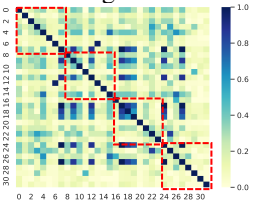
We sincerely thank the reviewers for the constructive comments and would like to address them as follows.

{R1.1 Computing GED$_E$} Yes, for the synthetic dataset, GED$_E$ may contain noises due to the reason mentioned by R1. However, for real-world datasets like ZINC, GED$_E$ will be the same as GED, yet with a much lower complexity.

{R1.2 Random method} The GED$_E$ and C-Score are 32.09 ± 4.85 and 0.315 ± 0.002 when using four factor graphs.

{R1.3 Visualization of DisenGCN} We add the correlation visualization of DisenGCN, shown in the figure below.



{R2.1 Graph classification methods} Thanks for the comment. In fact, we have indeed compared with GIN, published in 2019 and the SOTA graph-classification method on the datasets we used (10-fold c.v.). As suggested, we have also added results of DiffPool and GatedGCN, both tailored for graph-level tasks. On the ZINC dataset, the MAEs for DiffPool, GatedGCN, GIN, and Ours (FactorGCN) are 0.466, 0.437, 0.387, and 0.366, showing the superiority of our method, let alone its capability for graph-level disentanglement.

{R2.2 Improvement} The performance improvement is truly not trivial. On ZINC, a large-scale real-world dataset, FactorGCN achieves an MAE of 0.366, while GIN, MoNet, GAT achieve 0.387, 0.397, 0.479 under the same setup [1]. The improvement over the best model is 5%, which, given the nature of the task and SOTA results, can be indeed considered as significant. Such claim is also supported by the paired t-test.

{R2.3 Node classification and IPDGN(Liu et al. 2019)} We add node-classification experiments on the large "Pattern" dataset (14K graphs, 1,664,491 nodes) [1]. Results including IPDGN are shown below. All models contain four layers.

Method (Acc) | Random (50.0) GCN (63.9) GatedGCN (84.5) GIN (85.6) MoNet (85.5) DisenGCN (75.0) IPDGN (78.7) Ours (**86.6**)

{R2.4 Paired t-test} As suggested, we conduct paired t-test between our method and the second-best ones. The p values on the Synthetic, ZINC, IMDB-B, COLLAB, and MUTUG are < 0.0005, < 0.0005, > 0.25, > 0.25, and > 0.25. It shows that our method performs significantly better on the first two datasets and on par with SOTA for the rest.

{R2.5 Consistency of factor graphs} Thanks. Like any other disentanglement method [3], there is no absolute guaranty that the disentangled factors will be strictly consistent with the ground truth. However, by enforcing the diversity of factors, as also done in many disentanglement methods like [3], the model tends to generate factors that contain the natural patterns (ground-truth ones) of the input. This is empirically supported and validated by the higher GED$_E$.

{R2.6 High-level formulation & motivation} Thanks for the nice suggestion. The core idea can be formulated as $\text{argmax}_{\mathcal{F},\mathcal{D}} \left(\mathbf{P}(\mathbb{G} \subseteq \mathcal{F}(G), \mathcal{D}(\mathcal{F}(G)) = \mathbb{Y}) | G)\right)$, where $\mathcal{F}$ will disentangle G to a set a factor graphs, and $\mathcal{D}$ will generate the label of G based on the factor graphs. We seek the optimal $\mathcal{F}$ and $\mathcal{D}$ to maximize the probability that the generated factor graphs contain the ground truth ones and the predicted label equals to the true label. Our motivations are two-fold: disentangling the input globally will account for higher-order relations among nodes; multi-relational disentangling will allow us to discover various relations between the same pair of nodes. We will add these to revision.

{R2.7 Clarity} Thanks. We will remove bold fonts in Tab. 2. Readout in Eq. 2 and pooling method are mean pooling.

{R3.1 Modified GAT} Thanks for the nice suggestion. As advised, we add a discriminator to GAT; the Micro-F1(↑), GED$_E$(↓), and C-Score(↑) on synthetic dataset are 0.928, 12.35, 0.274, while those of ours are 0.995, 10.56, and 0.532. It shows that the factor-graph method indeed leads to better disentanglement performances.

{R3.2 More details} Thanks. The correlation is computed on all graphs. Fig.4 is conducted on the ZINC dataset. When varying λ, #factors is set to be eight; when varying #factors, λ is set to be 0.2. We will add the details to the revision.

{R4.1 Novelty} We kindly solicit R4, if possible, to re-assess the novelty from task- and evaluation-perspective. Task-wise, we introduce the first graph-level disentanglement method via GNN; metric-wise, for the first time we propose a quantitative evaluation protocol of the graph disentanglement. Both could potentially interest a large audience in GNN.

{R4.2 Multi-relational methods} In fact, the setup of disentanglement is different from that of multi-relational models: the former one aims to factorize a *single-relational* input graph into multiple graphs, while the latter requires a *multi-relational* graph as input. As a result, the former task is expected to be much challenging than the latter. Due to the setup difference, the proposed method does not support taking heterogeneous network as input. We will clarify this.

{R4.3 Enforcing classification} Thanks. The classifier will prevent model from collapsing to the point where all the factor graphs are the same (e.g. all equal to the input), and encourage different factor graph to focus on different sub-structures of the input. Without the classifier (i.e., setting $\lambda = 0$), the GED$_E$ will degrade from 12.6 to 13.0.

{R4.4 Clarity: goal and superiority} Our method aims to disentangle an input simple graph into several factor graphs via a GNN. For the first time, this is done via a graph-level factorization and quantitatively evaluated using the proposed graph-disentanglement metric, let alone the SOTA results. As suggested, we will clarify this in the revision.

{R4.5 Additional feedback/Relation to prior work} **[#node in Fig. 2]** Isolated nodes in Fig. 2 are removed for better visualization. The #node of each graph is in fact the same. **[Tested on different tasks]** The manuscript includes graph regression and classification tasks. We also add node classification task (please refer to R2.3), where **IPDGN** is added. **[Input]** For synthetic, IMDB-B, and COLLAB, uniform embedding is used as the node feature; for ZINC, each node has an one-hot vector representing the atom type (28 in total); for MUTAG, seven bios are used as node features. **[Graph-classification methods]** Thanks. Please refer to R2.1. **[σ in Eq. 3]** Yes, $G_i^e[c]$ in Eq. 3 contains an activation function (σ) implicitly. **[Typos]** Thanks and we will fix them.

[1] Dwivedi, Prakash, et al. Benchmarking graph neural networks. **[2]** Xu, Keyulu, et al. How powerful are graph neural networks? **[3]** Locatello, Francesco, et al. Challenging common assumptions in the unsupervised learning of disentangled representations