On the Influence of Connectivity to CTR Prediction

Zhongyu Ouyang [1] **Chunhui Zhang** [1] **Yaning Jia** [1] **Soroush Vosoughi** [1]

Abstract

Click-through rate (CTR) measures user engagement in recommender systems. Traditional CTR models predict interactions based on individual features of users, items, and the context in between. However, these models often overlook the complex and interconnected nature of user-item interactions, failing to capture valuable connectivity patterns. To address this, we explore the impact of leveraging user-item connectivity patterns on the predictive ability of existing CTR prediction models by incorporating the message passing (MP) mechanism exclusively in the embedding process. To quantify the cost of performance changes, we introduce Marginal Cost of Improvement (MCI), a metric inspired by marginal utility, that measures the utility required for incremental improvements. Our theoretical analysis and extensive experiments reveal three key insights: *First*, the benefits of MP increase proportionally with dataset density; *Second*, fine-tuning well-trained CTR models with MP is more effective than training with MP from scratch; *Third*, measured by MCI, MP-enhanced models allocate greater effort to under-served users with lower original CTR performance. These findings not only demonstrate the effectiveness of MP in improving CTR prediction but also highlight a promising direction for *democratizing* CTR prediction. By prioritizing under-served groups, MP has the potential to mitigate social media polarization and foster a more equitable and inclusive social media environment.

1. Introduction

In modern recommendation systems (Wang et al., 2021; Schafer et al., 1999; Ma et al., 2008; Jamali & Ester, 2010; Fan et al., 2019), click-through rate (CTR) models predict user interaction probability, assisting developers in enhancing user engagement and recommendation satisfaction. Specifically, user and item IDs, along with contextual features (e.g, user demographic information, item descriptions, interaction timestamp) are encoded into latent vectors (embeddings) through an embedding process. These embeddings are then concatenated and passed through into a deep neural network (DNN) that captures both low-order and high-order feature interactions. The model's output is a probability score indicating the likelihood of a user interacting with an item.

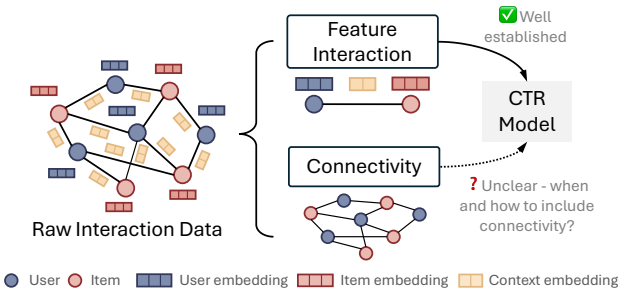

Figure 1. Building on well-established feature interaction modeling, we analyze *(i)* when incorporating user-item connectivity benefits CTR prediction, and *(ii)* how to effectively incorporate it.

Despite being widely adopted in social media recommendation research, current CTR prediction pipelines fail to fully account for the complexity of downstream tasks. Existing models primarily focus on feature interaction modeling, learning user, item, and context representations independently through typical linear or non-linear layers. However, user-item interactions—such as clicks, views, and purchases—can be naturally represented as a social graph that captures complex relationships among users (user-user), items (item-item), and between users and items (user-item). The impact of higher-order connectivity patterns on CTR prediction remains unclear and under-explored. This research gap is illustrated in Figure 1.

We systemically investigate how leveraging connectivity patterns impacts the predictive capabilities of exiting CTR models by incorporating message passing (MP), a mechanism that facilitates representation learning by utilizing the structural and relational context of raw interaction data. *We begin with a theoretical analysis on how MP affects model stability.* Using the Lipschitz constant as the indicator of model stability, we find that incorporating MP into the representation learning effectively decreases the Lipschitz

[1]Dartmouth College, Hanover, US.

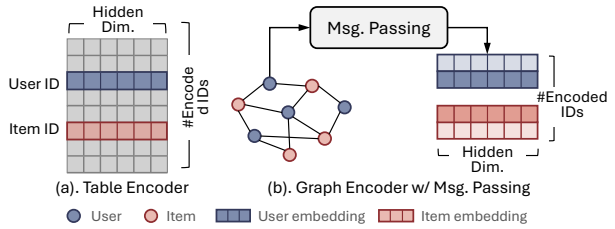

Figure 2. Encoders to encode user and item IDs.

constant of the representation mapping function. This reduction signifies lower model sensitivity to input variations, thereby enhancing the stability of representation learning.

Next, we explore the potential of connectivity patterns to enhance CTR prediction models with a focus on feature interaction modeling. To isolate the impact of connectivity, we integrate MP exclusively into the embedding process while preserving the original feature modeling structure. Our extensive experiments reveal that incorporating MP into the embedding process yields a more stable performance enhancement in dense datasets than in sparse ones. Furthermore, fine-tuning well-trained CTR models with MP proves to be more effective than training with MP from scratch.

Finally, we analyze how CTR models allocate effort across user groups in relation to their performance changes. We introduce Marginal Cost of Improvement (MCI), a metric inspired by marginal utility (Walras, 1900; Menger & Menger, 1923), to quantify the utility required for incremental improvements. Theoretically, MCI satisfies conditions derived from three natural marginal cost comparison cases (detailed in Section 4.4). Empirically, MCI reveals that MP assigns the greatest effort to under-served users with lower initial CTR performance, regardless of their activity levels or demographic profiles. These findings highlight a promising direction for democratizing CTR technology by prioritizing under-served groups and allocating greater effort to improve their experiences. Additionally, MP offers the potential to mitigate social media polarization, fostering a more equitable and inclusive social media ecosystem.

2. Background

Click-through Rate Prediction The problem of click-through rate (CTR) prediction is defined as predicting the interaction likelihood between a user and an item given the user ID, item ID, and optional context features. Formally, we denote IDs of user i and item j as x_i and x_j respectively, and the context features of the interaction in between as $\mathbf{c}_{ij} \in \mathbb{R}^{d^c}$, where d^c refers to the contextual feature dimension. Let the encoder of user/item IDs be $f(\cdot) : \mathbb{R} \to \mathbb{R}^d$, where d refers to the latent dimension of encoded ID embeddings. The traditional encoder $f(\cdot)$ in CTR models is a look-up table encoder, where $f(x_i)$ is the x_i-th entry in a matrix $E \in \mathbb{R}^{(|\mathcal{U}|+|\mathcal{I}|)\times d}$, where $\mathcal{U}$ is the user set and $\mathcal{I}$ is the item set. This table encoder does not utilize connectivity patterns between users and items, and is illustrated in Figure 2 (a). The encoder of contextual features is defined as $h(\cdot) : \mathbb{R}^{d^c} \to \mathbb{R}^{d'}$, where d' is the dimension of encoded contextual embeddings. Depending on the the feature type (i.e., categorical or numerical), the embedding process can be formulated as look-up tables for categorical features or a DNN for numerical features.

To predict the probability that user i interacts with item j, the input $\mathbf{z}_{ij} \in \mathbb{R}^{2d+d'}$ to a CTR model is defined as:

$$\mathbf{z}_i = f(x_i),\ \mathbf{z}_j = f(x_j),\ \mathbf{z}^c_{ij} = h(\mathbf{c}_{ij}), \tag{1}$$

$$\mathbf{z}_{ij} = \left[\mathbf{z}_i \parallel \mathbf{z}_j \parallel \mathbf{z}^c_{ij}\right], \tag{2}$$

where $\parallel$ refers to the concatenation operation, $\mathbf{z}_i$ and $\mathbf{z}_j$ refer to the latent ID embeddings of user i and item j generated by the encoder respectively, and $\mathbf{z}^c_{ij}$ is the encoded contextual features. With the encoded embeddings, each CTR model adopts its corresponding rating function $r(\cdot) : \mathbb{R}^{2d+d'} \to \mathbb{R}$ that models feature interaction to predict the likelihood of interaction. Specifically, let $p_{ij} = r(\mathbf{z}_{ij})$, where $p_{ij} \in [0, 1]$ represents how likely user i would interact with item j. The CTR model is trained with the binary cross entropy loss:

$$\mathcal{L} = -\frac{1}{N} \sum_{\{i,j\}\in T_r} y_{ij} \log(p_{ij}) + (1 - y_{ij}) \log(1 - p_{ij}) + \lambda \sum_l \|\mathbf{w}^{(l)}\|^2, \tag{3}$$

where T_r denotes the training set, y_{ij} refers to the binary label (1 for positive and 0 for negative), λ is the L_2 regularization coefficient, and l is the layer number of the model.

Message Passing As a widely studied framework for message passing (MP), graph neural networks enable the exchange of meaningful representations among nodes in relational data (Kipf & Welling, 2016; Veličković et al., 2017; Hamilton et al., 2017). To formulate the representation exchange in recommendation, let the interaction matrix be $R \in \{0, 1\}^{|\mathcal{U}|\times|\mathcal{I}|}$, where $r_{uv} = 1$ represents an observed valid interaction between user u and item v, and 0 otherwise. To extract collaborative signals, the interaction matrix R is abstracted to a bipartite graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$ is the set of nodes and $\mathcal{E} = \{(u, v)|u \in \mathcal{U}, v \in \mathcal{I}, r_{uv} = 1\}$ is the set of edges. A linear layer that updates node features using MP has the following update rule:

$$\mathbf{h}'_i = \sigma\left(W\left(\mathbf{h}_i + \mathrm{Agg}\left(\{\mathbf{h}_j : j \in \mathcal{N}_i\}\right)\right)\right), \tag{4}$$

where $\mathbf{h}'_i$ is node i's updated embedding from $\mathbf{h}_i$, $\mathcal{N}_i$ is the set of neighbors of node i, W is a shared weight matrix, σ is a Lipschitz continuous activation function with Lipschitz constant L_σ, and $\mathrm{Agg}(\cdot)$ is the averaging function. We depict this embedding process in Figure 2 (b).

