Large Scale Product Graph Construction for Recommendation in E-commerce

Xiaoyong Yang† Yadong Zhu†* Yi Zhang‡ Xiaobo Wang† Quan Yuan†
† Alibaba Inc, Beijing, China
‡ University of California, Santa Cruz, USA
{xiaoyong.yxy, edgewind.zyd, yongshu.wxb, yuanquan.yq}@alibaba-inc.com
{yiz}@soe.ucsc.edu

ABSTRACT

Building a recommendation system that serves billions of users on daily basis is a challenging problem, as the system needs to make astronomical number of predictions per second based on real-time user behaviors with O(1) time complexity. Such kind of large scale recommendation systems usually rely heavily on pre-built index of products to speedup the recommendation service so that the waiting time of online user is un-noticeable. One important indexing structure is the product-product index, where one can retrieval a list of ranked products given a seed product. The index can be viewed as a weighted product-product graph.

In this paper, we present our novel technologies to efficiently build such kind of indexed product graphs. In particular, we propose the Swing algorithm to capture the substitute relationships between products, which can utilize the substructures of user-item click bi-partitive graph. Then we propose the Surprise algorithm for the modeling of complementary product relationships, which utilizes product category information and solves the sparsity problem of user co-purchasing graph via clustering technique. Base on these two approaches, we can build the basis product graph for recommendation in Taobao. The approaches are evaluated comprehensively with both offline and online experiments, and the results demonstrate the effectiveness and efficiency of the work.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database applications–Data mining

Keywords

Product Graph, E-commerce, Recommender System

1. INTRODUCTION

As billions of users leverage e-commerce sites to fulfill their purchasing needs while saving time and voiding crowds, and how to better serve online customers is becoming an increasingly important problem. Meanwhile large e-commerce sites such as Taobao and Amazon often provide billions of products for consumers to choose. Recommender systems can help consumers to select the right products to meet their tastes and special needs, and play key role in modern e-commerce [10]. Existing recommendation approaches can be mainly divided into two categories: Collaborative Filtering approaches (CF) [14, 24] and Content-Based approaches (CB) [15, 17, 18]. There also exist other approaches such as hybrid methods that combines both, or context-aware methods [2, 1], which considers user's current context information to make recommendation more reasonable.

Besides the detailed recommendation approaches, another important problem is understanding and capturing relationships between products, which is the basis of modern e-commerce recommender systems [16]. We can view product relationship graph as a re-built index of products, which can return back a list of ranked products given a seed product. This index can dramatically speedup the recommendation service so that online user waiting time is un-noticeable.

There are two very important types of relationships between products: *substitute* and *complementary*. *Substitute* products are those that are interchangeable such as these shirts shown in Figure 1, while *complementary* products are those that might be purchased in addition such as shirts and trousers [16]. Different contextual environments may have different requirements or implications about recommendation relevance, thus different relationship graphs are needed to speed up the recommendation. For example, at different stages of a user session, the user's preferences can be very different. As illustrated in figure 1, when a user focuses on shirts and has not purchased any shirt in the session, the user may prefer to get *substitute* products recommended for him/her to compare with. Once a purchase is made, substitute products becomes less relevant, and some *complementary* products such as trousers or jackets would be more attractive and relevant.

Recently some researchers have recognized the importance of substitutes and complements, [16] utilizes the text of product reviews and descriptions to infer the relationships between products via a supervised approach. However, the existing approach doesn't apply to a huge recommendation system like Taobao's, because the text of product reviews and descriptions are huge and very noisy for billions of products from millions of sellers, thus the supervised approach

*Corresponding Author

WOODSTOCK '97 El Paso, Texas USA

DOI: 10.475/123_4

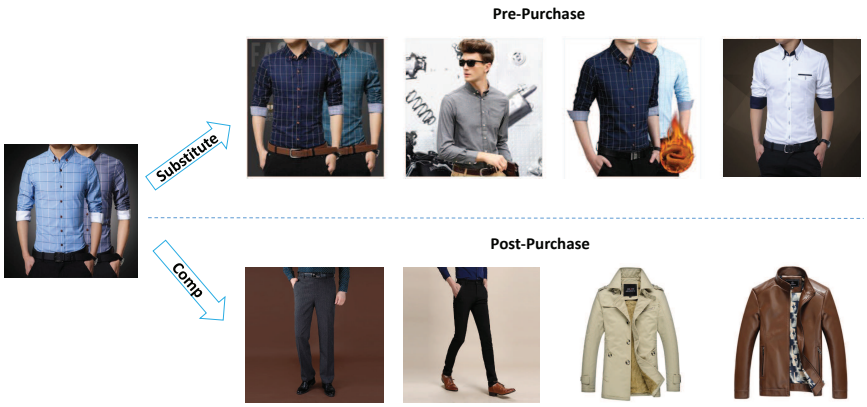


Figure 1: Application example of different relationships between products.

based on text is not cost effective.

On the other hand, Taobao has an enormous amount of real user behavior data from billions of customers, which are much stronger and reliable signals for capturing product relationships than text data. Thus we focus on utilizing the user behavior data directly to construct the product graph via an unsupervised approach similar to similarity based collaborative filtering.

When we try to adapt traditional approaches such as item-item CF with local similarities [23, 14], we face the following challenges:

- *Accuracy*: The traditional local similarity calculations do not consider any inner structure of user behavior graph, which has been shown to be useful for other link prediction problems. Thus the prediction accuracy is limited.

- *Robustness*: User behavior data (i.e. user-item click graph) contain many noisy, casual or accidental clicks, which could affect the reliability of predictions.

- *Sparsity*: Though the user behavior data in Taobao is huge, the ratio of user purchase is relatively small. User co-purchasing data is very sparse, thus capturing complementary relationships is very difficult.

- *Direction*: The complementary relationship is asymmetry. We also need to consider the direction of relationship on the basis of co-purchasing graph.

- *Scalability*: The computation complexity of traditional approaches grows with both the number of customers and products. Given billions of users and products in Taobao, scalability is a major challenge.

