# HGPNet: Hierarchical Granularity based Personalized Recommendation Network for Auxiliary Product Sales

Maolei Huang , Yao Yu, Detao Lv, Manwei Li, Huawei Cao, Shuhan Song*
Fliggy Alibaba Group, ICT, UCAS

August 9–13, 2026
KDD2026
Jeju, Korea

## Background

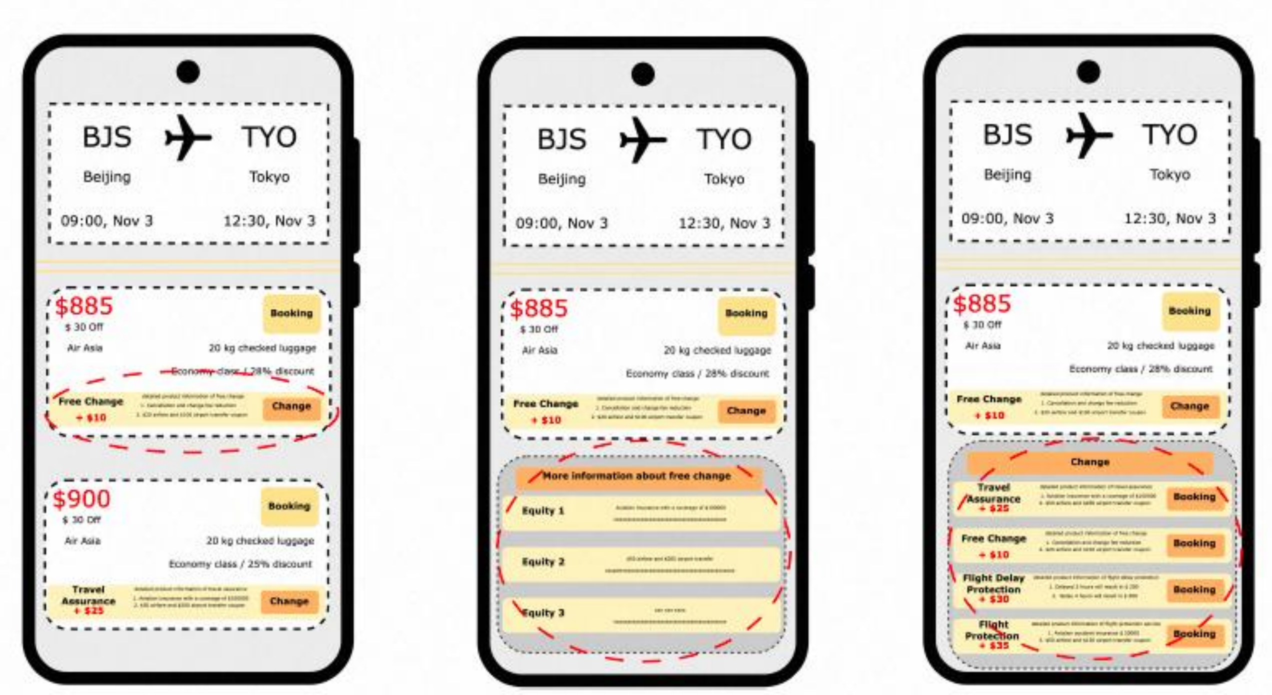


**Figure 1: Application page of Auxiliary Product (Left: The auxiliary product bundling interface; Middle: The auxiliary product details interface; Right: The auxiliary product replacement interface).**

With the rapid advancement of the tourism sector, Online Travel Platforms (OTPs) like Fliggy have seen the emergence of a diverse array of travel-related auxiliary products (e.g., flight delay protection, travel assurance, flexible change policies). These auxiliary products have become a significant revenue stream. Recommendation mechanisms play a crucial role in presenting these related product options to users during their primary travel service purchases. The success of these recommendations is often measured by Gross Merchandise Volume (GMV), which is influenced by predicted Click-Through Rate (CTR) and Conversion Rate (CVR). Optimization often involves balancing GMV and bundling rate, and considering the hierarchical granularity of products within the recommendation system.

## Challenge

### LIMITATIONS

Despite the importance of personalized auxiliary product recommendations, several limitations exist in current approaches:

1. **Sparse User Engagement:** A substantial proportion of users either do not interact with auxiliary products or exhibit sparse engagement, hindering the comprehensive modeling of individual user preferences.
2. **Single-Level Granularity:** Prior research predominantly represents items at a single level of granularity, overlooking the complex relationships and dependencies that may exist across multiple granularities (e.g., different levels within a hierarchical product structure like "xitem" and "skus").
3. **Ineffective Mutual Information Modeling:** Existing approaches have yet to effectively address the potential interference arising from mutual information between different items at a given granularity, which can impact the accuracy and robustness of recommendation models.

