# Appendix

## A Related Work [Extended]

### A.1 Optimal Transport Plan

The optimal transport (OT) distance (e.g., Wasserstein distance or Gromov-Wasserstein distance) provides a flexible way to compare and couple probability distributions. In machine learning, optimal transport distances can be used to perform various learning tasks over histograms (distributions) [2, 56]. With the increasing prevalence of graphs [50, 69, 75], OT is being increasingly applied for their processing. Existing studies directly apply the final result (e.g., optimal transport distance or plan) from the OT problem to corresponding tasks, such as graph alignment and partition [14, 54, 68], domain adaptation [11, 48] and shape matching [33, 43], rather than using the transport plan to design a specific model. Furthermore, to the best of our knowledge, studies focusing on considering multiple transport plans (*independently*) are currently found only in the design of OT algorithms [34, 46, 57]. Mémoli [34] implicitly introduces the decoupled variant of the Gromov-Wasserstein distance in the optimization procedure, which considers two coupling measures (plans) independently. [57] makes this process explicit that seeks to simultaneously learn two independent correspondences (i.e. rows and columns of matrices) by decoupling the GW distance. In contrast to these approaches that consider different plans independently, our work aims to align two closely related transport plans, which consider the same dependency and matching relationship that a pair of objects should have under different spaces (e.g., graph space and representation space).

## B Optimization, Complexity, and Implementation Details

### B.1 Optimization

The optimization problem for $\pi$ mainly includes the computation of the Wasserstein and the Gromov-Wasserstein terms, where the former is a linear program and the latter is a nonconvex quadratic program. To improve the efficiency of the algorithm with respect to the GW term, we use semi-relaxed Gromov-Wasserstein divergence

$$\pi^* = \underset{\pi \in \Pi(\mu,m)}{\operatorname{argmin}} \langle \boldsymbol{L}(\boldsymbol{A}_1, \boldsymbol{A}_2) \otimes \pi, \pi \rangle \tag{13}$$

where

$$\Pi(\mu, m) := \{\pi \in \mathbb{R}_+^{n \times m} | \pi \mathbf{1}_m = \mu\},$$

in [60] and optimize the transport plan with the conditional gradient (CG) solver. The gradient of $\pi$ with respect to Equation (13) is $\nabla_\pi = 2\boldsymbol{L}(\boldsymbol{A}_1, \boldsymbol{A}_2) \otimes \pi$. The conditional gradient algorithm [21] consists in solving a linearization $\langle \boldsymbol{X}, \nabla_\pi \rangle$ at each iteration $r$. It can be solved by gradient descent with a direction $\boldsymbol{X}^{(r)} - \pi^{(r)}$, followed by a line search for the optimal step. The details of the algorithm are summarized in Algorithm 1. As a result, the optimization for Equation (7) only requires replacing the gradient with $\nabla_\pi = \sigma \boldsymbol{K}(\boldsymbol{X}_1, \boldsymbol{X}_2) + 2(1-\sigma)\boldsymbol{L}(\boldsymbol{A}_1, \boldsymbol{A}_2) \otimes \pi$.

For the Wasserstein term, we can solve the linear program with OT network simplex solver [4] or using the Sinkhorn-Knopp algorithm [12] which allows a fast computation of the transport plan. The Sinkhorn-Knopp algorithm iteratively approximates the optimal solution $\pi^*$. Specifically, the Sinkhorn-Knopp algorithm add an additional entropy regularizer and perform a scheme of alternating Sinkhorn projections: $\pi^{(0)} = \exp(-\boldsymbol{J}(\boldsymbol{Z}_1, \boldsymbol{Z}_2)/\lambda)$ and $\pi^{(t+1)} = \mathcal{S}(\mathcal{T}(\pi^{(t)}))$, where $t$ denotes the number of iterations, $\lambda$ weights the regularization, $\mathcal{S}(\pi) = \pi \oslash (\mathbf{1}\mathbf{1}^\top \pi) \odot (\mathbf{1}\boldsymbol{b}^\top)$ and $\mathcal{T}(\pi) = \pi \oslash (\pi \mathbf{1}\mathbf{1}^\top) \odot (\boldsymbol{a}\mathbf{1}^\top)$, $\odot$ denotes the Hadamard product and $\oslash$ denotes element-wise division. As shown by [12], in the limit this scheme converges to a minimizer $\pi^{(t)} \overset{t\to\infty}{\longrightarrow} \pi^*$.