Lipschitz Constant of Deep Models The Lipschitz constant is a critical concept for understanding and controlling the stability and robustness of deep models. A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is said to be Lipschitz continuous on an input set $\mathcal{X} \subseteq \mathbb{R}^n$ if there exists a bound $K \geq 0$ such that for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$, f satisfies:

$$\|f(\mathbf{x}) - f(\mathbf{y})\| \leq K \|\mathbf{x} - \mathbf{y}\|, \quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{X}. \tag{5}$$

The smallest possible K in Equation 5 is defined as the Lipschitz constant of f, denoted as $\mathrm{Lip}(f)$:

$$\mathrm{Lip}(f) = \sup_{\mathbf{x},\mathbf{y} \in \mathcal{X}, \mathbf{x} \neq \mathbf{y}} \frac{\|f(\mathbf{x}) - f(\mathbf{y})\|}{\|\mathbf{x} - \mathbf{y}\|}. \tag{6}$$

In this context, f is referred to as a K-Lipschitz function, where the Lipschitz constant K quantifies the maximum change in the function's output caused by a unit-norm perturbation in its input. This constant serves as a crucial indicator of a deep model's stability wrt its inputs: a smaller Lipschitz constant indicates lower sensitivity to input variations.

3. How Connectivity Changes CTR Prediction

We begin by conducting a theoretical analysis that examines how connectivity patterns within user-item interactions reduces the Lipschitz constant of deep models, resulting in smoother outputs at the level of individual nodes wrt input variations. With this analysis, we design an experimental strategy leverages these connectivity patterns in the representation learning while reusing the existing CTR models.

3.1. MP Enhances CTR Model Stability

Theorem 1. *For any node $i \in \mathcal{V}$ and its node feature $\mathbf{h}_i$, consider the mapping function with the updating rule demonstrated in Equation 4. Then, the Lipschitz constant L of the mapping $f : \mathbf{h} \mapsto \mathbf{h}'$, where $\mathbf{h}, \mathbf{h}' \in \mathbb{R}^d$ are the collection of all node features in the graph before and after the mapping respectively, satisfies:*

$$L \leq L_\sigma \|W\|_F \sqrt{1 + \frac{1}{k_i}}, \tag{7}$$

where $\|W\|_F$ is the Frobenius norm of the weight matrix W, k_i denotes the degree of node i, L_σ is the Lipschitz constant of a Lipschitz continuous activation function $\sigma(\cdot)$, and d is the dimension of the node feature.

Insight. Within the MP mechanism, an increased node degree k_i for node i reduces the Lipschitz constant of the mapping function, thereby stabilizing the model output wrt input variations.

Proof Sketch. To bound the Lipschitz constant L of the MP update $f : \mathbf{h} \mapsto \mathbf{h}'$, we analyze how changes in input features $\mathbf{h}$ affect output features $\mathbf{h}'$. The update for node i combines its own features $\mathbf{h}_i$ and an average of its neighbors' features: $\mathbf{h}'_i = \sigma\left(W\left(\mathbf{h}_i + \frac{1}{k_i}\sum_{j \in \mathcal{N}_i} \mathbf{h}_j\right)\right)$, where σ is L_σ-Lipschitz and W has Frobenius norm $\|W\|_F$. The Jacobian $\mathbf{J}$ of this mapping has non-zero blocks only for $\mathbf{h}_i$ and its neighbors $\mathbf{h}_j$, with $\frac{\partial \mathbf{h}'_i}{\partial \mathbf{h}_i^\top} = D_i W$ and $\frac{\partial \mathbf{h}'_i}{\partial \mathbf{h}_j^\top} = D_i W \cdot \frac{1}{k_i}$, where D_i contains activation derivatives bounded by L_σ. The Frobenius norm of $\mathbf{J}$ is $\|\mathbf{J}\|_F \leq L_\sigma \|W\|_F \sqrt{1 + \frac{1}{k_i}}$, and since $L \leq \|\mathbf{J}\|_2 \leq \|\mathbf{J}\|_F$, we obtain $L \leq L_\sigma \|W\|_F \sqrt{1 + \frac{1}{k_i}}$. This shows that L decreases as the neighborhood size k_i grows, stabilizing the model. For full proof, please refer to Appendix D.

3.2. Integrating MP into CTR models

The user-item bipartite graph captures rich topological relationships such as co-purchase or shared interests. To investigate whether this topological information alone is beneficial for CTR prediction, we replace the traditional table encoder for ID encoding with a graph encoder to straightforwardly integrate connectivity patterns into existing CTR models, while preserving the feature interaction modeling structures. Unlike a table encoder which independently encodes the ID information for users and items, a graph encoder additionally leverages the topological relationships in the graph to generate user and item embeddings.

We adopt LightGCN (He et al., 2020), a well-studied linear MP mechanism, as the replacing graph encoder of exiting CTR methods. In LightGCN, Equation 4 is simplified by removing both the non-linearity $\sigma(\cdot)$ and the weight matrix W, focusing solely on linear aggregation of neighbor information. This minimalistic design streamlines the MP process, and has been proven effective in recommendation tasks. Specifically, let $f_g(\cdot, \cdot) : \mathcal{G} \times x \to \mathbb{R}^d$ be the graph encoder. For user i and item j, $f_g(\cdot, \cdot)$ conducts MP in each layer to propagate and aggregate information from the neighborhood. The graph-encoded embedding for node i (user or item) is obtained with the following:

$$\mathbf{z}_i = f_g(\mathcal{G}, x_i) = \sum_{l=0}^{L} a_l \mathbf{z}_i^{(l)}, \tag{8}$$

$$\text{where} \quad \mathbf{z}_i^{(l)} = \sum_{v \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_j|}} \mathbf{z}_j^{(l-1)} \quad \text{and} \quad \mathbf{z}_j^{(0)} = f(x_j). \tag{9}$$

In Equation 9, $\mathbf{z}_i^{(l)}$ is the embedding for node i in layer l, $\mathcal{N}_i$ is the set of neighbors for node i in $\mathcal{G}$, and a_l is the readout coefficient for each layer-l's embeddings. With the obtained graph-based user and item ID embeddings, we construct the input features following Equation 2, and fed them into any CTR method (e.g., DCN) to predict the interaction probabilities. We depict this encoding process in Figure 3.

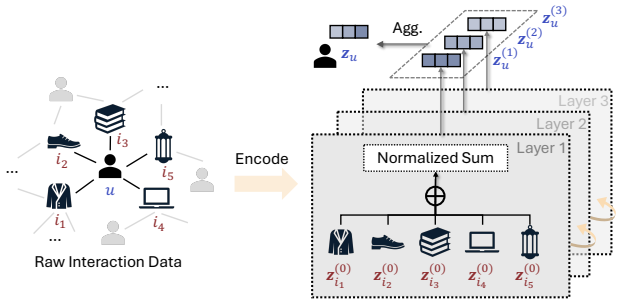

Figure 3. The message passing mechanism for recommendation.

4. Experiments

We investigate MP's influence to the CTR models from the following perspectives: *(i) Dataset Density*: How does dataset density affect the effectiveness of MP? *(ii) Training Paradigm*: When incorporating MP, should we train from scratch or adapt existing well-trained models? *(iii) User Activity*: How does MP affect the performance across users with varying activity levels? *(iv) Supervision Reliability*: Do samples with weak/noisy supervisions affect those with strong/reliable supervisions?

4.1. Setup

Datasets We select two publicly representative recommendation benchmark datasets with distinct *global* dataset density to conduct our experiments: *(1)* **MovieLens-1M**[1] dataset is a dense dataset sourced from the MovieLens website, containing 1 million ratings from 6,000 users on 4,000 movies; *(2)* **Yelp2022**[2] dataset is a sparse dataset which includes a large collection of user reviews, item information, and user-item interactions from the Yelp platform.

For each dataset, we rely on k-core filtering (He & McAuley, 2016), which retains users and items with at least k interactions, to vary the *local* dataset density. The higher k, the higher local dataset density. To analyze supervision reliability, we denote RM as *retaining middle-rated samples*, where RM=✓ represents retaining samples with middle ratings (i.e., samples with weak/noisy supervisions), and RM=✗ represents otherwise. We demonstrate the meta data of the curated datasets utilized in the experiments in Table 1.

For all datasets, we convert user-to-item ratings into binary labels through thresholding. We randomly split the datasets with a ratio of 0.8/0.1/0.1 for training, validation, and testing, respectively. Each experimental setting is repeated under five random seeds and the averaged performance is reported.