In this paper, we propose a new algorithm called Swing, which can utilize the inner structure–*Swing* of user-item behavior graph, to construct substitute product graph. Swing is a quasi-local structure and is designed to be much more stable than a single edge used in traditional CF approaches. It provides much more reliable calculation propagation over a user-item bi-partite graph, and helps to reduce the influence of noisy user behavior data. Then we propose a new algorithm called Surprise to construct complementary product graph. Surprise algorithm solves the sparsity problem on the user co-purchasing graph by utilizing products' category information and product clusters constructed based on Swing algorithm. Furthermore, it considers the time sensitivity and temporal order of co-purchased products. Both methods are implemented for parallel running with commonly used large scale distributed computing framework such as MapReduce or Spark, thus scalability is not a problem. Based on that, we can build the product substitute graph and product complementary graph in Taobao, which provide basic indexing service to generate candidate products for further recommendation ranking modules.

To evaluate the effectiveness of our approaches, we conduct extensive experiments with both offline data and online user studies. The offline experimental results on a large dataset demonstrate the proposed approaches can significantly outperform an existing well tuned baseline CF method in Precision, Recall and MAP metrics. Online experimental results with real world e-commerce users on Taobao also demonstrate the proposed approaches lead to significantly higher CTR (click through rate), CVR (conversion rate) and PPM (Payment Per Thousand Impressions). The efficiency of the new approaches on offline running time is also analyzed and demonstrated.

The main contribution of this papers are:

1. A new efficient and effective algorithm (i.e. Swing) that utilizes the quasi-local structure information of user behavior graph. It provides a more reliable calculation propagation and eliminates the effect of noisy data. Based on Swing, we can build more reliable similar relationships between products.

2. A new efficient and effective algorithm (i.e. Surprise) that utilizes product category information and clustering technique. It solves the sparsity problem, thus the complementary relationships between products are more reliable and reasonable when using the Surprise algorithm.

3. An efficient large scale industrial implementation for the proposed approaches via parallelization.

4. An empirical verification of the effectiveness and efficiency of the new approaches via comprehensive offline and online experiments and detailed analysis.

The rest of the paper is organized as follows. We introduce the Swing algorithm for substitute relationships in Section 2. We then describe the Surprise algorithm for complementary relationships in Section 3. Section 4 presents the experimental results, Section 5 describes the related work, and Section 6 concludes the paper.

2. SWING ALGORITHM FOR SUBSTITUTE RELATIONSHIPS

Computing similarity between two items is central to the task of building substitute graph. There are many similarity measures that only require local information and those methods are usually computationally efficient and can be applied to extremely large graphs. Frequently used methods in recommendation are shown as follows [23, **?**].

Vector Cosine Similarity

$$w_{i,j} = \frac{U_i \bigcap U_j}{\|U_i\| \times \|U_j\|}$$

where U_i is the set of users that have clicked item i.

Jaccard Coefficients

$$w_{i,j} = \frac{U_i \bigcap U_j}{U_i \bigcup U_j}$$

Correlation-Based Similarity

$$w_{i,j} = \frac{\sum_{u \in U_i \bigcap U_j} (r_{u,i} - \overline{r_i})(r_{u,j} - \overline{r_j})}{\sqrt{\sum_{u \in U_i \bigcap U_j} (r_{u,i} - \overline{r_i})^2} \sqrt{\sum_{u \in U_i \bigcap U_j} (r_{u,j} - \overline{r_j})^2}}$$

where $r_{u,i}$ is the rating of user u on item i, and $\overline{r_i}$ is item i's average rating. The similarity between item i and item j is measured by computing *Pearson Coefficients*.

Most traditional methods focus on item-user-item path when calculating neighborhood strength, with item based normalization to penalize popular items. These methods do not utilize the other inner structures such as user-item-user on the user behavior graph, thus the prediction accuracy is limited.

For example, suppose there are five users (Alice, Bob, Chris, David and Eric) and all of them are looking for T-shirts on Taobao. As shown in figure 2(a), each row represent a user and items the user clicked. Alice is choosing T-shirt for her boyfriend. Bob wants to buy a popular T-shirt for himself. Chris loves t-shirts of a special brand named *Mucunsanshe*. David is a big fan of Michael Jackson, and only clicks one Michael Jackson T-shirt. While Eric is looking for T-shirts in red color. All of them have clicked item h. The click information can be summarized as figure 2(b), where each capitalized letter represents a person(corresponding to five users above, we use A,B,C,D,E for short) and each lower case letter represents a clicked product. For simplicity and without loss of generality, we only show items in the same category: *T-shirts*. Users may click any objects. For example, after buying T-shirts, Alice may view and clicks dresses for herself, and David may click headphones, since he is a music enthusiast.

In this case, we have $|U_h| = 5$, $|U_t| = 15$, $|U_p| = 40$, $|U_q| = 60$ and $|U_z|$=4, which means there are totally 5 users that have clicked item h, and 15 users have clicked item t, etc. If we rank neighboring items for h using cosine similarity, the result would be t > z > p > q. However, we can see that z is not very similar to h. Besides the number of common-neighbors, it only depends on the degree of each item node on the denominator(namely, how many users have clicked each item), which could be affected by many factors and has big uncertainty. Noting that the CF score of z could be greater than p and q, as it is less popular.

2.1 Swing Algorithm

Since our user click data is noisy, with many accidental or random clicks, we need a node-proximity measure which will consider more robust inner-network structural information. Stable network structure has been studied in link prediction field, e.g. users tend to form triangles in social networks [11, 13]. Considering bipartite-networks in E-commerce, when there is only one user who clicks h and i together, it is more likely to be a coincidence. However, if two users both clicked h and i, the relation is much stronger. Thus for each seed product, considering the local graph that includes all users who clicked the seed product and all products clicked by those users, we call each *user-item-user* network structure on the local graph as a *swing*. For example, besides the seed item h, both D and E also clicked item q, so $[D, q, E]$ is a *swing*. while $[A, z], [C, y], [E, o], [E, x]$ are just single edges that are not part of any swing.