We compute the gradient $\nabla_{Z_i} \pi^{(t)}$ using backpropagation to update the parameters of GNNs in the optimization progress. It's noteworthy that extensive research has been conducted concerning the convergence properties of this differential mechanism. Recent advancements in this domain are highlighted in the paper [40], wherein theoretical proofs have been established. Theorem 3.3 of [40] imples that $\pi^{(t)}$ is continuously differentiable for all $t$ and the sequence of derivatives $\nabla_{Z_i} \pi^{(t)}$ converges at a linear rate. In particular, for all $Z_i$, $\nabla_{Z_i} \pi^{(t)} \overset{t\to\infty}{\longrightarrow} \nabla_{Z_i} \pi^*$, where $Z_i$ is the independent variable of the cost matrix $\mathbf{C}(Z_i)$ and it corresponds to $\theta$ in the original text of [40].

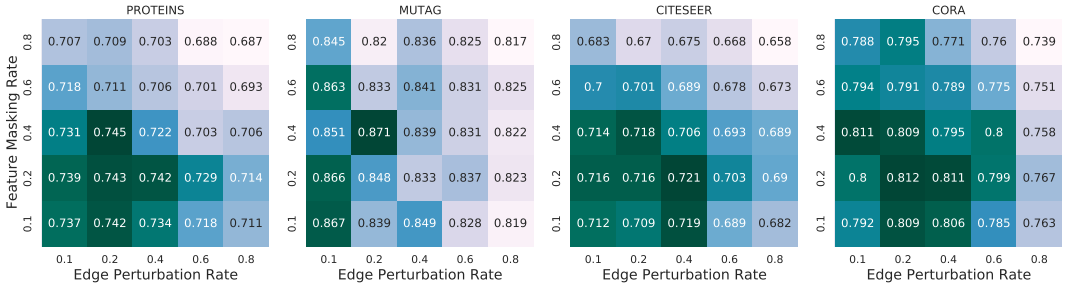
**Algorithm 1** Conditional Gradient Solver.

**repeat**
  $\nabla_\pi^{(r)} \leftarrow 2\boldsymbol{L}(\boldsymbol{A}_1, \boldsymbol{A}_2) \otimes \pi$.
  $\boldsymbol{X}^{(r)} \leftarrow \operatorname{argmin}_{\boldsymbol{X} \in \Pi(\mu, m)} \langle \boldsymbol{X}, \nabla_\pi^{(r)} \rangle$.
  Line search the optimal step size for the descent direction: $\gamma^\star = \operatorname{argmin}_{\gamma \in [0,1]} \langle \boldsymbol{L}(\boldsymbol{A}_1, \boldsymbol{A}_2) \otimes \boldsymbol{Z}^{(r)}(\gamma), \boldsymbol{Z}^{(r)}(\gamma) \rangle$, where $\boldsymbol{Z}^{(r)}(\gamma) = \pi^{(r)} + \gamma(\boldsymbol{X}^{(r)} - \pi^{(r)})$.
  $\pi^{(r+1)} \leftarrow (1 - \gamma^\star)\pi^{(r)} + \gamma^\star \boldsymbol{X}^{(r)}$.
**until** Convergence.

### B.2 Implementation details

The proposed model GALOPA mainly consists of three components: graph perturbation, the backbone model, and optimization of the OT plan. To perform graph augmentation, we use 4 types of operations: Edge Perturbation, Feature Masking, Node Dropping, and Graph Sampling. The encoder used for our model is GCN. We use Algorithm 1, OT network simplex solver [4], or Sinkhorn-Knopp algorithm [12] for the optimization of the OT plan. Our model is implemented with Pytorch Geometric [13] and Deep Graph Library [63]. All experiments are conducted on the Linux server (version 5.15.0-72-generic) with 12 Intel(R) Xeon(R) CPUs (E5-2603 v4 @1.70GHz) and two NVIDIA RTX A5000.

## C Perturbation on Graph Contrastive Learning




(a) GRAPHCL (b) GRAPHCL (c) BGRL (d) BGRL

Figure 7: The mean graph/node classification accuracy when contrasting with different perturbation rate under 4 datasets. Fix one of the augmentations as NoAug and the other augmentation be the combination of edge perturbation and feature masking. Darker colors indicate better performance.