Baselines We select six state-of-the-art CTR baseline models that emphasize advanced feature interaction modeling, spanning a variety of approaches such as deep cross networks (Wang et al., 2021; 2023), attention mechanisms (Song et al., 2019; Mao et al., 2023), and automated neural architecture designs (Tian et al., 2023; Cheng et al., 2020). Specifically, we select DCNV2 (Wang et al., 2021), AutoInt (Song et al., 2019), EulerNet (Tian et al., 2023),, AFN (Cheng et al., 2020), FinalMLP (Mao et al., 2023), and GDCNP (Wang et al., 2023). These models collectively represent the forefront of CTR prediction methodologies, providing a robust foundation for comparison. Focusing on models designed specifically for feature interaction modeling isolates the impact of MP effectively. Including diverse model types with differing objectives (e.g., behavior prediction, auxiliary tasks) could introduce modeling biases, confounding the true effects of connectivity patterns. More details of the baseline models are provided in Appendix B.

Table 1. The meta data of the curated datasets. k is the least number of interactions per user and item. Density = (#Inters / (#Users × #Items))×100%.

k	Users	Inter/User	Items	Inter/Item	Inters	Density%
ML-1M (dense)						
3	6K	165.6	3.5K	285.5	1M	4.72
6	6K	165.5	34K	296.0	1M	4.90
10	6K	165.3	3.3K	306.3	1M	5.07
Yelp2022 (sparse)						
3	529K	9.8	144K	36.2	5.2M	0.01
6	215K	17.9	96K	40.3	3.9M	0.02
10	99K	27.9	56K	48.9	2.8M	0.05

Training Configurations We adopt binary cross-entropy demonstrated in Equation 3 as the loss function to train models on the training set, and choose the AdamW (Loshchilov & Hutter, 2019) as the optimizer. We conduct grid searches over all the baseline models' provided hyper-parameters for their best AUC performance on the validation set. With the best hyper-parameters, we train the models under five random seeds and save all the checkpoints. Methods are evaluated by inferring the corresponding saved checkpoints.

Evaluation Metrics Following previous studies, we use the AUC score as the primary metric to evaluate CTR prediction performance. Logloss results are provided in Appendix C.1. We introduce the Marginal Cost of Improvement (MCI) metric, detailed in Section 4.4, to quantify the utility required for performance changes.

Three-tiered MP Integration to CTR models We propose a three-tier analysis framework that progressively integrates MP components at increasing levels of sophistication: (i) The baseline tier, denoted as Base, preserves the original CTR architectures without incorporating any MP mechanisms, serving as a fundamental performance benchmark;

[1] https://grouplens.org/datasets/movielens/1m/

[2] https://www.kaggle.com/datasets/yelp-dataset/yelp-dataset/

Table 2. The AUC comparison between models under TS (train from scratch with MP) and Base (the original CTR model). $\Delta\%$ shows the relative change brought by TS. Yelp2022 is the sparse dataset and ML-1M is the dense dataset. Color intensities show the extent of improvement or decrement and grey means no significant change. k is the minimum number of interactions per user and item.

ML-1M									
Model	DCNV2			EulerNet			FinalMLP		
k	3	6	10	3	6	10	3	6	10
Base	82.20	82.20	82.11	82.00	82.09	81.95	82.18	82.15	82.04
TS($\Delta\%$)	82.33 (0.16)	82.34 (0.18)	82.24 (0.16)	82.25 (0.30)	82.21 (0.15)	82.18 (0.28)	82.21 (0.03)	82.22 (0.09)	82.17 (0.15)
Model	GDCNP			AFN			AutoInt		
Base	82.07	82.12	81.97	82.02	82.08	81.93	82.11	82.14	81.99
TS($\Delta\%$)	82.33 (0.32)	82.35 (0.28)	82.24 (0.33)	82 (-0.02)	82.16 (0.10)	82.06 (0.16)	82.17 (0.08)	82.24 (0.12)	82.05 (0.07)
Yelp2022									
Model	DCNV2			EulerNet			FinalMLP		
Base	81.65	80.48	77.22	81.45	79.30	77.41	79.82	78.18	76.56
TS($\Delta\%$)	80.3 (-1.66)	79.19 (-1.61)	77.21 (-0.01)	81.65 (0.25)	79.71 (0.52)	77.52 (0.15)	79.87 (0.06)	78.27 (0.12)	76.97 (0.54)
Model	GDCNP			AFN			AutoInt		
Base	81.38	79.39	77.52	81.12	79.20	77.54	82.06	79.88	78.09
TS($\Delta\%$)	81.63 (0.31)	80.04 (0.83)	78.01 (0.63)	81.02 (-0.13)	79.64 (0.55)	77.77 (0.29)	81.57 (-0.60)	78.96 (-1.15)	76.41 (-2.16)

(ii) The second tier, denoted as FT, enhances the baseline by fine-tuning well-trained CTR models with graph encoders as MP modules, allowing for an evaluation of incremental performance gains from this augmentation; (iii) The third tier, denoted as TS, represents the most comprehensive integration, where CTR models are randomly initialized and trained from scratch with graph encoder-based MP throughout the entire training process.

4.2. Exploring the Benefits of MP for CTR Prediction

MP generally benefits CTR prediction. Within each dataset, we vary k in k-core filtering to adjust density locally among users and items. We compare TS, the third-tiered MP integration, with the baseline CTR models denoted as Base with varied k in $\{3, 6, 10\}$ of each dataset, where higher k represents higher dataset density. The AUC results are shown in Table 2. According to the table, TS demonstrates prominent beneficial performance in AUC score in 27 out of 36 experiments, while yielding considerable and inferior performance in 3 and 6 experiments, respectively, across six different models and three k-core values. These results indicates that MP generally benefits CTR prediction.

The benefits of MP are more pronounced in dense datasets. Among 18 experiments, 16 demonstrate an average AUC gain of 0.18%, while the remaining two exhibit less than 0.05% performance fluctuations compared to Base models. In contrast, for the sparse dataset Yelp2022, the beneficial ratio decreases to 11 out of 16, and the effectiveness of MP is model-dependent. This is because a higher interaction density provides richer connectivity patterns, enabling MP to effectively propagate meaningful information between users and items. The limited connectivity in sparse datasets reduces the quality of information propagation with more inherent noises, making the effectiveness of MP more dependent on the model's architecture and its ability to compensate for the lack of data.

Global dataset density drives MP effectiveness over local density. When examining local density among users and items, we notice that as k increases (i.e., higher local density), the benefits measured by relative performance does not consistently increase. There is no obvious pattern indicating that increased local density leads to more significant relative improvement. The effectiveness of MP depends more on the global dataset density (the dataset as a whole), rather than the local density among individual users or items.

4.3. Comparing between MP Integration Strategies

fine-tuning well-trained CTR models with MP is the better integration than training from scratch. To determine which level of connectivity integration is more beneficial for CTR prediction, we compare FT (second-tier integration by fine-tuning from Base) and TS (third-tier integration by training with MP from scratch) across varying k values. All the samples, whether they are strongly- or weakly-supervised, are retained in the training set. Focusing on the AUC results in Table 3 with EvalMR set to ✗ (evaluated on the complete test set), FT consistently outperforms TS in terms of relative performance improvement. On average, FT achieves a 0.22% higher relative performance change than TS across all datasets, with 0.11% on the dense dataset and 0.33% on the sparse dataset.

Importantly, the results on the dense and sparse datasets further reinforce our previous finding, which states that integrating MP more consistently benefits CTR prediction on

Table 3. The AUC comparison between TS (train from scratch with MP), FT (finetune from the original model with MP), and Base (the original CTR model). EvalRM denotes whether samples with middle ratings (weak supervisions) are retained (✓) or removed (✗) during evaluation. In training, samples with middle ratings (weak supervisions) are retained (✓). Yelp2022 is the sparse dataset and ML-1M is the dense dataset. Color intensities refer to the extent of improvement or decrement and grey means no significant change.

Model	EvalRM	Dataset	ML-1M			Yelp2022		
		k	3	6	10	3	6	10
DCNV2	✓	Base	82.20	82.20	82.11	81.65	80.48	77.22
		FT(Δ%)	82.51 (0.38)	82.53 (0.40)	82.25 (0.17)	80.49 (-1.42)	79.52 (-1.19)	77.61 (0.51)
		TS(Δ%)	82.33 (0.16)	82.34 (0.18)	82.24 (0.16)	80.3 (-1.66)	79.19 (-1.61)	77.21 (-0.01)
	✗	Base	89.57	89.56	89.41	86.35	85.73	82.30
		FT(Δ%)	89.87 (0.33)	89.84 (0.31)	89.6 (0.21)	85.93 (-0.48)	85.09 (-0.75)	83.11 (0.98)
		TS(Δ%)	89.69 (0.14)	89.67 (0.12)	89.58 (0.19)	85.72 (-0.72)	84.55 (-1.38)	82.7 (0.49)
AutoInt	✓	Base	82.11	82.14	81.99	82.06	79.88	78.09
		FT(Δ%)	82.28 (0.22)	82.32 (0.22)	82.2 (0.26)	81.84 (-0.26)	79.72 (-0.20)	78.13 (0.05)
		TS(Δ%)	82.17 (0.08)	82.24 (0.12)	82.05 (0.07)	81.57 (-0.60)	78.96 (-1.15)	76.41 (-2.16)
	✗	Base	89.47	89.50	89.28	86.95	85.08	83.39
		FT(Δ%)	89.66 (0.22)	89.7 (0.22)	89.47 (0.21)	86.76 (-0.23)	84.92 (-0.20)	83.48 (0.10)
		TS(Δ%)	89.56 (0.11)	89.59 (0.11)	89.37 (0.10)	86.56 (-0.45)	84.3 (-0.92)	81.88 (-1.82)
FinalMLP	✓	Base	82.18	82.15	82.04	79.82	78.18	76.56
		FT(Δ%)	82.3 (0.14)	82.31 (0.19)	82.15 (0.13)	79.52 (-0.37)	77.91 (-0.35)	76.73 (0.22)
		TS(Δ%)	82.21 (0.03)	82.22 (0.09)	82.17 (0.15)	79.87 (0.06)	78.27 (0.12)	76.97 (0.54)
	✗	Base	89.50	89.44	89.33	84.63	83.12	81.57
		FT(Δ%)	89.61 (0.12)	89.62 (0.20)	89.44 (0.12)	84.34 (-0.34)	82.77 (-0.43)	81.85 (0.34)
		TS(Δ%)	89.55 (0.06)	89.56 (0.14)	89.44 (0.12)	84.82 (0.23)	83.4 (0.33)	82.22 (0.79)