If there are many *swing*s between a user pair $< u, v >$, it usually indicates a wide range of products could meet their requirements. And the relation among these *swing*s is less intensive. Therefore, we weigh each *swing* through the total number of *swing*s formed between each user pair. The definition of *swing* score is given below.

$$s(i, j) = \sum_{u \in U_i \bigcap U_j} \sum_{v \in U_i \bigcap U_j} \frac{1}{\alpha + |I_u \bigcap I_v|} \tag{1}$$

where U_i is the set of users who have clicked item i, and I_u is the set of items that clicked by user u. α is a smoothing coefficient.

Consider the example in figure 2 and how to compute the similarity between item h and other items. Without loss of generality, let $\alpha = 1$. $[A, p, B]$ is a swing, while there are 3 swings on $[A, B]$ with $[A, t, B]$, $[A, r, B]$ and $[A, p, B]$. Therefore, the swing score p contributed by $[A, B]$ is $\frac{1}{(1+3)}$. $[B, p, C]$ is a swing, and there is only one swing on $[B, C]$, so the swing score of p contributed by [B, C] is $\frac{1}{(1+1)}$. Similarly, p also gets $\frac{1}{(1+1)}$ from $[A, C]$, as $[A, p, C]$ is a swing. The total swing score for p is:

$$swing(h, p) = \frac{1}{4} + \frac{1}{2} + \frac{1}{2} = 1.25$$

Similarly, we have:

$$swing(h, q) = \frac{1}{2} + \frac{1}{2} + \frac{1}{2} = 1.5$$

$$swing(h, t) = swing(h, r) = \frac{1}{2} = 0.25$$

Therefore, we have $q > p > r = t > others$. Item q is a Michael Jackson T-shirt with brand *Mucunsanshe* and red color, which is very similar to item h. With *swing*, we are able to rank item q up to the first place. At the same, all items connected with single edge are ranked to the bottom in a strict way.

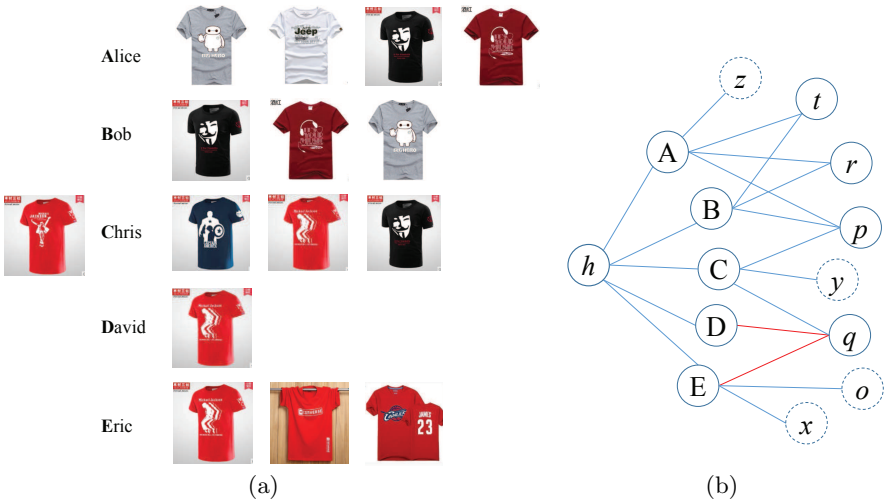


Figure 2: An Example of Swing score computing.

For each *swing* structure, we further add user weighting factor similar to the well-known Adamic/Adar algorithm [12] to penalize active users. The more items a user clicks, the smaller weight it gets. The final *swing* score is:

$$s(i,j) = \sum_{u \in U_i \bigcap U_j} \sum_{v \in U_i \bigcap U_j} w_u \cdot w_v \cdot \frac{1}{\alpha + |I_u \bigcap I_v|} \quad (2)$$

where

$$w_u = \frac{1}{\sqrt{|I_u|}}, w_v = \frac{1}{\sqrt{|I_v|}}$$

The detailed procedure to calculate swing score is described in Algorithm 1.

Noting that in this work, we mainly focus on the similarity computing based on user's clicking behavior only, other factors such as time decaying could also be incorporated into each method. Furthermore, extensive closely related clicks often happen in the same user session in Taobao, where time interval shows small impact.

Assuming the total number of items is T, the average node degree of an item is N, and the average node degree of a user is M, the traditional item-item similarity based CF approaches have a time complexity of $O(T \cdot N \cdot M)$. The time complexity of *Swing* is $O(T \cdot N^2 \cdot M)$, which is higher due to the consideration of inner network structures on user behavior data.

2.2 Parallelization Implementation Framework

In the Taobao recommendation system, we developed parallel implementation of *Swing* on a Map-Reduce programming framework in our own distributed platform of Open Data Processing Service[1]. The detailed procedure is presented in fig 3.

Specifically, for the original input data of user click matrix, each row is a clicked item list of a specific user. At the mapper stage, for each item clicked by user i in one row, we make a neighborhood broadcasting through u_i, and output the key-value pairs as $< i_{i1}, u1\ i_{12}\ \cdots\ i_{1n} >$. Then at the Reducer stage, we collect the user set U_i who clicked $item_i$, and the item set clicked by each user in U_i. Finally *Swing* is calculated to compute the most similar items for $item_i$ as described in Algorithm 1.

[1]https://www.aliyun.com/product/odps/

Algorithm 1 Swing Algorithm for Substitute Relationships

Input:
User and Item index table: **U**, **I**,
Smoothing coefficient: α
Output: Substitute item list for $item_i$
for each $u \in U_i$ **do**
  $w_u = \frac{1}{\sqrt{|I_u|}}$
  for each $v \in U_i \setminus u$ **do**
    $w_v = \frac{1}{\sqrt{|I_v|}}$
    $k = |I_u \bigcap I_v|$
    for each $j \in I_u \bigcap I_v$ **do**
      $Swing[j] = w_u * w_v * \frac{1}{\alpha + k}$
    end for
  end for
end for
return Swing$_i = (Swing_i[1], \cdots, Swing_i[n])$.