In this section, we propose to investigate the performance fluctuations of the traditional graph contrastive learning algorithms when the anchor graph is perturbed with different augmentation rates. As with Section 7, when augmenting the graph, we fixed one of the augmentations as NoAug and the other augmentation requires a hyper-parameter "*aug ratio*" that controls the portion of node attributes/edge that are selected for perturbing. We perform two augmentation policies, edge perturbation and feature masking, with different augmentation rates on 2 social network datasets, CORA and CITESEER [25] for node classification, and 2 graph classification data PROTEINS and MUTAG from TUDataset [36] for graph classification. We select two state-of-the-art GCL algorithms GRAPHCL [72] and BGRL [55] and evaluate them on graph data and network data, respectively. The results are shown in Figure 7. As shown in Figure 7, the performance of GRAPHCL and BGRL fluctuates considerably when the perturbation rate is too small or too large. For example, if both the edge perturbation rate and feature masking rate are set to 0.8, GRAPHCL is 5.8% below its optimal performance on the dataset PROTEINS. BGRL degrades 6.3% and 7.3% on the datasets CITESEER and CORA, respectively.

## D Comparing with the Model based on Graph Edit Distance

More recently, a self-supervised learning model D-SLA based on graph edit distance [22] aims to learn the discrepancy between the original and the augmented graphs and utilizes the graph edit

distance to train graph discriminator to predict whether a graph is an original graph or a perturbed one. Since the graph edit distance is not directly comparable with the embedding-level distance, the algorithm relaxes the constraint to let the ratio of two embedding-level distances in the representation space equal to the ratio of the corresponding two graph edit distances in the graph space. In other words, this is equivalent to assuming the existence of an appropriate linear mapping over the graph edit distance and forcing the embedding-level distance equal to the mapped edit distance. This model has the following limitations: (1) Inconsistency of discrete and continuous distances. The graph edit distance is actually a discrete distance, and deleting or adding nodes (edges) causes the distance to change by a constant cost. In contrast, the embedding-level distance (e.g. Euclidean distance) between graph representations obtained from the encoder is a continuous distance. Forcing a linear relationship between the two distances is not realistic; (2) Insufficiency. Since the same graph edit distance may correspond to multiple different perturbations, it is not sufficient to model the discrepancy between different graphs using the edit distance only; (3) Label-invariance. To prevent the computation of the edit distance between vastly different graphs, which is a complex problem, this method utilizes the principle of contrastive learning. It assumes that the distance between the original graph and the perturbed graph should be less than the distance between the original graph and the negative graph. However, positive and negative samples are still needed for this model, and it relies on the assumption of label invariance.

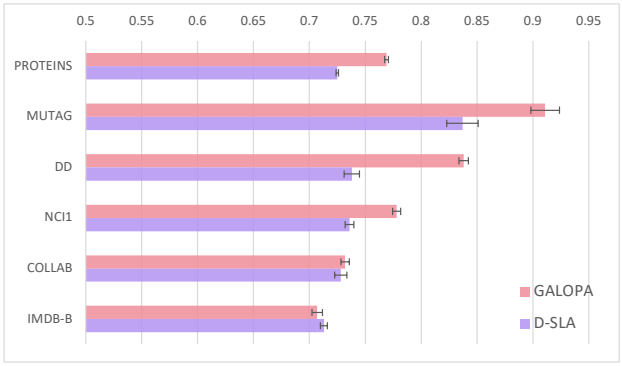



Figure 8: The mean graph classification accuracy on 4 datasets between GALOPA and D-SLA.

Next, we conduct experiments to compare our paradigm with this model based on graph edit distance. We evaluate the performance on 6 graph classification data NCI1, PROTEINS, DD, MUTAG, COLLAB, and IMDB-B from TUDataset [36], since the implementation of the model is mainly oriented towards graph classification. Figure 8 reports the averaged graph classification accuracy results over the graph-level datasets. From Figure 8, we can find that our method significantly outperforms the counterpart for most cases.

## E Experiments on Large Scale Datasets

We performed the experiments in Sections 5-7 on the dataset ogbn-arxiv [20], which contains 169,343 nodes with 1,166,243 edges. We first partition the graph and compute the plan between subgraphs. We perform self-supervised pre-training with all training data and supervised fine-tuning with 10% of them then evaluate the test sets, which is repeated 10 times.

In Section 5, we compare the plan and distance. Table 3 records the node classification accuracies on the ogbn-arxiv when pre-training using the Equations (11) and (12) as losses. 'NoPretrain' refers to direct fine-tuning without pre-training. We can find that using the plan as losses outperforms the counterpart using the distance, which is consistent with the conclusions in the paper. In Section 6, we compare node attributes and edges. If $\sigma = 1$, the model takes into account only node attributes. When $\sigma = 0$, it integrates only edge. We set $\rho = 0$ (or $\rho \neq 0$) to remove (or add) the implicit structure term $\mathcal{L}_{\text{(im)strc}}$. From the table, we can see similar conclusions to the paper. In Section 7, we test the robustness of GALOPA. Table 5 records the performance when both feature masking (vertical axis) and edge perturbations (horizontal axis) are used. We can find that our model is robust on ogbn-arxiv.