dense rather than sparse datasets. While adding MP enhances the performance of models trained on ML-1M, it can sometimes negatively impact models trained on Yelp2022, especially when local density is low (i.e., with small k values). However, we notice that even when integrating MP has a negative effect, FT generally mitigates the adverse impact more effectively than TS, which integrates MP comprehensively from the start of model training. Therefore, when incorporating MP to models trained on dense datasets, finetuning from well-trained CTR models proves to be more efficient than training with MP from scratch, by leading to more performance benefits with less computational costs.

Performance improvement in dense datasets is more consistent across weakly and strongly-supervised samples. We use EvalRM to assess the consistency of AUC performance among samples with varying levels of supervision reliability. Specifically, in Table 3, we focus on results evaluated on the results evaluated on the test set where weakly supervised samples are either retained (✓) or removed (✗).

For the dense dataset, the relative performance improvement is more consistent across samples providing strong and weak supervision compared to the sparse dataset. This is evidenced by the comparable relative performance observed regardless of whether weakly supervised samples are removed or retained. For example, for AutoInt evaluated by ML-1M with $k = 3$, the relative performance improvement by FT remains 0.22 in both cases. Similarly, for Final MLP evaluated by ML-1M with $k = 10$, the relative performance improvement by TS shows minimal variation (0.13 vs 0.12). In ML-1M, the difference in relative performance change between EvalRM set to ✗ and ✓ is within 0.03%. However, this difference increase to 0.29% in Yelp2022.

4.4. Democratizing CTR Prediction into Under-served Groups through MP

Under-served groups Under-served groups refer to users with lower prediction performance compared to others, regardless of their interaction activity levels. For instance, a highly active user who interacts with niche items might still receive lower prediction accuracy due to insufficient overlapping patterns with other users, making them under-served. Conversely, a highly active user engaging with popular items can be well-served due to the abundance of shared interaction data. Similarly, a low-active user may also be either under-served or well-served; an infrequent user interacting with niche items may suffer from low prediction accuracy, while another interacting with popular items may benefit from higher accuracy despite their low activity level.

Marginal Cost of Improvement While relative performance change is widely adopted to evaluate the extent of improvement, it can be misleading as it disproportionately amplifies small gains when baseline performance is low, making them appear more significant. Moreover, it overlooks the marginal cost required to achieve these improve-

Table 4. The AUC performance on ML-1M (6-core) and Yelp2022 (10-core) across users with varying activity levels. Base refers to the original CTR model, TS refers to training from scratch with MP, and FT refers to fine-tuning from the original model with MP.

ML-1M									
Model	DCNV2			AutoInt			FinalMLP		
Variant	Base	FT(Δ%)	TS(Δ%)	Base	FT(Δ%)	TS(Δ%)	Base	FT(Δ%)	TS(Δ%)
Overall ↑	82.20	82.53 (0.40)	82.34 (0.18)	82.14	82.32 (0.22)	82.24 (0.12)	82.15	82.31 (0.19)	82.22 (0.09)
Low-U.Act. ↑	76.51	77.02 (0.67)	77.00 (0.63)	76.61	76.97 (0.48)	76.76 (0.19)	76.47	76.60 (0.18)	76.66 (0.25)
Mid-U.Act ↑	79.91	80.34 (0.53)	80.12 (0.25)	79.87	80.10 (0.29)	79.97 (0.12)	79.98	80.16 (0.22)	80.03 (0.06)
High-U.Act. ↑	83.24	83.5 (0.31)	83.32 (0.10)	83.15	83.29 (0.17)	83.24 (0.12)	83.15	83.29 (0.17)	83.22 (0.08)
Var(L,M,H) ↓	11.30	10.48 (-7.28)	10.01 (-11.43)	10.69	9.97 (-6.73)	10.52 (-1.58)	11.17	11.2 (0.34)	10.76 (-3.62)
Yelp2022									
Overall ↑	77.22	77.61 (0.51)	77.21 (-0.01)	78.09	78.13 (0.05)	76.41 (-2.16)	76.56	76.73 (0.22)	76.97 (0.54)
Low-U.Act. ↑	79.34	79.25 (-0.11)	79.02 (-0.41)	79.70	79.61 (-0.10)	76.79 (-3.65)	76.84	76.89 (0.07)	77.48 (0.84)
Mid-U.Act. ↑	78.52	78.63 (0.13)	78.27 (-0.32)	79.14	79.11 (-0.04)	76.87 (-2.86)	77.02	77.18 (0.21)	77.55 (0.69)
High-U.Act. ↑	76.14	76.79 (0.85)	76.39 (0.32)	77.23	77.36 (0.17)	76.21 (-1.32)	76.30	76.49 (0.26)	76.60 (0.39)
Var(L,M,H) ↓	2.77	1.64 (-40.61)	1.84 (-33.60)	1.67	1.40 (-16.51)	0.13 (-92.17)	0.14	0.12 (-14.93)	0.28 (102.12)

ments, which is crucial for assessing the actual effort a model expends to deliver such gains.

To address this problem, inspired by the economic concept of *marginal utility* (Walras, 1900; Menger & Menger, 1923), we propose Marginal Cost of Improvement (MCI), a metric that quantifies the marginal utility required to achieve performance improvements. In our context, we use MCI to measure the effort allocated to each user group for their respective performance change. Specifically, let $p_b \in (0, 1)$ be the baseline AUC and c be the absolute change in AUC. A valid (p_b, c) satisfies $p_b + c \in (0, 1)$. MCI is defined as:

$$\text{MCI}(p_b, c) = \log\left(\frac{1 - p_b}{1 - p_b - c}\right). \tag{10}$$

Notice that $c > 0$ indicates a performance improvement with the corresponding MCI > 0; $c < 0$ indicates a performance decrement with the corresponding MCI < 0.

Importantly, MCI satisfies conditions derived from three natural marginal utility comparison cases: (i) Under the same baseline performance (p_b), the larger the absolute change in performance (c), the higher the cost (MCI); (ii) Under the same ultimate performance ($p_b + c$), the lower the baseline performance, the higher the cost; (iii) Under the same absolute change in performance, the lower the baseline performance, the higher the cost required or utility lost (i.e., |MCI| is larger). The corresponding MCI of various (p_b, c) pairs are plotted in Figure 4. We prove that MCI satisfies the above three conditions in Appendix E.

Impact of MP on Under-served Groups MP inherently allocates greater effort to under-served groups by propagating information from well-represented users those who are under-served. Using activity level as the grouping factor, we categorize users into low-active, medium-active, and highly-active groups based on their number of interactions.

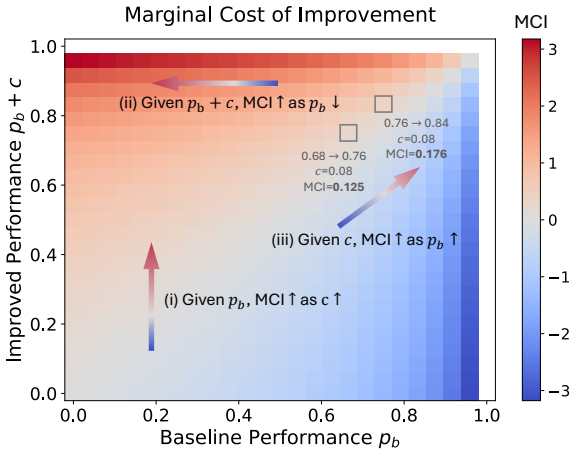

Figure 4. A plot of MCI values with varied performance. p_b is the baseline performance, c is the absolute performance change, and $p_b + c$ is the new performance. The MCI in between $MCI(p_b, c) = \log((1 - p_b)/(1 - p_b - c))$.

We first evaluate the AUC performance of each user group to identify the under-served users, and then utilize MCI to inspect the effort allocated to each group relative to their performance changes.

The AUC performance of each user group is presented in Table 4. For ML-1M, low-active users are identified as under-served, with an average performance decrement of 6.9% compared to the overall AUC. Conversely, in Yelp2022, highly-active users are under-served with on average 1.4% less than the overall performance. The above result highlights that being highly active does not guarantee a user will be well-served by the model. Furthermore, we observe that the variance in the AUC performance across user groups is generally reduced in both FT and TS compared to Base. Performance variance across user groups serves as a strong indicator of democratization—the smaller the variance, the greater the level of democratization and equity among users.

