{e}_r-\boldsymbol{f}_t^{\top}\boldsymbol{y}_t\right\}}{1+1/|S(\boldsymbol{\psi}_t)|}\\&=\frac{2\cdot 1\{\boldsymbol{f}_t\neq\boldsymbol{y}_t\}}{1+1/|S(\boldsymbol{\psi}_t)|}\\&\leq 2\left(1-\frac{1}{k}\right)1\left\{\boldsymbol{f}_t\neq\boldsymbol{y}_t\right\},\end{aligned}$$

where the last inequality is due to $|S(\boldsymbol{\psi}_t)|\leq k-1$ when $y_t\notin S(\boldsymbol{\psi}_t)$. □

Algorithm 6 RELAXATION($\mathcal{G},\lambda,\boldsymbol{K}_\beta^{-1}$) (Rakhlin & Sridharan, 2017)

1: $T_1=\operatorname{tr}\left((\boldsymbol{M})\right), A_1=0, \boldsymbol{G}=[\boldsymbol{0},\boldsymbol{0},\ldots,\boldsymbol{0}]\in\mathbb{R}^{k\times n}, \boldsymbol{M}_{\lambda,\beta}=\left(\frac{\boldsymbol{K}_\beta^{-1}}{2\lambda}+\frac{\boldsymbol{I}}{2n}\right)^{-1}$
2: **for** $t=1,\ldots,n$ **do**
3: $\quad\boldsymbol{\psi}_t=-\boldsymbol{G}^t\boldsymbol{M}[:,t]/\sqrt{A_t+D^2\cdot T_t}$
4: $\quad$Predict $\hat{y}_t\sim q_t(\boldsymbol{\psi}_t)$ and get loss gradient $\boldsymbol{\nabla}_t=\begin{cases}\frac{\max_{r:\boldsymbol{e}_r\neq\boldsymbol{y}}\{\boldsymbol{e}_r-\boldsymbol{y}\}}{1+1/|\mathcal{S}(\psi)|} & \text{if } y_t\notin\mathcal{S}\left(\psi_t\right)\\ \frac{1}{|\mathcal{S}(\psi_t)|}\boldsymbol{1}_{\mathcal{S}(\psi_t)}-\boldsymbol{y}_t & \text{otherwise}\end{cases}$
5: $\quad$Update $\boldsymbol{G}[:,t]=\boldsymbol{\nabla}_t$
6: $\quad A_{t+1}=A_t+2\boldsymbol{\nabla}_t^{\top}\boldsymbol{G}^t\boldsymbol{M}[:,t]+m_{tt}\cdot\|\boldsymbol{\nabla}_t\|^2$
7: $\quad T_{t+1}=T_t-m_{tt}$
8: **end for**

If we use the above Algorithm 6, then we have the following lemmas directly from (Rakhlin & Sridharan, 2017)

Lemma C.3. *(Rakhlin & Sridharan, 2017) Let $\boldsymbol{G} = [\boldsymbol{\nabla}_1, \dots, \boldsymbol{\nabla}_n] \in \mathbb{R}^{k\times n}$. We have the following*

$$-\inf_{\boldsymbol{F}\in\mathcal{F}_{\lambda,\boldsymbol{K}}} \sum_{t=1}^{n} \boldsymbol{\nabla}_t^\top \boldsymbol{f}_t = \sqrt{\sum_{i=1}^{k} \boldsymbol{G}_i^\top \boldsymbol{M}_{\lambda,\beta} \boldsymbol{G}_i}, \tag{36}$$

where the pre-computed matrix is $\boldsymbol{M}_{\lambda,\beta} = \left(\frac{\boldsymbol{K}_\beta^{-1}}{2\lambda} + \frac{\boldsymbol{I}_n}{2n}\right)^{-1}$. Then we could have

$$-\inf_{\boldsymbol{F}\in\mathcal{F}_{\lambda,\boldsymbol{K}}} \sum_{t=1}^{n} \boldsymbol{\nabla}_t^\top \boldsymbol{f}_{:,t} \le -\inf_{\boldsymbol{F}\in\mathcal{F}_{\lambda,\boldsymbol{K}}} \sum_{t=1}^{n} \boldsymbol{\nabla}_t^\top \boldsymbol{F}_{:,t}. \tag{37}$$

The above lemma is from the following fact:

Lemma C.4. *Consider the following optimization problem*

$$\min_{\boldsymbol{F}\in\mathbb{R}^{k\times n}} \operatorname{tr}\left(\boldsymbol{F}\boldsymbol{Y}^\top\right) \text{ subject to } \operatorname{tr}\left(\boldsymbol{F}\boldsymbol{M}^{-1}\boldsymbol{F}^\top\right) \le 1, \tag{38}$$

where $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix and $\boldsymbol{Y} \in \mathbb{R}^{k\times n}$. (38) obtains the optimal at $\boldsymbol{F}^* = -\frac{\boldsymbol{Y}\boldsymbol{M}}{\sqrt{\operatorname{tr}(\boldsymbol{Y}\boldsymbol{M}\boldsymbol{Y}^\top)}}$, that is

$$\min_{\boldsymbol{F}\in\mathbb{R}^{k\times n}} \operatorname{tr}\left(\boldsymbol{F}\boldsymbol{Y}^\top\right) \ge -\sqrt{\operatorname{tr}\left(\boldsymbol{Y}\boldsymbol{M}\boldsymbol{Y}^\top\right)} \text{ with equality holds at } \boldsymbol{F}^*. \tag{39}$$

Proof. A possible Lagrangian can be defined as

$$\mathcal{L}(\boldsymbol{F};\lambda) = \operatorname{tr}\left(\boldsymbol{F}\boldsymbol{Y}^T\right) + \lambda(\operatorname{tr}\left(\boldsymbol{F}\boldsymbol{M}^{-1}\boldsymbol{F}^T\right) - 1) \tag{40}$$

and differentiating and setting the gradient to be 0, we have

$$\boldsymbol{\nabla}_{\boldsymbol{F}}\mathcal{L} = \boldsymbol{Y} + 2\lambda\boldsymbol{F}\boldsymbol{M}^{-1} = \boldsymbol{0}$$
$$\boldsymbol{Y} = -2\lambda\boldsymbol{F}\boldsymbol{M}^{-1}.$$

Denote $\lambda^* = \frac{\sqrt{\operatorname{tr}(\boldsymbol{Y}\boldsymbol{M}\boldsymbol{Y}^\top)}}{2}$, $\boldsymbol{F}^* = -\frac{\boldsymbol{Y}\boldsymbol{M}}{2\lambda^*}$. The rest of the proof is to show $(\boldsymbol{F}^*, \lambda^*)$ satisfies the KKT conditions. First of all, $\boldsymbol{0}_{k\times n} \in \boldsymbol{\nabla}_F\mathcal{L}$ when $\boldsymbol{F} = \boldsymbol{F}^*$ hence stationarity is satisfied. $\lambda^* > 0$, so it is dual feasible. $\lambda^*\left(\operatorname{tr}\left(\boldsymbol{F}^*\boldsymbol{M}^{-1}\boldsymbol{F}^{*\top}\right) - 1\right) = 0$, so it satisfies complementary slackness. Clearly, $\operatorname{tr}\left(\boldsymbol{F}^*\boldsymbol{M}^{-1}\boldsymbol{F}^{*\top}\right) - 1 \le 0$, so it is primal feasible. Hence the problem obtains the optimal at $\boldsymbol{F}^*$.