3. SURPRISE ALGORITHM FOR COMPLEMENTARY RELATIONSHIPS

In this section, we propose a comprehensive framework to find complementary products by mining user purchasing data. When a user has just bought a T-shirt, it would be unappropriate to recommend T-shirts anymore. Instead, shorts and shoes may be better candidates. If he has just bought a mobile phone, showing accessories such as phone shells and portable power devices are more reasonable, since the user is no longer interested in mobile phones. This important product-product relationship is called *complementary* relationship, which links a seed product with a related product that the user is likely to buy additionally or together.

To find complementary relationships, we borrow the idea proposed in [24] and use purchasing data to find candidate products. Furthermore, we need to consider time sensitiv-

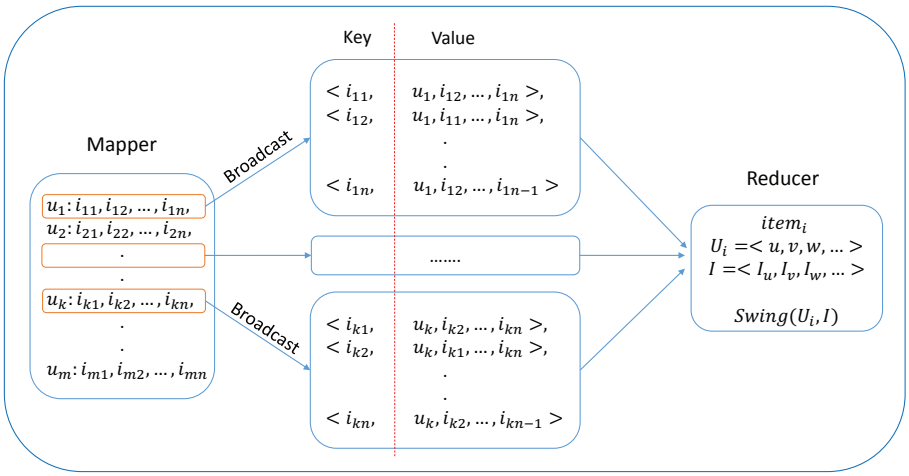


Figure 3: Parallelization implementation of Swing via MR framework

ity of products [8, 26, 20, 25] and data sparsity problem, especially in Taobao scenario. First, we introduce a continuous time decay factor to the behavioral relevance, which has been shown to improve accuracy. Then, instead of using content information, which exists in very high dimensional space and differs greatly across categories, we introduce a clustering method based on user click data to solve the problem of data sparsity. Note that implicit feedback data such as user clicks is actually not so sparse, although user purchasing data is sparse, because users normally browse and click lots of items and only buy much less. We call this framework with the name of *Surprise* to aim at helping users to find closed related items of different categories that users would be unaware.

3.1 Relevance of Category-level

To find the most related products to the seed item user purchased, firstly we try to find the most related categories on products' taxonomy, which is usually used in e-commerce systems. This removes a vast majority of cases derived from user purchase behaviors.

By mapping each item to its category, we get a user-category matrix. Then standard collaborative filtering techniques could be used to compute the relevance between categories. The probability of c_j is a related category for c_i is defined as follows:

$$\theta_{i,j} = p(c_{i,j}|c_j) = \frac{N(c_{i,j})}{N(c_j)} \tag{3}$$

where $N(c_j)$ is the total purchases in category c_j, and $N(c_{i,j})$ is the number of times c_j is purchased after c_i.

Supposing that the category lists obtained from equation 3 for c_i is $[c_{j_1}, c_{j_2}, ..., c_{j_m}]$, with a descending posterior probability list of $[\theta_{i,j_1}, \theta_{i,j_2}, ..., \theta i, j_m]$. Roughly take top percentage or a fixed number of candidates as relevant categories for c_i is not acurrate, since category varies greatly from each other. Instead, we compute a *relative drop* score for each category $c_{j,k}$.

$$\eta_k = \frac{(\theta_{i,j_{k+1}} - \theta_{i,j_k})}{\theta_{i,j_k}} \tag{4}$$

Categories ranked before the *maximum relative drop point* are selected as top-related categories for c_i, and we denote these categories as $\Gamma(c_i)$.

The probabilities of related categories for *T-shirt (male)* and *Mobile Phone* are shown in figure 4(a) and figure 4(b) respectively. From the two sub-figures, we can see there are change points in relevance distribution. By taking items before the maximum relative drop point, we get 8 top-related categories for *T-shirts (male)*: *Casual Pants*, *Jeans*, *Jacket*, *Shirt*, *Low Shoes*, *Sweater*, *Wei clothing*, *Cotton-padded Coat*, and 3 tightly related categories for *Mobile phone*: *Phone shell*, *Phone membrane* and *Portable power source*.

3.2 Relevance of Product-level

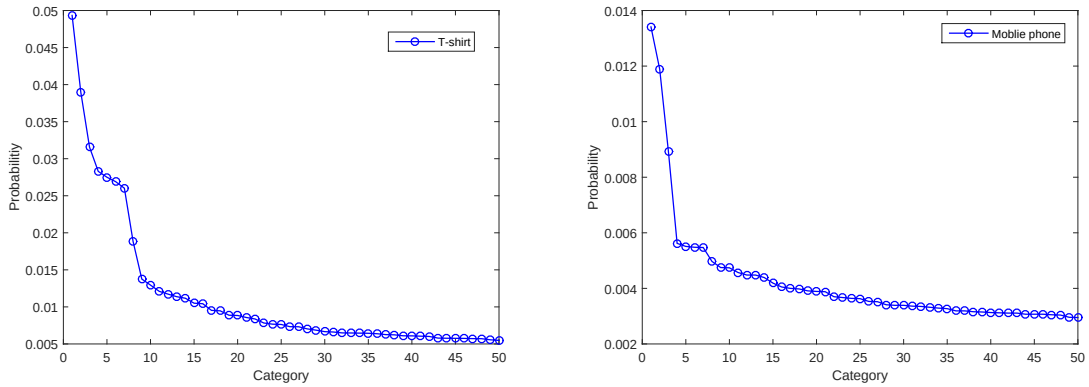
For each related category, we compute the relevance score for items in the category, where standard item-item based collaborative filtering techniques could be applied. In our system, we add the constraint that a candidate related item j should be purchased after item i. The *order* is very important in modeling *complementary*. For example, it is reasonable to recommend portable power devices when a user has just bought a cell phone, while it would be unappropriate if we recommend cell phones for him after a portable power device is purchased. Then the related score is defined as follows.

