ABSTRACT

In e-commerce, a group of similar or complementary products is recommended as a bundle based on the product category. Existing work on modeling bundle recommendations consists of graph-based approaches. In these methods, user-product interactions provide a more personalized experience. Moreover, these approaches require robust user-product interactions and cannot be applied to cold start scenarios. When a new product is launched or for products with limited purchase history, the lack of user-product interactions will render these algorithms inaccessible. Hence, no bundles recommendations will be provided to users for such product categories. These scenarios are frequent for retailers like Target, where much of the stock is seasonal, and new brands are launched throughout the year. This work alleviates this problem by modeling product bundles recommendation as a supervised graph link prediction problem. A graph neural network (GNN) based product bundles recommendation system, BundlesSEAL is presented. First, we build a graph using add-to-cart data and then use BundlesSEAL to predict the link representing bundles relation between products represented as nodes. We also propose a heuristic to identify relevant pairs of products for efficient inference. Further, we also apply BundlesSEAL for predicting the edge weights instead of just link existence. BundlesSEAL based link prediction leads to amelioration of the above-mentioned cold start problem by increasing the coverage of product bundles recommendations in various categories by 50% while achieving a 35% increase in revenue over behavioral baseline. The model was also validated over the Amazon product metadata dataset.

CCS CONCEPTS

• Information systems → Online shopping.

KEYWORDS

link prediction, graph neural networks, bundles recommendation

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

ACM Reference Format:

1 INTRODUCTION

Retailers often bundle products to offer an improved shopping experience to the customers and increase their sales by expanding the order size. The product bundles are highlighted in mannequins or other displays across the stores. Appropriately created bundles can lead to serendipitous, and more satisfying purchases for the customer, often called the ‘Target effect’ [24] in the retail jargon. Digitally, creating this experience requires a careful understanding of the product assortment and user behaviors. These recommendations are often labeled as ‘Frequently Bought Together’ items in the e-stores. For example, in Figure 1, we show an example of the product bundle formed for the car seat. When a user clicks on the car seat, a bundle with a back seat mirror, a support cushion, and a car seat cover is displayed. Depending on the retailer’s understanding of user intent, such recommendations can surface on the product display page when users add items to their cart or even when they initially enter the site. In the rest of the paper, we will call the item clicked by the user as the driver item and all the recommended items in the bundle as the go-together items.

Recommending bundles is different from recommending purely similar or complementary products. Product recommendation is a widely studied research problem, and many works focus on similar or complementary product recommendations [10, 20, 23, 26, 28, 31, 32] forming bundles using either of these relations. However, in practice, product bundles can have both similar and complementary products. For example, Figure 2 shows a bundle for a camping tent with other camping gear such as an air mattress, lantern, and burner (top), while a bundle for baby snacks can have similar snacks with different flavors (bottom). Thus, using just one relation for bundle formation limits the potential of bundling.

Many popular state-of-the-art models for bundles recommendations include personalized product bundles [3, 4, 15, 16, 19, 21, 22, 30, 37]. But they suffer from the cold start problem. They cannot produce satisfactory results for i) users with no historical data and ii) products with less or no purchase data. We need to build a robust system that works even in the cold start situation. Image-based approaches [13, 27] have also been used for bundles recommendation, but extending them to frequently bought together items is not feasible due to sparsely available labeled data.
Another popularly used method for bundles recommendation is by using behavioral signals such as customer co-browse or co-purchases [29]. But identifying the correct driver item leading to the purchase of relevant go-together items is challenging. In the car seat example in Figure 1, a customer looking for a car seat may likely find purchasing a back seat mirror helpful, or a customer purchasing an iPhone will find it bundled with a screen protector convenient. On the other hand, if a customer is looking for a screen protector, then a recommendation containing a phone might seem exasperating. Since co-purchase data lacks driver-item information, it is difficult to overcome such futile recommendations using them.

Bundles are also recommended based on association rules [6] but on applying the same to our co-purchase data we observed low or no coverage across categories. The non-empty (at least one product) in the recommendation are considered covered, and the proportion of products with non-empty bundles recommendation out of all the products in the category is defined as the coverage. In the association rule based methods, newly introduced items can be represented as a collection of category-specific product graphs, \( \{ (V_c, E_c) \} \in \text{Catalog} \). For example, if the catalog contains Baby, Beauty, Furniture, and Electronics product categories, it can be represented as

\[
\{ G_{\text{Baby}}, G_{\text{Beauty}}, G_{\text{Furniture}}, G_{\text{Electronics}} \}
\]

In the first step, for each category, the graph \( G_c \) consisting of \((V_c, E_c)\) is created. Nodes \( V_c \) represent products in the category \( c \), and edges \( E_c \) represent bundles relationship between the products.