## Architecture Design

### Main Module

The main module employs a dual-tower network to predict click-through rates (CTR) for xitem and conversion rates (CVR) for sku. It addresses three inter-related sub-tasks for pXitem, pSku, and pXitemSku.

$$\widehat{y_{xitem}} = f_{ctr}(W_{xitem} \cdot (e_u \oplus e_q \oplus e_{\mathcal{H}} \oplus e_{\mathcal{C}}) + b_{xitem})$$

$$\mathcal{L}_{xitem} = -y_{xitem} \log(\widehat{y_{xitem}}) - (1 - y_{xitem}) \log(1 - \widehat{y_{xitem}})$$

$$\widehat{y_{sku}} = f_{cvr}(W_{sku} \cdot (e_u \oplus e_q \oplus e_{\mathcal{H}} \oplus e_{\mathcal{C}}) + b_{sku})$$

$$\mathcal{L}_{sku} = -y_{sku} \log(\widehat{y_{sku}}) - (1 - y_{sku}) \log(1 - \widehat{y_{sku}})$$

$$\mathcal{L}_{main} = \lambda_1 \mathcal{L}_{xitem} + \lambda_2 \mathcal{L}_{sku}$$

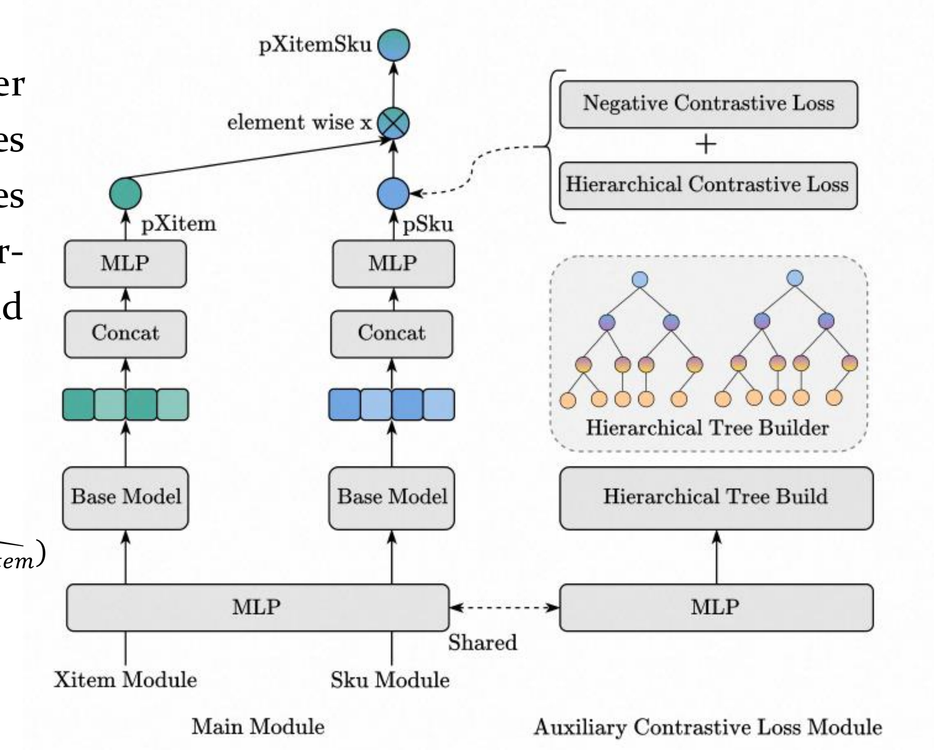


**Figure 2: Framework of the HGPNet model.**

### Auxiliarv Module

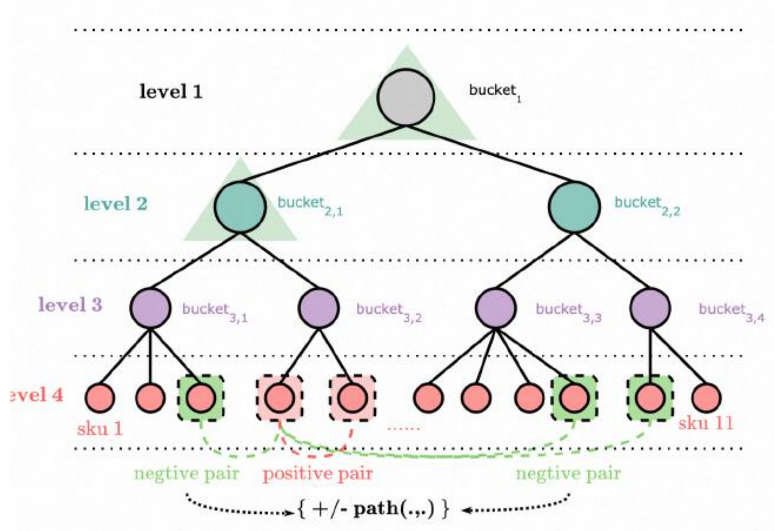


**Figure 3: The Structure of Bucket hierarchical Tree.**

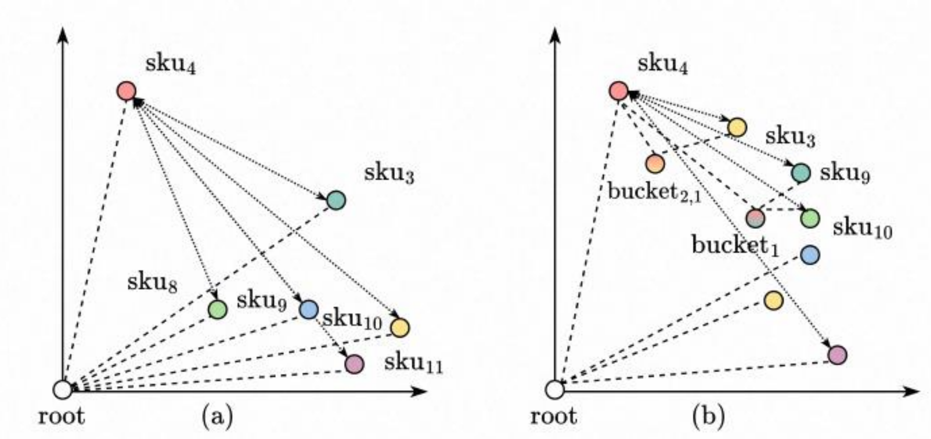


**Figure 4: (a) shows traditional contrastive learning, which treats all negatives uniformly. (b) incorporates hierarchical relationships to adjust distances across levels.**

### Bucket Hierarchical Tree & Hierarchical Contrastive Loss

The Hierarchical Contrastive Loss (HCL) aims to mitigate semantic overlap between SKUs and enhance shared semantic information within a bucket hierarchical