By applying $\boldsymbol{Y} = \boldsymbol{G}$, we have that $\boldsymbol{F}^* = -\frac{\boldsymbol{Y}_n\boldsymbol{M}}{\sqrt{\operatorname{tr}(\boldsymbol{Y}_n\boldsymbol{M}\boldsymbol{Y}_n^\top)}}$. □

From the above lemma C.4, we have the following upper bound of the defined regret.

Lemma C.5. *(Rakhlin & Sridharan, 2017) The regret has the following bound*

$$\mathop{\mathbb{E}}_{\widehat{\boldsymbol{Y}}\sim\mathcal{A}} \sum_{t=1}^{n} \ell(\hat{\boldsymbol{y}}_t, \boldsymbol{y}_t) - 2\left(1 - \frac{1}{k}\right) \min_{\boldsymbol{F}\in\mathcal{F}_{\lambda,\boldsymbol{K}}} \sum_{t=1}^{n} \ell(\boldsymbol{f}_t, \boldsymbol{y}_t) \le \sup_{\boldsymbol{F}\in\mathcal{F}_{\lambda,\boldsymbol{K}}} \left\{\sum_{t=1}^{n} \boldsymbol{\nabla}_1\ell(\boldsymbol{\psi}_t, \boldsymbol{y}_t)^\top(\boldsymbol{\psi}_t - \boldsymbol{f}_t)\right\}$$

We are ready to prove our main theorem.

Theorem 5.1 (Regret of RELAXATION with parameterized Kernel matrix $\boldsymbol{M}_\beta^{-1}$)**.** *Let $\hat{\boldsymbol{Y}}$ be the prediction matrix returned by* RELAXATION*, if the input label sequence $\boldsymbol{Y}$ has good pattern, meaning a strong assumption $\lambda = n^\gamma$ with $\gamma \in (0,1)$, then choosing $\beta = n^{\gamma-1}$ for kernel $\boldsymbol{K}_\beta^{-1} = \boldsymbol{I} - \beta\boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$ and $\beta = 1 - \frac{\lambda}{n}$ for $\boldsymbol{K}_\beta^{-1} = \beta\boldsymbol{I} + \boldsymbol{S}^{-1/2}\mathcal{L}\boldsymbol{S}^{-1/2}$, we have the following regret bound*

$$\mathop{\mathbb{E}}_{\widehat{\boldsymbol{Y}}\sim\mathcal{A}} \sum_{t=1}^{n} \ell(\hat{\boldsymbol{y}}_t, \boldsymbol{y}_t) - 2\left(1 - \frac{1}{k}\right) \min_{\boldsymbol{F}\in\mathcal{F}_{\lambda,\boldsymbol{K}}} \sum_{t=1}^{n} \ell(\boldsymbol{f}_t, \boldsymbol{y}_t) \le D\sqrt{2n^{1+\gamma}}. \tag{41}$$

Proof. Since $\mathcal{F}_{\lambda,\beta} \subseteq \bar{\mathcal{F}}_{\lambda,\beta}$, we continue to have an upper bound of the regret as

$$\begin{aligned}
\sup_{\boldsymbol{F}\in\mathcal{F}_{\lambda,\beta}} \left\{\sum_{t=1}^{n} \boldsymbol{\nabla}_1 \ell(\boldsymbol{\psi}_t, \boldsymbol{y}_t)^\top (\boldsymbol{\psi}_t - \boldsymbol{f}_t)\right\} &\leq \sup_{\boldsymbol{F}\in\bar{\mathcal{F}}_{\lambda,\beta}} \left\{\sum_{t=1}^{n} \boldsymbol{\nabla}_1 \ell(\boldsymbol{\psi}_t, \boldsymbol{y}_t)^\top (\boldsymbol{\psi}_t - \boldsymbol{f}_t)\right\} \\
&= \sum_{t=1}^{n} \boldsymbol{\nabla}_1 \ell(\boldsymbol{\psi}_t, \boldsymbol{y}_t)^\top \boldsymbol{\psi}_t - \inf_{\boldsymbol{F}\in\bar{\mathcal{F}}_{\lambda,\beta}} \left\{\sum_{t=1}^{n} \boldsymbol{\nabla}_1 \ell(\boldsymbol{\psi}_t, \boldsymbol{y}_t)^\top \boldsymbol{f}_t\right\} \\
&= \sum_{t=1}^{n} \boldsymbol{\nabla}_t^\top \boldsymbol{\psi}_t - \inf_{\boldsymbol{F}\in\bar{\mathcal{F}}_{\lambda,\beta}} \sum_{t=1}^{n} \boldsymbol{\nabla}_t^\top \boldsymbol{f}_t,
\end{aligned}$$

where we denote $\boldsymbol{\nabla}_1 \ell(\boldsymbol{\psi}_t, \boldsymbol{y}_t)$ as $\boldsymbol{\nabla}_t$ in the last equality. For the term $\sum_{t=1}^{n} \boldsymbol{\nabla}_t^\top \boldsymbol{\psi}_t$, from Lemma B.5, we know that if we choose

$$\boldsymbol{\psi}_t = -\frac{\boldsymbol{G}^{t-1}\boldsymbol{M}_{t,:}}{\sqrt{\sum_{i=1}^{k} \left(\boldsymbol{G}_i^{t-1}\right)^\top \boldsymbol{M}\boldsymbol{G}_i^{t-1} + D^2 \sum_{j=t}^{n} M(j,j)}},$$

where relaxation is defined as

$$\mathrm{Rel}_t\left(\boldsymbol{\nabla}_1, \ldots, \boldsymbol{\nabla}_t; \boldsymbol{M}\right) = \sqrt{\sum_{i=1}^{k} \left(\boldsymbol{G}_i^{t-1}\right)^\top \boldsymbol{M}\boldsymbol{G}_i^{t-1} + D^2 \sum_{j=t}^{n} M(j,j)}$$

satisfies

$$\boldsymbol{\nabla}_t^\top \boldsymbol{\psi}_t \leq \mathrm{Rel}_{t-1}(\boldsymbol{\nabla}_1, \ldots, \boldsymbol{\nabla}_{t-1}; \boldsymbol{M}) - \mathrm{Rel}_t(\boldsymbol{\nabla}_1, \ldots, \boldsymbol{\nabla}_t; \boldsymbol{M}), \quad \forall t = 1, 2, \ldots, n.$$

