A APPENDIX

A.1 Graph Learning Tasks

Here we define three levels of graph learning tasks.

(1) Node-level Task. In a graph $G = (V, E, X)$, given a node $v_i \in V$ with label y_i, the node-level task aims to learn a model $\mathcal{F}_{node}(\cdot)$ satisfying $\mathcal{F}_{node}(v_i, G) = y_i$. The node-level dataset is $D_{node} = \{\cdots, ((v_i, G), y_i), \cdots\}$.

(2) Edge-level Task. In a graph $G = (V, E, X)$, given two nodes $v_i, v_j \in V$ with a label $y_{i,j}$, the edge-level task aims to learn a model $\mathcal{F}_{edge}(\cdot)$ satisfying $\mathcal{F}_{edge}(v_i, v_j, G) = y_{i,j}$. The edge-level dataset is $D_{edge} = \{\cdots, ((v_i, v_j, G), y_{i,j}, \cdots\}$.

(3) Graph-level Task. For a graph $G = (V, E, X)$ with a label y, the graph-level task aims to learn a model $\mathcal{F}_{graph}(\cdot)$ satisfying $\mathcal{F}_{graph}(G) = y$. The graph-level dataset is $D_{graph} = \{\cdots, ((G_i), y_i), \cdots\}$.

A.2 Graph Inducing

To align three basic graph learning tasks, we follow the existing work [27] to reformulate node-level and edge-level tasks as graph-level tasks, respectively, by transforming their inputs into induced graphs, a κ-ego network. For node-level tasks, an induced graph includes a node and its neighbors within κ hops. For edge-level tasks, an induced graph includes two nodes and their neighbors within κ hops.

In our experiments, we follow [27] construct datasets. For edge-level tasks, we specifically choose samples where the endpoints have identical labels, setting the label of the edge-level sample to match that of its endpoints. For the graph-level tasks, we select the majority label of the subgraph nodes as the graph-level sample's label.

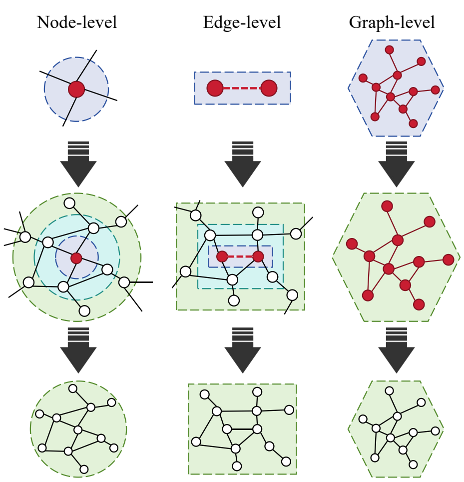

Figure 3: Graph inducing for three levels of tasks.

A.3 Differential Privacy

Here we define the Differential Privacy (DP) [36] for information privatization.

Definition 5. ***Differential Privacy.*** *A mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ that maps domain $\mathcal{M}$ to range $\mathcal{R}$ meets (ε, δ)-differential privacy if it satisfies:*

$$\Pr[\mathcal{M}(d) \in S] \le e^{\varepsilon} \Pr\left[\mathcal{M}\left(d'\right) \in S\right] + \delta \tag{10}$$

where $d, d' \in \mathcal{D}$ are two adjacent inputs, $S \subseteq \mathcal{R}$ are any subset of the mechanism's outputs.

A.4 Training Algorithm

Here we take the node classification task as an example to illustrate the training pipeline in Algorithm 1. We set the GNN as a pre-trained and learnable model, which will be optimized in the server after aggregation.

Algorithm 1: FedGPL training workflow.

Input: A server S with a GNN $\mathcal{G}$, and N clients with $f_P(\cdot|\cdot)$'s parameters θ_p and $f_h(\cdot)$'s parameters θ_h.
Output: Optimized prompt parameters and head parameters for each client.

