Robust Meta-Learning over Graphs with Graph Neural Networks

Alireza Sadeghi, Georgios B. Giannakis
Department of Electrical and Computer Engineering, University of Minnesota

***Abstract*—Graph neural networks (GNNs) have well documented success in various learning over graph problems, from drug discovery to recommender systems. These data hungry models however, are challenged in applications with limited available samples. Meta-learning paradigm provides principled methods to empower GNNs to learn rapidly from a few labeled data only, particularly beneficial for tasks with small datasets. Nonetheless, existing meta-learning frameworks for graphs encounter difficulties when the labeled nodes or links used during training are distant from those encountered during testing, rendering meta-learning in graph tasks challenging. In this context, current work leverages a suite of novel techniques to enhance generalization performance of learning over graphs with GNNs. A novel idea here is to further incorporate an extra adversarial update step during "adaptation" phase of conventional meta-learning algorithms. Additionally, our proposed method incorporates curvature information of the loss landscape during meta-updating to facilitate reliable knowledge transfer to downstream tasks with slightly different structures. The proposed methods are simple and with minimal computational costs on training process. Numerical experiments conducted on learning over graph problems validate the superiority of our algorithm over the existing counterparts, showcasing effectiveness when dealing with limited data resources.**

***Index Terms*—Robust learning, Meta-learning, GNNs.**

I. Introduction

Learning over graphs is prevalent in numerous real-world applications, particularly when the interconnections among data are represented by an underlying graph structure [1]. Examples include the analysis and inference tasks related to social, brain, power, communication, biological, transportation, and sensor networks [2]. Graph neural networks (GNNs) has recently emerged as a power full parametric models to efficiently learn complex functions defined over such graphs [3]–[5]. By combining graph-filters, topology information, and point-wise nonlinearities GNNs form nested expressive architectures. This has made them remarkably successful across a wide range of learning problems over irregular graph-structured inputs, including supervised, semi-supervised, and unsupervised learning ones [3]–[6].

Similar to other deep neural networks (DNNs), GNNs have their own shortcomings. A salient problem rises when having only a handful of training data samples — a typical case emerging in few-shot learning settings [7]. This becomes even more challenging when dealing with large-scale graphs. To tackle this, recent efforts have focused on few-shot learning with graph data by employing methods such as co-training and self-training, all tailored for specific few-shot learning tasks with predetermined classes [8], [9]. Another remedy to deal with these challenges is to invoke meta-learning algorithms [7], [10]; see also [11].

Meta-learning, also termed "learning to learn," aims to extract *shared prior knowledge* from a set of inter-related tasks, facilitating rapid adaptation to an unseen yet related new one with limited training samples [10], [12]. Meta-learning holds the promise to obtain domain-agnostic inductive biases that empower models to generalize better, adapt rapidly to new tasks, while at the same time require fewer data for training [8]. Applying meta-learning to GNNs for graph problems is particularly important due to the diverse downstream applications that GNNs may encounter for the unseen new tasks, spanning from node classification to link prediction. Meta-learning over graphs enables identifying suitable prior knowledge and encode such information into several hyper-parameters that can be learned across tasks, and be then fine-tuned using the validation data of all tasks [8]. The predominant challenge however still persists, that is a small subset of labeled nodes or links might not be directly relevant for the downstream new unseen task. In node classification problems for instance, the labeled nodes rarely resemble those in the so-called support and query sets of a given task [7], [13], [14]. Similarly in link prediction, the support and query edges might be far apart in the underlying graph [7]. It thus holds of paramount importance to investigate whether the traditional meta-learning algorithms can be further improved to effectively utilize the scarcely avaiable data samples per task, beyond simply extracting prior knowledge.

Contributions. This work proposes simple, yet computationally efficient methods to enhance performance of conventional meta-learning algorithms. To this aim, the novel algorithm enforces an extra *adversarial update* step during adaptation phase, generating *task*- and *model*-dependent adversarial data samples on-the-fly, while exploiting the underlying data distribution. Additionally, it incorporates loss curvature while updating meta-parameters to cope with potential variations in new upcoming task. To manage arising computational complexities of the proposed meta-update rule while maintaining performance, an exploration-exploitation type step is further introduced. While applicable to any meta-learning problems, we evaluate its performance on graph-based node-classification problems.