## F An Effort to Reduce Complexity

Fast algorithms would come with their associated trade-offs. Efficiency gains may entail some compromise in precision. In light of this observation, our subsequent comparison experiments (Tables 6-9) illustrate that the performance trade-off resulting from these fast algorithms is minor when compared to the significant time savings they offer. This holds particularly true for medium to large

Table 3: Experimental results w.r.t plan versus distance on large-scale dataset ogbn-arxiv.

| Algo. | Test Accuracy |
|---|---|
| NoPretrain | $0.512 \pm 0.31$ |
| Plan | $0.544 \pm 0.18$ |
| Dist | $0.527 \pm 0.26$ |

Table 4: Results on ogbn-arxiv under different values of parameters.

| Para | Test Accuracy |
|---|---|
| $\rho = 0, \sigma = 1$ | $0.523 \pm 0.24$ |
| $\rho \neq 0, \sigma = 1$ | $0.542 \pm 0.37$ |
| $\rho \neq 0, \sigma = 0.5$ | $0.544 \pm 0.34$ |
| $\rho \neq 0, \sigma = 0$ | $0.540 \pm 0.25$ |

Table 5: Results on ogbn-arxiv with diffrent perturbation rate.

| Aug Rate | 0.1 | 0.2 | 0.4 | 0.6 | 0.8 |
|---|---|---|---|---|---|
| **0.1** | $0.542 \pm 0.23$ | $0.543 \pm 0.36$ | $0.540 \pm 0.32$ | $0.538 \pm 0.31$ | $0.532 \pm 0.26$ |
| **0.2** | $0.543 \pm 0.30$ | $\mathbf{0.544 \pm 0.27}$ | $0.544 \pm 0.18$ | $0.543 \pm 0.35$ | $0.538 \pm 0.45$ |
| **0.4** | $0.539 \pm 0.16$ | $0.541 \pm 0.33$ | $\mathbf{0.544 \pm 0.29}$ | $0.542 \pm 0.17$ | $0.540 \pm 0.39$ |
| **0.6** | $0.534 \pm 0.38$ | $0.539 \pm 0.19$ | $0.543 \pm 0.26$ | $0.542 \pm 0.32$ | $0.539 \pm 0.19$ |
| **0.8** | $0.531 \pm 0.23$ | $0.535 \pm 0.25$ | $0.539 \pm 0.19$ | $0.537 \pm 0.36$ | $0.537 \pm 0.41$ |

datasets. Thus, the cost incurred by the use of fast algorithms is judiciously balanced against the benefits they provide.

To provide a more comprehensive evaluation, we devise a variant algorithm known as GALOPA(linear). This variant focuses on computing the transport plan solely based on the node attributes and employs the linear Sinkhorn algorithm [45] to optimize in both the graph space and representation space. The original version of our model retains the name GALOPA(cube). We conduct experiments, maintaining the experimental settings outlined in the paper, across node and graph classification datasets. The performance outcomes are recorded, and the results are presented in the subsequent table.

From the results on Tables 6 and 7 we can observe that the variant GALOPA(linear) exhibits comparable performance with the GALOPA(cube), especially on median/large graphs such as the Amz-Photo (with 7,650 nodes), PubMed (19,717 nodes), Coauthor-CS (18,333 nodes), and Amz-Comp. (13,752 nodes). In some cases, GALOPA(linear) performs better than GALOPA(cube) because the neural networks may get stuck at local optima resulting in a slight difference in performance, which is a side note to the good performance of GALOPA(linear).

