We thank the reviewers for their valuable comments. We will fix all typos and re-organize the contents in the final nine-page paper, as advised. We will add the suggested related work, clarify the iterative module, and include more descriptions of PathGNN and more experimental details in the main text of the final paper.

Comparison to Iter-Path and 3/10/100-layer GNNs (R1& R2 & R3): A: We first compare with Iter-Path (a baseline w.o. homogeneous prior) on more tasks. The conclusions remain unchanged. Currently, the available results are:

Shortest Path		Component Cnt.		Physical sim.		Image-based Navi.	
0.0005 (ER)	0.09 (Lob)	89.2% (ER)	89.5% (Lob)	0.13 (50)	5.78 (100)	89.4% (16×16)	78.6% (33×33)

We also evaluate 3/10/100-layer PathGNNs. Their generalization performances on the shortest path (Lob) are 0.52, 0.26, 0.13, respectively, which are not as accurate as our IterGNN (0.02). We will include other results in the revised version.

R1: Difference between our IterGNN and ACT [31]. A: Compared with ACT, several important differences enable IterGNN to achieve much larger iteration numbers so that it can generalize to very large graphs. As illustrated in Figure 3, our IterGNN can iterate for 2500 times during inference while ACT only iterates for less than 30 times (more analysis in Section E.1.2 and C.1.2). **Not perfect performance on component counting.** A: Ordinary GNNs need random node features to increase representation power for solving the component counting problem, which makes optimization difficult (due to the high variance) and causes imperfect performance. **Other shortest path problems.** A: Our model can easily solve one of the multi-source shortest path problems, i.e., finding the distance to the nearest source nodes. We do not know any straightforward way to find pairwise distances efficiently. **Related works reporting notable generalization performance w.r.t. graph scales.** A: We will include them in the revised version. [r1] only evaluated on graphs with limited diameters (≤ 5) so that their fixed-depth paradigm would work. [r2] manually set the number of iterations as large as possible for all samples to achieve more generalizability. This approach is computationally expensive and makes performance worse in our experiments (Shared-Homo-Path in Table 3) possibly because of the accumulated errors after unnecessary iterations.

R2: Reasons for not returning the last h & Rephrase the iterative module with examples. A: We cannot return the last h^K at the step K because if so, the terminal signal $c^i = g(h^i)$, where g is the stopping criterion function, is not involved in the computation of $h^K = f \circ f \circ \cdots f(h^0)$ and no gradients would be propagated to train the function g. Instead, we return the expected value of h as $h = \sum_{i=1}^{\infty} c^k \prod_{i=1}^{k-1}(1-c^i)h^k$ so that there are gradients to c^i, enabling us to train the function g end-to-end without extra supervisions. We can interpret h as the expectation of the following random process: at each step i, we stop the iteration and output h^i with probability c^i. Then, starting from step 1, the probability for the random process to stop at step k and output h^k is $p^k = c^k \prod_{i=1}^{k-1}(1-c^i)$. The expectation is then h. The iteration number is the number of iterations. We will include an example in the main text. **Training/using deep GNNs.** A: We agree it would not work well. The layer numbers of those deep GNNs are still fixed or bounded. Therefore, they cannot solve many simple graph-related tasks, as proved in [19]. Deep GNNs are also computationally expensive and difficult to train. Moreover, the works that aim at training deep GNNs are orthogonal to our work. We may incorporate those techniques, e.g. for better convergence. **Using L_∞ norm for Theorem 2.** A: Yes. We provide an arbitrary-width version of Theorem 2 (Theorem B.2.2 in Appendix) that can incorporate the L_∞ norm.

R3: Why does homogeneous prior seem to help with generalizing over graph size? A: In general, for graphs of larger sizes, we expect that the scale of some internal features will also be large to capture those graph properties related to the graph size, e.g., the distances or community numbers (Detailed example in L33-40). The homogeneous prior helps handle those out-of-range features which then improves the generalizability w.r.t. graph sizes.

R4: Contributions and generalizability of the elements. A: The generalization performances are shown in Table 1,2,3. We did extensive ablation studies by adding components one-by-one and by removing each part from our best model. We fixed all other hyper-parameters for a fair comparison, and the results showed the contribution of each element clearly. **Non-standard benchmarks with only a few very basic GNN variants compared.** A: We have already compared our models with the most popular GNN variants. As for the benchmark, there is no such standard benchmark evaluating the generalizability of GNNs w.r.t. scales, so we adopt common practices in related fields when designing experiments, such as diverse graph generators for the shortest path problem and the exact same TSP-problem generators as the benchmark paper mentioned by the reviewer R4. **Comparison with the gates in LSTM/GatedGNN.** A: Although our IterGNN involves multiplications of hidden representations (like the gates), it is significantly different from LSTM/Gated-GNNs in terms of internal structures and usages. LSTM and Gated-GNNs require extra supervisions of the stopping conditions during training, e.g., the End-of-Sequence (EoS) symbol. **Other related works.** A: Flow-based models (e.g., Graph ODE or Continuous GNNs), as stated in L99-101, do not explicitly *learn* the iteration controller. It is unclear whether they can solve the shortest path problem on graphs of arbitrary sizes. *By the way, Graph ODE and Continuous GNNs are not published at the submission time.* The Position-aware GNN is proposed to improve the representation power of GNNs, similar to our random node features, instead of improving the generalizability w.r.t. scales. **PathGNN is bespoke for specific types of problems.** A: PathGNN is not the focus and main contribution of our work. We focus on the IterGNN and the homogeneous prior to improve the generalizability of GNNs w.r.t. graph scales, as stated in the introduction.