Initialize each client's parameters $\theta_p^{(0)}$ and $\theta_h^{(0)}$;
for *n-th training round not converge* **do**
  foreach *i-th client C_i in parallel* **do**
    C_i prompts all the k-th input graphs $\{G_{i,k}\}$ by $\{\tilde{G}_{i,k}\} = \{f_P(G_{i,k}|\hat{G}_i)\}$;
    C_i sends differentially privatized $\{\tilde{G}_{i,k}\}$ to S;
    S computes representations of $\{\tilde{G}_{i,k}\}$ by $\{h_{i,k}\} = \{\mathcal{G}(\tilde{G}_{i,k})\}$;
    S sends $\{h_{i,k}\}$ to C_i;
    C_i computes estimated labels $\{\hat{y}_{i,k}\} = \{f_h(h_{i,k})\}$;
    C_i calculates loss values and enables backward propagation;
    C_i calculates gradients of $\theta_p^{(n-1)}$ and $\theta_h^{(n-1)}$;
    C_i optimizes $\theta_p^{(n)}$ and $\theta_h^{(n)}$;
  end
  S calculates $\{\tau_{i \leftarrow j}^{(n+1)} | i, j \in [1, N]\}$ according to $\theta_i^{(n)} = (\theta_{p,i}^{(n)}, \theta_{h,i}^{(n)})$ as Equation 5 and 6;
  S optimizes parameters of $\mathcal{G}$;
  S sends $\{\tau_{i \leftarrow j}^{(n)} | j \in [1, N]\}$ to C_i for $i \in [1, N]$;
  foreach *i-th client C_i in parallel* **do**
    C_i updates parameters $\theta_p^{(n)}$ and $\theta_h^{(n)}$ as Equation 7;
  end
end

A.5 Proof of Theorem 1

Proof. Basically, we suppose that FedGPL consists of two participants as a-th and b-th clients. Their local parameters are $\theta_a^{(l)}$ and $\theta_b^{(l)}$ at l-th step. We can estimate the optimal learning directions by local data as $\theta_a^{(l)\prime}$ and $\theta_b^{(l)\prime}$. ACCording to Definition 3 we can calculate pairwise transferability as $\tau_{a \leftarrow a}, \tau_{a \leftarrow b}, \tau_{b \leftarrow a}, \tau_{b \leftarrow b}$. Moreover, based on Equations 5 and 6, we can calculate their parameters after federated aggregation as

$$\begin{aligned} \theta_a^{(l+1)} &= \frac{\tau_{a\leftarrow a}}{\tau_{a\leftarrow a}+\tau_{a\leftarrow b}}\theta_a^{(l)} + \frac{\tau_{a\leftarrow b}}{\tau_{a\leftarrow a}+\tau_{a\leftarrow b}}\theta_b^{(l)}, \\ \theta_b^{(l+1)} &= \frac{\tau_{b\leftarrow a}}{\tau_{b\leftarrow a}+\tau_{b\leftarrow b}}\theta_a^{(l)} + \frac{\tau_{b\leftarrow b}}{\tau_{b\leftarrow a}+\tau_{b\leftarrow b}}\theta_b^{(l)}. \end{aligned} \tag{11}$$

Next, we assume the learning direction is consistent between two steps, thus we can estimate their optimal learning direction at $(l+1)$-th step as

$$\begin{aligned} \theta_a^{(l+1)\prime} &= \theta_a^{(l+1)} + \overrightarrow{\theta_a^{(l)\prime} - \theta_a^{(l)}}, \\ \theta_b^{(l+1)\prime} &= \theta_b^{(l+1)} + \overrightarrow{\theta_b^{(l)\prime} - \theta_b^{(l)}}. \end{aligned} \tag{12}$$

The 2-norm parameter difference between two tasks with our proposed aggregation algorithm can be calculated as

$$\begin{aligned} \|\theta_a^{(l+1)\prime} - \theta_b^{(l+1)\prime}\|_2 = & \|(\tfrac{\omega_a}{1+\omega_a} + \tfrac{\omega_b}{1+\omega_b})(\|\overrightarrow{\theta_b^{(l)} - \theta_a^{(l)}}\|) \\ & +(\theta_a^{(l+1)} - \theta_b^{(l+1)})\|_2, \end{aligned} \tag{13}$$

where $\omega_a = \frac{\overrightarrow{\theta_a^{(l)\prime}-\theta_a^{(l)}}\cdot\overrightarrow{\theta_b^{(l)\prime}-\theta_a^{(l)}}}{\|\overrightarrow{\theta_a^{(l)\prime}-\theta_a^{(l)}}\|_2}$ and $\omega_b = \frac{\overrightarrow{\theta_b^{(l)\prime}-\theta_b^{(l)}}\cdot\overrightarrow{\theta_a^{(l)\prime}-\theta_b^{(l)}}}{\|\overrightarrow{\theta_b^{(l)\prime}-\theta_b^{(l)}}\|_2}$. Obviously, if $\Delta_T^{a,b}(\theta_a^{(l+1)\prime}, \theta_b^{(l+1)\prime}) \leq \Delta_T^{a,b}(\theta_a^{(l)\prime}, \theta_b^{(l)\prime})$ that the task heterogeneity is minor after applying HiDTA, we have