Then we continue to have

$$\begin{aligned}
\sum_{t=1}^{n} \boldsymbol{\nabla}_t^\top \boldsymbol{\psi}_t &\leq \sum_{t=1}^{n} \{\mathrm{Rel}_{t-1}(\boldsymbol{\nabla}_1, \ldots, \boldsymbol{\nabla}_{t-1}; \boldsymbol{M}) - \mathrm{Rel}_t(\boldsymbol{\nabla}_1, \ldots, \boldsymbol{\nabla}_t; \boldsymbol{M})\} \\
&= \mathrm{Rel}_0(\emptyset; \boldsymbol{M}) - \mathrm{Rel}_n(\boldsymbol{\nabla}_1, \ldots, \boldsymbol{\nabla}_n; \boldsymbol{M}) \\
&= D\sqrt{\sum_{i=1}^{n} m_{ii}} - \sqrt{\sum_{i=1}^{k} (\boldsymbol{G}_i^n)^\top \boldsymbol{M}\boldsymbol{G}_i^n}.
\end{aligned}$$

We are ready to provide the whole bound

$$\begin{aligned}
\mathrm{Reg}(\mathcal{F}_{\lambda,\boldsymbol{K}}) &\leq \sum_{t=1}^{n} \boldsymbol{\nabla}_t^\top \boldsymbol{\psi}_t - \inf_{\boldsymbol{F}\in\bar{\mathcal{F}}_{\lambda,\boldsymbol{K}}} \left\{\sum_{t=1}^{n} \boldsymbol{\nabla}_t^\top \boldsymbol{f}_t\right\} \\
&\leq D\sqrt{\sum_{i=1}^{n} m_{ii}} - \sqrt{\sum_{i=1}^{k} (\boldsymbol{G}_i^n)^\top \boldsymbol{M}\boldsymbol{G}_i^n} - \inf_{\boldsymbol{F}\in\bar{\mathcal{F}}_{\lambda,\beta}} \left\{\sum_{t=1}^{n} \boldsymbol{\nabla}_t^\top \boldsymbol{f}_t\right\} \\
&= D\sqrt{\sum_{i=1}^{n} m_{ii}} - \sqrt{\sum_{i=1}^{k} (\boldsymbol{G}_i^n)^\top \boldsymbol{M}\boldsymbol{G}_i^n} + \sqrt{\sum_{i=1}^{k} (\boldsymbol{G}_i^n)^\top \boldsymbol{M}\boldsymbol{G}_i^n} \\
&= D\sqrt{\sum_{i=1}^{n} m_{ii}}.
\end{aligned}$$

1. For the first parameterized kernel $\boldsymbol{K}_\beta^{-1} = \boldsymbol{I} - \beta \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$, we have $\boldsymbol{M}_{\lambda,\beta} = \frac{2\lambda n}{n+\lambda-n\beta}\boldsymbol{D}^{-1/2}\boldsymbol{X}_{\boldsymbol{L}}\boldsymbol{D}^{1/2}$ where $1-\alpha = \frac{n\beta}{n+\lambda}$, and the parameter $\beta \in (0, \frac{n+\lambda}{n})$. Note that the eigenvalue of $\boldsymbol{X}_{\boldsymbol{L}}$ be

$$\lambda(\boldsymbol{X}_{\boldsymbol{L}}) = \lambda\left(\alpha\left(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}\right)^{-1}\right) \in \left[\frac{\alpha}{2-\alpha}, 1\right]$$

We have

$$\lambda(\boldsymbol{M}_{\lambda,\beta}) = \frac{2\lambda n}{n+\lambda-n\beta}\lambda(\boldsymbol{X_L}) \in \frac{2\lambda n}{n+\lambda-n\beta}\left[\frac{\alpha}{2-\alpha}, 1\right].$$

Notice that when $\lambda = n^\gamma$ with $\gamma \in (0,1)$, there always exists such β so that $\lambda(\boldsymbol{M}_{\lambda,\beta})$ could be bounded by $2n^\gamma$, that is

$$\begin{aligned}\lambda(\boldsymbol{M}_{\lambda,\beta}) &\leq \frac{2\lambda n}{n+\lambda-n\beta} \\ &= \frac{2\lambda}{1+\lambda/n-\beta} \\ &= \frac{2n^\gamma}{1+n^\gamma/n-\beta} \\ &= 2n^\gamma,\end{aligned}$$

where we always choose $\beta = n^{\gamma-1}$. Note β is a valid parameter for our problem setting where the kernel matrix requires $\beta \in (0,1)$ and $\beta \in (0, 1+\frac{\lambda}{n})$. Notice that the trace of $\boldsymbol{M}$ is then bounded by

$$\operatorname{tr}\left((\boldsymbol{M}_{\lambda,\beta})\right) \leq 2n^{1+\gamma}.$$

Therefore, the final regret can be upper bounded as the following

$$\operatorname{Reg}(\mathcal{F}_{\lambda,\boldsymbol{K}}) \leq D\sqrt{\sum_{i=1}^n m_{ii}} \leq D\sqrt{2n^{1+\gamma}}. \tag{42}$$

2. Notice that our kernel matrix is $\boldsymbol{K}_\beta^{-1} = \beta\boldsymbol{I} + \boldsymbol{S}^{-1/2}\boldsymbol{L}\boldsymbol{S}^{-1/2}$ and corresponding $\boldsymbol{M}_{\lambda,\beta} = 2\lambda\boldsymbol{S}^{-1/2}\boldsymbol{X}_{\mathcal{L}}\boldsymbol{S}^{1/2}$ with $\alpha = \frac{n\beta+\lambda}{n}$ (see Table 1). By Lemma B.1, we have

$$\begin{aligned}\lambda(\boldsymbol{X}_{\mathcal{L}}) &= \lambda\left((\alpha\boldsymbol{I}+\boldsymbol{D}-\boldsymbol{W})^{-1}\right) \in \left[\frac{1}{\alpha+2D_{\max}}, \frac{1}{\alpha}\right] \\ \lambda(\boldsymbol{M}_{\lambda,\beta}) &= \lambda\left(2\lambda\boldsymbol{S}^{-1/2}\boldsymbol{X}_{\mathcal{L}}\boldsymbol{S}^{1/2}\right) \in \left[\frac{2\lambda}{\alpha+2D_{\max}}, \frac{2\lambda}{\alpha}\right].\end{aligned}$$

Whenever $\lambda = n^\gamma$ with $\gamma \in (0,1)$, we choose $\beta = 1 - \frac{\lambda}{n}$ so that $\alpha = 1$ and $\beta \in (0,1)$. Hence, the trace of $\boldsymbol{M}_{\lambda,\beta}$ can be bound by

$$\begin{aligned}\operatorname{tr}\left((\boldsymbol{M}_{\lambda,\beta})\right) = \sum_{i=1}^n \lambda_i(\boldsymbol{M}_{\lambda,\beta}) &\in \left[\frac{2n\lambda}{\alpha+2D_{\max}}, \frac{2n\lambda}{\alpha}\right] \\ &\in \left[\frac{2n\lambda}{1+2D_{\max}}, 2n\lambda\right].\end{aligned}$$

Therefore, we have

$$\operatorname{Reg}(\mathcal{F}_{\lambda,\boldsymbol{K}}) \leq D\sqrt{\sum_{i=1}^n m_{ii}} \leq D\sqrt{2n^{1+\gamma}}. \tag{43}$$

Combining (42) and (43), we finish the proof. □

Our final step is to prove the approximated bound for FASTONL. We state it as in the following theorem. In the rest, we simply define $M_\epsilon(i,i) = m_{ii}$.