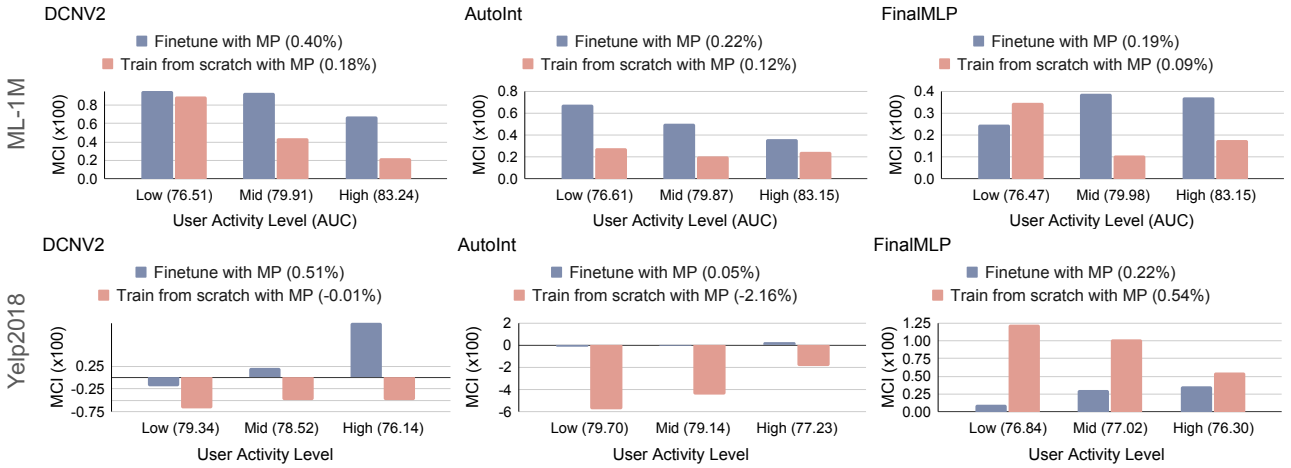

Figure 5. MCI-evaluated cost for performance improvement (MCI >0) and decrement (MCI <0) across users with varying activity levels. In ML-1M, low-active users are identified as the under-served group, while in Yelp2022, it is the group consists of highly-active users.

We further evaluate the MCI for each performance change across user groups, with the results illustrated in Figure 5. In ML-1M where both FT and TS exhibit performance improvements, the models consistently allocate greater effort to low-active users (the under-served group in ML-1M) compared to others. In Yelp2022 where the effectiveness of MP is more model-dependent, performance improvements primarily result in greater effort allocated to highly-active users (the under-served group in Yelp2022). Conversely, when MP leads to performance downgrades, under-served users experience the least utility lost measured demonstrated by the smallest absolute MCI values. These observations reinforce the notion that MP, whether integrated through FT or TS, inherently supports the democratization of CTR prediction by prioritizing under-served users.

4.5. MP Mitigates Social Media Polarization

Social media platforms often create *echo chambers*, where users are predominantly exposed to content that aligns with their existing beliefs (Pazzanese, 2017; Barberá, 2020). This selective exposure reinforces pre-existing opinions and fosters ideological segregation. Environments characterized by echo chambers tend to exacerbate societal polarization in the real world, influencing areas such as politics, wealth disparity, and democratic elections (Gillani et al., 2018).

In CTR prediction models, similar isolation patterns can occur. For example, models may inadvertently favor user groups with higher activity levels, leading to disparities in performance across different user segments. This bias can result in the under-representation of less active users, creating a form of "algorithmic echo chamber" within models' predictions (Centola, 2020; Piccardi et al., 2024). Beyond the numerical improvement from the integration of MP technique within CTR models, it potentially further facilitates the propagation of information across well-represented users to under-served ones, equitably bridging the gap between users with different activity levels. Our analysis with the MCI metric, demonstrates that MP inherently allocates greater effort to under-served groups. This allocation is evident in the favored AUC performance observed among these groups with MP integration. By prioritizing the improvement of under-served user segments, MP not only democratizes CTR prediction but also moves towards mitigating the risk of polarization within the model's outcomes.

5. Conclusion

In this work, we investigate how leveraging user-item connectivity patterns affect existing CTR models. Our theoretical analysis first shows that incorporating connectivity reduces the Lipschitz constant of a deep model, thereby enhancing its stability. We then apply the MP mechanism exclusively in the embedding process to enhance existing CTR models. To quantify the utility required for performance changes, we introduce the MCI metric that measures the marginal utility of incremental improvements. Our extensive experiments reveal that incorporating MP, especially in fine-tuning, provides a more stable performance enhancement in dense datasets than in sparse ones. Evaluated by MCI, we discover that MP democratizes CTR technology into under-served groups by allocating the greatest benefits to users with lower original CTR prediction performance, enhancing the inclusivity of recommendation environments.

Impact Statement

Our investigation into integrating MP into CTR prediction models has significant social and research impacts. *First*, MP facilitates the propagation of information from well-

represented to under-served users, leading to numerical performance improvements. Our findings offer practical guidance for both industry practitioners and researchers on when and how to effectively leverage connectivity patterns for optimal performance enhancement within existing CTR prediction pipelines. *Second*, MP reduces performance disparities across user groups by mitigating algorithmic biases that disproportionately favor certain users, leading to a more equitable and inclusive recommendation landscape. *Lastly*, democratizing CTR prediction into under-served groups helps counteract the formation of algorithmic echo chambers, where specific user groups receive disproportionately tailored content. By preventing the reinforcement of these echo chambers, MP reduces the risk of misinformation spreading, ultimately preserving trust in credible sources and mitigating the dominance of harmful narratives.

References

Araujo, A., Negrevergne, B., Chevaleyre, Y., and Atif, J. On Lipschitz regularization of convolutional layers using toeplitz matrix theory. In *AAAI*, 2021.

Barberá, P. Social media, echo chambers, and political polarization. *Social media and democracy: The state of the field, prospects for reform*, 2020.

Centola, D. Why social media makes us more polarized and how to fix it. *Scientific American*, 2020.

Cheng, W., Shen, Y., and Huang, L. Adaptive factorization network: Learning adaptive-order feature interactions. In *AAAI*, 2020.

Dasoulas, G., Scaman, K., and Virmaux, A. Lipschitz normalization for self-attention layers with application to graph neural networks. In *ICML*, 2021.

Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., and Yin, D. Graph neural networks for social recommendation. In *WWW*, 2019.

Gama, F. and Sojoudi, S. Distributed linear-quadratic control with graph neural networks. *Signal Processing*, 2022.

Gillani, N., Yuan, A., Saveski, M., Vosoughi, S., and Roy, D. Me, my echo chamber, and i: introspection on social media polarization. In *WWW*, 2018.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *ICML*, 2017.

Guo, H., Tang, R., Ye, Y., Li, Z., and He, X. Deepfm: a factorization-machine based neural network for CTR prediction. In *IJCAI*, 2017.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *NIPS*, 2017.

He, R. and McAuley, J. Vbpr: visual bayesian personalized ranking from implicit feedback. In *AAAI*, 2016.

He, X. and Chua, T.-S. Neural factorization machines for sparse predictive analytics. In *SIGIR*, 2017.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., and Wang, M. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*, 2020.

Horn, R. A. and Johnson, C. R. *Matrix Analysis*. Cambridge University Press, 1985.

Jamali, M. and Ester, M. A matrix factorization technique with trust propagation for recommendation in social networks. In *ACM Recommender Systems conference*, 2010.

Kim, H., Papamakarios, G., and Mnih, A. The Lipschitz constant of self-attention. In *ICML*, 2021.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2016.

Lian, J., Zhou, X., Zhang, F., Chen, Z., Xie, X., and Sun, G. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *SIGKDD*, 2018.

Lin, Z., Tian, C., Hou, Y., and Zhao, W. X. Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *WWW*, 2022.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *ICLR*, 2019.

Ma, H., Yang, H., Lyu, M. R., and King, I. Sorec: social recommendation using probabilistic matrix factorization. In *CIKM*, 2008.

Mao, K., Zhu, J., Su, L., Cai, G., Li, Y., and Dong, Z. Finalmlp: an enhanced two-stream mlp model for ctr prediction. In *AAAI*, 2023.

Menger, C. and Menger, K. *Grundsätze der volkswirtschaftslehre*. Hölder-Pichler-Tempsky, 1923.

Ouyang, Z., Zhang, C., Hou, S., Zhang, C., and Ye, Y. How to improve representation alignment and uniformity in graph-based collaborative filtering? In *ICWSM*, 2024.

Pauli, P., Gramlich, D., and Allgöwer, F. Lipschitz constant estimation for 1d convolutional neural networks. In *Learning for Dynamics and Control Conference*, 2023.

Pazzanese, C. Danger in the internet echo chamber. *Harvard Gazette*, 2017.

Piccardi, T., Saveski, M., Jia, C., Hancock, J. T., Tsai, J. L., and Bernstein, M. Social media algorithms can shape affective polarization via exposure to antidemocratic attitudes and partisan animosity, 2024.

Qi, X., Wang, J., Chen, Y., Shi, Y., and Zhang, L. Lipsformer: Introducing lipschitz continuity to vision transformers. In *ICLR*, 2023.

Rendle, S. Factorization machines. In *ICDM*, 2010.

Schafer, J. B., Konstan, J., and Riedl, J. Recommender systems in e-commerce. In *ACM conference on Electronic commerce*, 1999.