$$\begin{aligned} \|\theta_a^{(l+1)\prime} - \theta_b^{(l+1)\prime}\|_2 &\leq \|\theta_a^{(l)\prime} - \theta_b^{(l)\prime}\|_2, \\ (\tfrac{\omega_a}{1+\omega_a} + \tfrac{\omega_b}{1+\omega_b})(\|\overrightarrow{\theta_b^{(l)} - \theta_a^{(l)}}\|) &\leq 0, \\ \tfrac{\omega_a}{1+\omega_a} + \tfrac{\omega_b}{1+\omega_b} &\leq 0, \\ \tfrac{1}{\omega_a} + \tfrac{1}{\omega_b} + 2 &\geq 0, \\ \frac{\|\overrightarrow{\theta_a^{(l)\prime}-\theta_a^{(l)}}\|}{\tau_{a\leftarrow b}} + \frac{\|\overrightarrow{\theta_b^{(l)\prime}-\theta_b^{(l)}}\|}{\tau_{b\leftarrow a}} + 2 &\geq 0 \end{aligned} \tag{14}$$

where $\|\overrightarrow{\theta_b^{(l)} - \theta_a^{(l)}}\| \geq 0$. Therefore, when the pairwise transferability values between two clients $\tau_{a\leftarrow b}, \tau_{b\leftarrow a}$ are positive, we have $\Delta_T^{a,b}(\theta_a^{(l+1)\prime}, \theta_b^{(l+1)\prime}) \leq \Delta_T^{a,b}(\theta_a^{(l)\prime}, \theta_b^{(l)\prime})$, which means that the task heterogeneity decrease after aggregation. □

A.6 Proof of Theorem 2

PROOF. Given a graph $G(V, E, X)$, we assume that a graph representation computed by a GNN $\mathcal{G}$ follows $h_G \sim U[0, \eta]$. And the GNN is smooth as $h_G = \mathbb{E}_{E\to\emptyset}[\mathcal{G}(V, E, X)]$. The representation of prompted graph $\tilde{G}$ by a VPG $\hat{G} = ((\hat{V}^+, \hat{V}^-), (\hat{E}^+, \hat{E}^-), (\hat{X}^+, \hat{X}^-))$ can estimated as

$$h_{\tilde{G}} = \frac{|V| \cdot h_G + |\hat{V}^+| \cdot h_{G^+} - |\hat{V}^-| \cdot h_G}{|V| + |\hat{V}^+| - |\hat{V}^-|}, \tag{15}$$

where $h_{G^+} = \mathcal{G}(\hat{V}^+, \hat{E}^+, \hat{X}^+)$. Next, we assume $\hat{X}^+ \sim X$, $|\hat{V}^+| << |V|$, and $|\hat{V}^+| << |\hat{V}^-|$ in practice, thus we can get the distribution of prompted graphs as

$$h_{\tilde{G}} \sim \tilde{U}[\eta\alpha_n, \eta], \tag{16}$$

where $\alpha_n \in [0, 1]$ is a pre-defined percentage parameter of significance score borderline in VPG.

To measure the data heterogeneity between i-th and j-th clients, we denote their graph data following uniform distributions of $h_{G_a} \sim U^a[0, \eta^a]$ and $h_{G_b} \sim U^b[0, \eta^b]$, respectively. We can infer $h_{\tilde{G}_a} \sim \tilde{U}^a[\eta^a\alpha_n^a, \eta^a]$ and $h_{\tilde{G}_b} \sim \tilde{U}^b[\eta^b\alpha_n^b, \eta^b]$. Then, the expectation of the 2-norm embedding difference with and without graph prompting $\tilde{\delta}$ can be calculated by