Theorem 5.3 (Regret analysis of FASTONL with approximated parameterized kernel)**.** *Consider* FASTONL *presented in Algo. 3. If we call* FASTONL$(\mathcal{G}, \epsilon, \boldsymbol{K}_\beta, \lambda = n^\gamma)$ *and ϵ is chosen such that*

$$\left\|\boldsymbol{D}^{-1/2}\boldsymbol{R}_\epsilon\boldsymbol{D}^{1/2}\right\|_2 \leq 1, \quad \alpha\left\|\boldsymbol{D}^{-1/2}\boldsymbol{R}_\epsilon\boldsymbol{D}^{1/2}\right\|_2 \leq 1,$$

then we have the following regret bounded by

$$\mathrm{Reg} := \mathop{\mathbb{E}}_{\widehat{\boldsymbol{Y}}\sim\mathcal{A}} \sum_{t=1}^{n} \ell(\hat{\boldsymbol{y}}_t, \boldsymbol{y}_t) - \frac{2k-1}{k} \min_{\boldsymbol{F}\in\mathcal{F}_{\lambda,\boldsymbol{K}}} \sum_{t=1}^{n} \ell(\boldsymbol{f}_t, \boldsymbol{y}_t) \leq D\sqrt{(1+k^2)n^{1+\gamma}} \tag{44}$$

Proof. Since $\mathcal{F}_{\lambda,\boldsymbol{K}}$ has been relaxed to $\bar{\mathcal{F}}_{\lambda,\boldsymbol{K}}$, we have $\mathcal{F}_{\lambda,\boldsymbol{K}} \subseteq \bar{\mathcal{F}}_{\lambda,\boldsymbol{K}}$, the surrogate loss has been chosen, we continue to have an upper bound of the regret as

$$\begin{aligned}
\sup_{\boldsymbol{F}\in\mathcal{F}_{\lambda,\boldsymbol{K}}} \left\{ \sum_{t=1}^{n} \boldsymbol{\nabla}_1 \ell(\boldsymbol{\psi}_t, \boldsymbol{y}_t)^\top (\boldsymbol{\psi}_t - \boldsymbol{f}_t) \right\} &\leq \sup_{\boldsymbol{F}\in\bar{\mathcal{F}}_{\lambda,\boldsymbol{K}}} \left\{ \sum_{t=1}^{n} \boldsymbol{\nabla}_1 \ell(\boldsymbol{\psi}_t, \boldsymbol{y}_t)^\top (\boldsymbol{\psi}_t - \boldsymbol{f}_t) \right\} \\
&= \sum_{t=1}^{n} \boldsymbol{\nabla}_1 \ell(\boldsymbol{\psi}_t, \boldsymbol{y}_t)^\top \boldsymbol{\psi}_t - \inf_{\boldsymbol{F}\in\bar{\mathcal{F}}_{\lambda,\boldsymbol{K}}} \left\{ \sum_{t=1}^{n} \boldsymbol{\nabla}_1 \ell(\boldsymbol{\psi}_t, \boldsymbol{y}_t)^\top \boldsymbol{f}_t \right\} \\
&= \sum_{t=1}^{n} \boldsymbol{\nabla}_t^\top \boldsymbol{\psi}_t - \inf_{\boldsymbol{F}\in\bar{\mathcal{F}}_{\lambda,\boldsymbol{K}}} \sum_{t=1}^{n} \boldsymbol{\nabla}_t^\top \boldsymbol{f}_t,
\end{aligned}$$

where we denote $\boldsymbol{\nabla}_1 \ell(\boldsymbol{\psi}_t, \boldsymbol{y}_t)$ as $\boldsymbol{\nabla}_t$ in the last equality. For the term $\sum_{t=1}^{n} \boldsymbol{\nabla}_t^\top \boldsymbol{\psi}_t$, from Lemma B.5, we know that if we choose

$$\boldsymbol{\psi}_t = -\frac{\boldsymbol{G}^{t-1}(\boldsymbol{M}_\epsilon)_{:,t} + \boldsymbol{G}^{t-1}(\boldsymbol{M}_\epsilon)_{t,:}^\top}{\sqrt{\sum_{i=1}^{k} \left(\boldsymbol{G}_i^{t-1}\right)^\top \boldsymbol{M}_\epsilon \boldsymbol{G}_i^{t-1} + D^2 \sum_{j=t}^{n} M_\epsilon(j,j)}},$$

where relaxation is defined as

$$\mathrm{Rel}_t\left(\boldsymbol{\nabla}_1, \ldots, \boldsymbol{\nabla}_t; \boldsymbol{M}_\epsilon\right) = \sqrt{\sum_{i=1}^{k} \left(\boldsymbol{G}_i^{t-1}\right)^\top \boldsymbol{M}_\epsilon \boldsymbol{G}_i^{t-1} + D^2 \sum_{j=t}^{n} M_\epsilon(j,j)}$$

satisfies

$$\boldsymbol{\nabla}_t^\top \boldsymbol{\psi}_t \leq \mathrm{Rel}_{t-1}(\boldsymbol{\nabla}_1, \ldots, \boldsymbol{\nabla}_{t-1}; \boldsymbol{M}_\epsilon) - \mathrm{Rel}_t(\boldsymbol{\nabla}_1, \ldots, \boldsymbol{\nabla}_t; \boldsymbol{M}_\epsilon), \quad \forall t = 1, 2, \ldots, n.$$

Then we continue to have

$$\begin{aligned}
\sum_{t=1}^{n} \boldsymbol{\nabla}_t^\top \boldsymbol{\psi}_t &\leq \sum_{t=1}^{n} \{\mathrm{Rel}_{t-1}(\boldsymbol{\nabla}_1, \ldots, \boldsymbol{\nabla}_{t-1}; \boldsymbol{M}_\epsilon) - \mathrm{Rel}_t(\boldsymbol{\nabla}_1, \ldots, \boldsymbol{\nabla}_t; \boldsymbol{M}_\epsilon)\} \\
&= \mathrm{Rel}_0(\emptyset; \boldsymbol{M}_\epsilon) - \mathrm{Rel}_n(\boldsymbol{\nabla}_1, \ldots, \boldsymbol{\nabla}_n; \boldsymbol{M}_\epsilon) \\
&= \sqrt{D^2 \sum_{j=t}^{n} M_\epsilon(j,j)} - \sqrt{\sum_{i=1}^{k} (\boldsymbol{G}_i^n)^\top \boldsymbol{M}_\epsilon G_i^n}.
\end{aligned}$$