This work is supported in part by NSF grants 2312547, 2220292, 2126052 and 2212318. The authors are with the Dept. of ECE, University of Minnesota, Minneapolis, MN 55455, USA (e-mail: sadeghi@umn.edu; georgios@umn.edu).

II. Problem Formulation

Meta-learning is a bi-level optimization framework to extract prior knowledge from a collection of $t = 1, \ldots, T$ inter-related tasks indexed by t, to facilitate learning on a new unseen one. Each learning task t has an annotated dataset $\mathcal{D}_t := \{(\mathbf{x}_t^n, y_t^n)\}_{n=1}^{N_t}$ consisting of N_t data samples, drawn i.i.d from a distribution $(\mathbf{x}_t^n, y_t^n) \sim P_t$. The dataset is divided into a training subset $\mathcal{D}_t^{\rm trn} \subset \mathcal{D}_t$ and a validation one $\mathcal{D}_t^{\rm val} := \mathcal{D}_t / \mathcal{D}_t^{\rm trn}$. In addition to these tasks, an unseen new one indexed by $\star$ is given, with its training $\mathcal{D}_\star^{\rm trn}$, and unannotated test set $\mathcal{D}_\star^{\rm tst} := \{\mathbf{x}_\star^n\}_{n=1}^{N_\star^{\rm tst}}$ for which the corresponding labels $\{y_\star^n\}_{n=1}^{N_\star^{\rm tst}}$ are to be inferred. The major assumption in meta-learning is that the tasks are inter-related through e.g., their underlying data distributions. Such relationship facilitates training a unified large-scale model such as a DNN to fit all tasks, where each task tailored by its specific model parameter $\boldsymbol{\phi}_t \in \mathbb{R}^d$. However, as the cardinality $|\mathcal{D}_t^{\rm trn}|$ can be much smaller than d, directly optimizing $\boldsymbol{\phi}_t$ over $\mathcal{D}^{\rm trn}$ could be challenging, if not impossible without incurring overfitting.

To alleviate this, meta-learning capitalizes on the relationships among tasks. Specifically, as T is generally considerably large, a reliable *task-invariant* prior knowledge can be extracted from given tasks, thereby facilitating the data-scarce per-task training. Such structure of prior extraction and per-task yields a *bilevel optimization* problem. The inner-level (task-level) optimizes the per-task parameter $\boldsymbol{\phi}_t$ using $\mathcal{D}_t^{\rm trn}$ (a.k.a. adaptation), and the prior provided by outer-level, while the outer-level (meta-level) evaluates the trained $\{\boldsymbol{\phi}_t\}_{t=1}^T$ using $\{\mathcal{D}_t^{\rm val}\}_{t=1}^T$, and refines a parameterized prior $\boldsymbol{\theta} \in \mathbb{R}^D$. Formally, this convenntional bi-level optimization can be expressed as

$$\min_{\boldsymbol{\theta}} \sum_{t=1}^{T} \mathcal{L}(\boldsymbol{\phi}_t^*(\boldsymbol{\theta}); \mathcal{D}_t^{\rm val}) \tag{1a}$$

$$\text{s. to} \quad \boldsymbol{\phi}_t^*(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\phi}_t} \mathcal{L}(\boldsymbol{\phi}_t; \mathcal{D}_t^{\rm trn}) + \mathcal{R}(\boldsymbol{\phi}_t; \boldsymbol{\theta}) \tag{1b}$$

where the loss $\mathcal{L}$ assesses the fit of a task-specific model to the dataset $\mathcal{D}_t^{\rm trn}$, and the regularizer $\mathcal{R}$ quantifies the impact of task-invariant parameterized prior. A typical choice for this regularization can be e.g., $\mathcal{R}(\boldsymbol{\phi}_t; \boldsymbol{\theta}) := \|\boldsymbol{\phi}_t - \boldsymbol{\theta}\|_2^2$ to enforce proximity of the per-task parater $\boldsymbol{\phi}_t$ to $\boldsymbol{\theta}$, signifying the impact of learned prior from inter-related tasks.

Although intriguing, the problem in (1) even with prior leverage, may not significantly improve the generalization of task-level optimization, especially due to the scarcity of samples in the inner-level optimization—a common characteristic in few-shot learning over graph problems where nodes or links are far apart in the underlying graph [8]. Another significant challenge arises in practice from potential dissimilarity between the unseen downstream task and the previously encountered ones [13]. Therefore, it is imperative to design principled meta-level update rules promoting a more reliable prior, ensuring adaptability of prior to the slightly differing nature of incoming tasks. The forthcoming section addresses this issue.