$$\begin{aligned} \mathbb{E}[\tilde{\delta}] = & \mathbb{E}_{\begin{cases} h_{\tilde{G}_a}\sim\tilde{U}^a \\ h_{\tilde{G}_b}\sim\tilde{U}^b \end{cases}}[\|h_{\tilde{G}_a} - h_{\tilde{G}_b}\|_2] - \mathbb{E}_{\begin{cases} h_{G_a}\sim U^a \\ h_{G_b}\sim U^b \end{cases}}[\|h_{G_a} - h_{G_b}\|_2] \\ = & ((\tfrac{1-\alpha_n^a}{2}\eta^a)^2 + (\tfrac{1-\alpha_n^b}{2}\eta^b)^2 - 2(\tfrac{1-\alpha_n^a}{2}\eta^a)(\tfrac{1-\alpha_n^b}{2}\eta^b)) \\ & -((\tfrac{1}{2}\eta^a)^2 + (\tfrac{1}{2}\eta^b)^2 - 2(\tfrac{1}{2}\eta^a)(\tfrac{1}{2}\eta^b)) \\ = & \tfrac{(1-\alpha_n^a)^2-1}{4}\eta^{a2} + \tfrac{(1-\alpha_n^b)^2-1}{4}\eta^{b2} - 2(\tfrac{(1-\alpha_n^a)(1-\alpha_n^b)-1}{4}\eta^a\eta^b). \end{aligned} \tag{17}$$

Obviously, when $\alpha_n^a \approx \alpha_n^b$, we ensure

$$\mathbb{E}[\tilde{\delta}] \approx \frac{(1-\alpha_n^a)^2 - 1}{4}(\eta_a - \eta_b)^2 \leq 0, \tag{18}$$

because $(1-\alpha_n^a)^2 - 1 \leq 0$. Subsequently, according to Equation 1 that is monotonically increasing, the data heterogeneity can be reduced after graph prompting as

$$\mathbb{E}_{\begin{cases} h_{\tilde{G}_a}\sim\tilde{U}^a \\ h_{\tilde{G}_b}\sim\tilde{U}^b \end{cases}}[\Delta_D^{i,j}(h_{\tilde{G}_a}, h_{\tilde{G}_b})] \leq \mathbb{E}_{\begin{cases} h_{G_a}\sim U^a \\ h_{G_b}\sim U^b \end{cases}}[\Delta_D^{i,j}(h_{G_a}, h_{G_b})]. \tag{19}$$

□

A.7 Dataset Statistics

Table 6 are the statistics of graph datasets, including Cora, CiteSeer, DBLP, Photo, and Physics, which are transformed from traditional graph learning datasets for node classification.

Table 6: Statistics of datasets.

Dataset	# Nodes	# Edges	# Features	# Labels
Cora	2,708	5,429	1,433	7
CiteSeer	3,327	9,104	3,703	6
DBLP	17,716	105,734	602	6
Photo	7,650	238,162	745	8
Physics	34,493	495,924	8,415	5

A.8 Implementation Details

For the implementation of federated learning, we deploy 9 clients, and each level of tasks (node-level, edge-level, and graph-level) contains 3 clients, each client has 400 induced graphs on average, which are induced from the raw dataset with $\kappa = 5$, and each client participates in every communication round, during each round, clients undergo a single epoch of training, and the global communication round is set to 50. For the implementation of graph prompting methods, for GPF, we add additional feature vectors into the node features, for ProG, we set the number of tokens as 10. Each client contains its training dataset, prompt, and an answering layer that projects the embedding output by the GNN to the final results. The learning rate is set as 0.1 for all datasets. The results are averaged on all clients. The HiFGL framework is implemented based on PyTorch [24], and PyTorch-Lightning [6] runs on the machine with Intel Xeon Gold 6148 @ 2.40GHz, V100 GPU and 64G memory.

Table 7: Overall performance (ACC (%) and F1 (%)) of different federated algorithms and graph prompting methods in few-shot settings.

