On Maximizing Influence Spread for Social Item Hypergraph

Abstract—Product recommendation and viral marketing have been widely studied due to the huge business opportunities. However, existing works on seed selection in social networks do not take into account the effect of product recommendations in e-commerce stores. In this paper, we investigate the seed selection problem for viral marketing that considers both effects of social influence and item inference (for product recommendation). We develop a new model, *Social Item Graph (SIG)*, that captures both effects in the form of hyperedges. Accordingly, we formulate a seed selection problem, called *Social Item Maximization Problem (SIMP)* for a simplified hyperedge model, and prove the hardness of SIMP. We design an efficient algorithm with performance guarantee, called Hyperedge-Aware Greedy (HAG), for SIMP and develop a new index structure, called SIG-index, to accelerate the computation of diffusion process in HAG. Afterward, we extend the hyperedge model to the general case and formulate a seed selection problem, called *General Social Item Maximization Problem (GSIMP)*, and prove the hardness. To solve GSIMP, several pruning rules are proposed, as well as a multi-purposed hybrid structure, namely, *GSIG-index Lattice*, to accelerate the pruning process. Moreover, to construct realistic SIG models for SIMP, we develop a statistical inference based framework to learn the weights of hyperedges from data, and propose a weight sharing method to alleviate the severe data sparsity problem for general model. Finally, we perform a comprehensive evaluation on our proposals with various baselines. Experimental result validates our ideas and demonstrates the effectiveness and efficiency of the proposed model and algorithms over baselines.

Index Terms—viral marketing, product recommendation, frequent pattern

1 Introduction

Nowadays, with the fast growth of social media, people get used to explore information from friends' and celebrities newsfeeds. As the user personal behaviors and social interactions become observable and recordable, the ripple effect of social influence [4] has been explored for viral marketing via online social networks. In fact, studies prove that customers tend to receive product information from friends better than advertisements on traditional media [19]. To fully exploit the power of social influence, many research studies on *seed selection*, i.e., selecting a given number of influential customers to maximize the spread of social recommendation for a product, have been reported [7], [26].[1] Nevertheless, these works do not take into account the effect of product recommendations in online e-commerce stores. We argue that when a customer buys an item due to the social influence (e.g., via Twitter or Pinterest), there is a potential side effect due to the *item inference* recommendations from stores.[2] For example, when Alice buys a DVD of "Star War" due to the recommendation from friends, she may also pick up the original novel of the movie due to an in-store recommendation, which may in turn trigger additional purchases of the novel among her friends. To the best of our knowledge, this additional spread introduced by the item inference recommendations has not been considered in existing research on viral marketing.

Figure 1 illustrates the above joint effects in a toy example with two products and four customers, where a dash arrow represents the association rule behind the item inference recommendation, and a solid arrow denotes the social influence between two friends upon a product. In the two separate planes corresponding to DVD and novel, social influence is expected to take effect on promoting interests in (and potential purchases of) the DVD and novel, respectively. Meanwhile, the item inference recommendation by the e-commerce store is expected to trigger sales of additional items. Note that the association rules behind item inference are derived without considering the ripple effect of social influence. In the example, when Bob buys the DVD, he may also buy the novel due to the item inference recommendation. Moreover, he may influence Cindy to purchase the novel. However, the association rules behind item inference are derived without considering the ripple effect of social influence. On the other hand, to promote the movie DVD, Alice may be selected as a seed for a viral marketing campaign, hoping to spread her influence to Bob and David to trigger additional purchases of the DVD. Actually, due to the effect of item inference recommendation, having Alice as a seed may additionally trigger purchases of the novel by Bob and Cindy. This is a factor that existing seed selection

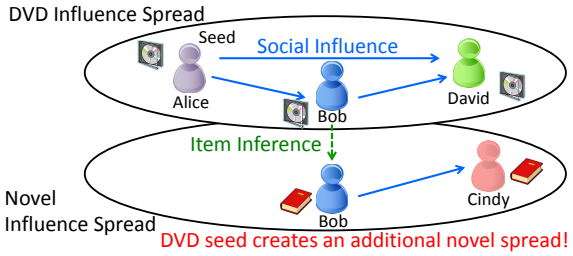


Fig. 1. A motivating example

1. All the top 5 online retailers, including Amazon, Staples, Apple, Walmart, and Dell, are equipped with sophisticated recommendation engines. They also support viral marketing by allowing users to share favorite products in Facebook.

2. In this paper, we refer product/item recommendation based on associations among items inferred from purchase transactions as item inference recommendation.

algorithms for viral marketing do not account for.