III. Sharpness-aware learning for Outer-level Updates

Instead of relying on conventional meta-learning update rule to find $\boldsymbol{\theta}^\star$ by solving the outer-level optimization in (1a), we advocate a *sharpness aware optimization* counter part [15] into our meta-update rule. That is, rather than finding $\boldsymbol{\theta}$ with small training loss, we prefer parameter $\boldsymbol{\theta}$ values within a *neighborhood* that have *uniformly low* training loss value across all the $\boldsymbol{\theta}$ within this neighborhood. This means searching for $\boldsymbol{\theta}$ that not only has small loss value, but more importantly resides in a neighborhood with a small curvature; see e.g., [15] and references therein. Such an update rule is essential when dealing with over-parameterized models such as DNNs, where typical optimization approaches such as SGD can easily obtain suboptimal solutions [15]. Such a sharpness-aware update rule can hopefully guarantee the learned prior encapsulated in $\boldsymbol{\theta}$ generalizes well to unseen and even slightly un-related tasks. Thus we advocate to solve the following optimization problem for the outer-level update

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\epsilon}} \sum_{t=1}^{T} \mathcal{L}(\boldsymbol{\phi}_t^*(\boldsymbol{\theta} + \boldsymbol{\epsilon}); \mathcal{D}_t^{\rm val}), \quad \text{s.t. } \|\boldsymbol{\epsilon}\|_2 \leq \rho. \tag{2}$$

To solve (2) one should note that typically neither the inner maximization over $\boldsymbol{\epsilon}$ nor the outer minimization over $\boldsymbol{\theta}$ does not admit closed forms. A remedy to solve (2) is thus to resort to iterative alternative maximization and minimization steps respectively to find $\boldsymbol{\epsilon}^*$ and $\boldsymbol{\theta}^*$. In practice, the optimal value for $\boldsymbol{\epsilon}^*$ necessarily depends on the value of the parameter $\boldsymbol{\theta}$ as well. Thus, such an intricate dependency may make it even more challenging to solve this objective. To deal with this intricate dependency, we resort to *two-step* Taylor series expansion of the objective in (2). Specifically, at an iteration while having a *fixed* given $\boldsymbol{\theta}$, it holds that

$$\begin{aligned}
\boldsymbol{\epsilon}^*(\boldsymbol{\theta}) &:= \arg\max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} \textstyle\sum_{t=1}^{T} \mathcal{L}(\boldsymbol{\phi}_t^*(\boldsymbol{\theta} + \boldsymbol{\epsilon}); \mathcal{D}_t^{\rm val}) \\
&\overset{(a)}{\approx} \arg\max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} \textstyle\sum_{t=1}^{T} \mathcal{L}(\boldsymbol{\phi}_t^*(\boldsymbol{\theta}) + \mathbf{J}_{\boldsymbol{\phi}_t^*}(\boldsymbol{\theta})\boldsymbol{\epsilon}; \mathcal{D}_t^{\rm val}) \\
&\overset{(b)}{\approx} \arg\max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} \textstyle\sum_{t=1}^{T} \mathcal{L}(\boldsymbol{\phi}_t^*(\boldsymbol{\theta}) + \boldsymbol{\epsilon}; \mathcal{D}_t^{\rm val}) \\
&\overset{(c)}{\approx} \arg\max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} \textstyle\sum_{t=1}^{T} \mathcal{L}(\boldsymbol{\phi}_t^*(\boldsymbol{\theta}); \mathcal{D}_t^{\rm val}) \\
&\qquad\qquad + \boldsymbol{\epsilon}^\top \nabla_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\phi}_t^*(\boldsymbol{\theta}); \mathcal{D}_t^{\rm val}) \Big|_{\boldsymbol{\phi}_t = \boldsymbol{\phi}_t^*(\boldsymbol{\theta})} \\
&\overset{(d)}{=} \arg\max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} \boldsymbol{\epsilon}^\top \left(\textstyle\sum_{t=1}^{T} \nabla_{\boldsymbol{\phi}_t} \mathcal{L}(\boldsymbol{\phi}_t^*(\boldsymbol{\theta}); \mathcal{D}_t^{\rm val}) \Big|_{\boldsymbol{\phi}_t = \boldsymbol{\phi}_t^*(\boldsymbol{\theta})} \right).
\end{aligned} \tag{3}$$

