R1 W1: The main contribution is the combination of the Monte-Carlo Propagation and the Reverse Push Propagation, which is the key to achieve sub-linear complexity. Furthermore, the proposed GBP algorithm is the first method that achieves good scalability and performance on the billion-scale dataset Friendster.

R1 W2: The value of r_{max} actually depends on the sparsity of the feature matrix $\mathbf{X}$. The features of Cora, Citeseer, and Pubmed are (sparse) bag-of-words features and thus can tolerate large error. The features of other datasets are dense representations, and thus we have to set r_{max} to be small for an acceptable error.

According to Theorem 1, the number of random walks per node w is automatically determined by r_{max}. In practice, for full-supervised learning tasks (PPI, Yelp, and Amazon) where most nodes are training nodes, we simply set $w = 0$ and only rely on the results of the Reverse Push Propagation. For semi-supervised tasks on Friendster, we set $w = 10,000$, which significantly reduces the Reverse Push Propagation error and improves the model accuracy.

R1 W3: The default value of r is $1/2$, and we perform a grid search to find the best value for r. In most cases, setting $r = 1/2$ can achieve satisfying results. Line 436 presents the choice of w_ℓ, please see **R3 W1** for a detailed discussion.

R1 Correctness: We will use the suggested evaluation methods in the final version of the paper.

R2 W1 & Major comment 1: Decoupling prediction and propagation sometimes improve performance (see the ablation study on small datasets in the APPNP paper). We will include an ablation study on the larger datasets.

R2 W2 & Major comment 2: GBP can support edges weights by treating the adjacency matrix $\mathbf{A}$ as a weighted matrix. Supporting edge features, however, is beyond the scope of this paper.

R2 Major comment 3: We follow the original setting of SGC to compute the prediction $\mathbf{Z} = SoftMax(\mathbf{S}^K\mathbf{X}\mathbf{W})$, which means there is no hidden layer. We will include the results of SGC running with a multi-layer network.

R2 Minor comments 1-8: Thanks for these insightful comments! Our response: 1) For GNN, the $O(m)$ term is multiplied by L and F, so it makes sense to optimize the $O(m)$ term. 2) We agree that the scalability limitations arise from neighborhood explosion and memory consumption of storing activations. Line 72 is merely a theoretical argument to show why we are not trying to improve the $O(LnF^2)$ term. 3) We will include the number of clusters as a hyperparameter for Cluster-GCN and analyze its complexity in the Stochastic Block Model. 4) We only analyzed the complexity of GraphSAINT with node sampling for simplicity. 5) We will include VRGCN in Table 1. 6) To the best of our knowledge, there is no study on the expressiveness of the PageRank-based GNNs in terms of the Weisfeiler-Lehman test. 7) We combine the results of the Monte-Carlo Propagation with each Reverse Propagation vectors, leading to the extra F term. 8) The technical reason is that the codes of LADIES and GraphSAINT store multiple views of the graphs to enable efficient subgraph sampling, which leads to memory overflow when the graph size is large.

R3 W1 & R1 W3: At line 436, we mention that GBP uses Personalized PageRank (PPR, $w_\ell = \alpha(1-\alpha)^\ell$) for all datasets other than Friendster. In fact, we can simply use PPR for datasets with real-world features. Friendster is a rare exception, where we have to set the w_ℓ to be the L-th transition probability ($w_L = 1, w_0 =, \ldots, = w_{L-1} = 0$). This is because PPR emphasizes each node's original feature (with w_0 being the maximum weight among $w_0, w_1, \ldots$) and, yet, the original feature of Friendster is random noise. We will include the above discussion in the final version of the paper.

R3 W2: We will include a discussion on PPRGo (KDD2020). Note that PPRGo precomputes the approximate Personalized PageRank (PPR) matrix $\mathbf{S}$ and uses matrix multiplication to calculate the propagation matrix $\mathbf{S} \cdot \mathbf{X}$. In contrast, GBP estimates $\mathbf{S} \cdot \mathbf{X}$ by the reverse push from $\mathbf{X}$ and thus avoids the computation of $\mathbf{S}$.

R3 W3: The main focus of this paper is to improve the scalability of GNN. APPNP is unable to scale on large graphs.

R4 W1: As suggested, we present the performance of GDC on the three small datasets in Table 1, which is similar to that of GBP. However, GDC is unable to run on the larger datasets due to its $O(n^2)$ space/time complexity. Note that GDC performs the sparsification operation after computing the $n \times n$ diffusion matrix, which inherently leads to the $O(n^2)$ space/time complexity. In fact, one of the main contributions of this paper is to improve the scalability of PageRank-based GNN such as APPNP and GDC.

Table 1: GDC on three small datasets.

Method	Cora	Citeseer	Pubmed
GDC	83.3 ± 0.2	72.2 ± 0.3	78.6 ± 0.4
GBP	$\mathbf{83.9 \pm 0.7}$	$\mathbf{72.9 \pm 0.5}$	$\mathbf{80.6 \pm 0.4}$

Table 2: LADIES with varying number of samples per node.

#Samples	256	512	1024	2048
PPI	57.8 (205)	**59.4** (206)	57.5 (215)	58.1 (206)
Yelp	27.1 (34)	28.5 (39)	26.1 (44)	**29.6** (37)
Amazon	84.8 (793)	**85.2** (784)	85.1 (787)	84.8 (799)

R4 W2: We present the performance of LADIES with a varying number of samples in Table 2. There is no significant change in performance and training time as we tune the number of samples per node from 256 to 2048.