Adaptive Online Graph Learning

Xiang Zhang, *Student Member, IEEE* and Qiao Wang, *Senior Member, IEEE*

***Abstract*—We consider the problem of learning graphs from smooth signals in an online fashion. A major challenge of this task is that the underlying graph may be time-varying. Existing works typically assume fixed model parameters to learn dynamic graphs, which may suffer the from model mismatch problem and obtain suboptimal results. To this end, we propose an online graph learning approach that handles model mismatch problem by combining multiple expert algorithms via an expert-tracking algorithm. Theoretically, we prove that our algorithm can obtain sublinear dynamic regret, a common metric for evaluating online algorithms. Experimental results of synthetic and real data exhibit the superiority of our method.**

***Index Terms*—Graph learning, online learning, dynamic environments, regret analysis**

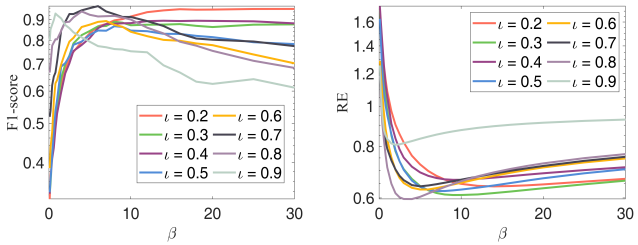
I. Introduction

Graph learning (GL) is a fundamental task that can infer graph topologies behind irregular datasets [1], [2] and plays a critical role in graph signal processing (GSP) [3] and graph machine learning [4]. In the literature, extensive studies have been conducted on GL, including statistical models [5], [6] and causality-based models [7], [8]. Recently, a plethora of works have attempted to learn graphs based on the assumption that the observed signals are smooth over the underlying graphs [9], [10]. In practice, miscellaneous signals enjoy the smoothness property, such as meteorology data [9] and image data [10], indicating many potential applications.

Here, we focus on the problem of learning graphs from smooth signals in an online fashion. The desideratum of this learning schema naturally arises when data are sequentially available and the underlying graphs are time-varying, such as tracking social networks [11] and financial relationships [12] from streaming data. Existing online graph learning (OGL) methods include statistical [13], [14] methods and causality-based [11], [15], [16] methods. The first work on OGL from smooth data is [12]. The subsequent works [17], [18] utilize a prediction step to better track topological changes. Then, [19] proposes an online dual-based proximal-gradient algorithm, and [20] proposes an online proximal ADMM algorithm.

Albeit effective, existing methods may suffer from model (parameter) mismatch. Specifically, the smoothness-based GL can be viewed as solving a regularized Laplacian minimization problem, in which some parameters are used to control graph properties such as sparsity [9] and connectivity [10] of the learned graph. Figure 1 depicts the performance of the GL method in [10] with different β values, which are used to control the sparsity of the learned graphs. It is observed that the optimal β depends on the sparsity levels, ι, of the real graph and may vary widely for different ι. However, it is impractical to choose a sequence of optimal parameters in advance when the environmental variability is unknown a priori. Existing OGL methods typically pre-select model parameters and keep them constant during algorithm execution. This practice is unreasonable and may cause suboptimal results since the dynamic graph may evolve far from the initial one.

The authors are with the School of Information Science and Engineering, Southeast University, China ({xiangzhang369, qiaowang}@seu.edu.cn).



(a) F1-score, the higher the better (b) Relative error, the lower the better

Fig. 1: The performance of the graphs learned by the method in [10], where β is the model parameter. We generate underlying graphs by the method in [9], where ι is used to control the sparsity. The larger ι is, the sparser the real graph is. It is observed that for real graphs with different sparsity, the corresponding optimal β varies widely. Thus, it is not reasonable to learn dynamic graphs with fixed β.

To address this issue, we propose an algorithm to handle the model mismatch problem in online graph learning. Our algorithm maintains a set of expert algorithms equipped with different model parameters. We propose an expert-tracking algorithm [21]–[25] to combine these expert algorithms, where the combination weights are adjusted according to the dynamic environment. Furthermore, we theoretically prove that our algorithm can reach an $\mathcal{O}\left(\sqrt{T}(1+V_T)\right)$ dynamic regret, a common metric for evaluating online algorithms [26], [27], where T is the number of iterations and V_T is a metric measuring environmental variability (see Section IV). Thus, the proposed algorithm can achieve sublinear dynamic regret with sublinear V_T. Finally, experimental results on synthetic and real data exhibit the superiority of our method.