The key contributions of this paper are as follows:

- We introduce a novel approach of forming a directed product graph by considering the sequence in which products are added to the cart by users in a session. Adding directed edges helps us identify the driver and go-together items in the bundles.
- We resolve the coverage issue by formulating bundles recommendation as a supervised link prediction problem in the directed product graph.
- We compare the performance of both links existence prediction and link weight prediction for bundles recommendation. We also evaluate how the model performs under different scenarios of initial node representation by incorporating i) BERT embedding of the product description, ii) pre-trained node embedding based on the graph structure, and iii) different node labeling techniques.
- We deployed this framework in production using a novel heuristic for selecting edges for inference and observed a 50% increase in coverage and a 35% increase in revenue.

The rest of the paper is organized as follows: Section 2 provides our proposed approach. Section 3 provides the details of the experiments, while Section 4 concludes the work with directions for future work.

## 2 PROPOSED METHOD

In this section, we describe the steps for building the bundles recommendation system BundlesSEAL. First, we build a graph for a category by representing the products in the category as nodes and the directed edges representing the bundles relation between pairs of products. Then we train a GNN model for link prediction on this graph. Once the model is trained, we use a heuristic to select the edges for bundle link prediction. We also train and apply a GNN model for edge weight prediction. Figure 3 represents the overall architecture.

### 2.1 Bundles Product Graph Creation

The catalog, Catalog, is represented as a collection of category-specific product graphs, \( \{ (V_c, E_c) \} \in \text{Catalog} \). For example, if the catalog contains Baby, Beauty, Furniture, and Electronics product categories, it can be represented as

\[
\{ G_{\text{Baby}}, G_{\text{Beauty}}, G_{\text{Furniture}}, G_{\text{Electronics}} \}
\]

Figure 1: Sample example of a product bundle presented to a customer intending to purchase a car seat.

Figure 2: Top - Product bundle of camping gear comprising complementary products. Bottom - Product bundle of baby food comprising similar products.
and the product added to the cart in a session contains an outgoing edge to the same category products added later to the cart in that session. We also assign weight to the edges, which is the count of users who added the source product before the target.

In Figure 4, we show an example of items added to the cart by a customer. The customer has sequentially added four items, t-shirts, pants, a TV, and a TV mount, respectively. Based on this cart, an edge can be formed from TV to the mount in the Electronics category graph, \( (TV, mount) \) \( \in E_{Electronics} \), and an edge from t-shirts to pants in the Kids category graph, \( (t-shirts, pants) \) \( \in E_{Kids} \). We also increment the weight of the edges (TV, mount) and (t-shirts, pants) by one based on this session. All edge weights are initialized by 0.

These edge weights are used to rank all nodes from each source and decide if the destination node will be present in the recommendation of the source node. The higher the weight more likely the destination product should be included in the source product’s bundles recommendation. We normalize edge weight by dividing it by the sum of all the edges’ weight from that edge’s source. We want to include only the significant edges in the graph, so we threshold the edges on an empirically chosen edge weight. If the edge weight is higher than the threshold, we keep that edge; otherwise, we omit it from the graph. The direction of an edge from the source to the destination product means the destination product can be part of the bundles’ recommendation for the source product. In Figure 5, we show the product graph for the recommendations of Figure 1. Each product from Figure 1: car seat, back seat mirror, support cushion, and car seat cover is represented as a node in this graph, and the edges from the car seat to the other three products shows we can recommend the other three items for the car seat.

The bundles product graph is directed. Often, one cannot include the source product as a recommendation for the destination product in bundles formation. In contrast, an undirected edge will denote...
The notion of co-purchase. For example, a co-purchase-based recommendation may recommend a car seat when the user is interested in buying a backseat mirror, but bundles recommendation cannot have a car seat as a recommendation for the backseat mirror. If a customer is checking the backseat mirror, it is highly likely that he has already chosen a car seat and is now checking the backseat mirror that goes along with the car seat.