Federated Method	Prompt Method	Cora ACC	Cora F1	CiteSeer ACC	CiteSeer F1	DBLP ACC	DBLP F1	Photo ACC	Photo F1	Physics ACC	Physics F1
Local	GPF	80.06	79.84	82.34	81.84	78.46	78.15	74.51	73.14	86.15	86.21
	ProG	84.25	84.31	83.19	83.04	79.84	79.73	69.84	69.47	85.89	85.03
	VPG	85.81	85.10	84.52	84.37	82.31	82.17	77.97	76.84	88.03	88.14
FedAvg	GPF	79.81	79.57	82.79	82.87	79.64	78.84	79.03	78.14	86.87	86.65
	ProG	80.44	80.26	84.54	83.81	80.14	80.20	70.12	70.03	86.84	86.67
	VPG	86.14	85.82	85.01	84.77	80.72	80.07	79.31	79.01	88.72	88.12
FedProx	GPF	77.37	76.41	81.04	80.97	78.07	77.91	80.57	80.19	84.58	84.29
	ProG	78.59	78.14	80.08	80.14	77.48	77.97	70.16	70.07	85.17	85.01
	VPG	84.32	84.23	81.24	80.88	78.14	78.45	84.25	84.12	90.05	90.03
SCAFFOLD	GPF	72.94	72.44	72.48	70.84	77.61	77.31	72.15	70.13	82.14	82.23
	ProG	70.67	68.47	60.41	60.87	75.14	75.10	68.72	68.12	83.17	83.04
	VPG	79.74	79.55	75.85	84.76	77.22	77.07	73.17	72.94	86.14	86.07
HiDTA	GPF	80.68	80.81	82.99	82.83	80.75	80.66	84.10	84.07	87.02	87.06
	ProG	80.22	79.28	83.48	83.45	80.67	80.17	69.47	69.38	88.87	88.17
FedGPL		**86.45**	**86.42**	**85.59**	**85.56**	**82.72**	**81.87**	**85.28**	**85.26**	**90.14**	**90.09**

A.9 Overall Performance for Supervised and Prompt Learning

We evaluate the performance by training modules from scratch (*i.e.*, Supervised) and freezing the pre-trained GNN (*i.e.*, Prompt) in Table 8. Beyond fine-tuning, the results of the two training schemes depict that FedGPL behaves more competitively than others on two datasets.

Table 8: ACC (%) and F1 (%) of different federated algorithms and graph prompting methods. Supervised: training models from scratch; Prompt: fine-tuning models based on a frozen pre-trained GraphTransformer.

Training Scheme	Federated Method	Prompt Method	Cora ACC	Cora F1	CiteSeer ACC	CiteSeer F1
Training	FedAvg	GPF	85.67	85.34	84.62	84.2
		ProG	87.96	87.61	85.71	85.87
		SUPT	86.31	86.24	85.98	85.51
	FedProx	GPF	81.34	80.51	83.22	82.71
		ProG	85.61	85.40	84.41	83.85
		SUPT	82.31	82.45	83.49	83.8
	FedGPL		**89.14**	**88.57**	**88.12**	**87.94**
Prompt	FedAvg	GPF	80.50	79.72	83.22	82.71
		ProG	84.02	83.77	84.41	83.85
		SUPT	81.83	80.61	83.49	83.8
	FedProx	GPF	77.81	76.97	82.61	82.33
		ProG	82.54	81.81	82.64	81.67
		SUPT	78.67	78.34	82.45	82.37
	FedGPL		**87.61**	**87.13**	**87.58**	**87.74**

A.10 Overall Performance with Pre-trained GCN

We test the fine-tuned FGL performance based on a pre-trained GCN on Cora and Citeseer datasets in Table 9. FedGPL outperform baseline models in terms of ACC and F1.

Table 9: ACC (%) and F1 (%) of different federated algorithms and graph prompting methods based on a pre-trained GCN [15].

Federated Method	Prompt Method	Cora ACC	Cora F1	CiteSeer ACC	CiteSeer F1
Local	GPF	83.14	82.64	81.63	81.37
	ProG	84.91	84.33	82.21	81.44
	SUPT	82.41	82.01	82.14	82.07
FedAvg	GPF	84.61	84.37	84.24	83.67
	ProG	85.61	85.03	84.81	84.02
	SUPT	83.17	83.34	83.61	83.44
FedProx	GPF	79.47	79.31	82.67	82.33
	ProG	83.44	82.97	83.41	83.01
	SUPT	78.67	78.34	82.76	82.03
SCAFFOLD	GPF	76.61	75.51	72.51	72.03
	ProG	72.61	72.81	73.67	73.11
	SUPT	73.35	72.85	72.61	72.17
FedGPL		**86.87**	**86.27**	**85.34**	**85.22**

Table 10: Comparison of ACC (%) for different graph prompting methods and training schemes. N, E, and G: node-, edge-, and graph-level task, (f): few-shot settings.