We are ready to provide the whole bound

$$\begin{aligned}
\mathrm{Reg}(\mathcal{F}_{\lambda,\mathcal{K}}) &\leq \sum_{t=1}^{n} \boldsymbol{\nabla}_t^\top \boldsymbol{\psi}_t - \inf_{\boldsymbol{F}\in\bar{\mathcal{F}}_{\lambda,\boldsymbol{K}}} \left\{ \sum_{t=1}^{n} \boldsymbol{\nabla}_t^\top \boldsymbol{f}_t \right\} \\
&\leq D\sqrt{\sum_{i=1}^{n} M_\epsilon(i,i)} - \sqrt{\sum_{i=1}^{k} (\boldsymbol{G}_i^n)^\top \boldsymbol{M}_\epsilon \boldsymbol{G}_i^n} - \inf_{\boldsymbol{F}\in\bar{\mathcal{F}}_{\lambda,\beta}} \left\{ \sum_{t=1}^{n} \boldsymbol{\nabla}_t^\top \boldsymbol{f}_t \right\} \\
&= D\sqrt{\sum_{i=1}^{n} M_\epsilon(i,i)} \underbrace{- \sqrt{\sum_{i=1}^{k} (\boldsymbol{G}_i^n)^\top \boldsymbol{M}_\epsilon \boldsymbol{G}_i^n} + \sqrt{\sum_{i=1}^{k} (\boldsymbol{G}_i^n)^\top \boldsymbol{M} \boldsymbol{G}_i^n}}_{E}.
\end{aligned}$$

Notice that when $E \leq 0$, we have the regret $D\sqrt{\sum_{i=1}^n M_\epsilon(i,i)}$, which is less than $D\sqrt{\sum_{i=1}^n m_{ii}}$. In the rest, we will assume $E > 0$ and show E is not too large when $\boldsymbol{R}_\epsilon$ is small enough.

$$
\begin{aligned}
E &\leq \sqrt{\sum_{i=1}^k \left(\boldsymbol{G}_i^n\right)^\top \left(\boldsymbol{M}_{\lambda,\beta} - \boldsymbol{M}_\epsilon\right) \boldsymbol{G}_i^n} \\
&= \sqrt{\sum_{i=1}^k (\boldsymbol{G}_i^n)^\top \boldsymbol{G}_i^n \cdot \frac{\left(\boldsymbol{G}_i^n\right)^\top \left(\boldsymbol{M}_{\lambda,\beta} - \boldsymbol{M}_\epsilon\right) \boldsymbol{G}_i^n}{(\boldsymbol{G}_i^n)^\top \boldsymbol{G}_i^n}} \\
&\leq \sqrt{knD^2 \cdot \sum_{i=1}^k \frac{\left(\boldsymbol{G}_i^n\right)^\top \left(\boldsymbol{M}_{\lambda,\beta} - \boldsymbol{M}_\epsilon\right) \boldsymbol{G}_i^n}{(\boldsymbol{G}_i^n)^\top \boldsymbol{G}_i^n}},
\end{aligned} \tag{45}
$$

where the first inequality is because $\|\boldsymbol{\nabla}_t\|_2^2 \leq D^2$. Hence, $\|\boldsymbol{G}_i^n\|_2^2 \leq nD^2$. The last inequality is due to the Rayleigh quotient property. Recall from Thm. B.3, we have

$$
\boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{x} - \boldsymbol{x}^\top \boldsymbol{M}_\epsilon \boldsymbol{x} \leq \boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{x} \cdot \sqrt{\frac{\lambda_{\max}(\boldsymbol{M})}{\lambda_{\min}(\boldsymbol{M})}} \|\boldsymbol{D}^{-1/2}\boldsymbol{R}\boldsymbol{D}^{1/2}\|_2 \leq \boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{x},
$$

which means

$$
\sum_{i=1}^k \frac{\boldsymbol{G}_i^{n\top} \boldsymbol{M} \boldsymbol{G}_i^n - \boldsymbol{G}_i^{n\top} \boldsymbol{M}_\epsilon \boldsymbol{G}_i^n}{\boldsymbol{G}_i^{n\top} \boldsymbol{G}_i^n} \leq \sum_{i=1}^k \frac{\boldsymbol{G}_i^{n\top} \boldsymbol{M} \boldsymbol{G}_i^n}{\boldsymbol{G}_i^{n\top} \boldsymbol{G}_i^n} \leq k\lambda_{\max}(\boldsymbol{M}) \leq kn^\gamma. \tag{46}
$$

Combining (45) and (46), we have

$$
\mathrm{Reg}(\mathcal{F}_{\lambda,\boldsymbol{K}}) \leq \sqrt{D^2 \sum_{i=1}^n M_\epsilon(i,i) + D^2k^2n^{1+\gamma}} \leq D\sqrt{(1+k^2)n^{1+\gamma}}.
$$

□

C.1. Practice Implementation

In practice, we use $T_0 = k \cdot n^2$ and use $\boldsymbol{x}_t$ to estimate the both t-th column and row vector of $(\boldsymbol{M}_\epsilon + \boldsymbol{M}_\epsilon^\top)/2$. This upper bound works well in all our experiments. In the following, $m_{tt} = (M_\epsilon)_{tt}$.

Algorithm 7 FASTONL($\mathcal{G}, \epsilon, \boldsymbol{K}_\beta^{-1}, \lambda, \Gamma$)

$\boldsymbol{G}^0 = [\boldsymbol{0}, \boldsymbol{0}, \ldots, \boldsymbol{0}] \in \mathbb{R}^{k\times n}$
$A_1 = 0$ and let $\alpha, \boldsymbol{M}_\epsilon$ be defined in Table 3
$T_1 = \Gamma$ // Big enough
for $t = 1, 2, \ldots, n$ **do**
  $\boldsymbol{x}_t, \boldsymbol{r}_t =$FIFOPUSH$(\mathcal{G}, \epsilon, \alpha, t)$
  Compute $(\boldsymbol{M}_\epsilon)_{:,t}$ on based on Tab. 1
  $\boldsymbol{\psi}_t = -\boldsymbol{G}^t(\boldsymbol{M}_\epsilon)_{:,t}/\sqrt{A_t + k \cdot T_t}$
  $\boldsymbol{\nabla}_t = \begin{cases} \frac{\max_{r:\boldsymbol{e}_r \neq \boldsymbol{y}}\{\boldsymbol{e}_r - \boldsymbol{y}\}}{1+1/|\mathcal{S}(\psi)|} & \text{if } y_t \notin \mathcal{S}(\psi_t) \\ \frac{1}{|\mathcal{S}(\psi_t)|}\boldsymbol{1}_{\mathcal{S}(\psi_t)} - \boldsymbol{y}_t & \text{otherwise} \end{cases}$
  Update gradient $\boldsymbol{G}_{:,t}^t = \boldsymbol{\nabla}_t$
  $A_{t+1} = A_t + 2\boldsymbol{\nabla}_t^\top \boldsymbol{v} + m_{tt} \cdot \|\boldsymbol{\nabla}_t\|_2^2$
  $T_{t+1} = T_t - m_{tt}$
end for

D. More experimental details and results

D.1. Experimental setups

We implemented all methods using Python and used the inverse function of scipy library to compute the matrix inverse. All experiments are conducted on a server with 40 cores and 250GB of memory.