II. Problem Formulation

In this section, we first present the background of GL from smooth signals and then formally state the problem of concern.

A. Batch Graph Learning From Smooth Signals

We consider undirected graphs with non-negative edges and no self-loops. The topology of such a graph $\mathcal{G}$ can be represented by its adjacency matrix $\mathbf{W} \in \mathbb{R}^{d\times d}$, and the set $\mathcal{W}$ containing all possible adjacency matrices is defined as

$$\mathcal{W} := \left\{\mathbf{W} : \mathbf{W} \geq 0, \mathbf{W} = \mathbf{W}^\top, \mathrm{diag}(\mathbf{W}) = \mathbf{0}\right\}, \quad (1)$$

where $\mathbf{W} \geq 0$ means that all elements of $\mathbf{W}$ are non-negative, and $\mathbf{0} \in \mathbb{R}^d$ is a vector of zeros. We study a graph signal $\mathbf{x} = [\mathbf{x}[1], ..., \mathbf{x}[d]]^\top \in \mathbb{R}^d$ associated with $\mathcal{G}$, where $\mathbf{x}[i]$ is the signal value of the i-th node in $\mathcal{G}$. Given T observed signals $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T] = [\widetilde{\mathbf{x}}_1^\top, \ldots, \widetilde{\mathbf{x}}_d^\top]^\top \in \mathbb{R}^{d\times T}$, the smoothness

of the observed signals $\mathbf{X}$ over $\mathcal{G}$ is [10]

$$\sum_{t=1}^{T} \mathbf{x}_t^\top \mathbf{L}\mathbf{x}_t = \text{trace}\left(\mathbf{X}^\top \mathbf{L}\mathbf{X}\right) = \frac{1}{2}\|\mathbf{W} \circ \mathbf{Z}\|_{1,1}, \quad (2)$$

where $\mathbf{L} = \mathbf{W} - \mathbf{D}$ is the Laplacian matrix of $\mathcal{G}$, $\mathbf{D} = \text{diag}(\mathbf{W}\mathbf{1})$ is the degree matrix, and $\mathbf{1}$ is a vector of ones. Furthermore, $\circ$ is the Hadamard product, $\|\cdot\|_{1,1}$ is the element-wise ℓ_1 norm of a matrix, and $\mathbf{Z} \in \mathbb{R}^{d\times d}$ is a pairwise distance matrix whose (i, j) entry is calculated as

$$\mathbf{Z}[ij] = \|\widetilde{\mathbf{x}}_i - \widetilde{\mathbf{x}}_j\|_2^2. \quad (3)$$

The quadratic form (2) quantifies how much the signals change w.r.t. $\mathcal{G}$ [9]. Smaller values of (2) are indicative of smoother signals. Therefore, the problem of batch graph learning based on the smoothness assumption is formulated as [10]

$$\begin{aligned} &\min_{\mathbf{W}} \frac{1}{2}\|\mathbf{W} \circ \mathbf{Z}\|_{1,1} - \alpha \mathbf{1}^\top \log(\mathbf{W}\mathbf{1}) + \frac{\beta}{2}\|\mathbf{W}\|_{\text{F}}^2 \\ &\text{s.t. } \mathbf{W} \in \mathcal{W},\ \mathbf{1}^\top \mathbf{W}\mathbf{1} = d. \end{aligned} \quad (4)$$

The second and third terms in the objective function are leveraged to control the connectivity and sparsity of the learned graphs [10], where α and β are model parameters. Inspired by [9], we add an extra constraint $\mathbf{1}^\top \mathbf{W}\mathbf{1} = d$ to control the scale of edge weights. Note that the number of free variables of $\mathbf{W}$ is $p := \frac{d(d-1)}{2}$ due to the constraints of $\mathbf{W}$ in (1). Thus, we rewrite the problem (4) in a vector form

$$\begin{aligned} &\min_{\mathbf{w}} \mathbf{z}^\top \mathbf{w} - \alpha \mathbf{1}^\top \log(\mathbf{S}\mathbf{w}) + \beta\|\mathbf{w}\|_2^2 \\ &\text{s.t. } \mathbf{w} \geq 0, \mathbf{w}^\top \mathbf{1} = d/2. \end{aligned} \quad (5)$$

where $\mathbf{S}$ is a linear operator satisfying $\mathbf{S}\mathbf{w} = \mathbf{W}\mathbf{1}$, and $\mathbf{w}, \mathbf{z} \in \mathbb{R}^p$ are the vector forms of the upper triangle variables of $\mathbf{W}$ and $\mathbf{Z}$, respectively.

