A The Algorithm of CPTPP

The complete training procedure of CPTPP is illustrated by Algorithm 1. We first initialise the user and item embedding tables (line 2). Then, we apply a GCL model to conducting embedding pre-training (line 4 ~ 7). Next, we step into the prompt-tuning phase and assign the pre-trained embeddings to $\mathbf{U}_E^*$ and $\mathbf{I}_E^*$ (line 9). Following, we input the user profile to the prompt generator to produce the personalised prompts (line 12) and combine them with $\mathbf{U}_E^*$ (line 13). Finally, we use $\mathbf{U}_E^*$ and $\mathbf{I}_E^*$ to calculate the loss and update them accordingly (ling 14 ~ 15). The update procedure will repeat until the termination condition is achieved (line 11 ~ 17).

Algorithm 1: CPTPP algorithm

Input: User embedding table $\mathbf{U}_E$; Item embedding table $\mathbf{I}_E$; User-item interaction graph adjacency matrix $\mathbf{A}$; Graph contrastive learning model $f(*)$; User profile $\mathbf{X}^u$; Prompt generator $g(\cdot)$; Multi-layer perceptron $\texttt{MLP}\ (\cdot)$; Pre-train epoch i; Prompt-tune epoch j.

Output: User and item embedding tables $\mathbf{U}_E^*$ and $\mathbf{I}_E^*$.

1 *Pre-train phase:*
2 *Initialize* $\mathbf{U}_E$, $\mathbf{I}_E$; $\mathbf{U}_E^{'}$, $\mathbf{I}_E^{'} \leftarrow \mathbf{U}_E$, $\mathbf{I}_E$;
3 $count = 0$;
4 **while** $count < i$ **do**
    `// Update user and item embedding tables.`
5 $\mathbf{U}_E^{'}, \mathbf{I}_E^{'} = f(\mathbf{U}_E^{'}; \mathbf{I}_E^{'}; \mathbf{A})$;
6 $count = count + 1$;
7 **end**
8 *Prompt-tune phase:*
9 $\mathbf{U}_E^* \leftarrow \mathbf{U}_E^{'}$; $\mathbf{I}_E^* \leftarrow \mathbf{I}_E^{'}$;
10 $count = 0$;
11 **while** $count < j$ **do**
    `// Personalized prompt generation.`
12 $\mathbf{P}^u = g(\mathbf{X}^u)$;
    `// Concatenate & fusion.`
13 $\mathbf{U}_E^* = \texttt{MLP}([\mathbf{P}^u; \mathbf{U}_E^*]^T) \in \mathbb{R}^{n \times d}$;
14 *Optimise* $\mathcal{L} = \sum_{i \in \mathcal{U}} \mathcal{L}_{rec}^i + \lambda ||\Theta||_2^2$;
15 *Update* $\mathbf{U}_E^*$, $\mathbf{I}_E^*$;
16 $count = count + 1$;
17 **end**
18 **return** $\mathbf{U}_E^*$, $\mathbf{I}_E^*$

B Reproducibility

This section provides supplementary details about our experimental settings for reproducibility.

B.1 Datasets

Three publicly available datasets are used in this work to examine the performance of the proposed CPTPP. The detailed statistics about the three datasets are listed in Table 2.

Table 2: Dataset Statistics

Dataset	#Users	#Items	#Interactions	Density
Douban	2,848	39,586	894,887	0.794%
ML-1M	6,040	3,900	1,000,209	4.246%
Gowalla	29,858	40,981	1,027,370	0.084%

Here, we provide the links for downloading these datasets for readers to retrieve, which are as follows:

- **Douban**: https://pan.baidu.com/s/1hrJP6rq#list/path=%2F
- **ML-1M**: https://grouplens.org/datasets/movielens/1m/
- **Gowalla**: https://github.com/kuandeng/LightGCN/tree/master/Data/gowalla

B.2 Baselines

We select several baselines for comparison experiments, and a brief introduction to them is listed below:

- **BPR-MF** [10] adopts a matrix factorization framework to learn embeddings for users and items via optimizing the BPR loss function.
- **BUIR** [11] only uses positive user-item interactions to learn representations following a bootstrapped manner, consisting of an online encoder and a target encoder.
- **SelfCF** [33] follows a similar strategy that BUIR adopts, which drops the momentum encoder to simplify the method.
- **NCL** [15] utilizes neighbour clustering to enhance GCL methods to acquire enhanced embeddings for users and items in the recommendation system.
- **SimGCL** [30] discusses the role of augmentations in GCL for recommendation tasks and proposes a simplified GCL method for recommendations.