Table 6: Node classification accuracy (%) for GALOPA(cube) and GALOPA(linear).

| Models | Cora | CiteSeer | PubMed | WiKiCS | Amz-Comp. | Amz-Photo | Coauthor-CS |
|---|---|---|---|---|---|---|---|
| GALOPA(cube) | $84.21 \pm 0.30$ | $74.34 \pm 0.18$ | $84.57 \pm 0.34$ | $81.23 \pm 0.19$ | $88.65 \pm 0.11$ | $92.77 \pm 0.40$ | $93.04 \pm 0.25$ |
| GALOPA(linear) | $82.73 \pm 0.29$ | $72.12 \pm 0.35$ | $84.39 \pm 0.19$ | $81.15 \pm 0.39$ | $88.49 \pm 0.17$ | $92.82 \pm 0.27$ | $92.76 \pm 0.22$ |

Table 7: Graph classification accuracy (%) for GALOPA(cube) and GALOPA(linear).

| Models | PROTEINS | DD | MUTAG | NCI1 | COLLAB | IMDB-B |
|---|---|---|---|---|---|---|
| GALOPA(cube) | $76.93 \pm 0.18$ | $83.87 \pm 0.42$ | $91.11 \pm 1.27$ | $77.86 \pm 0.36$ | $73.20 \pm 0.37$ | $70.72 \pm 0.48$ |
| GALOPA(linear) | $76.77 \pm 0.32$ | $82.39 \pm 0.45$ | $90.88 \pm 1.29$ | $76.59 \pm 0.24$ | $73.33 \pm 0.41$ | $70.71 \pm 0.39$ |

Additionally, we count the average elapsed time per epoch for training these two models on all datasets. The results are shown in the table below. These tables underscore the substantial reduction in time consumption associated with GALOPA(linear) compared to GALOPA(cube), especially evident in medium to large datasets such as PubMed, Amz-Comp., DD, etc.

We record in the following table the average elapsed time per epoch taken to pre-train on the ogbn-arxiv with the algorithm GALOPA(cube), the variant algorithm GALOPA(linear), and the baseline BGRL (with linear complexity), and the fine-tuning accuracies obtained on the supervised algorithms. Note that for GALOPA(cube) and GALOPA(linear) we first partition the graph and compute the plan between subgraphs, where the average size of each subgraph is $\sim$5000 nodes. The results of our

Table 8: The average elapsed time per epoch of the models on Node classification datasets.

| Models | Cora | CiteSeer | PubMed | WiKiCS | Amz-Comp. | Amz-Photo | Coauthor-CS |
|---|---|---|---|---|---|---|---|
| GALOPA(cube) | 1.53s | 2.18s | 74.58s | 25.66s | 34.73s | 11.60s | 71.17s |
| GALOPA(linear) | 0.29s | 0.80s | 10.20s | 3.14s | 5.24s | 1.66s | 32.03s |

Table 9: The average elapsed time per epoch of the models on Graph classification datasets.

| Models | PROTEINS | DD | MUTAG | NCI1 | COLLAB | IMDB-B |
|---|---|---|---|---|---|---|
| GALOPA(cube) | 7.05s | 300.50s | 1.21s | 18.15s | 74.64s | 3.59s |
| GALOPA(linear) | 3.36s | 20.07s | 0.47s | 11.75s | 21.46s | 2.76s |

experiments on the ogbn-arxiv dataset, presented in Table 10, showcase the substantial reduction in running time achieved through the implementation of the complexity reduction approach. Importantly, this efficiency enhancement is coupled with comparable performance to the original model.

Table 10: The average elapsed time per epoch and the fine-tuning accuracies on ogbn-arxiv.

| Models | Time | Test Accuracy |
|---|---|---|
| GALOPA(cube) | 60.4s | $0.544 \pm 0.18$ |
| GALOPA(linear) | 2.79s | $0.541 \pm 0.29$ |
| BGRL | 1.02s | $0.535 \pm 0.19$ |

## G Ablation Study

In this section we perform loss ablation experiments. We compare the performance of the algorithms when using Equation (9) and Equation (10) alone as losses below

Table 11: Ablation study on the losses $\mathcal{L}_{\text{match}}$ and $\mathcal{L}_{\text{(im)strc}}$.

| Loss | Cora | CiteSeer | PROTEINS | MUTAG |
|---|---|---|---|---|
| $\mathcal{L}_{\text{match}} + \mathcal{L}_{\text{(im)strc}}$ | $0.842 \pm 0.30$ | $0.743 \pm 0.18$ | $0.769 \pm 0.18$ | $0.911 \pm 1.27$ |
| $\mathcal{L}_{\text{match}}$ | $0.820 \pm 0.36$ | $0.689 \pm 0.34$ | $0.758 \pm 0.21$ | $0.879 \pm 1.14$ |
| $\mathcal{L}_{\text{(im)strc}}$ | $0.812 \pm 0.25$ | $0.684 \pm 0.24$ | $0.762 \pm 0.23$ | $0.869 \pm 1.21$ |

From Table 11 we can see that using only one loss alone leads to performance degradation, which verifies that each loss is indispensable. In addition we find that the plan matching loss $\mathcal{L}_{\text{match}}$ gives relatively better performance on most datasets compared to the implicit structure loss $\mathcal{L}_{\text{(im)strc}}$, which also suggests that the former may contribute more.