where $\mathbf{J}_{\boldsymbol{\phi}_t^*}(\boldsymbol{\theta})$ is the Jacobean of vector valued function $\boldsymbol{\phi}_t^*$ at $\boldsymbol{\theta}$. The (a) is obtained using Taylor series expansion of $\boldsymbol{\phi}_t^*$ at current $\boldsymbol{\theta}$, that is $\boldsymbol{\phi}_t^*(\boldsymbol{\theta} + \boldsymbol{\epsilon}) \approx \boldsymbol{\phi}_t^*(\boldsymbol{\theta}) + \mathbf{J}_{\boldsymbol{\phi}_t^*}(\boldsymbol{\theta})\ \boldsymbol{\epsilon}$, (b) is due to our assumption that $\boldsymbol{\phi}_t^*$ can be approximated by a diagonal matrix (essentially meaning its entries are sufficiently independent) and all the constant values in its diagonal entries can be lumped into the corresponding entries of vector $\boldsymbol{\epsilon}$, and

(c) is obtained by applying Taylor series expansion to scaler valued function $\mathcal{L}(\boldsymbol{\phi}_t^*(\boldsymbol{\theta})+\boldsymbol{\epsilon};\mathcal{D}_t^{\text{val}})$ at $\boldsymbol{\phi}_t^*(\boldsymbol{\theta})+\boldsymbol{\epsilon}$, that is

$$\mathcal{L}(\boldsymbol{\phi}_t^*(\boldsymbol{\theta})+\boldsymbol{\epsilon};\mathcal{D}_t^{\text{val}}) \approx \mathcal{L}(\boldsymbol{\phi}_t^*(\boldsymbol{\theta});\mathcal{D}_t^{\text{val}})+\boldsymbol{\epsilon}^\top\nabla_{\boldsymbol{\phi}_t}\mathcal{L}(\boldsymbol{\phi}_t^*(\boldsymbol{\theta});\mathcal{D}_t^{\text{val}})+\mathcal{O}(\|\boldsymbol{\epsilon}\|_2^2). \tag{4}$$

$\mathcal{O}(\|\boldsymbol{\epsilon}\|_2^2)$ captures higher order dependencies on $\boldsymbol{\epsilon}$, which are simply ignored when deriving (c). Finally, (d) is obtained as the first term in (c) does not depend on $\boldsymbol{\epsilon}$.

Clearly, the optimal solution for (4) can be represented as

$$\boldsymbol{\epsilon}^*(\boldsymbol{\theta}) = \rho\,\frac{\sum_{t=1}^T\nabla_{\boldsymbol{\phi}_t}\mathcal{L}(\boldsymbol{\phi}_t^*(\boldsymbol{\theta});\mathcal{D}_t^{\text{val}})}{\|\sum_{t=1}^T\nabla_{\boldsymbol{\phi}_t}\mathcal{L}(\boldsymbol{\phi}_t^*(\boldsymbol{\theta});\mathcal{D}_t^{\text{val}})\|_2}. \tag{5}$$

Replacing $\boldsymbol{\epsilon}^*(\boldsymbol{\theta})$ back in (2) and apply one step gradient descent to update the outer level minimization over $\boldsymbol{\theta}$ enables to iteratively solve (2), and find an approximant for $\boldsymbol{\theta}^*$.

Although one can iteratively find $\boldsymbol{\theta}^*$ using automatic differentiation methods, unfortunately, the update rule resulted from this process to find $\boldsymbol{\epsilon}^*$ remains computationally expensive, particularly due to the necessity of determining $\boldsymbol{\epsilon}^*$ per iteration of updating $\boldsymbol{\theta}$. Furthermore, empirical observations reveal that consistently perturbing $\boldsymbol{\theta}$ does not consistently yield enhanced performance. To address these challenges, we introduce an exploration-exploitation type approach to update $\boldsymbol{\theta}$. Specifically, we sporadically update model parameters using a sharpness-aware approach described in (2) with a *very small* probability of η. However, all updates involve also directly minimizing (1a) using $\boldsymbol{\epsilon}^* = \mathbf{0}$. This approach offers two main benefits: (i) updating $\boldsymbol{\theta}$ towards desirable regions in the feasible set; and at the same time, (ii) reducing computational burdens, since most updates have $\boldsymbol{\epsilon}^* = \mathbf{0}$.