D.2. Dataset Description

Table 4: Datasets Statistics

	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{Y}\rvert$	Weighted
Political	1,222	1,222	16,717	2	No
Cora	2,485	2,485	5,069	7	No
Citeseer	2,110	2,110	3,668	6	No
PubMed	19,717	19,717	44,324	3	No
MNIST	12,000	12,000	97,089	10	Yes
BlogCatalog	10,312	10,312	333,983	39	No
Flickr	80,513	80,513	5,899,882	195	No
OGB-Arxiv	169,343	169,343	1,157,799	40	No
YouTube	1,134,890	31,684	2,987,624	47	No
OGB-Products	2,385,902	500,000	61,806,303	47	No
Wikipedia	6,216,199	150,000	177,862,656	10	No

We list all ten graph datasets in Tab. 4. where $|\mathcal{L}|$ is the number of available labeled nodes.

1. Political (Adamic & Glance, 2005). This political blog graph contains 1,490 nodes and 16,715 edges. Each node represents a web blog on US politics, and the label belongs to either Democratic or Republican. We collect the largest connected component, including 1,222 nodes and 16,717 edges.

2. Cora (Sen et al., 2008). The Cora graph has 2,708 nodes and 5,278 edges. The label of each node belongs to a set of 7 categories of computer science research areas, including Neural Networks, Rule Learning, Reinforcement Learning, Probabilistic Methods, Theory, Genetic Algorithms, Case Based.

3. Citeseer (Sen et al., 2008). The Citeseer graph contains 3,312 nodes, including label sets (Agents, IR, DB, AI, HCI, ML). Since it contains many small connected components, we remain the largest connected component with 2,110 nodes and 3,668 edges as the input graph.

4. PubMed (Namata et al., 2012). The PubMed graph includes 19,717 edges and 44,324 edges. Each node belongs to one of three types of diabetes.

5. MNIST (Rakhlin & Sridharan, 2017). We downloaded MNIST with background noise images from `https://sites.google.com/a/lisa.iro.umontreal.ca/public_static_twiki/variations-on-the-mnist-digits`, which includes 12,000 noisy images from digit 0 to 9. To create edges between these images, we first create a 10-nearest neighbors graph and then create the edge weights by using the averaged ℓ_2 distance as suggested in Cesa-Bianchi et al. (2013).

6. BlogCataglog, Flickr, and Youtube datasets are found in (Perozzi et al., 2014).

7. OGB-Arix and OGB-Products are from OGB dataset (Hu et al., 2020)

8. Wikipedia. We download the raw corpus of English Wikipedia from `https://dumps.wikimedia.your.org/enwiki/20220820/` until the end of the year 2020. We create the inner-line edges for each Wikipedia article by checking the interlinks from these articles. Thus, we created the Wikipedia graph with 6,216,199 nodes and 177M edges. We then collect labels from `https://dbpedia.org/ontology/` where we can get 3,448,908 available node labels (only use first 150,000 nodes in our experiments) from ten categories including, Person, Place, Organisation, Work, MeanOfTransportation, Event, Species, Food, TimePeriod, Device.

D.3. Baseline Methods

We describe all four baseline methods and our method as follows

1. RELAXATION (Rakhlin & Sridharan, 2017). It has a parameter λ. To have the best performance, we choose $\boldsymbol{M}_{\lambda,\beta} = 2n\boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{1/2}$ as the underlying kernel matrix. In our experiments, we tune the parameter λ from $\{0.1 * n, 0.2 * n, \ldots, 0.9 * n\}$.

2. APPROXIMATE as defined in (7). It has parameter $p = 5$ in our small graph experiments. The reason that APPROXIMATE works well is that we use our FASTONL framework to predict labels. In other words, the only difference from our method is that APPROXIMATE use (7) to obtain kernel vectors. In contrast, we use FIFOPUSH to obtain kernel vectors. Similarly, we find the second kernel works great. Hence, we choose the second kernel as the underlying kernel to approximate. We found memory issues when we apply this approximation method to middle-scale graphs. As illustrated in Fig. 15, the approximated matrix becomes dense when $p \geq 10$ for most of the small graphs. For example, the approximated solution for the Blogcatalog dataset becomes dense only after 3 iterations. This suggests that APPROXIMATE is unsuitable for large-scale networks.

3. WM is the weighted-majority method. We implemented it as follows: 1) If u_t is the target node to be predicted, then the algorithm first finds all its available neighbors (nodes that have been seen in some previous iterations). Based on its neighbors, one can create a distribution of it. We just the maximal likelihood to choose the prediction label. We randomly select a label from the true label sets if no neighbors are found.

4. WTA is the weighted tree algorithm (Cesa-Bianchi et al., 2013). The essential idea of WTA is that the algorithm first constructs a random spanning tree (in our implementation, we choose to implement Wilson's algorithm described in Wilson (1996) to generate a random spanning tree. In expectation, the run time is linear to the number of nodes. One important parameter of WTA, we use $s = 5$ for all small-scale graphs. Our Python implementation is too slow for middle-scale graphs to finish one graph without several hours. However, we still see a large gap between WTA and ours.

5. FASTONL has two important parameter, including the label smoothness parameter λ where we choose the same as did in RELAXATION, and the precision ϵ to control the quality of kernel vectors. Except for the small-scale graphs. In all middle-scale graphs, we choose $\epsilon = .1/n$, which is good enough for node labeling tasks. To see this ϵ works well in general, we fix to use the second kernel and set λ based on the best value shown in Fig. 6. The results are shown in Fig. 16. This precision setting is good enough for most of these small datasets. In Fig. 6, we use the first 4 kernel matrices defined in Tab. 1. We name these kernels as $\boldsymbol{K}_1, \boldsymbol{K}_2, \boldsymbol{K}_3$, and $\boldsymbol{K}_4$, respectively. For $\boldsymbol{K}_3$ and $\boldsymbol{K}_4$, we directly use the defined β for our experiments.