Method	Cora			CiteSeer			Physics		
	N	E	G	N	E	G	N	E	G
Supervised	80.72	95.21	95.14	81.00	97.45	88.83	83.07	92.78	99.78
Pre-train	77.06	87.52	88.61	80.68	93.19	72.25	75.44	78.94	95.36
Fine-tune	81.01	95.36	95.57	81.33	97.27	85.67	83.81	92.47	99.87
GPF	76.17	88.24	87.05	82.11	93.81	74.17	76.51	86.66	98.42
ProG	77.56	89.49	89.48	82.51	95.57	75.94	76.42	87.12	97.45
VPG	79.19	90.35	91.02	82.94	96.48	77.49	83.37	91.03	99.50
GPF(f)	74.35	82.33	85.62	81.02	92.88	73.11	75.27	79.74	96.04
ProG(f)	75.14	87.51	89.57	81.41	93.43	74.75	80.78	78.41	97.47
VPG(f)	77.64	89.46	90.35	81.89	94.82	76.84	82.14	83.78	98.17

A.11 Privacy Analysis

We evaluate the impact of using differential privacy on the performance of FedGPL. Here we conduct experiments by varying the privacy scale of ϵ and test it on the Cora dataset. The results are shown in Figure 4, which demonstrates the relationship between ACC and privatization parameters. We observe that the ACC is reduced as we increase the privacy constraints (decreasing ϵ). Similar conclusions are drawn from different settings. The finding indicates that there is a trade-off between performance and privacy in FedGPL, which is common in FL [44]. In practical utilization, we will choose proper ϵ for different degrees of privacy protection demands on GNN and graph data.

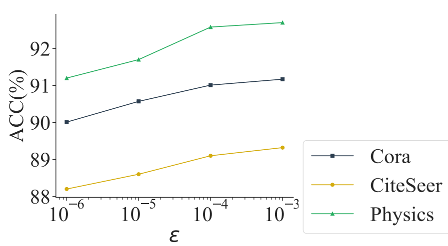

Figure 4: Performance-privacy correlation curves.

A.12 Case Study: Transferability

To further discuss how HiDTA becomes robust against two kinds of heterogeneity, we conduct a case study on transferability according to the aggregating weight matrix, to find out how transferability models the heterogeneous knowledge transfer. Specifically, Figure 5 shows the averaged and normalized transferability values in the last 20 training epochs on CiteSeer. The results suggest that HiDTA prioritizes learning directions with higher transferability. Besides, the optimization process of a target task mainly depends on itself, aligned with our motivations that aim to model asymmetric knowledge transfer.

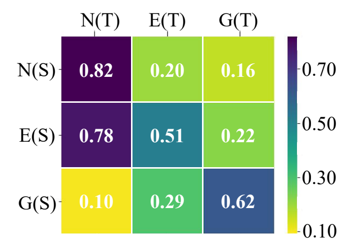

Figure 5: Transferability. N, E, and G: node-, edge-, and graph-level task, S: source task, and T: target task.

A.13 Performance on Few-shot Settings

Besides the standard setting with full training samples, we follow GPL works [27] to deploy a few-shot setting **(f)**. Few-shot settings are usually incorporated in prompt learning, which acts as an efficient learning manner. It lets each client have only a limited number of data samples. In our experiments, we utilize a 100-shot scenario, where each client possesses only 100 labeled samples. The prediction results are shown in Table 7. We observe that FedGPl still outperforms baselines in few-shot settings in a prompt tuning way, which demonstrates that its generalization ability successfully address FGL in a task- and data- heterogeneous scenario.

A.14 Experiments on Graph Prompting

To further evaluate the effectiveness of VPG, the graph prompting module of FedGPL, we compare VPG with GPF and ProG under the local graph prompting setting on Cora, CiteSeer, and Physics datasets, as shown in Table 10. We also compared with (1) supervised training: training a graph model from scratch, (2) fine-tuning: fine-tuning the GNN with labeled data, and (3) pre-training: freezing GNN with only a learnable task head. We observe that VPG performs better than ProG and GPF by 0.43% to 7.28% across all tasks on three datasets. Generally, prompting unleashes more potential of GNN on different downstream tasks with small-scale trainable parameters. In some cases, prompting even surpasses fine-tuning and supervised learning, which may be attributed to a lightweight model for more effective optimization.

Table 11: Comparison of ACC (%) for different graph prompting methods and federated algorithms. N, E, and G: node-, edge-, and graph-level task.