B. Problem Statement

The problem of concern in this study has two characteristics. i) The observed data arrive sequentially, and we need to update a graph immediately after each data $\mathbf{x}_t$ arrives, rather than waiting for all T data to be ready. ii) The underlying graph is time-varying, and the optimal parameters of smoothness based GL model (5) for the real graph may vary significantly. This work aims to online track graphs based on the smoothness GL model (5) while handling the model mismatch problem caused by dynamic environments.

III. The Proposed Algorithm

In this section, we propose our adaptive online GL algorithm, where we run N expert algorithms equipped with different model parameters in parallel. To address the problem of model mismatch, we propose an expert-tracking algorithm to adaptively combine the outputs of all expert algorithms.

A. Expert algorithm

For the i-th expert, the model parameters are α^i and β^i, as shown in (5). We exploit the online mirror descent (OMD) framework [28] to develop an online version of the batch GL algorithm as an expert algorithm. Specifically, after receiving $\mathbf{x}_t$, we first calculate $\mathbf{Z}_t$ using $\mathbf{x}_t$ via (3), i.e., $\mathbf{Z}_t[ij] = |\mathbf{x}_t[i] - \mathbf{x}_t[j]|^2$. We then extract the upper triangular elements of $\mathbf{Z}_t$ to obtain $\mathbf{z}_t$[1] and update $\bar{\mathbf{z}}_t$ via

$$\bar{\mathbf{z}}_t = \gamma \bar{\mathbf{z}}_{t-1} + (1-\gamma)\mathbf{z}_t, \quad (6)$$

where $\gamma \in [0, 1)$ is a forgetting factor, and $\bar{\mathbf{z}}_0 = \mathbf{0}$. The update (6) is called the exponential moving average.

Based on the batch formulation (5), we define

$$f_t^i(\mathbf{w}) := \bar{\mathbf{z}}_t^\top \mathbf{w} - \alpha^i \mathbf{1}^\top \log(\mathbf{S}\mathbf{w} + \delta\mathbf{1}) + 2\beta^i\|\mathbf{w}\|_2^2, \quad (7)$$

where δ is a small constant to avoid zero node degree. Under OMD framework, we update a new graph $\mathbf{w}_{t+1}^i$ through

$$\mathbf{w}_{t+1}^i = \underset{\mathbf{w}\geq 0, \mathbf{w}^\top\mathbf{1}=d/2}{\text{argmin}} \langle \nabla f_t^i(\mathbf{w}_t^i), \mathbf{w}\rangle + \frac{1}{2\eta_t^i}\|\mathbf{w} - \mathbf{w}_t^i\|_2^2, \quad (8)$$

where $\nabla f_t^i(\mathbf{w})$ is the gradient of f_t^i w.r.t. $\mathbf{w}$, and η_t^i is the stepsize of the i-th expert at t. In our problem, $\nabla f_t^i(\mathbf{w}_t^i)$ is

$$\nabla f_t^i(\mathbf{w}_t^i) = \bar{\mathbf{z}}_t + 2\beta^i \mathbf{w}_t^i - \alpha^i \mathbf{S}^\top \left(\frac{1}{\mathbf{S}\mathbf{w}_t^i + \delta\mathbf{1}}\right). \quad (9)$$

With the optimal condition of (8), we update $\mathbf{w}_{t+1}^i$ as

$$\mathbf{w}_{t+1}^i = \text{Proj}_{\mathcal{U}}\left(\mathbf{w}_t^i - \eta_t^i \nabla f_t^i(\mathbf{w}_t^i)\right), \quad (10)$$

where $\text{Proj}_{\mathcal{U}}$ represents projection onto the simplex $\mathcal{U} := \left\{\mathbf{w} : \mathbf{w} \geq 0,\ \mathbf{w}^\top\mathbf{1} = d/2\right\}$. Inspired by [12], the stepsize is selected as $\eta_t^i = \left(2\beta^i + \frac{2\alpha^i(d-1)}{\min(\mathbf{S}\mathbf{w}_t^i)^2}\right)^{-1}$. The model parameters of the i-th expert are α^i and β^i. Fortunately, as stated in Proposition 2 of [10], α^i and β^i have a scaling effect, meaning that we can fix α^i as a constant and only search for the best β^i. However, as depicted in Fig.1, the optimal β^i corresponding to different underlying graphs may vary widely. It is impractical to select in advance a series of parameters that matches the dynamic environments. Thus, we utilize the following expert-tracking algorithm to address this issue.