## H Sensitivity Analysis

For the selection of $\rho$ and $\sigma$, we search the optimal configuration for them from the set $\{10^{-3}, 10^{-2}, \ldots, 10^{2}, 10^{3}\}$ and $\{0, 0.1, 0.2, \ldots, 0.9, 1\}$. In this section we conducted sensitivity analysis on these two parameters. Table 12 shows the average node classification accuracy on the data Cora for different values of the parameter $\rho$ (vertical axis) and the parameter $\sigma$ (horizontal axis).

From the table, we can see that when we remove the implicit structure constraint $\mathcal{L}_{\text{(im)strc}}$ ($\rho = 0$), the performance of GALOPA drops dramatically if we do not use the explicit edge structure ($\sigma = 1$) at the same time. Whereas, if we use the edge structure ($\sigma < 1$) to a greater extent, i.e., the smaller the $\sigma$, the better performance of the algorithm. Additionally, we discuss the case where only the node attributes are considered without using explicit edge structure ($\sigma = 1$). In this case, if we add implicit structure constraints ($\rho \neq 0$) we can get superior performance. Combining these two cases, it can be concluded that implicit structural constraint $\mathcal{L}_{\text{(im)strc}}$ do capture the internal structure of the graph.

Table 12: The sensitivity analysis of the method to the hyperparameters $\sigma$ and $\rho$.

| $\rho$ vs. $\sigma$ | **0** | **0.3** | **0.5** | **0.8** | **1** |
|---|---|---|---|---|---|
| **0** | $0.813 \pm 0.35$ | $0.809 \pm 0.26$ | $0.801 \pm 0.38$ | $0.784 \pm 0.22$ | $0.776 \pm 0.30$ |
| $\mathbf{10^{-3}}$ | $0.816 \pm 0.21$ | $0.823 \pm 0.18$ | $0.834 \pm 0.31$ | $0.836 \pm 0.37$ | $0.838 \pm 0.30$ |
| $\mathbf{10^{-2}}$ | $0.814 \pm 0.27$ | $0.828 \pm 0.45$ | $0.831 \pm 0.36$ | $0.840 \pm 0.32$ | $0.838 \pm 0.21$ |
| $\mathbf{10^{-1}}$ | $0.818 \pm 0.34$ | $0.830 \pm 0.31$ | $0.829 \pm 0.31$ | $0.839 \pm 0.43$ | $0.842 \pm 0.34$ |
| $\mathbf{10^{0}}$ | $0.823 \pm 0.24$ | $0.834 \pm 0.26$ | $0.835 \pm 0.15$ | $0.838 \pm 0.20$ | $0.841 \pm 0.27$ |
| $\mathbf{10^{1}}$ | $0.819 \pm 0.29$ | $0.838 \pm 0.40$ | $0.834 \pm 0.38$ | $0.833 \pm 0.27$ | $0.832 \pm 0.38$ |
| $\mathbf{10^{2}}$ | $0.821 \pm 0.18$ | $0.826 \pm 0.29$ | $0.829 \pm 0.34$ | $0.841 \pm 0.14$ | $0.839 \pm 0.38$ |
| $\mathbf{10^{3}}$ | $0.820 \pm 0.33$ | $0.824 \pm 0.36$ | $0.833 \pm 0.26$ | $0.832 \pm 0.28$ | $0.841 \pm 0.31$ |

Furthermore, we find that the algorithm is robust to the parameters $\rho$ and $\sigma$, which fluctuate slightly for different values ($>0$).

# I GALOPA on Heterophilic Graphs

To better appreciate our work, in this section, we encapsulate several crucial aspects, which are further elucidated through empirical evaluations of the heterophilic graphs.

(i) *Universality of* GALOPA: Our proposed GALOPA framework possesses the flexibility to be employed on both homophilic and heterophilic graphs. To adapt GALOPA for heterophilic graphs, we only need to replace the current backbone encoder with a suitable one tailored for heterophilic graph scenarios.