B.3 Hyper-parameter Settings

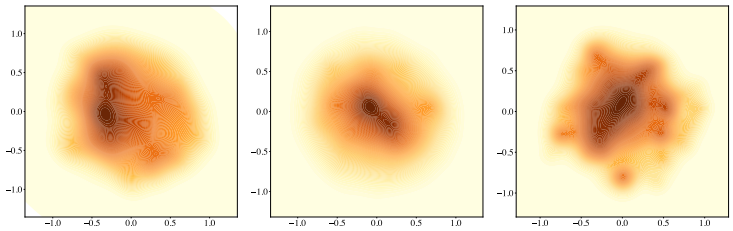
Table 3: Summary of hyper-parameter settings of CPTPP.

Hyper-parameter	Notation	Dataset		
		Douban	ML-1M	Gowalla
Hidden dimension size	d	64	64	64
Pre-train epoch	-	10	10	10
Prompt-tune epoch	-	100	100	100
Batch size	-	512	512	2048
Learning rate	-	0.003	0.001	0.001
Regularizer weight	λ	0.0001	0.0001	0.0001
Number of GNN layers	-	2	2	2
Dropout rate	-	0.1	0.1	0.1
Temperature parameter	τ	0.2	0.2	0.2
Prompt size	p	{8, 16, 32, 64, 128, 256}	{8, 16, 32, 64, 128, 256}	{8, 16, 32, 64, 128, 256}

We list detailed hyper-parameter settings here for reproducibility. The dimensionality of the representation embeddings of users and items is set to 64, and the personalized prompt size is chosen from $\{8, 16, 32, 64, 128\}$. For the pre-train phase, the maximum training epoch is 10, and for the prompt-tune stage, the training epoch is set to 100. The training batch size is 512 for the relatively smaller datasets, including Douban and ML-1M. For Gowalla, it is set to 2048. The learning rate and λ are set to $1e^{-3}$ and $1e^{-4}$, where λ is the weight for the $l2$-norm term in the overall training objective. The default number of layers of graph neural networks used in the models is set to 2. These settings are summarised in Table 3. More details can be found in the source codes by visiting this page: https://anonymous.4open.science/r/CPTPP-F8F4.

C Supplementary Experiment

In this section, several supplementary experiments are provided. Due to the page limit, the supplementary experiment results are listed and analyzed in the following sections.



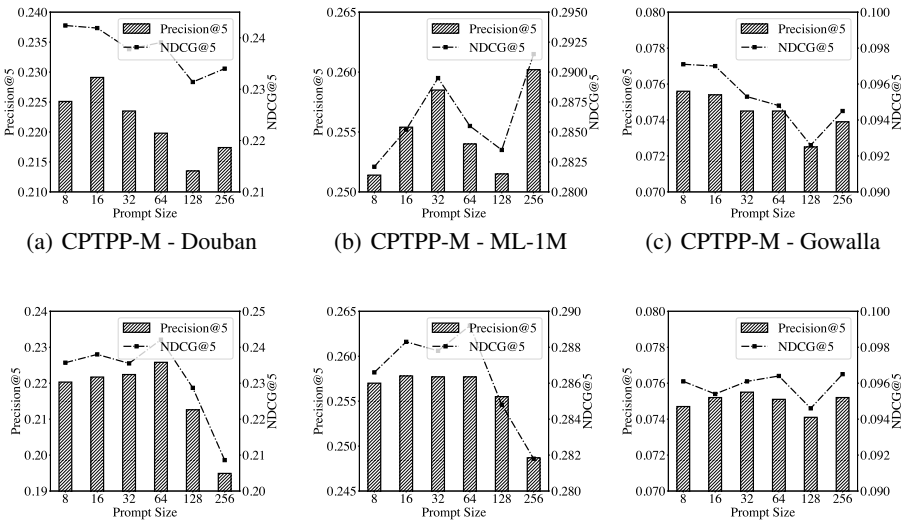
(a) NCL - Douban (b) SimGCL - Douban (c) NCL - ML-1M

(d) SimGCL - ML-1M (e) NCL - Gowalla (f) SimGCL - Gowalla

Figure 4: The visualization results of the user embeddings generated by baselines.