$$s_1(i,j) = \frac{\sum_{u \in U_i \bigcap U_j} 1/(1 + |t_{ui} - t_{uj}|)}{\| U_i \| \times \| U_j \|} \tag{5}$$

where $c_j \in \Gamma(c_i)$ and $t_{uj} \geq t_{ui}$. In Equation 5, we add a time decay factor in the numerator. If the time interval between purchase of item i and item j is long, it will be less likely to exist a strong relationship between them.

3.3 Relevance of Cluster-level

Supposing that Bob buys jeans j after a T-shirt i, we are not sure that j is a good match to i. While if there are several users who have bought j after i, we'll be more confident that j is very likely to be related to i. We also introduce a threshold on the co-occurrence number of (i, j) when selecting candidate items. For example, when we make recommendation after the purchase of item i, we only take the candidate items with $Co(i, j) > \gamma$, which can be viewed



(a) (b)

Figure 4: The examples of Top related categories selection via Max_drop.

as a kind of behavior confidence. However, this constraint will further bring the problem of data sparsity, since the user co-purchasing data is very sparse.

Considering a different situation, where Bob bought j after i and David bought k after i, meanwhile item j and item k are very similar, for instance they are both blue levis jeans. This kind of co-purchasing is informative about product relationships. This motivates us to also calculate the relevance score for items at the cluster level, which help to alleviate the data sparsity problem of co-purchasing at item level.

we attempt to project similar items into the same cluster by utilizing the similar items graph constructed by the *Swing* algorithm in the previous section. Then we compute the co-purchasing relevance at cluster-level.

3.3.1 Clustering using label propagation

Traditional clustering methods such as k-means and density based methods are not feasible in Taobao scenario, as scalability is an issue with billions of products. Inspired by the community detection method proposed in [19], we perform a similar label propagation process in the substitute graph created before.

For each item, top-similar items computed by *Swing* is added as its neighbors, with a directed link pointing from a similar item to the seed item. The weight of the edge $e_{j,i}$ is set to the similarity score $swing_{ij}$. Note that this weight is not necessarily symmetric. Initially, the *label* for each item node is setting to its node-id. The $p(\cdot)$ denotes the current probability of the corresponding *label* that a neighbor node belongs to. For each item node, we consider all its neighbors when updating the label probability based on the edge weight, then we choose the label with maximal probability value for the item node based on a certain probability (i.e. $random() > \beta$).

We implement this algorithm using a large scale graph computing framework in the Aliyun platform[2], and it converges after about ten iterations. Finally items with the same label are grouped into the same cluster. The algorithm is proved to be a very fast yet effective clustering method. It takes only *15 minutes* to cluster billions of items, and the detailed algorithm is presented in Algorithm 2.

[2]www.aliyun.com

Algorithm 2 Clustering using label propagation

Input:
Similarity graph $G(V, E)$,
item node $x \in V$ and its neighbors $\Gamma(x)$,
damping factor β=0.25
Output: Unique label $L(x)$

```
init L(x) = x
for t = 1, ..., n do
    for each x ∈ V do
        init p[L[y]] = 0,  y ∈ Γ(x)
        for each y ∈ Γ(x) do
            p[L[y]] += e_{y,x}
        end for
        if random() > β then
            Set L(x) = k, where p[k] = max(p[1 : m])
        end if
    end for
end for
return L.
```

3.3.2 Cluster-level relevance

After clustering items into different groups, we compute the relevance score at the cluster level. Let $L(i)$ be the cluster to which item i belongs. Then

$$s_2(i, j) = s_1(L(i), L(j)) \tag{6}$$

where s_1 is computed as equation 5 described before. That is, we compute a relevance score that purchases of item cluster $L(j)$ happened after item cluster $L(i)$.

3.4 Compute Surprise Score

Based on two relevance scores $s_1(i, j)$ and $s_2(i, j)$, we calculate the final related score by combining them linearly:

$$s(i, j) = \omega * s_1(i, j) + (1 - \omega) * s_2(i, j) \tag{7}$$

where ω is the combination weight that can be set manually or estimated from the data. The default value of ω is 0.8 in our experiment unless otherwise noted.

4. EXPERIMENTS

We will evaluate the proposed new approaches empirically. We first introduce the experimental setup, then report the offline evaluation with different metrics. The system is also deployed on Taobao.com to verify the results with online customers. Finally, efficiency analysis based on the running time is provided.

4.1 Experimental Setup

We collected a user behavior dataset with more than 400 million users and 500 million products from Dec 16th to Dec 30th 2015 in Taobao. The user click data is used to capture the substitute relationships, and the user purchasing data is used to capture the complementary relationships.

We utilize a classical item-based CF with cosine similarity as our main baseline method, as it fits well in the e-commerce scenario [14]. Other similarity measures are also evaluated, and the cosine similarity performs best among existing choices.