IV. Robust adaptation

Conventional adaptation phase of meta-learning algorithms entails solving (1b) for all t. Due to the limited number of available samples during adaptation – typical in few-shot learning problems, the resultant models often suffer from overfitting or exhibit poor generalization performance. This challenge is further exacerbated by the intricate relationship between the learned $\boldsymbol{\phi}_t^*$ and the meta-parameter $\boldsymbol{\theta}$ in the outer iteration (1a). To offer robustness, we consider the following robust adaptation rule, and approximately solve it as follows

$$\boldsymbol{\phi}_t^*(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\phi}_t}\ \mathcal{L}(\boldsymbol{\phi}_t;\mathcal{D}_t^{\text{trn}})+\mu_1\mathcal{R}(\boldsymbol{\phi}_t;\boldsymbol{\theta}) \tag{6a}$$

$$+\mu_2\mathcal{L}(\boldsymbol{\phi}_t;\mathcal{D}_t^{\text{adv}}). \tag{6b}$$

here $\mathcal{D}_t^{\text{adv}}$ represents adversarial data samples for task t. It essentially utilizes data augmentation techniques to craft synthetic data, which are *model*-dependent and can improve the performance of the learned model $\boldsymbol{\phi}_t^*(\boldsymbol{\theta})$. Specifically, for each task t, a simple remedy is to pre-process each data sample $\{(\mathbf{x}_t^n, y_t^n)\}_{n=1}^{N_t^{\text{trn}}}$ and create corresponding adversarial samples $\{(\mathbf{x}_t^{\text{adv,n}}, y_t^n)\}_{n=1}^{N_t^{\text{trn}}}$. Note that here, only the features are generated while the labels are kept unaltered. Clearly, a simple technique to obtain $\mathcal{D}_t^{\text{adv}} := \{(\mathbf{x}_t^{\text{adv,n}}, y_t^n)\}_{n=1}^{N_t^{\text{trn}}}$ is to rely on a notion of distance and craft samples that maximize training loss subject to a constraint as

$$\mathbf{x}_t^{\text{adv},n} = \arg\max_{\mathbf{x}}\ \mathcal{L}(\boldsymbol{\phi}_t;(\mathbf{x},y_t^n)),\forall n \quad \text{s.t. } \texttt{dist}(\mathbf{x}_t^{\text{adv},n},\mathbf{x}_t^n)\le\rho. \tag{7}$$

One can for instance invoke ℓ-2 to form $\texttt{dist}(\mathbf{x}_t^{\text{adv},n},\mathbf{x}_t^n) = \|\mathbf{x}_t^{\text{adv},n}-\mathbf{x}_t^n\|_2$ and characterize the constraint. Stochastic gradient ascent with a small step size can be leveraged to finding $\mathbf{x}_t^{\text{adv},n}$. Such augmentation of data although interesting, simply ignores most important aspect of the data – the underlying data distribution. Here, instead of relying on per-datum based data augmentation as proposed above, we consider data distribution as well when generating $\mathcal{D}_t^{\text{adv}}$. To formalize this, assume data samples for task t are drawn from a nominal data distribution $\boldsymbol{z}_t^n \sim P_t$, where $\boldsymbol{z}_t^n := (\mathbf{x}_t^n, y_t^n)$. Targeting adversarially generated data samples $\boldsymbol{z}_t^{\text{adv}} \in \mathcal{D}_t^{\text{adv}}$, we form an adversarial data distribution, that is $\boldsymbol{z}_t^{\text{adv}} \sim P_t^{\text{adv}}$. To characterize P_t^{adv} based on already available P_t, one can rely on momentum [16], [17], KL divergence [18], statistical test [19], and Wasserstein distance [19], [20]; see e.g., [21] for a recent overview. Among all choices, it has been shown that the Wasserstein distance results in a tractable approaches to generate samples $\boldsymbol{z}_t^{\text{adv}}$ using the given $\boldsymbol{z}_t$ ones, thanks to the strong duality result of [19] and [20]. To formalize this, consider two probability measures P and Q supported on set $\mathcal{Z}$ and $\mathcal{Z}'$, and let $\Pi(P,Q)$ be the set of all joint measures supported on $\mathcal{Z}\times\mathcal{Z}'$, with marginals P and Q. Let $c:\mathcal{Z}\times\mathcal{Z}'\to[0,\infty)$ measure the cost of transporting a unit of mass from $\boldsymbol{z}$ in P to another element $\boldsymbol{z}'$ in Q. The Wasserstein distance of order p between probability measures P and Q [22, Definition 6.1] is defined as