As part of the business rule requirements, we filter the edges further in the product graph based on the price. To ensure the bundles’ recommendations do not include significantly costlier items than the source product, we keep only those edges in the product graph that satisfy the condition mentioned below

\[ p_S \geq p_R - \epsilon \]

where \( p_S \) is the price of the source item, \( p_R \) is the price of the recommended item and \( \epsilon \) is a small variable greater than zero.

Each category graph can have several hundred thousand nodes in practice depending on the number of products in the catalog under that category. However, creating such product graphs by behavioral data mining results in a very sparse graph. Only a small fraction of products in the graph have edges based on add-to-cart edges. The distribution of coverage in the product graphs for different categories is shown in Figure 6. We observe that the coverage is extremely small for some categories, even less than 10%. We propose to resolve the sparsity issue and improve the recommendations coverage by predicting links between nodes.

### 2.2 Graph Neural Networks for Link Prediction

Most retailers fix the number of products to form the bundles, let \(|\text{Bundles}| = r\). This allows for a uniform user interaction experience across the products for the guests. For example, our business rules allow for at most three recommended items in the bundles. Thus we should have three outgoing links from each source product in a perfect bundles graph. However, we have low product coverage and have many products without any edge and less than three outgoing edges. We train a graph neural network model for link prediction to complete the graph. To build and compare the predictors, we consider the links in the product graph as true positive samples. We divide the existing edges in the product graph in an 8:2 ratio for training and testing. We build different predictors by learning from the training split of positive edges and an equal number of randomly sampled non-existing links as negative edges. The number of edges in a perfectly complete product bundles graph will contain the number of bundles recommendations from each product. Considering \(N\) products in the graph, the perfect product bundle graph will have \(r \times N\) edges, while the total number of edges is in \(O(N^2)\). Since \(r\) is a small constant, the perfect bundles graph inherently contains very few links than the total number of edges. So, using non-existing links as negatives will induce an almost insignificant error concerning the impact of the problem.

We want the predictors to recommend the most relevant products and form edges, especially for products that do not already have edges. So, we only test those edges from the positive test subset that isolates the source product in the graph after removing the test edge. We compare each selected test edge’s predicted score with all the edges from the source to all connected nodes in the product graph for evaluating predictors’ performance. We consider that the graph contains only the positive training edges during evaluation and the connected nodes are those nodes that have at least one edge in the positive training edges. We compare the score for selected test edges only with the edges from the connected subgraph because the products in the connected subgraph are the popular items, and including them in bundles increases the recommendation value.

We leverage the state-of-the-art GNN framework SEAL [35] for predicting the links and capturing potential bundle relations. We use Graph Isomorphism Network (GIN) [33], \(G\), as the base GNN architecture in the SEAL framework. In section 3.1 we explain further the rationale behind selecting GIN. For a graph, \(G(V, E)\), \(X\) is the node feature matrix representing the features of each node \(V\) in the graph. The input to the GIN model is \(X\) and graph edges (positive labels) along with the sampled negative edges, \(E \cup E_{\text{neg}}\).

The GIN model computes the node representation for each node in the graph and takes the global mean pool, \(\hat{y}_i\), as represented in the equation 1.

\[
\hat{y}_i = \text{MLP}(\text{GMP}(G(X, E \cup E_{\text{neg}})))
\]  

(1)

We formulate the link prediction task in two ways: link existence prediction (classification) and link weight prediction (regression) and compare their effectiveness in improving coverage. For the binary classification task, the true label, \(y_i\), is assigned 1 for all the positive training edges and 0 for the randomly sampled negative edges. We obtain the probability of the link existence by applying sigmoid to the model output, and the predictors are trained using binary cross-entropy loss in this task stated in equation 2. For the regression task, we use the edge weights for the positive training samples and 0 edge weights for the randomly selected negative training samples. In this task, the predictors are trained to predict the edge weights based on the input using mean squared error loss, expressed in equation 3. We compare the predicted probability and edge weights, respectively, for model evaluation. We pick those edges with a predicted score (probability or weight) above a threshold and form bundles containing at most \(r\) items from each source having the top \(r\) predicted score from that source.

\[
\text{loss} = \sum_{i=1}^{n} -y_i \log(\sigma(y_i)) - (1 - \sigma(y_i)) \log(1 - \sigma(\hat{y}_i))
\]

(2)
The SEAL framework (github.com/muhanzhang/SEAL), shown in Figure 7, is a GNN based approach that takes node attributes and the subgraph information as input and predicts the probability of the link existence. There are three types of inputs (i) node labels, (ii) node embedding based on the graph structure and (iii) product attributes that could be direct features or pre-trained embeddings.