For fair comparison, we also introduce a user weighting factor into the original CF to penalize the active users, which is similar to the way used in *Swing* algorithm. This improvement further improves the performance of CF, and is used as a stronger baseline in this paper. Thus the final CF *baseline method* is defined as follows.

$$w_{i,j} = \frac{\sum_{u \in U_i \cap U_j} w_u^2}{\sqrt{\sum_{u \in U_i} w_u^2} \sqrt{\sum_{v \in U_j} w_v^2}}$$

$$w_u = \frac{1}{\sqrt{|I_u|}}, w_v = \frac{1}{\sqrt{|I_v|}}$$

where $|I_u|$ is the product number of user clicks/purchases.

4.2 Offline Evaluation

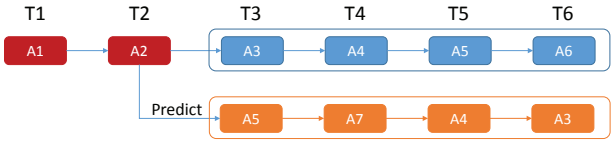


Figure 5: Offline evaluation example.

We propose an offline strategy to evaluate the performance of different techniques. The basic idea is to see how the relations mined from historical data hit (i.e. match) the real user behavior sequence in the future. For example, as shown in figure 5, the top row is an actual user click sequence. We randomly choose an intermediate product in the sequence, such as $A2$ at time $T2$, as the seed product for substitute/similar products recommendation. The bottom row contains similar items recommended given product $A2$. In this case, the recommendations *hit* $A3$, $A4$ and $A5$, the *hit* number is 3. The larger the *hit* number, the better the recommendation quality.

Given a set of product recommendation $predict_i$ of a given type (click or purchase) for a user i, and a set of known ground-$truth$, we can use traditional evaluation metrics, including Precision, Recall and MAP [3], for offline evaluation. The definitions of the metrics are:

$$precision = \frac{1}{n} \sum_{i}^{n} \frac{|hit_i|}{|predict_i|}$$

$$recall = \frac{1}{n} \sum_{i}^{n} \frac{|hit_i|}{|truth_i|}$$

$$MAP = \frac{1}{n} \sum_{i}^{n} \sum_{k=1}^{m} precision_i@k$$

where $hit_i = predict_i \cap truth_i$, and $predict_i$ denotes the algorithm prediction sequence for i-th user, $truth_i$ denotes the real behavior (click/purchase) sequence. $precision_i@k$ denotes the precision of $predict_i$ with a cutoff at position k.

We utilize the data from 16th Dec to 30th Dec to mine the product relations with the proposed approaches and baseline methods, then we use the actual user behavior sequences generated from 31th Dec as the ground-truth[3]. Evaluation results of *Swing* are shown in Table 1. The results show that the *Swing* algorithm significantly outperforms the classical item-based CF approach. Specially, the relative improvement of Swing over CF is up to 67.6% and 46.1% in terms of Precision and Recall respectively. It indicates that our method could find more relevant substitute products with higher precision. Furthermore, our approach also outperforms CF in terms of MAP with orders of magnitude, which indicates relevant products are ranked higher in the recommendation lists by *Swing*.

The evaluation results for *Surprise* are shown in Table 2, and we have similar findings.

4.3 Online Evaluation

Online experiments with real world E-commerce users are carried out to study the effects of the proposed approaches. We conduct A/B tests based on recommendation scenarios in the mobile APP of Taobao.

The online metrics we utilize are Click Through Rate (CTR), Click Conversion Rate (CVR) and Payment Per Thousand Impressions (PPM), which are commonly used in e-commercial systems. The definitions are given as follows.

$$CTR = \frac{\#item\_click}{\#show\_pv}$$

$$CVR = \frac{\#item\_trade}{\#item\_click}$$

$$PPM = \frac{\#Payment}{\#show\_pv} * 1000$$

4.3.1 Swing Online

We conducted A/B test with the *Swing* algorithm and the baseline approach in pre-purchase scenarios, and the results are shown in Figure 6. We find that the *Swing* bucket significantly outperforms ($p - value < 0.05$) the CF bucket in all three metrics. Specifically, the relative improvement of Swing over CF is up to 9.3% and 17.6% on average in terms of CTR and CVR, respectively. It means that users

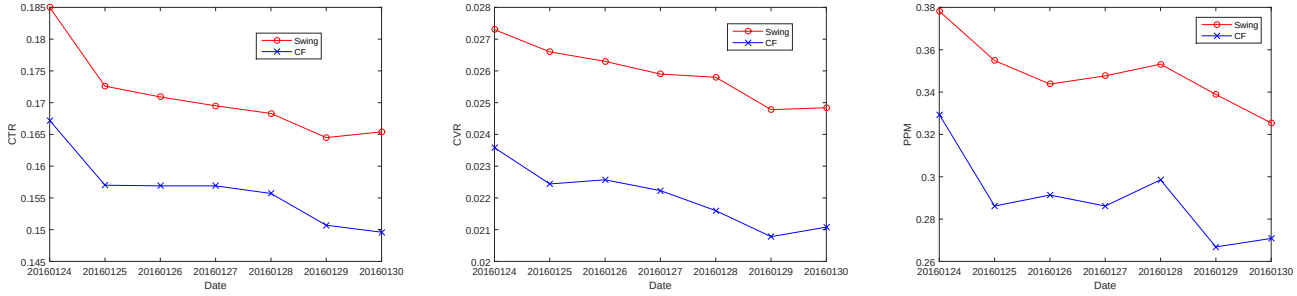
[3]The dataset will be released later in our official website: https://tianchi.shuju.aliyun.com/datalab/index.htm

Table 1: Offline evaluation of Swing.

	Precision	Recall	MAP
CF	0.01471	0.1093	0.01177
Swing	**0.02466** (+67.6%)	**0.1597** (+46.1%)	**0.06109** (+419%)

Table 2: Offline evaluation of Surprise.

	Precision	Recall	MAP
CF	0.01188	0.1231	0.06242
Surprise	**0.02519** (+111.9%)	**0.23875** (+93.9%)	**0.109558** (+75.5%)



(a) (b) (c)

Figure 6: Online Evaluation of Swing in pre-purchase scene via CTR, CVR and PPM metrics

find more interested products with *Swing*, and the probability of converting the click behavior to real purchases is improved significantly. *Swing* achieves an improvement of 20.3% on average for PPM. PPM is the key criteria for recommendation in e-commerce, thus the improvement has a huge commercial value.