$$W_p(P,Q) := \inf_{\pi\in\Pi}\mathbb{E}_\pi\big[c^p(\boldsymbol{z},\boldsymbol{z}')\big]. \tag{8}$$

Upon relying on this definition, the *worst-case* adversarial data set $\mathcal{D}_t^{\text{adv}}$ can be determined by solving the following optimization problem to obtain P^{adv} and then draw samples from the derived distribution

$$P^{\text{adv}} := \sup_Q\ \mathbb{E}_{\boldsymbol{z}\sim Q}[\mathcal{L}(\boldsymbol{\phi}_t;\boldsymbol{z})]\ \text{, s.t. } W_p(P,Q)\le\rho. \tag{9}$$

Constraint here entails solving an infinite-dimensional optimal transport problem characterized in (8), which is intuitively challenging, if not impossible to solve. In addition, even if solvable, there is no guarantee one can efficiently draw samples from resulting P^{adv}. Fortunately, these challenges can be alleviated for a broad range of losses as well as transport costs. Specifically, it has been shown that optimization problem (9) satisfies strong duality condition under some mild conditions; that is, the optimal objective of this maximization and its Lagrangian dual optimal objective, are equal [21], [23] . Interestingly. the dual problem involves optimization over a one-dimensional dual variable, rather than infinite dimensional-fictional learning. In addition, one can directly arrive at the adversarial samples obtained from P^{adv} rather than forming the P^{adv}. These two observations make it possible to construct $\mathcal{D}_t^{\text{adv}}$ in the *dual domain* by levering this

Datasets	Cora		Citeseer	
few-shot / Models	1-shot	3-shot	1-shot	3-shot
DeepWalk	16.06 %	25.67 %	14.52 %	21.18 %
Node2Vec	15.15 %	25.66 %	12.98 %	20.02 %
GraphSAGE-Mean	50.89 %	53.12 %	53.49 %	55.01 %
GraphSAGE-Pool	48.53 %	50.15 %	51.02 %	53.98 %
SGC	61.64 %	**75.67** %	56.91 %	65.67 %
Meta-SGC	56.06 %	70.98 %	63.86 %	82.34 %
Proposed-Meta-SGC	**67.27** %	69.01 %	**70.59** %	**85.27** %

TABLE I: Accuracy in % for node classification.

strong duality condition. The following Theorem elaborates this in further details; see e.g., [21], [23].

Theorem. Let $\mathcal{L} : \boldsymbol{\phi}_t \times \mathcal{Z} \to [0, \infty)$ be upper semi-continuous and integrable, and $c : \mathcal{Z} \times \mathcal{Z}' \to [0, \infty)$ be a lower semi-continuos function. Then, for any given P, and $\rho > 0$, it holds that

$$\sup_{Q} \mathbb{E}_{\boldsymbol{z}\sim Q}[\ell(\boldsymbol{\phi}_t; \boldsymbol{z})] = \inf_{\gamma \geq 0} \Big\{ \mathbb{E}_{\boldsymbol{z}\sim P}\big[\sup_{\boldsymbol{\zeta}\in\mathcal{Z}} \{\ell(\boldsymbol{\theta}; \boldsymbol{\zeta}) - \gamma(c(\boldsymbol{z}, \boldsymbol{\zeta}) - \rho)\} \big] \Big\} \quad (10)$$

where $Q \in \{P | W_p(P_t, P) \leq \rho\}$.