We argue that to select seeds for maximizing the spread of product information to a customer base (or maximizing the sale revenue of products) in a viral marketing campaign, both effects of item inference and social influence need to be considered. To incorporate both effects, we propose a new model, called *Social Item Graph (SIG)* in the form of hyperedges, for capturing "purchase actions" of customers on products and their potential influence to trigger other purchase actions. Different from the conventional approaches [7], [17] that use links between customers to model social relationship (for viral marketing) and links between items to capture the association (for item inference recommendation), SIG represents a *purchase action* as a node (denoted by a tuple of a customer and an item), while using hyperedges among nodes to capture the influence spread process used to predict customers' future purchases. Unlike the previous influence propagation models [7], [17] consisting of only one kind of edges connecting two customers (in social influence), the hyperedges in our model span across tuples of different customers and items, capturing both effects of social influence and item inference.

Based on SIG, we first consider the social item graph of which the hyperedges are with only single destination node, and formulate the *Social Item Maximization Problem* (SIMP) to find a seed set, which consists of selected products along with targeted customers, to maximize the total adoptions of products by customers. Note that SIMP takes multiple products into consideration and targets on maximizing the number of products purchased by customers.[3] SIMP is a very challenging problem, which does not have the submodularity property. We prove that SIMP cannot be approximated within n^c with any $c < 1$, where n is the number of nodes in SIMP, i.e., SIMP is extremely difficult to approximate with a small ratio because the best approximation ratio is almost n.[4]

To tackle SIMP, two challenges arise: 1) numerous combinations of possible seed nodes, and 2) expensive on-line computation of influence diffusion upon hyperedges. To address the first issue, we first introduce the *Hyperedge-Aware Greedy (HAG)* algorithm, based on a unique property of hyperedges, i.e., a hyperedge requires all its source nodes to be activated in order to trigger the purchase action in its destination node. HAG selects multiple seeds in each seed selection iteration to further activate more nodes via hyperedges.[5] To address the second issue, we exploit the structure of Frequent Pattern Tree (FP-tree) to develop *SIG-index* as a compact representation of SIG in order to accelerate the computation of activation probabilities of nodes in online diffusion.

Afterward, we extend the model to hyperedges with multiple destination nodes. Consider the following motivating example. When Alice buys a DVD of "Star Wars", in addition to spreading her influence to Bob so that Bob purchases the DVD, it may turns out that Bob also needs a DVD player to watch it. One may argue that this can be represented by introducing two independent hyperedges: one from (Alice, DVD) to (Bob, DVD) and another from (Alice, DVD) to (Bob, DVD player). However, this is actually inappropriate since Bob has no intention to get the DVD player if he does not purchase the DVD. To support the general framework of maximizing influence spread on SIG, we first propose a number of pruning rules, namely *Inclusive Diffusion Pruning*, *Subtractive Diffusion Pruning*, *Exclusive Diffusion Pruning* and *Maximum Diffusion Pruning*, to accelerate the seed selection process by ignoring unpromising source combinations and avoiding to compute their effects on diffusion. Furthermore, we propose a multi-purposed hybrid structure, *GSIG-index Lattice*, which not only speeds up the diffusion computation in GSIG, but also tracks the inclusion relationships among the numerous source and destination combinations in GSIG. By exploiting these relationships, we propose an efficient strategy for examining the hyperedges, *Source Combination Ordering*, which can be effectively combined with the pruning strategies to enhance the efficiency of the seed selection process. Last but not least, we also propose a local search strategy, namely the *Seed Replacement Rule*, to tweak the selected seed set by making profitable modifications to the selected source combinations in each round.

Moreover, to construct a realistic SIG model for SIMP, we also develop a statistical inference based framework to learn the weights of hyperedges from logs of purchase actions. Identifying the hyperedges and estimating the corresponding weights are major challenges for constructing of an SIG due to data sparsity and unobservable activations. To address these issues, we propose a novel framework that employs smoothed expectation and maximization algorithm (EMS) [21], to identify hyperedges and estimate their values by kernel smoothing. A dimension reduction technique is also proposed to accelerate the process of finding similar hyperedges for collaborative estimation. Moreover, the number of possible hyperedge combinations for multiple destination hyperedges are large, which not only increases the computation complexity but also results in severe data sparsity problem, i.e., the number of activations for a user to buy an item is much smaller than the possible combinations. To tackle this challenge, we further propose a joint user-item mixture model and dimension reduction technique to make the SIG model workable for hyperedges with multi-destination nodes. We first formulate the EMS for hyperedges with multi-destination nodes and then present the user-item mixture model to deal with the data sparsity issue.