tree. It first defines a `path` function to identify the nearest common parent `bucket` for any given pair of SKUs $(sku_i, sku_j)$. To reduce semantic overlap, the representation of this common ancestor $e_{\text{bucket}_k}$ is subtracted from the SKU representations to derive an adjusted similarity score s-. Conversely, to emphasize shared characteristics, the common ancestor representation $e_{\text{bucket}_k}$ is added to the SKU representations, resulting in an enhanced similarity score s+. These two strategies collectively form the basis for the HCL.

$$\begin{aligned}\textbf{path}(sku_i, sku_j) &= \text{Nearest bucket}_k\\ s.t.\textbf{way}(\text{bukcet}_k, sku_i) &= 1\\ \textbf{way}(\text{bukcet}_k, sku_j) &= 1\end{aligned}$$

$$s^-(e_{sku_i}, e_{sku_j}) = \frac{(e_{sku_i} - e_{bucket_k})^T (e_{sku_j} - e_{bucket_k})}{||e_{sku_i} - e_{bucket_k}||||e_{sku_j} - e_{bucket_k}||}$$

$$s^+(e_{sku_i}, e_{sku_j}) = \frac{(e_{sku_i} + e_{bucket_k})^T (e_{sku_j} + e_{bucket_k})}{||e_{sku_i} + e_{bucket_k}||||e_{sku_j} + e_{bucket_k}||}$$

In our scenario, to eliminate the semantic influence of product types (xitems bucketing), we employ the similarity measure s- conversely, to amplify the discriminative signal of key price attributes, we adopt s+ for price bucketing.

$$\mathcal{L}^{+/-} = -\sum_{i \in batch} \log \frac{\exp(s^{+/-}(e_{sku_i}, e_{sku'_i})/\tau)}{\sum_{j \in batch} \exp(s^{+/-}(e_{sku_i}, e_{sku_j})/\tau)}$$