Relying on this Theorem, and *assuming* knowledge of the optimal dual variable $\gamma^\star$, one can craft adversarially generated samples by solving

$$\mathbf{x}_t^{\text{adv},n} = \arg\max_{\boldsymbol{\zeta}} \mathcal{L}(\boldsymbol{\phi}_t; \boldsymbol{\zeta}) + \gamma^\star(\rho - c(\boldsymbol{z}_t^n, \boldsymbol{\zeta})), \quad \forall n. \quad (11)$$

Solving this problem to its global maximizer is not possible, specifically if the model characterize by $\boldsymbol{\phi}_t$ is a DNN. Instead, one can approximately solve it using K iterates of gradient ascent with a constant step size β

$$\boldsymbol{\zeta}^n(k+1) = \boldsymbol{\zeta}^n(k) + \beta \nabla_{\boldsymbol{\zeta}} \psi(\boldsymbol{\phi}_t, \boldsymbol{\zeta}; \boldsymbol{z}^n)|_{\boldsymbol{\zeta}=\boldsymbol{\zeta}^n(k)}, k = 1 \cdots K$$

where $\boldsymbol{\zeta}^n(0) = \boldsymbol{z}^n$, and $\psi(\boldsymbol{\phi}_t, \boldsymbol{\zeta}; \boldsymbol{z}^n) := \mathcal{L}(\boldsymbol{\phi}_t; \boldsymbol{\zeta}) + \gamma^\star(\rho - c(\boldsymbol{z}_t^n, \boldsymbol{\zeta}))$. Having solved (11), one can form $\mathcal{D}_t^{\text{adv}}$. Creating an entire adversarial dataset $\mathcal{D}_t^{\text{adv}}$ in advance can be cumbersome, specifically as it should happen in a batch form for each training epoch. Here we instead generate adversarial samples on-the-fly for each datum individually. This process involves first applying gradient updates to the model based on the clean input, then using the updated model to generate an adversarial sample, followed by another round of gradient updates based on this adversarially generated data. This approach *approximately* solves the regularized learning problem in (6) by relying on two steps of back propagation.

V. Numerical Tests

To assess the performance of the proposed algorithm, we consider node-classification problem within a graph-based learning framework. Objective is to classify nodes into new classes with only a small K-number of samples available for each class using two datasets, namely Cora and Citeseer [24]; see [7] for more details. In their original format, these data sets have not been designed for few-shot learning problems over graphs. We thus relied on [7] to make the necessary changes to the dataset partitioning, tailoring them for few-shot learning.

K-shot / h_d	1	4	8	16	32	64
1-shot	63.63%	70.37%	66.06%	76.21%	77.04%	82.80%
3-shot	65.27%	70.83%	72.34%	78.86%	81.36%	82.65%

TABLE II: Ablation on hidden dimension on Citeseer dataset.

The details of the modifications can be found in [7, Sec. 2.2]. Specifically, nodes in each dataset are partitioned into two distinct sets, one for meta-training, the other for meta-testing. The labels of the nodes in the meta-training set are chosen to belong one of the classes captured in $\mathcal{C}^{\text{trn}}$, while the labels of the meta-testing nodes are coming from different classes not seen during training, denoted by set $\mathcal{C}^{\text{test}}$. This means that nodes in the training set belong to known classes, while nodes in the test set are associated with entirely new, unseen ones. This setup allows to evaluate generalization performance to unseen classes. In this setup, if having K labeled nodes per class, the task is referred to as $|\mathcal{C}^{\text{test}}|$-way K-shot learning [7].

Our proposed method is compared against Meta-GNN implemented in [7]. Specifically, we consider the Simplified Graph Convolutional Network (SGC) as the backbone for our GNN [5]. Conventional MAML algorithm was used for gradient updates during Meta-GNN training. To implement proposed method, we have tailored the vanilla MAML algorithm to further incorporate the sharpness aware optimization step as discussed in Sec. III, as well as the robust adaptation discussed in Sec. IV. For the sharpness-aware updating rule, we used a small $\eta = 0.01$ for exploration-exploitation probability. Besides Meta-GNN, we further report the performance agains embedding based methods of DeepWalk [25], Node2Vec [26], SGC, and GraphSAGE [4]. In addition, we set $\gamma^\star = 0.76$ for adversarial data generation, and thus ρ is no longer needed.