Our contributions of this paper are summarized as follows.

- We propose the new *Social Item Graph (SIG)* that captures both effects of social influence and item inference in the prediction of potential purchase actions. To the best of our knowledge, this is the first work considering the joint effect of social influence and iterm inference.
- Based on SIG of single destination nodes, we formulate a new problem, called *Social Item Maximization*

3. SIMP can be extended to a weighted version with different profits from each product. In this paper, we focus on maximizing the total sales.

4. While there is no good solution quality guarantee for the worst case scenario, we empirically show that the algorithm we developed achieves total adoptions on average comparable to optimal results.

5. A hyperedge requires all its source nodes to be activated to diffuse its influence to its destination node.

Problem (SIMP), to select the seed nodes for viral marketing that effectively facilitates the recommendations from both friends and stores simultaneously. In addition, we analyze the hardness of SIMP. We design an efficient algorithm with performance guarantee, called Hyperedge-Aware Greedy (HAG), and develop a new index structure, called SIG-index, to accelerate the computation of diffusion process in HAG.

- We extend the hyperedge model to the general case and formulate a seed selection problem, called *General Social Item Maximization Problem (GSIMP)*, and prove the hardness. To solve GSIMP, several pruning rules are proposed, as well as a multi-purposed hybrid structure, namely, *GSIG-index Lattice*, to accelerate the pruning process.
- To construct realistic SIG models for SIMP, we develop a statistical inference based framework to learn the weights of hyperedges from data. Moreover, to alleviate the severe data sparsity problem for general model, we propose a weight sharing method, as well as a dimension reduction technique to accelerate the process of finding similar hyperedges for collaborative estimation.
- We conduct a comprehensive evaluation on our proposals with various baselines. Experimental result validates our ideas and demonstrates the effectiveness and efficiency of the proposed model and algorithms over baselines.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 details the SIG model and its influence diffusion process. Section 4 formulates SIMP and designs new algorithms to efficiently solve the problem. Section 5 describes our approach to construct the SIG. Section 9 reports our experiment results and Section 10 concludes the paper.

2 Related Work

To discover the associations among purchased items, frequent pattern mining algorithms find items which frequently appear together in transactions [3]. Some variants, such as closed frequent patterns mining [20], maximal frequent pattern mining [16], have been studied. However, those existing works, focusing on unveiling the common shopping behaviors of individuals, disregard the social influence between customers [27]. On the other hand, it has been pointed out that items recommended by item inference may have been introduced to users by social diffusion [26]. In this work, we develop a new model and a learning framework that consider both the social influence and item inference factors jointly to derive the association among purchase actions of customers. In addition, we focus on seed selection for prevalent viral marketing by incorporating the effect of item inference.

With a great potential in business applications, social influence diffusion in social networks has attracted extensive interests recently [7], [17]. Learning algorithms for estimating the social influence strength between social customers have been developed [10], [18]. Based on models of social influence diffusion, identifying the most influential customers (seed selection) is a widely studied problem [7], [17]. Precisely, those studies aim to find the best k initial seed customers to target on in order to maximize the population of potential customers who may adopt the new product. This seed selection problem has been proved as NP-hard [17]. Based on two influence diffusion models, Independent Cascade (IC) and Linear Threshold (LT), Kempe et al. propose a $1 - 1/e$ approximation greedy algorithm by exploring the submodularity property under IC and LT [17]. Some follow-up studies focus on improving the efficiency of the greedy algorithm using various spread estimation methods, e.g., MIA [7] and TIM+ [24]. However, without considering the existence of item inference, those algorithms are not applicable to SIMP. Besides the IC and LT model, Markov random field has been used to model social influence and calculate expected profits from viral marketing [9]. Recently, Tang et al. propose a Markov model based on "confluence", which estimates the total influence by combining different sources of conformity [23]. However, these studies only consider the diffusion of a *single* item in business applications. Instead, we incorporate item inference in spread maximization to estimate the influence more accurately.

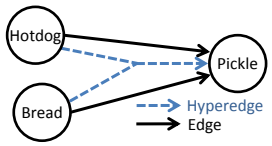


Fig. 2. A hyperedge example

3 Social Item Graph Model

Here we first present the social item graph model and then introduce the diffusion process in the proposed model.

3.1 Social Item Graph

We aim to model user purchases and potential activations of new purchase actions from some prior. We first define the notions of the social network and purchase actions.

Definition 1. A *social network* is denoted by a directed graph $G = (V, E)$ where V contains all the nodes and E contains all the directed edges in the graph. Accordingly, a social network is also referred to as a *social graph*.

