R #1 and #5: "computation cost". Our bandit sampler only requires additionally E floats to store the alternative sampling distribution that will be optimized, where E is the number of edges used for the message passing operations in GNN. Beyond that, no further storage is required. This is true even for more sophisticated architectures where messages are passed between neighbors beyond one hop. The bandit sampler has the same time complexity as any other node-wise sampling approaches. Fig. 1 (left) shows the runtime on PPI using deeper GCNs.

R #2 and #5: "combinatorial bandits" or "how other simple bandit algorithm such as eps-greedy can benefit, and how to apply". The major contribution of this work focuses on the formulation of neighbor selection as a bandit problem. We believe using other more sophisticated combinatorial bandits should be promising, and is left as future study. We can simply adapt epsilon greedy algorithm to our bandit framework. We have alternative sampling probability for the arm with the largest reward as $\frac{\epsilon}{K} + 1 - \epsilon$, and each of rest $(K-1)$ arms as $\frac{\epsilon}{K}$. We can estimate the reward of arm j by $\frac{\text{sum of rewards when arm } j \text{ taken before } t}{\text{number of times when arm } j \text{ taken before } t}$ to determine the arm with largest reward before sampling. We setup a simple test. We have a list of numbers [1,2,3,4,5,6,7,8] with mean 4.5. We aim to estimate the mean by sampling with sample size 2. After 1000 times of sampling, we hope to have unbiased estimate with low variance. We compare three sampling approaches, i.e. random, EXP3 and eps-greedy. All of them are unbiased estimators, however, the variance of EXP3 is 0.25, while the variances of eps-greedy and random are 0.63 and 1.22 respectively. Finally, We show the convergence of our bandit approach with epsilon-greedy and EXP3 on PPI in Fig. 1 (middle).

R #1 and #5: "hyperparameter", "how to select sample size k". We perform grid search or follow previous literature for setting the hyparameters. We elaborate the hyperparameter setting in the supplement. We note that Theorem 1 shows that our algorithm can converge to optima using any sample size k. We simply select a small sample size k to report the numbers, and do the sample size analysis in section 7.3 as well.

R #1: **(1)** "More baseline approaches are needed." To the best of our knowledge, we have tried our best to include all state-of-the-art approaches in the related works and experiments. **(2)** "Table 3 is confused." The comparison is fair only if we compare the proposed sampling approach with other sampling approaches under the *identical* neural architectures. So we report the results of GCNs and attentive GNNs in table 2 and 3 respectively. There are no sampling approaches work for attentive GNNs that can guarantee sampling variance and unbiased estimates to our best knowledge. We promise to include more analyses on those numbers. **(3)** "more experiments on large dataset". We have tested our approach on several large OGBN datasets. For example, our bandit sampler obtained 0.7825 on OGBN-proteins (132,534 nodes) dataset and 0.803 on OGBN-products (2,449,029 nodes) dataset, which significantly outperform other sampling approaches using the same GNN architecture. We will include these numbers in the paper. **(4)** "running time". We have reported the convergence in terms of running time in the supplementary material.

R #2: **(1)** "insight into why bandit works". Optimize the variance help the convergence. However, optimal variance requires the knowledge of all the neighbors' embeddings that are computation infeasible. Our chance is to exploit the sampled good neighbors. This motivates us using bandit algorithms.

R #3: **(1)** "why set gradient of the variance as the reward". The variance difference (lhs on eq.7) to be minimized is upper bounded by the rhs due to convex property. The rhs on eq.7 can be interpreted as expected loss (negative reward) with policy Q_i^t minus expected loss with optimal policy. Hence, the negative derivative of variance is reward in case we optimize this rhs. Moreover, this upper bound yields lemma 1 in the supplement and lets us prove the result in theorem 1. **(2)** "plots on toy data to show the convergence as degree increases". We set up the following experiments. We randomly sample 100 labeled nodes {1,...,i,...,100} with each μ_i uniformly sampled from [-10, 10]. For each labeled node i we generate k neighbors, and its neighbors' features are 1-dimensional scalars in real field that are sampled from $\text{uniform}(\mu_i - \sigma, \mu_i + \sigma)$. Each node i's label is generated by simply averaging its neighbors' 1-dimensional scalar features. We compare the convergence with a random sampler by increasing $k = 50$ to 100 and 200 in Fig. 1 (right). All samplers use a fixed sample size 10. **(3)** "a plot comparing the empirical regret with the upper bound". The optimal variance is hard to compute, but can be viewed as a constant. We will show the variances w.r.t epochs in revision.

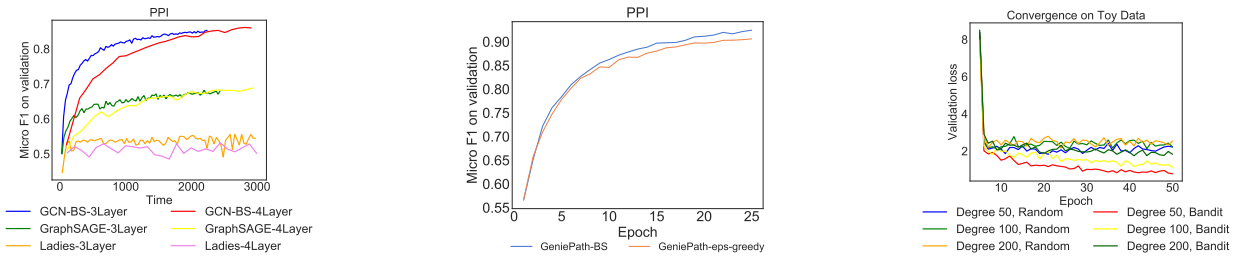

Figure 1: **left**: deeper arch running time; **middle**: EXP3 vs. eps-greedy; **right**: convergence on toy data.