4.3.2 *Surprise Online*

The *Surprise* algorithm is deployed in post-purchase scenarios, and the evaluation results are shown in Figure 7. For the *Surprise* method, we also conducted another version without the relevance score at the cluster level, i.e. $s(i,j) = s_1(i,j)$, and it is represented as *Surprise-NCR*.

From the figures we can find that both *Surprise* and *Surprise-NCR* show significant improvements over the original purchasing based CF in all evaluation measures, which indicates the importance of top-related categories and the time-based weighting mechanism. Meanwhile, compared with *Surprise-NCR*, *Surprise* achieved 27.9% CTR improvement on average, largely due to better relevance estimation and its ability to generate more product recommendations. On the CVR metric, *Surprise-NCR* has very similar performance to Surprise. Our hypothesis is that once a user has clicked a certain interested product, the willingness to purchase is largely depends on the product properties itself, which is not captured by the cluster-level relevance score. With the measure of PPM, *Surprise* outperforms *Surprise-NCR* and the original purchasing CF by 35% and 183% on average, respectively. In summary, the *Surprise* algorithm achieved the best commercial impact.

4.3.3 *Hybrid Application*

We also conduct an overall evaluation of the product graphs in an integrated scenario, which are constructed by Swing (SW) and Surprise(SP). The recommendation list is generated by combing substitute items for user past clicks and complementary items for user current order. The detailed evaluation results are shown in Figure 8. The combination of our proposed methods outperforms the original CF + CF by 33.2%, 26.7%, 62.9% on average in CTR, CVR and PPM, respectively. The results further prove the effectiveness of our proposed approaches.

4.4 Efficiency Evaluation

We further study the efficiency of the proposed approaches. As discussed in section 2.1, the time complexity of Swing is higher than CF due to the consideration of inner structures on user click graph. While for both methods, we developed practical parallelization implementation for large scale applications, it is still worth to compare the offline calculation time of all the methods. The results are shown as following (unit: hour):

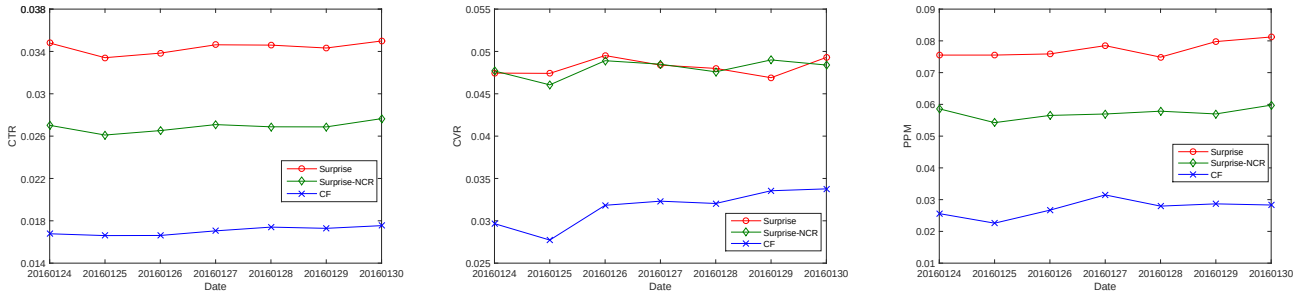
$$\text{Click-CF } (\sim 2.0h) \prec \text{Swing } (\sim 3.5h)$$

$$\text{Purchase-CF } (\sim 1.3h) \prec \text{Surprise } (\sim 2.5h)$$

The computation of Surprise includes 3 parts: the inference of top-related categories, the item-level relevance calculation and the cluster-level relevance calculating. With top relevant categories, we can pre-filter unrelated items at the Mapper stage and reduce the calculation cost greatly. Our approach takes longer, however, acceptable offline computing time, while achieves significant improvements in online experiments.

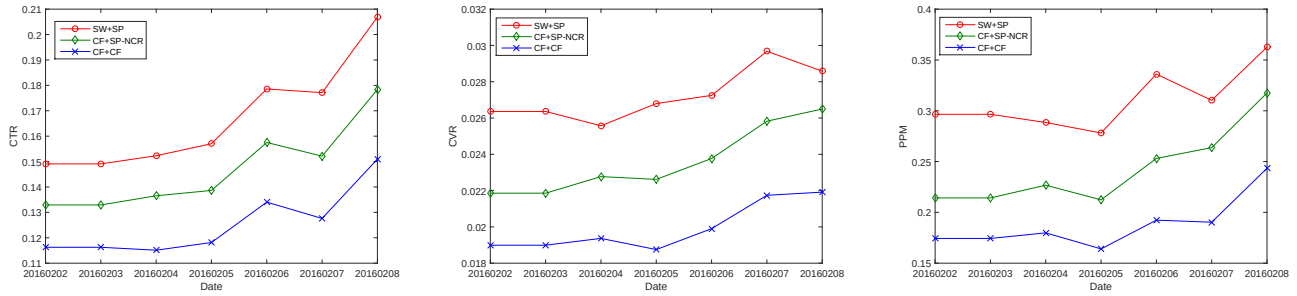
5. RELATED WORK

Recommender systems can help consumers to select the right products to match their tastes and meet their special needs, and play a key role in modern e-commerce systems. Most of existing recommendation approaches can be divided into two categories: Content-Based approaches and Collaborative Filtering approaches.



(a) (b) (c)

Figure 7: Online Evaluation of Surprise in post-purchase scene via CTR, CVR and PPM metrics



(a) (b) (c)

Figure 8: Online Evaluation of Hybrid application in Taobao via CTR, CVR and PPM metrics

The content-based approaches make recommendation based on item features. Several approaches have been proposed [17, 18, 15], such as vector space models, bayesian classifiers and clustering models. However, the application of CB methods in e-commerce is limited, because content features such as product title words, brands can hardly describe all the important hidden characters of products and the fine-grained preferences of users. Therefore, CB methods are usually utilized as part of a solution in a hybrid recommender system [5, 24].