B. Expert-tracking Algorithm

Suppose N experts $\mathcal{E}^1, ..., \mathcal{E}^N$ are prepared, whose corresponding parameters lie in $\mathcal{S} = \{\beta^1, ..., \beta^N\}$. Without loss of generality, we assume $\beta^1 < \beta^2 <, ..., < \beta^N$. Each expert $\mathcal{E}^i$ is assigned an initial weight φ_1^i to represent the "importance" of its output. The weight of $\mathcal{E}^i$ is online updated using

$$\widetilde{\varphi}_{t+1}^i = \frac{\varphi_t^i \exp(-\kappa h_t(\mathbf{w}_t^i))}{\sum_j^N \varphi_t^j \exp(-\kappa h_t(\mathbf{w}_t^j))}, \quad (11)$$

$$\varphi_{t+1}^i = \frac{\xi}{N} + (1-\xi)\widetilde{\varphi}_{t+1}^i, \quad (12)$$

where κ and ξ are predefined constants whose values will be discussed in section IV. Furthermore, $h_t(\mathbf{w}_t^i)$ is a function to evaluate the learned graphs. Here, $h_t(\mathbf{w}_t^i)$ is selected as

$$h_t(\mathbf{w}_t^i) = \bar{\mathbf{z}}_t^\top \mathbf{w}_t^i. \quad (13)$$

The metric is similar to [29], with the intuition that if the model parameter of $\mathcal{E}^i$ matches the current environment, the smoothness (13) should be small. The smoothness (13) is not normalized like [29] since the constraint, $\mathbf{w}^\top\mathbf{1} = d/2$, eliminates the effect of edge weight scales. We use an extra update (12) to avoid too small φ_{t+1}^i. The weight φ_{t+1}^i is at least ξ/N, which is called fixed share weight [22]. Thanks to the fixed share weight, a previously underperforming expert can regain sufficient weight and quickly become a leader when its parameters match the current environment. Using (11) and (12), the weights of different experts are adaptively adjusted according to their real-time performance [21]. We combine the results of all experts to output the final graph as

$$\mathbf{w}_{t+1} = \sum_{i=1}^{N} \varphi_{t+1}^i \mathbf{w}_{t+1}^i. \quad (14)$$

[1] $\mathbf{z}_t$ is referred to as a data vector hereafter since it is calculated from $\mathbf{x}_t$.

Algorithm 1 Adaptive online graph learning (AOGL)

Input:

γ, κ, ξ, $\mathcal{S} = \{\beta^1, ..., \beta^N\}$

Output:

The learned graph $\mathbf{w}_t, t = 1, ..., T$.

Initialize $\varphi_1^i = \frac{1}{N}$ and $\mathbf{w}_1^i = \mathbf{w}_1, i = 1, ..., N$
for $t = 1, ..., T$ **do**
  Receive data $\mathbf{x}_t$ and update $\bar{\mathbf{z}}_t$ using (6)
  for $i = 1, ..., N$ in parallel **do**
    Update φ_{t+1}^i using (11) and (12)
    Update $\mathbf{w}_{t+1}^i$ using (10).
  end for
  Update $\mathbf{w}_{t+1}$ using (14)
end for

It is not difficult to conclude that the expert whose parameter best matches the current environment will obtain increasing weights. Thus, its output occupies a larger position in the final result. In this way, we can learn graphs online that adapt to dynamic environments. The algorithm flow is in Algorithm 1.

In our algorithm, the complexity of calculating the gradient $\nabla f_t^i(\mathbf{w}_t^i)$ is $\mathcal{O}(p)$. Furthermore, it is well known that the complexity of projection onto the simplex $\mathcal{U}$ is $\mathcal{O}(p + \mathrm{nnz}(\mathrm{Proj}_{\mathcal{U}}(\mathbf{w})) \cdot \log p)$ [30], where $\mathrm{nnz}(\cdot)$ is the number of non-zero elements. Thus, the complexity of our algorithm per time instant is $\mathcal{O}\left(N\left(p + \mathrm{nnz}(\mathrm{Proj}_{\mathcal{U}}(\mathbf{w})) \cdot \log p\right)\right)$. Usually, $N \ll p$, and N expert algorithms are run in parallel, meaning that we can omit the effect of N.

The proposed framework is flexible enough that any existing OGL algorithm can serve as the individual expert algorithm. Thus, the main novelty of our work lies in providing an approach to handle the model mismatch problem, which has not been explored, rather than devising a new OGL algorithm.