Song, W., Shi, C., Xiao, Z., Duan, Z., Xu, Y., Zhang, M., and Tang, J. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *CIKM*, 2019.

Terris, M., Repetti, A., Pesquet, J.-C., and Wiaux, Y. Building firmly nonexpansive convolutional neural networks. In *ICASSP*, 2020.

Tian, Z., Bai, T., Zhao, W. X., Wen, J.-R., and Cao, Z. Eulernet: Adaptive feature interaction learning via euler's formula for CTR prediction. In *SIGIR*, 2023.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. In *ICLR*, 2017.

Walras, L. *Éléments d'économie politique pure: ou, Théorie de la richesse sociale*. F. Rouge, 1900.

Wang, C., Yu, Y., Ma, W., Zhang, M., Chen, C., Liu, Y., and Ma, S. Towards representation alignment and uniformity in collaborative filtering. In *SIGKDD*, 2022.

Wang, F., Gu, H., Li, D., Lu, T., Zhang, P., and Gu, N. Towards deeper, lighter and interpretable cross network for ctr prediction. In *CIKM*, 2023.

Wang, R., Fu, B., Fu, G., and Wang, M. Deep and cross network for ad click predictions. In *Proceedings of the Workshop on Data Mining for Online Advertising (ADKDD)*, 2017.

Wang, R., Shivanna, R., Cheng, D., Jain, S., Lin, D., Hong, L., and Chi, E. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *WWW*, 2021.

Wang, X., He, X., Wang, M., Feng, F., and Chua, T.-S. Neural graph collaborative filtering. In *SIGIR*, 2019.

Wu, J., Wang, X., Feng, F., He, X., Chen, L., Lian, J., and Xie, X. Self-supervised graph learning for recommendation. In *SIGIR*, 2021.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *In ICLR*, 2018.

Yu, J., Yin, H., Xia, X., Chen, T., Cui, L., and Nguyen, Q. V. H. Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *SIGIR*, 2022.

Zhu, J., Jia, Q., Cai, G., Dai, Q., Li, J., Dong, Z., Tang, R., and Zhang, R. Final: Factorized interaction layer for ctr prediction. In *SIGIR*, 2023.

Zou, D., Balan, R., and Singh, M. On lipschitz bounds of general convolutional neural networks. *IEEE Transactions on Information Theory*, 2019.

A. Related Work

A.1. Click-through Rate Prediction

Early CTR research focused on efficiently modeling interactions between user-item pairs and contextual features. Low-order feature interactions were preserved using methods like Factorization Machines (Rendle, 2010), while high-order feature interactions were explored through deep neural networks (He & Chua, 2017; Guo et al., 2017; Lian et al., 2018). Later studies refined DNN architectures to automatically learn bounded-degree feature interactions (Wang et al., 2017; Cheng et al., 2020; Wang et al., 2021; Zhu et al., 2023; Mao et al., 2023). More recent approaches have proposed projecting features into predefined hyperspaces (Song et al., 2019; Tian et al., 2023), enabling more efficient and complex feature interactions. Building upon previous research, this study explores the potential of the additional information embedded in the connectivity patterns among users, items, and between users and items in benefiting the predictive ability in CTR models.

A.2. Message Passing

Graph Neural Networks (GNNs) are robust learning frameworks designed to extract meaningful representations from graph-structured data (Kipf & Welling, 2016; Veličković et al., 2017; Hamilton et al., 2017). These networks map input nodes into low-dimensional vectors, which can be applied to various tasks, such as graph-level classification (Xu et al., 2018) or node-level predictions (Kipf & Welling, 2016). Most GNNs utilize a layer-wise MP mechanism (Gilmer et al., 2017), where each node iteratively aggregates information from its first-order neighbors. By stacking multiple layers, these models capture information from multi-hop neighbors, enabling a richer contextual understanding. GNNs are also widely adopted to abstract collaborative filtering signals from user-item interactive graphs. Improving upon NGCF (Wang et al., 2019), one of the pioneering efforts to apply GNNs to ranking recommendation, LightGCN (He et al., 2020) identifies the redundancies in the non-linearity and over-parametrization of traditional GNN architectures, and has spurred numerous following works with noticeable performance improvements (Wu et al., 2021; Yu et al., 2022; Wang et al., 2022; Lin et al., 2022; Ouyang et al., 2024).

A.3. Lipschitz Constant in Deep Models

Prior research on Lipschitz constants has primarily focused on general deep models incorporating convolutional or attention layers (Zou et al., 2019; Terris et al., 2020; Kim et al., 2021; Araujo et al., 2021; Pauli et al., 2023). In the context of attention, Dasoulas et al. (2021) discuss the Lipschitz constant for their designed a Lipschitz continuous Transformer Qi et al. (2023). More recently, (Gama & Sojoudi, 2022) estimated the filter Lipschitz bound using the infinite norm of a matrix.

Table 5. The corresponding Logloss to the AUC performance in Table 2. Lower logloss's (↓) indicate better prediction confidence.

Model	Dataset	ML-1M			Yelp2022		
	k	3	6	10	3	6	10
DCNV2	Base	0.5095	0.5095	0.5115	0.5110	0.5091	0.5829
	TS	0.5064	0.5069	0.5102	0.5031	0.5076	0.5286
	Δ% ↓	-0.60	-0.51	-0.25	-1.55	-0.29	-9.31
EulerNet	Base	0.5123	0.5137	0.5161	0.4774	0.4935	0.5100
	TS	0.5113	0.5104	0.5105	0.4712	0.4975	0.5150
	Δ% ↓	-0.21	-0.64	-1.08	-1.29	0.81	0.97
FinalMLP	Base	0.5079	0.5083	0.5090	0.4980	0.5051	0.5143
	TS	0.5077	0.5071	0.5086	0.4917	0.5046	0.5118
	Δ% ↓	-0.03	-0.23	-0.09	-1.27	-0.11	-0.48
GDCNP	Base	0.5105	0.5112	0.5124	0.4755	0.4955	0.5064
	TS	0.5091	0.5092	0.5095	0.4722	0.4851	0.5015
	Δ% ↓	-0.27	-0.39	-0.58	-0.69	-2.10	-0.97
AFN	Base	0.5098	0.5091	0.5103	0.5228	0.5295	0.5338
	TS	0.5100	0.5083	0.5088	0.5166	0.5166	0.5398
	Δ% ↓	0.02	-0.16	-0.29	-1.19	-2.44	1.13
AutoInt	Base	0.5099	0.5091	0.5128	0.4681	0.4945	0.5011
	TS	0.5080	0.5079	0.5112	0.4739	0.6053	0.5452
	Δ% ↓	-0.38	-0.24	-0.33	1.25	22.41	8.80

B. Baselines

DCNV2 (Wang et al., 2021) learns explicit and implicit feature interactions through a cross-network and a deep neural network, respectively. It improves DCN with a low-rank cross-network that enhances the efficiency and interpretability of the model. AutoInt (Song et al., 2019) utilizes self-attentive neural networks to learn more effective feature interactions. EulerNet (Tian et al., 2023) learns high-order feature interactions by transforming their exponential powers into linear combinations of the modulus and phase of complex features. AFN (Cheng et al., 2020) focuses on modeling feature interactions through an adaptive logarithmic transformation, capturing both low- and high-order interactions effectively. FinalMLP (Mao et al., 2023) simplifies the learning process by leveraging a lightweight, fully connected structure to model feature interactions with high efficiency. GDCNP (Wang et al., 2023) enhances deep cross networks by integrating graph-based connectivity to refine feature representation and interaction modeling.

C. Further Analysis

C.1. Logloss Results

We provide the Logloss results in Table 5 and 6, in relative to the AUC performance in Table 2 and 3. Note that different from AUC, the lower Logloss, the more reliable the output.

The results in Table 5 generally align with those in Table 2. In the dense dataset ML-1M, incorporating MP under the TS

Table 6. The corresponding Logloss results to the AUC performance in Table 3. Lower logloss's (↓) indicate better prediction confidence. Base refers to the original CTR model, TS refers to training from scratch with MP, and FT refers to fine-tuning from the original model with MP. k is the least number of interactions per user and item.