The collaborative Filtering approaches make recommendations based on a user's past behaviors [14, 24]. These approaches usually operate on the user-item matrix, where each row is a user vector and each column is a item vector. User-based methods try to find similar user neighbors to the current user [22, 4]. Item-based approaches find item neighbors that are similar to the item a user clicked or purchased based on the item vectors [23, 7, 6]. Among these methods, various similarity measures have been proposed to find the top-n similar neighbors, such as cosine similarity, Jaccard similarity, Pearson correlation coefficient and conditional probability based similarity. Compared with the user-based approaches, the item-based methods are more commonly used in large scale e-commerce systems due to their effectiveness and efficiency. Recently, factorization models such as Singular Value Decomposition (SVD) [9] or Probabilistic Matrix Factorization (PMF) [21] have gained much attention due to their good performance on some benchmark datasets, and those methods can also incorporate additional information such as implicit user feedback and temporal information.

Rather than providing a single recommendation algorithm or solution, this paper mainly focuses on finding the relationships between products, which serve as the basis for quickly returning candidate products for further computationally expensive recommendation algorithms, such as hybrid filtering or learning to rank. A recent work [16] also focuses on the same problem. It utilizes the text of product reviews and descriptions to infer the relationships between products via a supervised approach. In contrast to their work, we focus on directly utilizing the user behavior data such as user co-click and co-purchase, which are stronger and more reliable signals for the capturing of product relationships.

Although we are constructing item-item similarity matrix, traditional item-based CF methods based on standard local similarity measures [23, 14, **?**] are 'pointwise' measures, which ignores the inner structure of user behavior data. On the other hand, existing work on social network and graph analysis have shown inner structure could be useful for predictions in other domains. These motivate us to introduce inner structures to recommendation system.

6. CONCLUSIONS

This paper focus on the product graph with substitute and complementary relationships. Construction of such kind of large scale product graphes in e-commerce has several major challenges: robustness, data noise, data sparsity, relationship directions and scalability. To tackle those challenges, we propose the Swing algorithm, which can utilize the inner stable structures of user behavior data, to capture the substitute relationships of products. It performs significantly better as it eliminates noisy information and makes the predicted relationship more robust. Then we propose an advanced Surprise algorithm for the modeling of

complementary relationship. Surprise utilizes product category information and solves the sparsity problem in user co-purchasing graph via clustering technique of label propagation. Furthermore, it also considers the time sensitivity and time sequential of propagation to guarantee the complementary relationship is reasonable. Base on the two approaches, we build two product graphs to support fast recommendations in Taobao. Finally, we verify the effectiveness and efficiency of our approaches with comprehensive offline and online evaluations.

The product-product graph construction methods proposed in this paper is very general and can be applied to other applications, such as advertising and personalized search in e-commerce. This can be explored in the future. Besides, we will consider combing the content information with the proposed work more effectively and efficiently. This paper is a first step towards introducing quasi-local user-user edge based inner structure into recommendation systems. The approach is simple, efficient and extremely effective in practice. We plan to consider different inner structures and introduce them in other ways such as Markov random fields or Bayesian network in the future, and explore efficient solutions to adapt those more theoretical solutions to our large scale system.

7. REFERENCES

[1] G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin. Context-aware recommender systems. *AI Magazine*, 32(3):67–80, 2011.

[2] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 335–336, 2008.

[3] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[4] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. UAI'98, pages 43–52, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[5] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, Nov. 2002.

[6] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, Jan. 2004.

[7] G. Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the 10th CIKM*, pages 247–254, New York, NY, USA, 2001. ACM.

[8] Y. Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 53(4):89–97, Apr. 2010.

[9] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.

[10] Y. Koren and R. M. Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 145–186. Springer, 2011.

[11] G. Kossinets and D. Watts. Empirical analysis of an evolving social network. *Science*, 311(5757):88–90, 2006.

[12] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the 12th CIKM*, pages 556–559, New York, NY, USA, 2003. ACM.

[13] R. Lichtenwalter and N. V. Chawla. Vertex collocation profiles: subgraph counting for link analysis and prediction. In A. Mille, F. L. Gandon, J. Misselis, M. Rabinovich, and S. Staab, editors, *WWW*, pages 1019–1028. ACM, 2012.

[14] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, Jan. 2003.

[15] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer US, 2011.

[16] J. McAuley, R. Pandey, and J. Leskovec. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD*, pages 785–794, New York, NY, USA, 2015. ACM.

[17] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, DL '00, pages 195–204, New York, NY, USA, 2000. ACM.

[18] M. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27(3):313–331, 1997.

[19] U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E*, 76:036106, Sep 2007.

[20] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th WWW*, pages 811–820, New York, NY, USA, 2010. ACM.

[21] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th ICML*, pages 880–887, New York, NY, USA, 2008. ACM.

[22] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2Nd ACM Conference on Electronic Commerce*, EC '00, pages 158–167, New York, NY, USA, 2000. ACM.

[23] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th WWW*, pages 285–295, New York, NY, USA, 2001. ACM.

[24] J. Wang, B. Sarwar, and N. Sundaresan. Utilizing related products for post-purchase recommendation in e-commerce. In *Proceedings of the 5th RecSys*, pages 329–332, New York, NY, USA, 2011. ACM.

[25] J. Wang and Y. Zhang. Opportunity model for e-commerce recommendation: Right product; right time. In *Proceedings of the 36th International ACM SIGIR*, pages 303–312, New York, NY, USA, 2013.

[26] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun. Temporal recommendation on graphs via long- and short-term preference fusion. In *Proceedings of the 16th ACM SIGKDD*, pages 723–732, New York, NY, USA, 2010. ACM.