IV. Dynamic Regret Analysis

Dynamic regret is a common metric for evaluating online algorithms [22]. For OGL algorithms, we consider the dynamic regret of the smoothness loss, i.e.,

$$Reg_d(T) := \sum_{t=1}^{T} h_t(\mathbf{w}_t) - \sum_{t=1}^{T} h_t(\mathbf{w}_t^*), \tag{15}$$

where $\mathbf{w}_t^*$ is the minimizer of the expert algorithm corresponding to the best-matching parameters in $\mathcal{S}$ at t. By definition, the regret quantifies the cumulative smoothness loss incurred by an online algorithm relative to the instantaneous best-matching expert. An OGL algorithm is admissible only if it yields a sublinear regret since online algorithms with sublinear regret asymptotically perform as well as the best experts on average [31]. Before theoretical analysis, we make some assumptions.

A1. All received data are bounded, i.e., there exists a constant $B_z > 0$ such that $\|\mathbf{z}_t\|_2 \leq B_z$ for $t = 1, ..., T$.

A2. The normalized smoothness loss is bounded, i.e., there exists a constant $B_h > 0$ such that $\left|h_t(\mathbf{w}_t^i)\right| \leq B_h$ for $i = 1, ..., N$ and $t = 1, ..., T$.

Naturally, **A1** holds in real-world applications, and **A2** holds for graphs with finite edge weights. Based on **A1** and Proposition 1 in [18], we conclude that $\|\bar{\mathbf{z}}_t\|_2$ is also bounded by B_z. Next, we define $v_t = \max_{i=1,...,N} \|(\mathbf{w}_{t+1}^i)^* - (\mathbf{w}_t^i)^*\|_2$ to measure environmental variability, where $(\mathbf{w}_t^i)^*$ is the minimizer of the i-th expert at t. We have Theorem 1 based on **A1**- **A2**.

Theorem 1. *Under **A1**-**A2**, suppose the expert corresponding to the smallest smoothness switches at most m times and*

$$\xi = \frac{m}{T-1},\ \kappa = \sqrt{\frac{8\left((m+1)\ln N + (T-1)R\left(\frac{m}{T-1}\right)\right)}{TB_h^2}}, \tag{16}$$

where $R(x) = -x\ln(x) - (1-x)\ln(1-x)$. If the smallest stepsize, $\eta_{min} = \min_{t,i} \eta_t^i$, $t = 1, ..., T, i = 1, ..., N$, satisfies $\eta_{min} \propto \frac{1}{\sqrt{T}}$, the dynamic regret of our algorithm is

$$\begin{aligned} Reg_d(T) \leq & \sqrt{T}\sqrt{\frac{B_h^2}{2}\left((m+1)\ln N + m\ln T + 1\right)} \\ & + \sqrt{T}\left(\frac{B_z B_w}{2\beta^1} + \frac{B_z}{2\beta^1}\sum_{t=1}^{T-1} v_t\right) \\ = & \mathcal{O}\left(\sqrt{T}(1 + V_T)\right), \end{aligned} \tag{17}$$

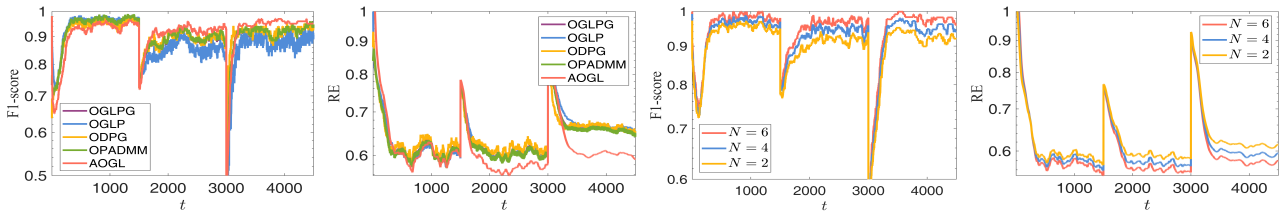
where $V_T = \sum_{t=1}^{T-1} v_t$. Furthermore, $B_w = \max_i \|\mathbf{w}_1 - (\mathbf{w}_1^i)^*\|_2$, $i = 1, ..., N$, where $\mathbf{w}_1$ is the initial graph.