The node labels include subgraph information defined using different node labeling techniques. SEAL framework originally uses the Double Radius Node Labeling (DRNL) technique. In DRNL both source and destination nodes of an edge are labeled as 1. In this node labeling technique, the node labels for all the other nodes in the subgraph (n-hop neighbors from the source and destination) are based on a hash function that uses the node’s distance from the source and destination. Since the DRNL node label is designed for undirected graphs and considers both source and destination the same, we use another node labeling technique called Distance Encoding (DE). In the DE node labeling technique, we assign a tuple of distance from source and destination as the label to all the nodes. In our modified version of DE, we fix the index for source and destination in the tuple and assign different labels for the source and destination nodes. These modifications ensure that the differences based on source and destination are stored in the node labels. We use the shortest path distance for weighted directed graphs to calculate the modified DE labels to take weights and direction into account.

We evaluated the following variations of the inputs:

- **Node labeling techniques**: We experiment with DRNL and modified DE labeling techniques.
- **Node embeddings**: We also create the node embeddings (NE) from node2vec [7] to capture the graph structure. Node embeddings essentially capture the representation of graph/graph topology, hence they are a powerful way of representing the users’ behavior associated with adding items to the cart together.
- **Product attributes**: We use the sub-category of the products and the brand name as the categorical node features (PA). In empirical studies, we found that guests are most sensitive to these attributes when purchasing items from the bundles. Hence we incorporated these features. We also generate the BERT embeddings [5] of the textual description, including the product title (PA-BE). BERT embeddings for a given node are obtained by passing the product title (and description) to a BERT model.

We use these three different input sets and their combinations as input for our predictors for link prediction. All the above enhancements to the SEAL framework, including the regression model for edge weight prediction, modified DE node labeling, and different node features, form the BundlesSEAL. For the GNN model, we tried four different architectures, Graph Convolutional Network (GCN) [14], GraphSAGE (SAGE) [8], Deep Graph Convolution Network (DGCNN) [36] and, Graph Isomorphism Network (GIN) [33].

### 2.3 Efficient Edge Selection for Inference

Once BundelsSEAL is trained, we use it for predicting hidden links. We cannot exhaustively select all pairs of nodes without a link in the original sparse graph since that would be \( O(N^2) \) edges and is practically not feasible in the production environment. Therefore, we apply a heuristic to preselect pairs of nodes for link prediction during inference. At first, we add similarity edges from the bundles graph to the substitutes graph. The substitutes graph was created based on the weighted sum of the similarity score from the product attributes, including text and image embeddings using approximate nearest neighbor algorithm [25]. Now, we have a product graph with both similar and bundles relationship. Next, we select all pairs of nodes for the inference that are separated by two-hop paths where at least one edge represents a bundles relationship. The later condition omits the possibility of selecting a pair of only similar nodes, which represents the substitutes relation and might not represent bundles relation. We illustrate the edge selection...
We choose these three heuristics as they perform well in several open benchmark datasets for link prediction. Table 2 contains the results for binary classification using these three heuristics. We compare four different base GNNs in the SEAL framework: DGCNN, SAGE, GCN, and GIN for link existence prediction. We train and test these classifiers on positive and randomly sampled negative training and test edges, respectively. We evaluate them for link existence prediction using the Area under the ROC Curve (AUC) metric. We use the AUC value to evaluate these classifiers because it is agnostic to the choice of threshold and gives us a measure of separability of the two classes by the model. We run each experiment multiple times and report the mean and the standard deviation across the three runs in this paper.

The heuristics are unable to predict the existence of an edge if the source node is isolated. In Table 2, we observe that the AUC of all the three heuristics is significantly lower than those of GNN based approaches. Among the heuristics, PPR works better than CN and AA. The relatively global information captured by PPR over the local neighborhood in CN and AA could be the reason for the superior performance of PPR. Comparing the AUC mean values in Table 2 suggests the GNN models can predict link existence better than simple heuristic-based methods on internal datasets. Because GNN based models have better link existence prediction capability, we used it to build our model for improving the coverage. We also observe that GIN, the most expressive GNN model, has the best performance among all four base models used for link existence prediction. So, we select GIN as our base for building BundlesSEAL.