Model	EvalRM	Dataset	ML-1M			Yelp2022		
		k	3	6	10	3	6	10
DCNV2	✗	Base	0.5095	0.5095	0.5115	0.5110	0.5091	0.5829
		FT(Δ%)	0.5037 (-1.14)	0.5032 (-1.24)	0.5075 (-0.77)	0.4983 (-2.49)	0.497 (-2.38)	0.5325 (-8.64)
		TS(Δ%)	0.5064 (-0.6)	0.5069 (-0.51)	0.5102 (-0.25)	0.5031 (-1.55)	0.5076 (-0.29)	0.5286 (-9.31)
	✓	orig	0.4082	0.4129	0.4138	0.4021	0.4615	0.5070
		FT(Δ%)	0.4012 (-1.72)	0.4009 (-2.92)	0.4101 (-0.87)	0.4209 (4.68)	0.4149 (-10.11)	0.4825 (-4.84)
		TS(Δ%)	0.4103 (0.52)	0.4058 (-1.73)	0.4151 (0.31)	0.4262 (6)	0.416 (-9.86)	0.464 (-8.48)
AutoInt	✗	Base	0.5099	0.5091	0.5128	0.4681	0.4945	0.5011
		FT(Δ%)	0.5071 (-0.56)	0.5059 (-0.64)	0.5084 (-0.86)	0.4696 (0.33)	0.4918 (-0.55)	0.4992 (-0.37)
		TS(Δ%)	0.508 (-0.38)	0.5079 (-0.24)	0.5112 (-0.33)	0.4739 (1.25)	0.6053 (22.41)	0.5452 (8.8)
	✓	Base	0.4139	0.4109	0.4132	0.3914	0.4257	0.4027
		FT(Δ%)	0.4116 (-0.54)	0.4094 (-0.37)	0.4224 (2.22)	0.3968 (1.38)	0.4173 (-1.97)	0.4062 (0.87)
		TS(Δ%)	0.4199 (1.47)	0.4034 (-1.84)	0.4293 (3.9)	0.4042 (3.27)	0.607 (42.58)	0.4886 (21.33)
FinalMLP	✗	Base	0.5079	0.5083	0.5090	0.4980	0.5051	0.5143
		FT(Δ%)	0.5061 (-0.35)	0.506 (-0.45)	0.5074 (-0.32)	0.4933 (-0.95)	0.5046 (-0.11)	0.511 (-0.63)
		TS(Δ%)	0.5077 (-0.03)	0.5071 (-0.23)	0.5086 (-0.09)	0.4917 (-1.27)	0.5046 (-0.11)	0.5118 (-0.48)
	✓	Base	0.4129	0.4079	0.4159	0.4109	0.4074	0.4029
		FT(Δ%)	0.4095 (-0.81)	0.4076 (-0.06)	0.4097 (-1.51)	0.4072 (-0.89)	0.4087 (0.33)	0.4026 (-0.07)
		TS(Δ%)	0.4116 (-0.31)	0.4102 (0.57)	0.4161 (0.04)	0.4078 (-0.75)	0.4083 (0.23)	0.3959 (-1.73)

Table 7. The meta data for the paired curated datasets.

TrRM	k	#users	#inter/user	#items	#inter/item	#inters	Density%
ML-1M							
✓	1	6.0K	165.6	3.7K	269.9	1.0M	4.47
✗	19	5.7K	128.6	2.9K	251.4	0.7M	4.44
✓	3	6.0K	165.6	3.5K	285.5	1.0M	4.72
✗	22	5.4K	133.1	2.8K	256.5	0.72M	4.74
Yelp2022							
✓	3	0.5M	9.8	0.14M	36.2	5.2M	0.01
✗	3	0.5M	9.2	0.14M	32.0	4.5M	0.01
✓	6	0.2M	17.9	96K	40.3	3.8M	0.02
✗	6	0.2M	16.8	89K	36.3	3.2M	0.02

configuration not only improves the AUC performance but also enhances prediction confidence. Similar to the observations from Table 2, in the sparse dataset Yelp2022, we find the benefits of MP in enhancing the reliability of the predictions is model-dependent. For instance, in DCNV2 (Wang et al., 2021), MP consistently improves prediction reliability enhancement, whereas in EulerNet (Tian et al., 2023), MP proves beneficial only when local dataset is low. However, for AutoInt (Song et al., 2019), incorporating MP negatively impacts prediction reliability across all k values.

The results in Table 6 also align with those in Table 3. Comparing with TS, incorporating MP under FT is more likely to enhance model prediction confidence, as reflected in lower Logloss values. Even when MP decreases confidence, FT mitigates the negative effect brought by MP more effectively than TS. Notably, we observe that prediction confidence in strongly supervised samples is lower than the overall confidence, suggesting that these samples exhibit more general patterns, making them easier for the model to distinguish. Consequently, improving confidence in predicting strongly supervised samples is more challenging, as their confidence levels are already high, whereas noisy, weakly supervised samples offer more room for confidence improvement.

C.2. Impact of Supervision Reliability on MP in CTR Prediction

We vary the inclusivity of weakly-supervised samples during both training and testing to further investigate how supervision reliability affects the effectiveness of integrating MP to CTR prediction. To control for the influence of data density, we group curated datasets into pairs with similar densities. By comparing results within each pair, we can isolate and analyze the impact of supervision reliability during training while minimizing the effects of the confounding factor of dataset density. The meta data for the the paired curated datasets is provided in Table 7.

The results in Table 8 indicate that excluding weakly supervised samples during training improves the performance of strongly supervised samples. In the dense dataset ML-1M, MP consistently enhances performance regardless of the inclusion of weakly supervised samples, with relative improvements remaining stable across both weakly and strongly supervised samples. However, in sparse datasets, training with weakly supervised samples affects the integra-

Table 8. The AUC comparison between TS (train from scratch with MP) and Base (the original CTR model) across different middle-rating processing configurations. Samples with middle ratings (weakly-supervised) are either retained (✓) or removed (✗) during training (TrRM) and evaluation (EvalRM). k is the least number of interactions per user and item. $\Delta\%$ shows the relative change brought by TS. Yelp2022 is the sparse dataset and ML-1M is the dense dataset. Color intensities refer to the extent of improvement or decrement and grey means no significant change.

Dataset		ML-1M					Yelp2022				
Model	TrRM	EvalRM	✓		✗		EvalRM	✓		✗	
		k	Base	TS($\Delta\%$)	Base	TS($\Delta\%$)	k	Base	TS($\Delta\%$)	Base	TS($\Delta\%$)
DCNV2	✓	1	82.00	82.12 (0.14)	89.44	89.55 (0.11)	3	81.65	80.3 (-1.66)	86.35	85.72 (-0.72)
	✗	19		-	90.15	90.34 (0.21)	3		-	88.14	86.96 (-1.33)
	✓	3	82.20	82.33 (0.16)	89.57	89.69 (0.14)	6	80.48	79.19 (-1.61)	85.73	84.55 (-1.38)
	✗	22		-	90.14	90.21 (0.08)	6		-	86.26	85.83 (-0.49)
FinalMLP	✓	1	81.95	81.97 (0.03)	89.35	89.38 (0.04)	3	79.82	79.87 (0.06)	84.63	84.82 (0.23)
	✗	19		-	90.22	90.32 (0.11)	3		-	85.90	85.75 (-0.17)
	✓	3	82.18	82.21 (0.03)	89.50	89.55 (0.06)	6	78.18	78.27 (0.12)	83.12	83.4 (0.33)
	✗	22		-	90.26	90.4 (0.16)	6		-	84.56	84.48 (-0.09)
GDCNP	✓	1	81.86	82.15 (0.35)	89.27	89.58 (0.35)	3	81.38	81.63 (0.31)	86.37	86.69 (0.37)
	✗	19		-	90.04	90.34 (0.33)	3		-	86.70	87.1 (0.47)
	✓	3	82.07	82.33 (0.32)	89.38	89.7 (0.35)	6	79.39	80.04 (0.83)	84.51	85.24 (0.86)
	✗	22		-	90.11	90.39 (0.31)	6		-	85.58	86.16 (0.68)
AutoInt	✓	1	81.93	81.96 (0.04)	89.34	89.38 (0.04)	3	82.06	81.57 (-0.6)	86.95	86.56 (-0.45)
	✗	19		-	90.13	90.19 (0.06)	3		-	87.09	86.97 (-0.14)
	✓	3	82.11	82.17 (0.08)	89.47	89.56 (0.11)	6	79.88	78.96 (-1.15)	85.08	84.3 (-0.92)
	✗	22		-	90.14	90.21 (0.08)	6		-	85.35	85.85 (0.59)

tion of MP—when MP leads to performance gains, strongly supervised samples experience higher relative improvement, while in cases of performance degradation, the negative effects on strongly supervised samples are mitigated. Furthermore, we again observe that the effectiveness of MP in the sparse Yelp2022 dataset is highly model-dependent, emphasizing the need for careful consideration when integrating MP in sparse datasets.

D. Proof of Theorem 1

Theorem 1 *For any node $i \in \mathcal{V}$ and its node feature $\mathbf{h}_i$, consider the mapping function with the updating rule demonstrated in Equation 4. Then, the Lipschitz constant L of the mapping $f : \mathbf{h} \mapsto \mathbf{h}'$, where $\mathbf{h}, \mathbf{h}' \in \mathbb{R}^d$ are the collection of all node features in the graph before and after the mapping respectively, satisfies:*

$$L \leq L_\sigma \|W\|_F \sqrt{1 + \frac{1}{k_i}},$$

where $\|W\|_F$ is the Frobenius norm of the weight matrix W, k_i denotes the degree of node i, L_σ is the Lipschitz constant of a Lipschitz continuous activation function $\sigma(\cdot)$, and d is the dimension of the node feature.

Proof. We aim to estimate the Lipschitz constant L of the mapping $f : \mathbf{h} \mapsto \mathbf{h}'$, where $\mathbf{h}$ and $\mathbf{h}'$ denotes the collection of all node features in the graph before and after the mapping, respectively. The Lipschitz constant measures how much the output of the function can change with respect to changes in the input, and can be estimated by analyzing the Jacobian matrix $\mathbf{J}$ of the mapping.