Proof. The proof is placed in the supplementary material. □

The points of Theorem 1 are two-fold. First, it tells us that, when graphs do not vary fast, i.e., V_T at least does not increase linearly as T, our algorithm can reach sublinear regret $\mathcal{O}\left(\sqrt{T}(1 + V_T)\right)$. Second, the theorem also provides some disciplines to determine the parameters of our algorithm to achieve sublinear regret. Eq.(16) indicates that ξ and κ depend on m, which is a parameter that reflects environmental variability. When the graph varies rapidly, more switches will be incurred. In this case, m is a large value, bringing large ξ and κ to track rapidly-changing graphs. One drawback of our algorithm is that m must be known a priori. However, we believe that it is easier to determine m based on prior knowledge of environmental variability than to pre-select a sequence of optimal parameters. When m is unknown, we can resort to alternative approaches in [32], [33], which, however, may lead to higher regret or higher computational complexity.

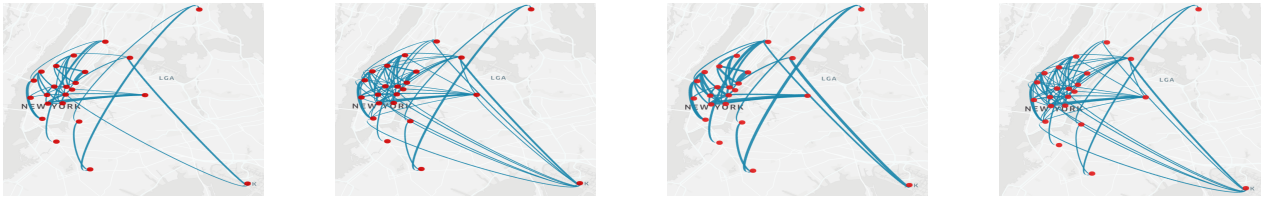
V. Numeric Tests

Synthetic data. We first generate Gaussian radial basis function (RBF) random graphs [9] as underlying graphs. Specifically, we generate 20 vertices whose coordinates are randomly in a unit square. The edge weights are calculated using $\exp(-\mathrm{dist}(i,j)^2/2\sigma_r^2)$, where $\mathrm{dist}(i,j)$ is the distance between vertices i and j, and $\sigma_r = 0.5$ is a kernel width parameter. Then, we remove the edges whose weights are smaller than a threshold ι. Hence, ι represents the sparsity of the generated graph. In this experiment, ι is set to be $0.5, 0.2$, and 0.8 when t in $[1, 1500]$, $[1501, 3000]$, and $[3001, 4500]$, respectively. We generate smooth signal $\mathbf{x}_t$ from the Gaussian distribution $\mathcal{N}(\mathbf{0}, (\mathbf{L}_t^{real})^\dagger + \sigma_e^2\mathbf{I})$ [9], where $(\mathbf{L}_t^{real})^\dagger$ is the pseudo inverse of the real Laplacian matrix at t, and $\sigma_e = 0.1$ is the noise level. We employ four existing smoothness-based OGL algorithms in the literature as baselines, i,e, OGLPG [12], OGLP [16], [18], ODPG [19], and OPADMM [20]. It is observed from Fig.2 that when $\alpha = 1$ and $\beta = 5$, the model



(a) (b) (c) (d)

Fig. 2: The figure shows: (a)-(b) The results of our methods and baselines. (c)-(d) The results of different expert sets.



(a) Sep.10th (Monday) 6 a.m. (b) Sep.10th (Monday) 10 p.m. (c) Sep.17th (Monday) 6 a.m. (d) Sep.17th (Monday) 10 p.m.

Fig. 3: The visualizations of the travel relation graph of different time stamps.

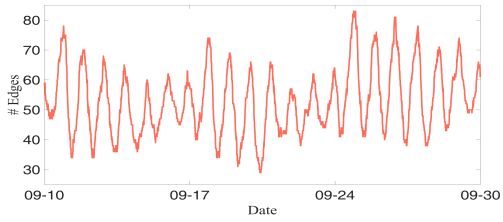


Fig. 4: The number of edges of the learned graph.

(5) performs well in the case of $\iota = 0.5$ (the initial case). Thus, we fix $\alpha = 1$ and set $\beta = 5$ for baseline algorithms. For our algorithm, we let $\mathcal{S} = \{0.1, 0.5, 1, 2, 5, 10\}$ which can almost cover all optimal parameters as suggested in Fig.1. We then set $\gamma = 0.99$, $m = 2$, and calculate ξ and κ using (16). Identifying whether two vertices are connected is a binary classification problem. Therefore, we first utilize the F1-score to evaluate the learned graphs. The second metric is the relative error (RE) $\|\mathbf{w}_t - \mathbf{w}_t^{real}\|_2 / \|\mathbf{w}_t^{real}\|_2$, where $\mathbf{w}_t^{real}$ is the real graph at t. We remove edges from the learned graph whose weights are less than 10^{-4}. As depicted in Fig.2(a)-(b), our algorithm fails to yield the best performance when $t \in [0, 1500]$, especially the F1-score. The reason may be that the selected β for baselines matches the ground-truth, and our algorithm may introduce some small edges weights caused by the models with suboptimal parameters, which are still considered connected edges. However, it is superior to other methods in $[1500, 300]$ and $[3000, 4500]$. The reason is that β of baselines is fixed, which may not be consistent with the real graphs. Our algorithm adjusts the weights of N experts via real-time performance, which can better track dynamic graphs.