Run time comparison between FASTONL and RELAXATION. We conduct the experiment to compare these two methods. The table below shows the run time (in seconds) where the total time is to predict all n nodes. Recall the size of the graph is between $n = 1,222$ (Political) to $n = 169,343$ (OGB-Arxiv). We fix $\lambda = 0.15n, \alpha = 1 - \frac{n}{n+\lambda}$, and use $\boldsymbol{K}_2$. The parameter $\epsilon = 0.1/n$. We perform experiments per data 10 times and take the average. As presented in Tab. 5 and 6, it is evident that FASTONL is more efficient in most of datasets. Importantly, this increased efficiency does not come with a significant trade-off in accuracy.

Table 5: Run time of two methods in seconds

	Political	Citeseer	Cora	Blogcatalog	MNIST	Pubmed	OGB-Arxiv
RELAXATION	**0.1417**	1.0222	1.5480	87.8246	135.3061	585.3558	Out of Memory
FASTONL	0.1557	**0.5883**	**1.0971**	**5.9510**	**60.0465**	**50.7256**	**3199.6773**

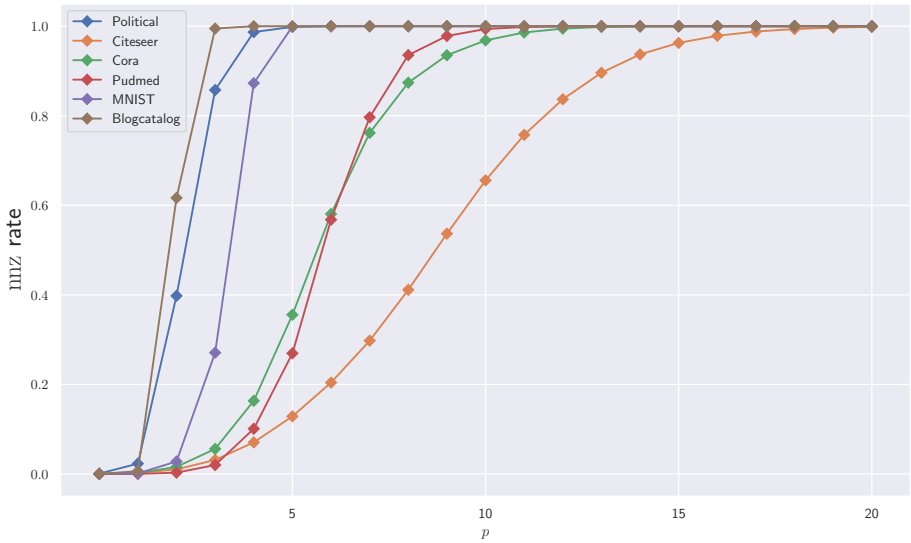


Figure 15: The nnz rate as a function of iteration p for APPROXIMATE. The nnz rate, i.e., the sparsity rate of the approximated matrix, is defined as $\mathrm{nnz}\left(\boldsymbol{M}_p\right)/n^2$ where $\mathrm{nnz}\left(\boldsymbol{M}_p\right)$ is the number of nonzero entries in $\boldsymbol{M}_p$ after p iterations. Most of these approximations become dense after 10 iterations.

Table 6: Accuracy of two methods

	Political	Citeseer	Cora	Blogcatalog	MNIST	Pubmed	OGB-Arxiv
RELAXATION	0.9493	0.7415	0.8404	0.2192	0.7930	0.8254	-
FASTONL	0.9418	0.7404	0.8420	0.2921	0.7960	0.8257	0.7089

Power law of magnitudes of $\boldsymbol{x}_s$. We close up this section by showing the magnitudes of $\boldsymbol{x}_s$ of both $\boldsymbol{X}_{\mathcal{L}}$ and $\boldsymbol{X}_{\boldsymbol{L}}$ follow the power law distribution as we illustrate in Fig. 17 and 18.

Method	Regret	Per-Iteration	Total-Time	Memory
WM	-	$\mathcal{O}\left(d_u\right)$	$\mathcal{O}(m)$	$\mathcal{O}(m)$
Perceptron	$\Phi G(\mathbf{y}) \cdot \left(\frac{n}{n^{\dagger} n^{-}}\right)^2$	$\mathcal{O}(n)$	$\mathcal{O}\left(n^3\right)$	$\mathcal{O}\left(n^2\right)$
WTA	$\Phi G(\mathbf{y}) \cdot \log n$	$\mathcal{O}(s)$	$\mathcal{O}(s \cdot n \log n)$	$\mathcal{O}(s \cdot n)$
RELAXATION	$\mathcal{O}(\sqrt{\mathrm{tr}(\mathbf{M})})$	$\mathcal{O}(n)$	$\mathcal{O}\left(n^3\right)$	$\mathcal{O}\left(n^2\right)$
Ours	$\mathcal{O}(\sqrt{n^{1+\gamma}})$	$\mathcal{O}\left(\frac{\mathcal{S}_T}{\alpha \cdot \eta T} \log^{3/2} n\right)$	$\mathcal{O}\left(\frac{n \cdot \mathcal{S}_T}{\alpha \cdot \eta T} \log^{3/2} n\right)$	$\mathcal{O}(\mathrm{supp}(\mathbf{x}) \cdot n)$

Table 7: Comparison of time and memory complexity between these online node labeling algorithms.