According to [76], the heterophily restricts the learning ability of existing homophilic GNNs on general graph-structural data, resulting in significant performance degradation on heterophilic graphs. GALOPA is a flexible OT-based self-supervised framework. The choice of a backbone for graph encoding in GALOPA is not rigidly tied to the proposed framework. This flexibility empowers users to select different backbones based on the specific context. For instance, transitioning GALOPA from homophilic to heterophilic graph settings involves a straightforward substitution of the current homophilic encoder with a suitable heterophilic encoder, serving as the backbone for GALOPA.

To demonstrate this, we conducted a new set of experiments on 4 heterophilic graph data, i.e., Chameleon, Wisconsin, Cornell, and Squirrel. In these experiments, we compared GALOPA against state-of-the-art homophilic graph methods (BGRL) and heterophilic graph methods (SP-GCL [62]). For both BGRL and GALOPA, we assessed two scenarios by employing both traditional GNNs encoders (HoGNN) used in the paper as well as specialized heterophilic GNNs encoders (HeGNN) based on the structure proposed in [32]. We also retained SP-GCL's original encoder design as it is specifically tailored for heterophilic graphs. The results are presented in Table 13.

Table 13: Node classification accuracy on four heterophilic datasets.

| **Alg.** | **Wisconsin** | **Cornell** | **Squirrel** | **Chameleon** |
|---|---|---|---|---|
| BGRL(HoGNN) | $0.523 \pm 0.27$ | $0.561 \pm 0.34$ | $0.462 \pm 0.31$ | $0.634 \pm 0.51$ |
| BGRL(HeGNN) | $0.685 \pm 0.22$ | $0.579 \pm 0.29$ | $0.468 \pm 0.36$ | $0.636 \pm 0.45$ |
| SP-GCL | $0.635 \pm 0.18$ | $0.586 \pm 0.33$ | $0.522 \pm 0.47$ | $0.653 \pm 0.36$ |
| GALOPA(HoGNN) | $0.627 \pm 0.24$ | $0.577 \pm 0.25$ | $0.428 \pm 0.39$ | $0.598 \pm 0.42$ |
| GALOPA(HeGNN) | $0.731 \pm 0.26$ | $0.682 \pm 0.31$ | $0.473 \pm 0.28$ | $0.654 \pm 0.39$ |

The results demonstrate clear performance enhancements in GALOPA when transitioning the backbone from HoGNN to HeGNN across all heterophilic data. Notably, instances like the Wisconsin data exhibit a notable 16.6% enhancement, while Chameleon showcases a 9.4% uplift. Importantly, GALOPA consistently surpasses BGRL in performance. Additionally, in comparison to SP-GCL, a leading heterophilic graph solution, GALOPA outperforms it on three out of four datasets. This robust performance reinforces the efficacy of GALOPA on heterophilic graphs.

(ii) *Versatile performance across graph types*: In this part, we evaluate the performance of GALOPA across both homophilic and heterophilic graph data, utilizing a single adaptable backbone. We demonstrate that if a graph encoder performs effectively on both homophilic and heterophilic graphs,

the same holds true for GALOPA when utilizing this encoder as its backbone. The adaptability of HeGNN in encoding both homophilic and heterophilic graphs is evident [32]. To verify this, we evaluate the performance of GALOPA(HeGNN) on three homophilic graph data in Table 14.

Table 14: Node classification accuracy on homophilic graph datasets.

| Alg. | Cora | CiteSeer | PubMed |
|---|---|---|---|
| MVGRL | $0.834 \pm 0.68$ | $0.732 \pm 0.48$ | $0.800 \pm 0.62$ |
| BGRL | $0.813 \pm 0.54$ | $0.720 \pm 0.63$ | $0.805 \pm 0.30$ |
| GALOPA(HoGNN) | $0.842 \pm 0.30$ | $0.743 \pm 0.18$ | $0.845 \pm 0.34$ |
| GALOPA(HeGNN) | $0.839 \pm 0.21$ | $0.745 \pm 0.34$ | $0.836 \pm 0.27$ |

The findings illustrate that GALOPA(HeGNN) exhibits comparable performance to GALOPA(HoGNN) on homophilic graphs while outperforming baselines. This can largely be attributed to its capacity to adeptly utilize the low-pass, high-pass, and identity channels within GNNs, effectively addressing the variations in both homophilic and heterophilic scenarios. These results further affirm GALOPA's capability to achieve strong performance across distinct graph types by utilizing a unified backbone.