We next test the effect of N. We consider three cases, i.e., $N = 2, 4, 6$. The value range of the elements in the three sets is $[0.1, 10]$. As depicted in Fig.2 (c)-(d), if the range of β is fixed, large N means a finer division of the range. Thus, it is more likely that there is an expert in $\mathcal{S}$ that matches current environments, resulting in better performance. However, more experts also incur more computational costs.

Real-world data. We leverage the Yellow Taxi Trip data of the New York City [2], which was used in [34] to learn a graph that conveys some information about the traffic patterns in the city. Our study is the first to use this dataset to track dynamic traffic graphs online. The city is divided into 23 zones, and we choose taxi trip data in September 2018. We count pickups every 5 minutes in divided zones as graph signals, which are the only data at hand. Finally we obtain 8640 signals, i.e., $T = 8640$. All parameters are determined by the principle in Theorem 1, and we set $\gamma = 0.99$ and $m = 3$. Figure 4 displays the number of edges of the learned graph from Sep.10th to Sep.30th, and some periodic patterns are observed. In a day, the number of edges decreases from midnight until it reaches a minimum in the morning, then increases again until the next midnight. This can be explained by that people are asleep in the early morning, leading to sparser interconnections between different zones. The maximum occurs around 10 p.m. since people tend to take a taxi home to rest. Furthermore, the average number of edges on weekends is less than that on workdays because people are prone to stay home on weekends. The figure also shows that the sparsity of the underlying graph varies widely during a day and a week. Our algorithm is not affected by the changes in sparsity due to the expert-tracking algorithm. Fig.3 (a) and (d) display the learned graphs visually. Compared with 6 a.m, graphs of 10 p.m contain more connections in Manhattan and residential areas. Moreover, graphs of the same day in two different weeks are similar, e.g., 6 a.m. and 10 p.m. in Sep.10th and Sep.17th. Our method successfully tracks daily and weekly patterns of travel relationships in New York.

VI. Conclusion

We proposed an algorithm to online learn dynamic graphs from smooth signals. The algorithm consists of multiple expert algorithms equipped with different model parameters and an expert-tracking algorithm to address the model mismatch problem. Theoretically, we proved that our algorithm can reach sublinear dynamic regret. Experimental results on synthetic and real data validated the superiority of our method.

[2]The data is available at https://data.cityofnewyork.us/Transportation

REFERENCES

[1] X. Dong, D. Thanou, M. Rabbat, and P. Frossard, "Learning graphs from data: A signal representation perspective," *IEEE Signal Process. Mag.*, vol. 36, no. 3, pp. 44–63, 2019.

[2] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Process. Mag.*, vol. 36, no. 3, pp. 16–43, 2019.

[3] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, 2013.

[4] X. Dong, D. Thanou, L. Toni, M. Bronstein, and P. Frossard, "Graph signal processing for machine learning: A review and new perspectives," *IEEE Signal Process. Mag.*, vol. 37, no. 6, pp. 117–127, 2020.

[5] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–441, 2008.

[6] M. Yuan and Y. Lin, "Model selection and estimation in the Gaussian graphical model," *Biometrika*, vol. 94, no. 1, pp. 19–35, 2007.

[7] A. Bolstad, B. D. Van Veen, and R. Nowak, "Causal network inference via group sparse regularization," *IEEE Trans. Signal Process.*, vol. 59, no. 6, pp. 2628–2641, 2011.

[8] J. Songsiri and L. Vandenberghe, "Topology selection in graphical models of autoregressive processes," *J. Mach. Learn. Res.*, vol. 11, pp. 2671–2705, 2010.

[9] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning Laplacian matrix in smooth graph signal representations," *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6160–6173, 2016.

[10] V. Kalofolias, "How to learn a graph from smooth signals," in *Proc. Int. Conf. Artif. Intell. Stat., AISTATS*. PMLR, 2016.