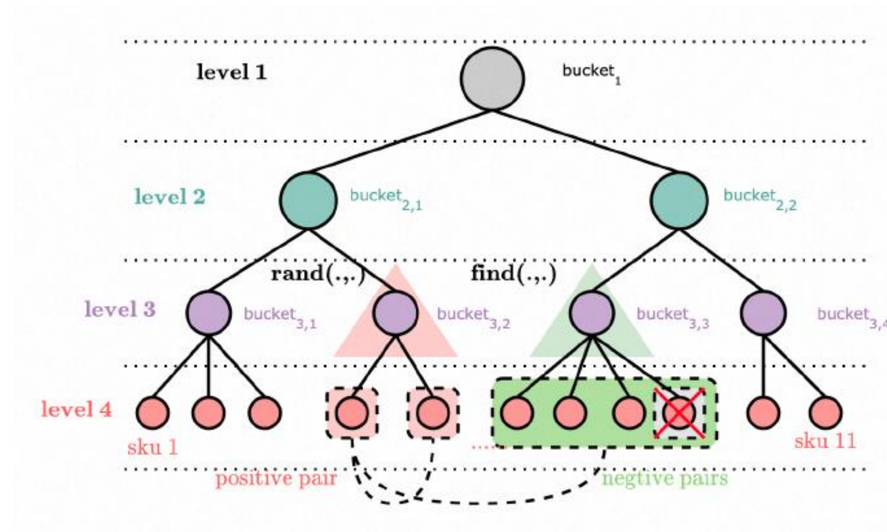


**Figure 5: Examples of Negative Contrastive Loss.**

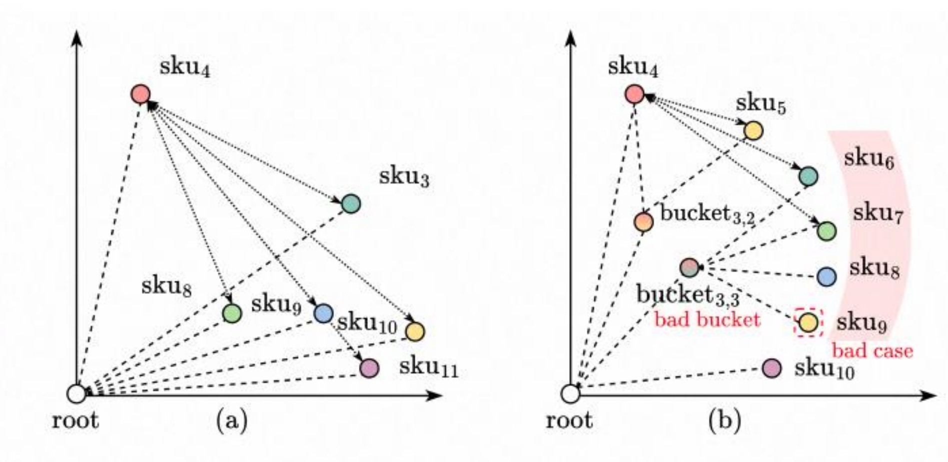


**Figure 6: (a) shows traditional contrastive learning, which treats all negatives uniformly. (b) incorporates negative relationships to adjust distances across bad buckets.**

### Negative Contrastive Loss

1) **find(i)**: returns the nearest parent bucket of the leaf node $sku_i$;

$$\begin{aligned}\textbf{find}(sku_i) &= \text{Nearest bucket}_i\\ s.t.\ \textbf{way}(\text{bukcet}(i), sku_i) &= 1\end{aligned} \tag{15}$$

2) **rand(i)**: retrieves an arbitrary sku from a different bucket at the same hierarchical level as the parent bucket of $sku_i$.

$$\begin{aligned}\textbf{rand}(sku_i) &= \text{bucket}'_i\\ s.t.\ \textbf{way}(\text{bucket}'_i, sku_i) &= 0\\ s.t.\ \text{level}(\text{bucket}'_i) &= \text{level}(\text{bucket}_i)\end{aligned} \tag{16}$$

$$s'(e_{sku_i}, e_{sku_j}) = \frac{(e_{sku_i} + e_{\text{gap}}^{\text{bucket}}(i,j))^T (e_{sku_i} + e_{\text{gap}}^{\text{bucket}}(i,j))}{||e_{sku_i} + e_{\text{gap}}^{bucket}(i,j)||||e_{sku_i} + e_{\text{gap}}^{\text{bucket}}(i,j)||}$$

$$e_{\text{gap}}^{\text{bucket}}(i,j) = e_{\text{bucket}_i} - e_{\text{bucket}_j} - e_{neg}$$

$$e_{\text{bucket}_j} \in \text{bucket}'_i$$

$$\mathcal{L}' = -\sum_{i \in batch} \log \frac{\exp(s'(e_{sku_i}, e_{sku'_i})/\tau)}{\sum_{j \in batch} \exp(s'(e_{sku_i}, e_{sku_j})/\tau)}$$

$$\mathcal{L}_{final} = \alpha \mathcal{L}_{main} + \beta \mathcal{L}^{+/-} + \gamma \mathcal{L}'$$