C.1 Supplementary Comparison Study

According to Figure 4, We can see that (i) embeddings learned by SimGCL fall into several hot areas on dataset ML-1M, and they are centralized in a small area on datasets Douban and Gowalla. (ii) NCL exhibits better performance as the distribution of the user embeddings expands to a relatively larger area than that of SimGCL. Compared to our proposed method CPTPP, we can observe that CPTPP has a more uniform distribution of the produced user embeddings, illustrated by the uniformity of the colour maps, especially on dataset ML-1M and Gowalla. As suggested in [15], the more uniform the embedding distribution is, the more capability to model the diverse preferences of users the method has, which reflects CPTPP's superiority.



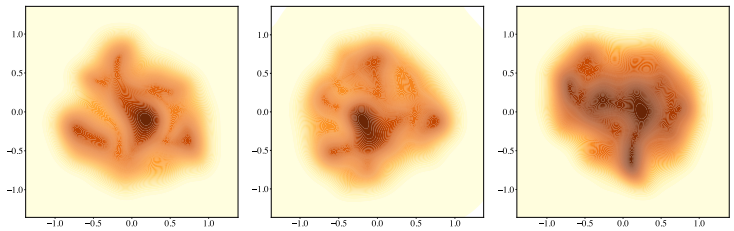
(a) CPTPP-M - Douban (b) CPTPP-M - ML-1M (c) CPTPP-M - Gowalla

(d) CPTPP-R - Douban (e) CPTPP-R - ML-1M (f) CPTPP-R - Gowalla

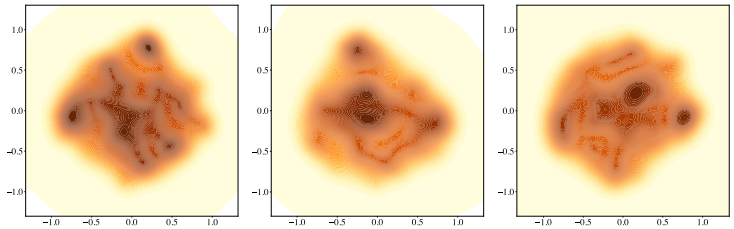
Figure 5: The performance, demonstrated by the metrics *Precision*@5 and *NDCG*@5, of CPTPP-M and CPTPP-R on the selected datasets.

C.2 Supplementary Hyper-Parameter Study

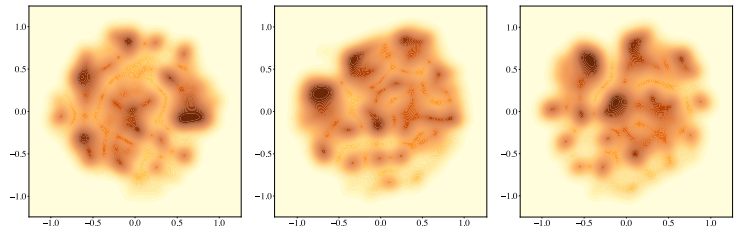
In Section 3.2.2, we solely illustrate the performance of CPTPP-H with different prompt sizes on the three datasets. We show the rest of the hyper-parameter study results in Figure 5. For easy reading, we list our findings here again: (i) The first thing we can observe is that, in most cases, CPTPP has the best performance when the prompt size is not larger than the dimensionality of user embeddings. A potential reason is that sizeable prompt dimensions would introduce more noise into pre-trained user embeddings, disturbing the structural semantics extracted from the user-item interaction graph by graph contrastive learning. (ii) We also notice a significant performance improvement when prompt size is 256 in several cases, such as CPTPP-M on dataset ML-1M and CPTPP-R on dataset Gowalla. However, they still fail to significantly outperform the CPTPP model, which has a much smaller prompt size. Therefore, a small prompt size for prompt-tuning is a better option in practice as they achieve a relatively good recommendation quality and higher efficiency.



(a) CPTPP-H - Douban (b) CPTPP-M - Douban (c) CPTPP-R - Douban



(d) CPTPP-H - ML-1M (e) CPTPP-M - ML-1M (f) CPTPP-R - ML-1M



(g) CPTPP-H - Gowalla (h) CPTPP-M - Gowalla (i) CPTPP-R - Gowalla

Figure 6: The visualisations of the user embeddings generated by different versions of CPTPP.

C.3 Supplementary Ablation study

The impacts of different personalized prompts on CPTPP are investigated. We visualise the user embeddings produced by all three variations of the proposed CPTPP as shown in Figure 6. We can observe that both CPTPP-H and CPTPP-R have a more uniform distribution, especially on datasets Douban and ML-1M. Such an observation indicates that personalised prompts generated from trainable user profiles can produce user embeddings that have more uniform distributions to demonstrate diverse user preferences better.