Federated Method	Prompt Method	Cora			CiteSeer			Physics		
		N	E	G	N	E	G	N	E	G
Local	GPF	76.17	88.24	87.05	82.11	93.81	74.17	76.51	86.66	98.42
	ProG	77.56	89.49	89.48	80.74	91.57	73.03	78.09	80.83	98.03
	VPG	79.19	90.35	91.05	82.94	96.48	77.49	83.37	91.03	99.51
FedAvg	GPF	75.87	87.21	86.74	82.44	93.71	75.03	77.94	88.96	98.57
	ProG	77.61	89.14	90.13	82.86	86.42	86.27	79.74	82.84	99.13
	VPG	79.34	90.57	91.26	83.31	96.78	78.21	83.42	91.21	99.34
FedProx	GPF	72.21	82.84	83.11	82.86	87.33	75.86	75.64	87.24	96.48
	ProG	75.84	87.31	85.74	81.03	91.91	73.13	81.87	83.13	98.32
	VPG	76.61	88.16	86.81	81.37	94.47	76.61	83.25	91.12	99.04
SCAFFOLD	GPF	69.28	78.48	79.51	75.41	79.18	69.37	72.57	84.47	94.42
	ProG	58.97	62.41	65.48	61.47	68.81	64.23	73.51	85.24	94.57
	VPG	70.57	77.04	75.52	72.61	84.76	67.11	79.88	87.34	95.21
HiDTA	GPF	74.13	85.91	86.01	81.47	94.51	76.68	79.92	87.87	98.82
	ProG	74.24	86.04	86.11	82.83	95.84	77.74	82.55	88.16	98.72
FedGPL		**82.19**	**92.49**	**92.02**	**83.41**	**96.81**	**78.41**	**83.86**	**91.79**	**99.47**

A.15 Hyperparameter Sensitivity

We also investigate the impact of different α_n, as shown in Figure 6, where we conclude that optimal α_n of tasks are distinct. For example, $\alpha_n = 0.3$ is optimal for the graph-level task, while $\alpha_n = 0.5$ is optimal for the edge-level task. Results further reveal the existence of heterogeneity between different levels of tasks. This heterogeneity indicates personalized prompting is essential and motivates us to design a selective cross-client knowledge sharing method to improve prediction accuracy.

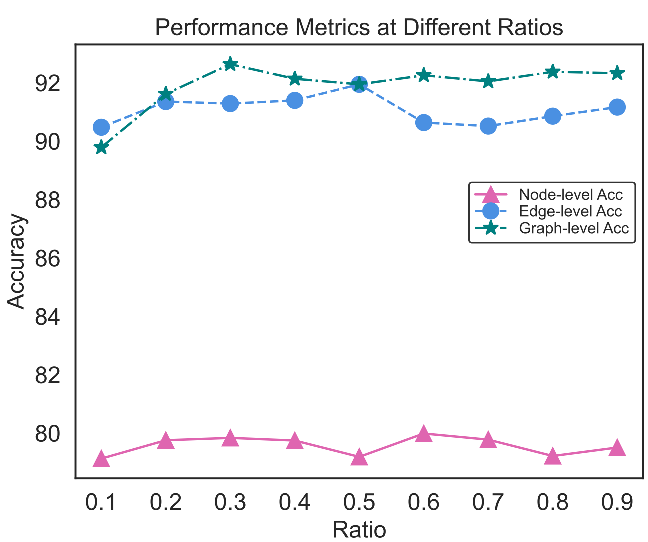

Figure 6: Parameter sensitivity of α_n.

A.16 Task Performance

Table 11 is a detailed comparison of the accuracy of different federated learning algorithms combined with graph prompting methods on three datasets across three types of tasks. From the table, it is evident that FedGPL outperforms other methods in all three tasks. For example, on the Cora dataset, compared to local training, FedGPL shows improvements of 4.02%, 2.14%, and 0.97% in node-level, edge-level, and graph-level tasks, respectively. This demonstrates that FedGPL enhances prompting performance by facilitating knowledge sharing across different tasks. On the other hand, compared to FedAvg, HiDTA exhibits better performance with VPG. For instance, on the Cora dataset, HiDTA shows improvements of 3.00%, 2.14%, and 0.97% in the three tasks compared to FedAvg, indicating that HiDTA better mitigates the heterogeneity between different tasks, enabling more effective knowledge sharing.