We appreciate the valuable feedback from all the reviewers and will include the following discussion into our work.

Q1. Discussion on GNNExplainer for multi-instance case (Reviewers #1 and #4). As discussed in lines 47-52, to explain a set of instances, GNNExplainer first interprets a representative instance and then adopts ad-hoc post-analysis to generalize to multiple instances. We believe that this is not an elegant way to have a global view of the GNN model. "Since the explanatory motifs are not learned end-to-end, the model however may suffer from sub-optimal generalization performance." Instead, the explanation in PGExplainer is generated with a parameterized network, where parameters are shared among populations. PGExplainer is natively designed for collectively explaining multiple instances.

Q2. Clarification for technical detail (Reviewer #1). The inner workings of the method (how to sample the subgraph) are mainly described by equations. Note that index notations, such as i, in different contexts may have different meanings. Using graph classification as an example, in Algorithm 2, i is the index of an instance (graph) to be explained. In line 8 of the algorithm, we iterate over all edges in the graph and each corresponding latent variable is calculated with Eq. (11). Since the index (i, j) is used to indicate an edge in Eq. (11), we don't expand Eq. (11) to avoid confusion. The source code of PGExplainer can be found in GitHub with the name "PGExplainer".

Q3. The usage of AUC (Reviewer #1). We follow the experimental setting in GNNExplainer. In their paper, "explanation accuracy" is not formally defined. From their source code, we find that they actually used AUC (line 327, explainer/explain.py in their GitHub repository). Besides, comparing to classification accuracy, AUC is a more comprehensive measurement to evaluate binary classification.

Q4. Std in Table 1 (Reviewer #1). We didn't report std for baselines because they don't have sampling processes in subgraph generation. GNNExplainer doesn't report std in their paper either. Since PGExplainer includes sampling, std is reported to show its stableness. Baselines' stds are shown in the table below. (GRAD is deterministic with 0 std.)

	BA-Shapes	BA-Community	Tree-Cycles	Tree-Grid	BA-2motifs	MUTAG
ATT	0.815 ± 0.005	0.739 ± 0.015	0.824 ± 0.012	0.667 ± 0.005	0.674 ± 0.040	0.765 ± 0.013
GNNExplainer	0.925 ± 0.002	0.836 ± 0.001	0.948 ± 0.005	0.875 ± 0.012	0.742 ± 0.001	0.727 ± 0.014

Q5. Discussion on related work (Reviewer #1). Methods in the CVPR paper mentioned only explain a specific type of GNN: GCN, on a specific task: graph classification. PGExplainer is a general model compatible with different GNNs and diverse learning tasks. Besides, instead of edge-level important scores, they just calculate node-level important scores. Thus, this paper was not cited and compared by GNNExplainer. We select a method "Gradient" in the CVPR paper which doesn't require the global average pooling layer to calculate the importance of each node, then calculate the importance of an edge by average the connected nodes' importance scores. The AUC scores on BA-2motifs and MUTAG are 0.773 and 0.653, respectively, much lower than PGExplainer. The KDD paper mentioned just showed up (June 3). It only applies to graph classification. Second, it only provides model-level explanations without preserving the local fidelity. Thus, the generated explanation may not be a substructure of the real input graph. Instead, PGExplainer can provide an explanation for each instance with a global view of the GNN model, which can preserve the local fidelity.

Q6. Usage of "global" (Reviewers #2 and #3). We provide an explanation for each instance, which is generated from a parameterized network. Parameters are shared among instances in the population, which equips PGExplainer with a global view of the explained GNN model. That's why we call it a global method. For quantitative evaluation, we follow the setting in GNNExplainer to evaluate the explanation for each instance and then report the average AUC. In Fig. 1, we use a case study to show a high frequent motif in mutagens. For MUTAG dataset, we only evaluate with mutagens containing NH_2 or NO_2. In Table 1, "house" motif is used as GT in BA-Community (line 287).

Q7. Discussion on "local fidelity" and "general understanding" (Reviewer #2). Both "local fidelity" and "general understanding" are important to interpret a GNN model. As discussed in [38], "local fidelity" requires an explanation corresponds to how the model behaves in the vicinity of the predicted instance. For example, in MUTAG dataset, ubiquitous motifs for mutagens include NH_2 and NO_2 [50,9]. Some mutagens only have NH_2 and some only have NO_2. Thus, it is important to give an explanation for each instance. Meanwhile, "global perspective", or "general understanding" is important to ascertain trust in the mode [38]. However, by generating a painstakingly customized explanation for each instance **individually and independently**, it is challenging for GNNExplainer to have a "general understanding" of the GNN model, because the GNN model is trained with multiple instances. Thus, we proposed PGExplainer to preserve the "local fidelity", at the same time, with a global view of the GNN model. Lacking of "global" view of GNN model explains why some important edges are missing in the explanations given by GNNExplainer.

Q8. Our contributions (Reviewer #4). GNNExplainer is a pioneer to provide explanations for GNN's predictions. We include a parameterized network to enable explainer a global view of the GNN model. By collectively explaining a set of instances, PGExplainer can achieve more consistent and accurate explanations. Besides, since the parameterized network is shared among the population, PGExplainer is able to explain new instances without retraining. Empirically, PGExplainer is much more effective and efficient than the state-of-the-art method. It is non-trivial to significantly improve the accuracy performance with much less running time.

Q9. Discussion on total training time of PGExplainer (Reviewer #4). As discussed in Appendix D.1, "PGExplainer can achieve relatively good performance with a small number of trained instances, which makes PGExplainer more practical in large datasets. The results also explain why we dismiss the training time of PGExplainer and only count the inference time in Section 5" (line 541-543).