Step 1: Define the Jacobian of the Mapping
Consider the mapping for node i:

$$\mathbf{h}'_i = \sigma\left(W\left(\mathbf{h}_i + \frac{1}{k_i}\sum_{j \in \mathcal{N}(i)} \mathbf{h}_j\right)\right) = \sigma\left(W\mathbf{s}_i\right), \quad (11)$$

where $\mathbf{s}_i = \mathbf{h}_i + \mathbf{h}_{\text{agg}}$ and $\mathbf{h}_{\text{agg}} = \frac{1}{k_i}\sum_{j \in \mathcal{N}(i)} \mathbf{h}_j$. The Jacobian matrix $\mathbf{J}$ of this mapping is the matrix of partial derivatives of $\mathbf{h}'_i$ wrt all node features $\mathbf{h}_j$ for $j \in \mathcal{V}$:

$$\mathbf{J} = \frac{\partial \mathbf{h}'_i}{\partial \mathbf{h}^\top}, \quad (12)$$

where $\mathbf{h}'_i \in \mathbb{R}^d$ is the updated feature vector of node i, and $\mathbf{h} \in \mathbb{R}^{|\mathcal{N}|d}$ denotes the concatenation of all node features in the graph.

Step 2: Express the Jacobian matrix

The Jacobian matrix can be partitioned into blocks corresponding to derivatives for $\mathbf{h}_i$ and $\mathbf{h}_j, \forall j \in \mathcal{N}(i)$:

(i) For the partial derivative of $\mathbf{h}'_i$ wrt $\mathbf{h}_i$, we have

$$\frac{\partial \mathbf{h}'_i}{\partial \mathbf{h}_i^\top} = D_i W, \tag{13}$$

where $D_i = \operatorname{diag}(\sigma'(W\mathbf{s}_i))$ is a diagonal matrix containing the derivatives of the activation function applied element-wise, and $\sigma'(\cdot)$ is the elementwise derivative of the activation function $\sigma(\cdot)$.

(ii) For the partial derivative of $\mathbf{h}'_i$ wrt $\mathbf{h}_j, \forall j \in \mathcal{N}(i)$:

$$\frac{\partial \mathbf{h}'_i}{\partial \mathbf{h}_j^\top} = \begin{cases} D_i W \cdot \frac{1}{k_i}, & \text{if } j \in \mathcal{N}(i), \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

This distinction of the two cases arises because $\mathbf{h}'_i$ depends only on the features of node i and its neighbors $j \in \mathcal{N}(i)$.

Step 3: Bound the Partial Derivatives
To estimate the Lipschitz constant, we need to estimate the spectral norm of the Jacobian $\|\mathbf{J}\|_2$. Since the spectral norm is less than or equal to the Frobenius norm (Horn & Johnson, 1985), we seek to compute the Frobenius norm of $\mathbf{J}$:

$$\|\mathbf{J}\|_F = \sqrt{\left\| \frac{\partial \mathbf{h}'_i}{\partial \mathbf{h}_i^\top} \right\|_F^2 + \sum_{j \in \mathcal{N}(i)} \left\| \frac{\partial \mathbf{h}'_i}{\partial \mathbf{h}_j^\top} \right\|_F^2}. \tag{15}$$

We now simplify the expression separately. We simplify the first Frobenius norm as:

$$\begin{aligned} \left\| \frac{\partial h'_i}{\partial \mathbf{h}_i^\top} \right\|_F &= \|D_i W\|_F \le \|D_i\|_F \|W\|_F \\ &\le L_\sigma \|W\|_F. \end{aligned} \tag{16}$$

Similarity, the second Frobenius norm can be simplified as:

$$\begin{aligned} \left\| \frac{\partial \mathbf{h}'_i}{\partial \mathbf{h}_j^\top} \right\|_F &= \left\| D_i W \cdot \frac{1}{k_i} \right\|_F = \frac{1}{k_i} \|D_i W\|_F \\ &\le \frac{\|D_i\|_F \|W\|_F}{k_i} \\ &\le \frac{L_\sigma \|W\|_F}{k_i} \quad (\|D_i\|_F \le L_\sigma). \end{aligned} \tag{17}$$

Summing over the norms, we have:

$$\sum_{j \in \mathcal{N}(i)} \left\| \frac{\partial \mathbf{h}'_i}{\partial \mathbf{h}_j^\top} \right\|_F^2 \le k_i \left(\frac{L_\sigma \|W\|_F}{k_i} \right)^2 = \frac{(L_\sigma \|W\|_F)^2}{k_i}. \tag{18}$$

Step 4: Combine Terms for the Frobenius Norm
Combining Equation 16 and 18, we then have the Frobenius norm following the constraint:

$$\|\mathbf{J}\|_F^2 \le (L_\sigma \|W\|_F)^2 \left(1 + \frac{1}{k_i}\right). \tag{19}$$

Taking the square root:

$$\|\mathbf{J}\|_F \le L_\sigma \|W\|_F \sqrt{1 + \frac{1}{k_i}}. \tag{20}$$

Step 5: Frobenius Norm Bound to Lipschitz Constant
Since the Lipschitz constant L is the supremum of the spectral norm of the Jacobian over all inputs $\|\mathbf{J}\|_2$, and the spectral norm is always bounded above by the Frobenius norm $\|\mathbf{J}\|_F$, any upper bound on the Frobenius norm immediately provides an upper bound on L:

$$L \le L_\sigma \|W\|_F \sqrt{1 + \frac{1}{k_i}}. \tag{21}$$

□

E. Proof of the Three Conditions for MCI

We define $p_b \in (0, 1)$ as the baseline AUC and c as the absolute change in AUC. A valid (p_b, c) satisfies $p_b + c \in (0, 1)$. We define MCI as:

$$\text{MCI}(p_b, c) = \log\left(\frac{1 - p_b}{1 - p_b - c}\right). \tag{22}$$

We now prove that the design of MCI satisfies three conditions derived from three natural marginal utility comparison cases: (i) Under the same baseline performance (p_b), the larger the new performance is, the higher the cost is; (ii) Under the same ultimate performance ($p_b + c$), the lower the baseline performance is, the higher the cost is; (iii) Under the same absolute change in performance, the lower the baseline performance, the higher the cost required or lost (i.e., |MCI| is larger).

Proof. It is trivial to show that with the same p_b, the larger c is, the higher the value of MCI is, satisfying condition (i). The definition of MCI can be reformulated as:

$$\text{MCI}(p_b, c) = \log\left(\frac{1 - p_b}{1 - (p_b + c)}\right).$$

From the above equation, it is trivial to tell that with the same $p_b + c$, the smaller p_b is, the larger the MCI is. Therefore, condition (ii) is satisfied.

We now prove that MCI satisfies condition (iii). Let $p_b^1, p_b^2 \in (0, 1)$ be two baseline AUCs satisfying $p_b^1 < p_b^2$. Also consider a value for c such that $p_b^1 + c, p_b^2 + c \in (0, 1)$. Therefore, (p_b^1, c) and (p_b^2, c) are valid pairs for MCI calculation. The difference between $\text{MCI}(p_b^1, c)$ and $\text{MCI}(p_b^2, c)$ can be formulated as:

$$\text{MCI}(p_b^1, c) - \text{MCI}(p_b^2, c) = \log \frac{(1 - p_b^2)(1 - p_b^1 - c)}{(1 - p_b^1)(1 - p_b^2 - c)}. \tag{23}$$

Subtracting the numerator with the denominator, we have:

$$(1-p_b^2)(1-p_b^1-c)-(1-p_b^1)(1-p_b^2-c)=c(p_b^2-p_b^1).$$

Case 1: performance improvement with $c>0$:
Since $c>0$ and $p_b^2>p_b^1$, we know $0<c(p_b^2-p_b^1)<1$, which means the numerator is larger than the denominator in Equation 23. Therefore, we have:

$$\begin{aligned}&\log\left(\frac{(1-p_b^2)(1-p_b^1-c)}{(1-p_b^1)(1-p_b^2-c)}\right)>0\\ \Rightarrow\quad &\text{MCI}(p_b^1,c)-\text{MCI}(p_b^2,c)>0\\ \Rightarrow\quad &\text{MCI}(p_b^1,c)>\text{MCI}(p_b^2,c).\end{aligned}$$

Since we also have $p_b^{1,2}+c>p_b^{1,2}$, which means

$$\text{MCI}(p_b^{1,2},c)=\log\left(\frac{1-p_b^{1,2}}{1-(p_b^{1,2}+c)}\right)>0.$$

Therefore, we have:

$$|\text{MCI}(p_b^1,c)|>|\text{MCI}(p_b^2,c)|,$$

indicating condition (iii) is satisfied in this case.

Case 2: performance decrement with $c<0$:
Since $c<0$ and $p_b^2>p_b^1$, we know $c(p_b^2-p_b^1)<0$, which means the numerator is smaller than the denominator in Equation 23. Therefore, we have:

$$\begin{aligned}&\log\frac{(1-p_b^2)(1-p_b^1-c)}{(1-p_b^1)(1-p_b^2-c)}<0\\ \Rightarrow\quad &\text{MCI}(p_b^1,c)-\text{MCI}(p_b^2,c)<0\\ \Rightarrow\quad &\text{MCI}(p_b^1,c)<\text{MCI}(p_b^2,c).\end{aligned}$$

Since we also have $p_b^{1,2}+c<p_b^{1,2}$, which means

$$\text{MCI}(p_b^{1,2},c)=\log\left(\frac{1-p_b^{1,2}}{1-(p_b^{1,2}+c)}\right)<0.$$

Therefore, we have:

$$|\text{MCI}(p_b^1,c)|>|\text{MCI}(p_b^2,c)|,$$

indicating condition (iii) is satisfied in this case.

□