[11] Y. Shen, B. Baingana, and G. B. Giannakis, "Tensor decompositions for identifying directed graph topologies and tracking dynamic networks," *IEEE Open J. Signal Process.*, vol. 65, no. 14, pp. 3675–3687, 2017.

[12] S. S. Saboksayr, G. Mateos, and M. Cetin, "Online graph learning under smoothness priors," in *Proc. Eur. Signal Process. Conf.* IEEE, 2021, pp. 1820–1824.

[13] E. Kummerfeld and D. Danks, "Tracking time-varying graphical structure," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013.

[14] S. Yang, H. Xiong, Y. Zhang, Y. Ling, L. Wang, K. Xu, and Z. Sun, "Ogm: Online Gaussian graphical models on the fly," *Appl. Intell.*, vol. 52, no. 3, pp. 3103–3117, 2022.

[15] Y. Shen and G. B. Giannakis, "Online identification of directional graph topologies capturing dynamic and nonlinear dependencies," in *IEEE Data Sci. Workshop*. IEEE, 2018, pp. 195–199.

[16] A. Natali, E. Isufi, M. Coutino, and G. Leus, "Online graph learning from time-varying structural equation models," in *Proc. IEEE Asilomar Conf. Signals Syst. Comput.* IEEE, 2021, pp. 1579–1585.

[17] ——, "Learning time-varying graphs from online data," *IEEE Open J. Signal Process.*, vol. 3, pp. 212–228, 2022.

[18] X. Zhang and Q. Wang, "Online graph learning in dynamic environments," in *Proc. Eur. Signal Process. Conf.* IEEE, 2022, pp. 2151–2155.

[19] S. S. Saboksayr and G. Mateos, "Dual-based online learning of dynamic network topologies," *arXiv preprint arXiv:2211.07449*, 2022.

[20] H. Chahuara and G. Mateos, "Online proximal admm for graph learning from streaming smooth signals," in *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2025, pp. 1–5.

[21] A. Daniely, A. Gonen, and S. Shalev-Shwartz, "Strongly adaptive online learning," in *Proc. Int. Conf. Mach. Learn.* PMLR, 2015, pp. 1405–1411.

[22] E. C. Hall and R. M. Willett, "Online convex optimization in dynamic environments," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 4, pp. 647–662, 2015.

[23] P. M. Ghari and Y. Shen, "Graph-aided online learning with expert advice," in *Proc. IEEE Asilomar Conf. Signals Syst. Comput.* IEEE, 2020, pp. 470–474.

[24] T. Van Erven and W. M. Koolen, "Metagrad: Multiple learning rates in online learning," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016.

[25] L. Zhang, S. Lu, and Z.-H. Zhou, "Adaptive online learning in dynamic environments," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

[26] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 928–936.

[27] A. Jadbabaie, A. Rakhlin, S. Shahrampour, and K. Sridharan, "Online optimization: Competing with dynamic comparators," in *Proc. Int. Conf. Artif. Intell. Stat., AISTATS*. PMLR, 2015, pp. 398–406.

[28] E. Hazan *et al.*, "Introduction to online convex optimization," *Found. Trends Mach. Learn.*, vol. 2, no. 3-4, pp. 157–325, 2016.

[29] H. Araghi, M. Sabbaqi, and M. Babaie-Zadeh, "k-graphs: An algorithm for graph signal clustering and multiple graph learning," *IEEE Signal Processing Letters*, vol. 26, no. 10, pp. 1486–1490, 2019.

[30] L. Condat, "Fast projection onto the simplex and the l 1 ball," *Mathematical Programming*, vol. 158, no. 1-2, pp. 575–585, 2016.

[31] B. Zaman, L. M. L. Ramos, D. Romero, and B. Beferull-Lozano, "Online topology identification from vector autoregressive time series," *IEEE Trans. Signal Process.*, pp. 210–225, 2020.

[32] A. Gyorgy, T. Linder, and G. Lugosi, "Efficient tracking of large classes of experts," *IEEE Trans. Inf. Theory*, vol. 58, no. 11, pp. 6709–6725, 2012.

[33] D. Adamskiy, W. M. Koolen, A. Chernov, and V. Vovk, "A closer look at adaptive regret," in *Int. Conf. on Alg. Learn. Theory*. Springer, 2012, pp. 290–304.

[34] D. Thanou, X. Dong, D. Kressner, and P. Frossard, "Learning heat diffusion graphs," *IEEE Trans. Signal. Inf. Process. Netw.*, vol. 3, no. 3, pp. 484–499, 2017.