## J GALOPA with Different Backbones

In this section, we evaluate the stability of the proposed GALOPA across various GNNs backbones. The final results show that GALOPA exhibits consistent stability when employing different similar GNNs as its backbone. We conducted an examination of the performance impact on GALOPA by employing diverse GNNs, specifically GCN (as used in the original paper) and SGC (with 1- or 2-hops, denoted as SGC-1 and SGC-2) [65] as encoders. The GCN structure employs a 2-layer design, while the SGC structure utilizes a 1-layer configuration by default. The hidden layer dimension for both models is set to 512. The experimental results are shown in Table 15. The results obtained from these experiments highlight the stability of GALOPA's performance when different GNNs are employed as its backbone. This consistency across diverse GNN architectures underscores the robustness and versatility of our proposed approach.

Table 15: Node classification accuracy of GALOPA and baselines on homophilic graph datasets.

| Alg. | Cora | CiteSeer | PubMed |
|---|---|---|---|
| MVGRL | $0.834 \pm 0.68$ | $0.732 \pm 0.48$ | $0.800 \pm 0.62$ |
| BGRL | $0.813 \pm 0.54$ | $0.720 \pm 0.63$ | $0.805 \pm 0.30$ |
| GALOPA(GCN) | $0.842 \pm 0.30$ | $0.743 \pm 0.18$ | $0.845 \pm 0.34$ |
| GALOPA(SGC-2) | $0.831 \pm 0.24$ | $0.732 \pm 0.38$ | $0.851 \pm 0.41$ |
| GALOPA(SGC-1) | $0.822 \pm 0.36$ | $0.735 \pm 0.34$ | $0.840 \pm 0.18$ |

## K Wilcoxon Signed Rank Test

We performed the Wilcoxon signed rank test on GALOPA and baseline on the node classification dataset and the graph classification dataset, respectively. Tables 16 and 17 report the $p$-values for the Wilcoxon signed-ranks test for GALOPA at 0.05 significance level with node classification baselines and graph classification baselines, respectively. If the $p$-value is small, it can reject the idea that the difference is due to chance and conclude that the population has a median distinct from the performance of the baseline model. As shown in the table, GALOPA achieves superior performance against all the baselines.

Table 16: The $p$-values for the Wilcoxon test for GALOPA on node dataset at 0.05 significance level.

| BGRL | GCA | GRACE | MVGRL | DGI | VGAE | GAE | NODE2VEC | DEEPWALK |
|---|---|---|---|---|---|---|---|---|
| 0.015 | 0.007 | 0.007 | 0.007 | 0.007 | 0.007 | 0.007 | 0.007 | 0.007 |

Table 17: The $p$-values for the Wilcoxon test for GALOPA on graph data at 0.05 significance level.

| SIMGRACE | RGCL | JOAOV2 | AD-GCL | GRAPHCL | INFOGRAPH | GRAPH2VEC | SUB2VEC | DGK | WL | GK |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.078 | 0.078 | 0.046 | 0.015 | 0.078 | 0.078 | 0.031 | 0.015 | 0.031 | 0.078 | 0.015 |

## L Statistics of Datasets

For evaluation purposes, we choose 7 node classification benchmark datasets: CORA, CITESEER, PUBMED [25] and Wiki-CS, Amazon-Computers, Amazon-Photo, and Coauthor-CS [47]. Additionally, we select 6 public graph classification benchmark datasets from TUDataset [36]: NCI1, PROTEINS, DD, MUTAG, COLLAB, and IMDB-B. The statistics for these datasets can be found in Tables 18 and 19.

Table 18: The statistical information of node classification datasets.

| Dataset | Nodes | Edges | Classes | Feat. |
|---|---|---|---|---|
| Cora | 2708 | 10556 | 7 | 1433 |
| CiteSeer | 3327 | 9228 | 6 | 3703 |
| PubMed | 19717 | 88651 | 3 | 500 |
| WikiCS | 11701 | 216123 | 10 | 300 |
| Coauthor-CS | 18333 | 327576 | 15 | 6805 |
| Amz-Comp. | 13752 | 574418 | 10 | 767 |
| Amz-Photo | 7650 | 287326 | 8 | 745 |

Table 19: The statistical information of graph classification datasets.

| Dataset | Graphs | Avg. Nodes | Avg. Edges | Classes |
|---|---|---|---|---|
| PROTEINS | 1113 | 39.06 | 72.82 | 2 |
| DD | 1178 | 284.32 | 715.66 | 2 |
| MUTAG | 188 | 17.93 | 19.79 | 2 |
| NCI1 | 4110 | 29.87 | 32.30 | 2 |
| COLLAB | 5000 | 74.49 | 2457.78 | 3 |
| IMDB-B | 1000 | 19.77 | 96.53 | 2 |