Numerical Test Results. Table I demonstrates performance in classification accuracy (in %) of the proposed method compared with the alternatives using 1000 epochs of training. It is clear that the proposed method achieves the best accuracy across both Cora and Citeseer datasets. GNN-based models including ours effectively outperform the others on the few-shot learning problems when dealing with new classes. In contrast, GraphSAGE with Mean and Pool variants, although being inductive methods, do not exhibit superior performance. A probable reason for this is that inductive methods may not generalize well to *new unseen* classes, but rather only to new nodes (see also [4], [7]), as was the case in this toy experiment. Finally, we have further investigated the impact of hidden dimension of the SGC model on the performance using only 100 epochs of training. The results of this ablation study are reported in Table II, where it can be seen that by increasing model capacity (i.e., h_d), better performance is generally expected.

VI. Conclusions

This work addressed meta-learning in graph-based problems, where limited labeled data samples were available. A light weight sharpness-aware rule for updating parameterized priors was proposed, along with an adversarially robust adaptation to further enhance generalization.

REFERENCES

[1] G. B. Giannakis, Y. Shen, and G. V. Karanikolas, "Topology identification and learning over graphs: Accounting for nonlinearities and dynamics," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 787–807, 2018.

[2] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 249–270, 2020.

[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *Int. Conf. Learn. Represent.*, 2017.

[4] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[5] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Int. Conf. Mach. Learn.*, 2019.

[6] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI open*, vol. 1, pp. 57–81, 2020.

[7] F. Zhou, C. Cao, K. Zhang, G. Trajcevski, T. Zhong, and J. Geng, "Meta-GNN: On few-shot node classification in graph meta-learning," in *Proc. ACM Int. Conf. Inf. Knowl. Manag.*, 2019, pp. 2357–2360.

[8] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. AAAI Conf. Artif. Intell.*, 2018.

[9] R. Xu, L. Xing, S. Shao, L. Zhao, B. Liu, W. Liu, and Y. Zhou, "GCT: Graph co-training for semi-supervised few-shot learning," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 12, pp. 8674–8687, 2022.

[10] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Int. Conf. Mach. Learn.*, 2017.

[11] M. Abbas, Q. Xiao, L. Chen, P.-Y. Chen, and T. Chen, "Sharp-maml: Sharpness-aware model-agnostic meta learning," in *Intl. Conf. Machine Learn.* PMLR, 2022, pp. 10–32.

[12] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv:1803.02999*, 2018.

[13] K. Huang and M. Zitnik, "Graph meta learning via local subgraphs," *Adv. Neural Inf. Process. Syst.*, 2020.

[14] A. Sankar, X. Zhang, and K. C.-C. Chang, "Meta-GNN: Metagraph neural network for semi-supervised learning in attributed heterogeneous information networks," in *Proc. IEEE/ACM Int. Conf. Adv. Soc. Netw. Anal. Min.*, 2019, pp. 137–144.

[15] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, "Sharpness-aware minimization for efficiently improving generalization," *Int. Conf. Learn. Represent.*, 2020.

[16] E. Delage and Y. Ye, "Distributionally robust optimization under moment uncertainty with application to data-driven problems," *Oper. Res.*, vol. 58, no. 3, pp. 595–612, 2010.

[17] W. Wiesemann, D. Kuhn, and M. Sim, "Distributionally robust convex optimization," *Oper. Res.*, vol. 62, no. 6, pp. 1358–1376, 2014.

[18] Z. Hu and L. J. Hong, "Kullback-Leibler divergence constrained distributionally robust optimization," *Available at Optimization Online*, 2013.

[19] C. Bandi and D. Bertsimas, "Robust option pricing," *Eur. J. Oper. Res.*, vol. 239, no. 3, pp. 842–853, 2014.

[20] A. Sinha, H. Namkoong, and J. Duchi, "Certifying some distributional robustness with principled adversarial training," in *Intl. Conf. Learn. Represent.*, Apr. 2018.

[21] J. Blanchet and K. Murthy, "Quantifying distributional model risk via optimal transport," *Math. Oper. Res.*, vol. 44, no. 2, pp. 565–600, 2019.

[22] C. Villani, *Optimal Transport: Old and New*. Springer Science & Business Media, 2008, vol. 338.

[23] A. Sinha, H. Namkoong, R. Volpi, and J. Duchi, "Certifying some distributional robustness with principled adversarial training," *Int. Conf. Learn. Represent.*, 2018.

[24] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[25] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2014, pp. 701–710.

[26] A. Grover and J. Leskovec, "Node2Vec: Scalable feature learning for networks," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2016, pp. 855–864.