Next, we experiment rigorously with the BundlesSEAL framework with four different models 1. DRNL_R, 2. DRNL_C, 3. DE_R, and 4. DE_C. These experiments compare the relative prediction score of selected positive test edges that isolate the source by removing positive test edges with edges between that source and all other nodes in the connected subgraph. Since we want the predictors to predict the probability or weight of the positive test edge higher than the comparison edges, we rank the predicted value for it against other comparison edges and compute the Mean Reciprocal Rank (MRR) to evaluate their performance. We perform hyperparameter tuning using skopt [11] and plot the results as percentage across different evaluations in figure 9.

In figure 9, the results for the classification are represented as DRNL_C and DE_C, and regression are represented as DRNL_R and DE_R for different categories, different types of inputs, and node labeling techniques in BundlesSEAL. We observe in figure 9 that the MRR values are mostly higher for the classification task as compared to regression. This observation suggests that the number of customers adding a product after another does not correlate as well compared to just identifying if customers add a product after another based on the product attributes and graph structure.

We also perform an ablation study on BundlesSEAL with a GIN base to check the impact of different components. We evaluate the performance of the models by comparing different node features: Categorical containing product attributes - PA (sub-category and brand name), BERT embeddings - BE for the textual product description, and Node embeddings - NE generated from node2vec graph embeddings. We create six different input sets: i) without any attributes (WoA), ii) categorical alone containing product attributes (PA), iii) node2vec embeddings (NE), iv) categorical with BERT embeddings (PA-BE), v) categorical with node embeddings

<table>
<thead>
<tr>
<th>Categories</th>
<th>Baby</th>
<th>Beauty</th>
<th>Toys</th>
<th>Arts &amp; Crafts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coverage(%)</td>
<td>27</td>
<td>69</td>
<td>22</td>
<td>24</td>
</tr>
<tr>
<td>Nodes</td>
<td>24780</td>
<td>28895</td>
<td>13035</td>
<td>27089</td>
</tr>
<tr>
<td>Edges</td>
<td>77981</td>
<td>116775</td>
<td>19931</td>
<td>39238</td>
</tr>
</tbody>
</table>

Table 1: Statistics of the dataset for different categories

...
Improving Bundles Recommendation Coverage in Sparse Product Graphs

WWW ’22 Companion, April 25–29, 2022, Virtual Event, Lyon, France

Figure 9: MRR% along y-axis for baby and beauty categories on internal dataset, and toys and arts & crafts categories on Amazon dataset using BundlesSEAL.

<table>
<thead>
<tr>
<th>Category</th>
<th>CN</th>
<th>AA</th>
<th>PPR</th>
<th>DGCNN</th>
<th>SAGE</th>
<th>GCN</th>
<th>GIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baby</td>
<td>93.83 ± 0.29</td>
<td>89.4 ± 1.45</td>
<td>95.36 ± 0.28</td>
<td>99.44 ± 0.07</td>
<td>99.29 ± 0.04</td>
<td>97.81 ± 0.06</td>
<td>99.45 ± 0.02</td>
</tr>
<tr>
<td>Beauty</td>
<td>77.18 ± 0.66</td>
<td>75.15 ± 0.84</td>
<td>87.70 ± 0.17</td>
<td>97.61 ± 0.51</td>
<td>97.14 ± 0.41</td>
<td>94.00 ± 0.22</td>
<td>97.68 ± 0.17</td>
</tr>
</tbody>
</table>

Table 2: AUC mean and standard deviation for binary classification

(PA-NE), and vi) categorical with both BERT and node embeddings (PA-NE-BE).

The following key observations can be made from Figure 9:

- Based on the results of the classification models, we learn that the models without node attributes (WoA) containing the sub-graph information have comparable predictive values.
- We observe that the product attributes alone (PA) usually perform better than models without any node attributes (WoA), suggesting that node attributes containing product sub-category and brand name with the sub-graph information help in better bundle formation than just the sub-graph information.
- Similar to the results obtained in [35], the performance of BundlesSEAL does not always improve with the inclusion of node embeddings. Node embeddings represent the behavioral graph structure/patterns and hence adding them doesn’t always provide extra information to the models over the WoA models. We observed that the performance is similar to the WoA models. In some categories, the performance even degraded over the WoA models, which may be attributed to added complexity in adding node embeddings. Running over more epochs might converge the models with node embeddings to the same performance as the WoA models.
- The inclusion of BERT embeddings for nodes improves the performance of almost all models. This implies that product description shares a good clue about product attributes and hence suggests bundling products. We tried using the off-the-shelf BERT-small model in this experiment. It may be worth trying other natural language models, pretrained on our product catalog.
- We tried DRNL and modified DE node labeling techniques in these experiments, as explained before. We observe that the results in most cases do not improve by using the modified DE node labeling technique as compared to DRNL. This comparison suggests that the inclusion of direction and weights in the node labeling technique does not help boost the results of BundlesSEAL further.

3.2 Deployment and real-world experiments

The online performance of BundlesSEAL is measured using A/B tests. We deployed the best offline performing model to production for the online evaluation on four categories: Baby, Grocery, Home, and Beauty. The current recommendations are produced offline on our GPU framework based on the predictions on the selected edges using the strategy described in the section 2.3 and exported to the cloud layer to be cached and served in real-time. We use the average percentage increase in revenue across these categories as the online evaluation metric. An increase in revenue is a direct measure of improvement in the business by adopting the new model [12].

The baseline model in this study is a legacy model based on behavioral co-purchase data. In this baseline model, a frequent co-occurring itemset in the basket was mined using FP-trees without
We compare the performance of the legacy model and bundles recommendations rendered from the BundlesGraphs (without predicting new links using GNN). Next, we compare the performance of the BundlesGraphs with the best offline performing BundlesSEAL, GIN classifier with DRNL node labeling, and product attributes and BERT embeddings defining the node features. The attributable demand is measured as the revenue attributed to a recommendation shown to a user when the user clicks on one of the given recommendations, adds it to the cart, and purchases it within the same session.

As shown in Table 3, we can see that by rendering bundles recommendations using the BundlesGraph, there is a significant increase in the revenue of 15% over the baseline. Further, by predicting new links using BundlesSEAL, we observed an increase in the revenue by another 20%. Thus, by using BundlesSEAL, Bundles graph built using add-to-cart data and link prediction using GNN, we get an improvement of 35% over the behavioral baseline model based on co-purchase. We also observe an increase in coverage of 50% by using BundlesSEAL. All the results were statistically significant with p-value <0.05. This confirms the results observed in the offline test and the effectiveness of BundlesSEAL in recommending products for the bundles formation.

### 3.3 Case Study

In this section, we discuss some screenshots of product bundles (available on the website) associated with the launch of a self-owned brand, Brightroom’s launch in January 2022. When new products are launched, the lack of customer shopping behavior associated with them means that we can’t predict bundles using BundlesGraph. However, using BundlesSeal, we are able to generate relevant bundles recommendations. Over a few days, as add-to-cart data becomes available for different products, the bundles relationships are known and not inferred or predicted. Figure 10 gives snapshots of a few bundle recommendations after a few days of launch. It can be seen that bundlesSEAL (green box) gives a significant coverage boost for such newly launched items. The green box recommendations are relevant and also comprise items from other popular brands at Target.

### 4 CONCLUSION AND FUTURE WORK

In this paper, we propose a novel graph based approach for product bundle recommendation by mining the sequence in which customers add products to their cart in a session. We observe in A/B tests that this add-to-cart sequence behavior based recommendation provides a lift of 15% in revenue over purely co-purchase behavior based recommendations. Since behavior based data is sparse, we apply a graph neural network model BundlesSEAL to predict new edges or recommendations for products which had no coverage in the add-to-cart behavior based data. We also propose a heuristic that helps to apply BundlesSEAL for inferencing in a scalable way and thereby expand the coverage of recommendation. This approach has led to an increase in the number of products which have bundle recommendation by nearly 50% and has boosted the revenue in A/B tests by 35%. Our work showcases how GNNs can have an impact on boosting the performance of real world recommender systems. In the future, we would like to compare BundlesSEAL framework with other existing product bundle recommendation systems.

### ACKNOWLEDGEMENTS

The authors would like to thank the following individuals from Target for their insight and help in shaping this project: Ashes Das, Gregory Olsen, Farhan Iqbal, Anupama Joshi, and Janet Keel.
REFERENCES


[9] Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining frequent patterns without candidate generation. ACM sigmod record 29, 2 (2000), 1–12.