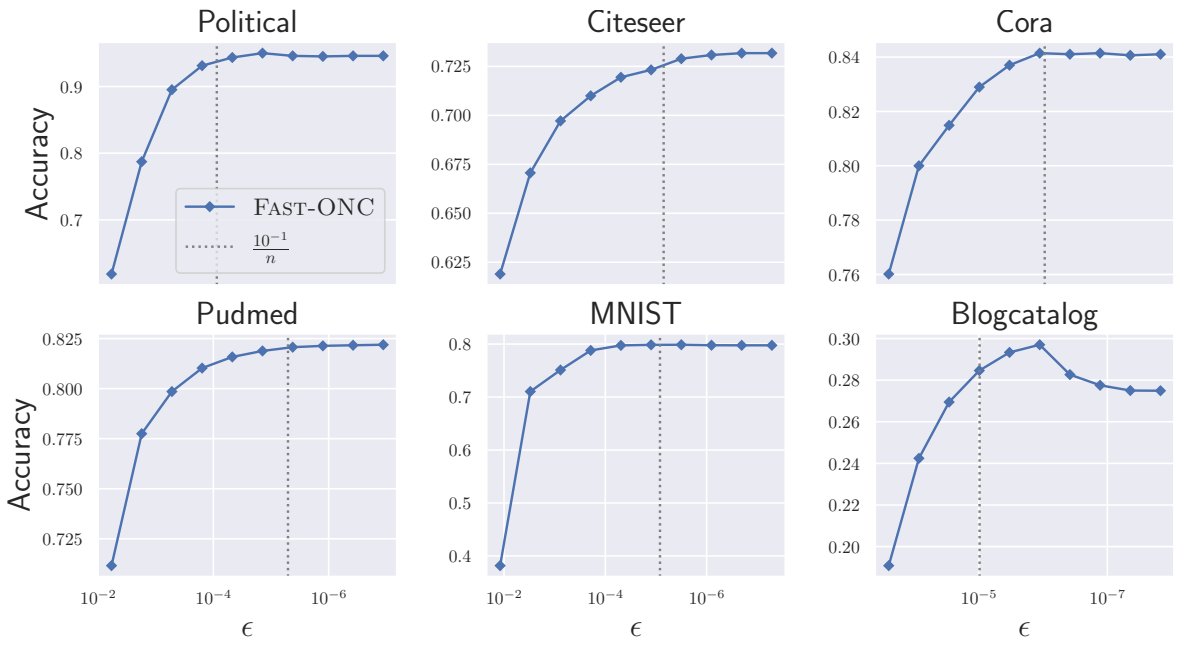


Figure 16: The overall node labeling accuracy as a function of ϵ over all six small graphs. The vertical line is when $\epsilon = \frac{10^{-1}}{n}$ corresponding to the ϵ we set in our experiments. This rough parameter estimation is good enough for most of the datasets.

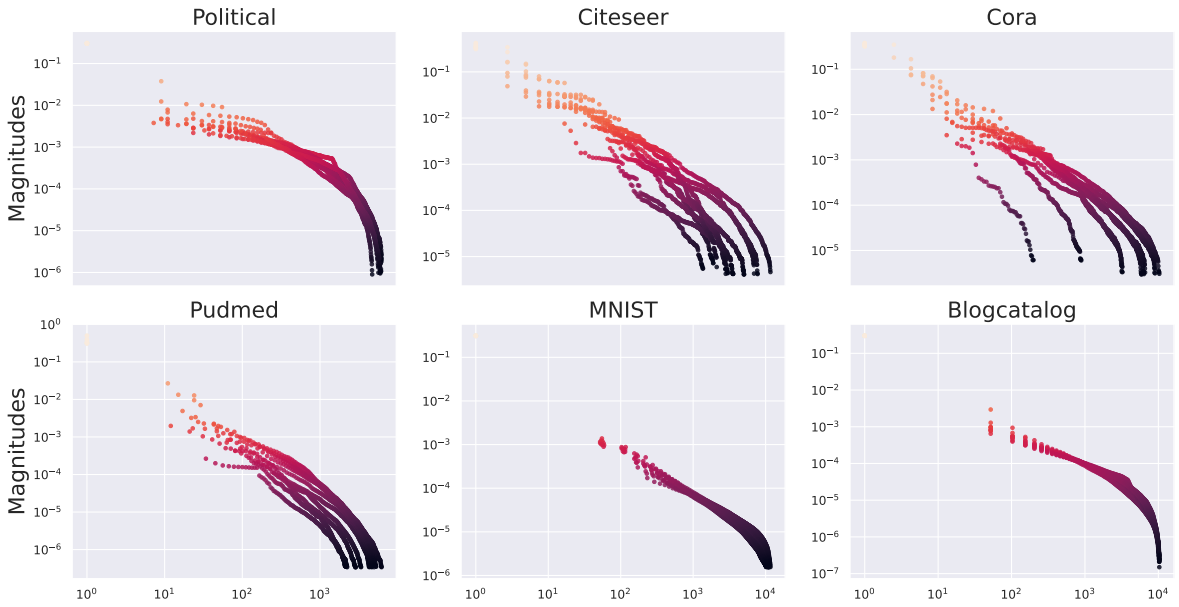


Figure 17: The power law distribution of magnitudes of $\boldsymbol{X_L}$ on six small graphs. We set $\epsilon = 10^{-1}/m$ and fix $\alpha = 0.2$. We then randomly select 20 nodes corresponding to 20 columns of $\boldsymbol{X_L}$, sort the magnitudes of $\boldsymbol{x}_s$, and plot them according to their rankings. These magnitudes follow the power law distribution.

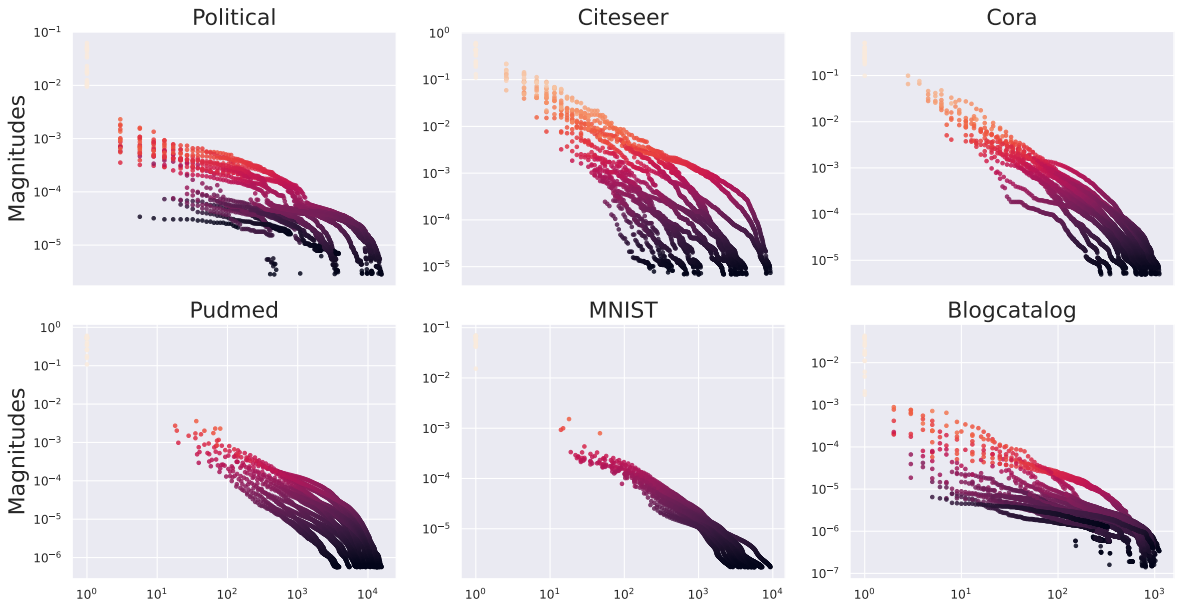


Figure 18: The power law distribution of magnitudes in $\boldsymbol{X}_{\mathcal{L}}$ on six small graphs. We set $\epsilon = 10^{-1}/m$ and fix $\alpha = 0.2$. We then randomly select 20 nodes corresponding to 20 columns of $\boldsymbol{X}_{\mathcal{L}}$, sort the magnitudes of $\boldsymbol{x}_s$, and plot them according to their rankings. These magnitudes follow the power law distribution.