## Offline Evaluation

**Table 2: Results of offline experiments (Best values are i bold; next-best values are underlined).**

| Method | Model | NDCG@10 | NDCG@15 | AUC |
|---|---|---|---|---|
| rule-based | Cheapest | 0.8435 | 0.9146 | 77.15% |
| | Priciest | 0.8357 | 0.9120 | 73.73% |
| | Policies | 0.8558 | 0.9231 | 77.17% |
| single-task | LR | 0.8563 | 0.9301 | 75.71% |
| | GBDT | 0.8661 | 0.9316 | 76.30% |
| | WDL | 0.8697 | 0.9331 | 76.07% |
| | DIN | 0.8791 | 0.9401 | 75.36% |
| | ECAU | 0.8709 | 0.9413 | 75.99% |
| multi-task | ESMM | 0.8801 | 0.9512 | 75.77% |
| | PLE | 0.8817 | 0.9556 | 75.50% |
| ours | HGPNet | **0.8956** | **0.9667** | **78.20%** |

**Table 3: Results of offline experiments.**

| Model | NDCG@10 | NDCG@15 | AUC |
|---|---|---|---|
| w/o HCL | 0.8895 | 0.9603 | 77.78% |
| w/o NCL | 0.8901 | 0.9617 | 77.29% |

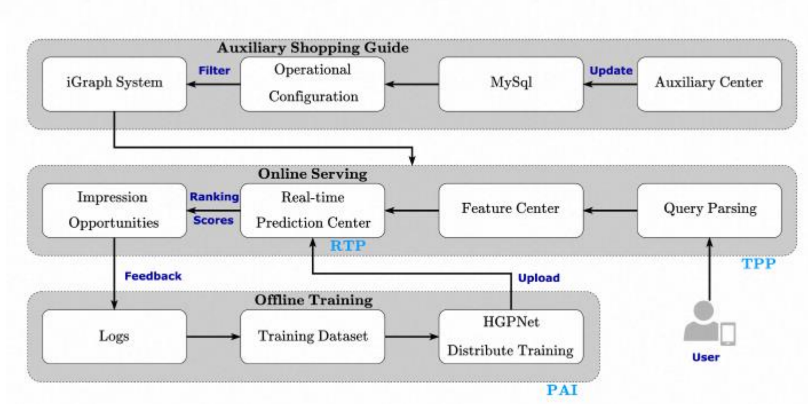


**Figure 7: Illustration of deployment of HGPNet at Fliggy.**

## System Deployment Design & Online A/B Test

**Table 4: Results of offline experiments(Best values are in bold; next-best values are underlined).**

| Model | - | Ticket | Buddle Rate | Revenue |
|---|---|---|---|---|
| rule-based | Cheapest | 8690 | **34.11%** | 143457 |
| | Priciest | 8482 | 32.17% | 153671 |
| | Policies | 8561 | 33.83% | 159764 |
| single-task | LR | 8571 | 33.16% | 163228 |
| | GBDT | 8599 | 33.18% | 161124 |
| | WDL | 8602 | 33.27% | 164378 |
| | DIN | 8611 | 33.22% | 169922 |
| | ECAU | 8671 | 33.32% | 171927 |
| multi-task | ESMM | 8627 | 33.25% | 164635 |
| | PLE | 8699 | 33.41% | 170753 |
| ours | HGPNet | **8766** | 33.32% | **173260** |

**Table 6: Comparison of efficiency of different methods (m-minute; s-second; ms-millisecond).**

| Method | Training Time | Inference Time |
|---|---|---|
| **Cheapest** | - | 2ms |
| **Priciest** | - | 2ms |
| **Policies** | - | 2ms |
| **LR** | 34m19s | 14ms |
| **GBDT** | 34m19s | 14ms |
| **WDL** | 58m28s | 27ms |
| **DIN** | 75m30s | 29ms |
| **ECAU** | 77m46s | 25ms |
| **ESMM** | 77m33s | 26ms |
| **PLE** | 73m28s | 27ms |
| **HGPNet** | 73m30s | 26ms |