Definition 2. Given a list of commodity items I and a set of customers V, a *purchase action* (or *purchase* for short), denoted by (v, i) where $v \in V$ is a customer, and $i \in I$ is an item, refers to the purchase of item i by customer v.

Definition 3. A *purchase log* is a database consisting of all the purchase actions in a given period of time.

Association-rule mining (called item inference in this paper) has been widely exploited to discover correlations between purchases in transactions. For example, the rule $\{\text{hotdog}, \text{bread}\} \rightarrow \{\text{pickle}\}$ obtained from the transactions of a supermarket indicates that if a customer buys hotdogs and bread together, she is likely to buy pickles. To model the above likelihood, the confidence of a rule $\{\text{hotdog}, \text{bread}\} \rightarrow \{\text{pickle}\}$ is the proportion of the transactions with hotdogs and bread to those also including

pickles [12]. It has been regarded as the conditional probability that a customer buying *both* hotdogs *and* bread would trigger the additional purchase of pickles. To model the above rule in a graph, a possible way is to use two separate *edges* (see Figure 2; one from hotdog to pickle, and the other from bread to pickle, respectively), while the probability associated with each of these edges is the confidence of the rule. In the above graph model, however, *either one* of the hotdog or bread may trigger the purchase of pickle. This does not accurately express the intended condition of purchasing *both* the hotdog *and* bread. By contrast, the *hyperedges* in Graph Theory, by spanning multiple source nodes and one destination node, can model the above association rule (as illustrated in Figure 2). The probability associated with the hyperedge represents the likelihood of the purchase action denoted by the destination node when *all* purchase actions denoted by source nodes have happened.

On the other hand, in viral marketing, the traditional IC model activates a new node by the social influence probabilities associated with edges to the node. Aiming to capture both effects of item inference and social influence, we propose a new *Social Item Graph* (SIG). SIG models the likelihood for a purchase (or a set of purchases) to trigger another purchase in the form of *hyperedges*, which may have one or multiple source nodes leading to one destination node. We define a social item graph as follows.

Definition 4. Given a social graph of customers $G = (V, E)$ and a commodity item list I, a *social item graph* is denoted by $G_{SI} = (V_{SI}, E_H)$, where V_{SI} is a set of purchase actions and E_H is a set of hyperedges over V_{SI}. A node $n \in V_{SI}$ is denoted as (v, i), where $v \in V$ and $i \in I$. A hyperedge $e \in E_H$ is of the following form:

$$\{(u_1, i_1), (u_2, i_2), \cdots, (u_m, i_m)\} \rightarrow (v, i)$$

where u_i is in the neighborhood of v in G, i.e., $u_i \in N_G(v) = \{u | d(u, v) \leq 1\}$.[6]

Note that the conventional social influence edge in a social graph with one source and one destination can still be modeled in an SIG as a simple edge associated with a corresponding influence probability. Nevertheless, the influence probability from a person to another can vary for different items (e.g., a person's influence on another person for cosmetics and smartphones may vary.). Moreover, although an SIG may model the purchases more accurately with the help of *both* social influence and item inference, the complexity of processing an SIG with hyperedges is much higher than simple edges in the traditional social graph that denote only social influence.

For simplicity, let u and v (i.e., the symbols in Typewriter style) represent the nodes (u, i) and (v, i) in SIG for the rest of this paper. We also denote a hyperedge as $e \equiv \mathtt{U} \rightarrow \mathtt{v}$, where U is a set of source nodes and v is the destination node. Let the associated edge weight be p_e, which represents the *activation probability* for v to be activated if all source nodes in U are activated. Note that the activation probability is for one single hyperedge $\mathtt{U} \rightarrow \mathtt{v}$. Other hyperedges sharing the same destination may have different activation probabilities. For example, part of the source nodes in a hyperedge $\{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\} \rightarrow \mathtt{x}$ can still activate x, e.g., by $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\} \rightarrow \mathtt{x}$ with a different hyperedge with its own activation probability.

3.2 Diffusion Process in Social Item Graph

Next, we introduce the diffusion process in SIG, which is inspired by the probability-based approach behind *Independent Cascade* (IC) to capture the word-of-mouth behavior in the real world [7].[7] This diffusion process fits the item inferences captured in an SIG naturally, as we can derive conditional probabilities on hyperedges to describe the trigger (activation) of purchase actions on a potential purchase.

The diffusion process in SIG starts with all nodes inactive initially. Let S denote a set of seeds (purchase actions). Let a node $\mathtt{s} \in \mathtt{S}$ be a seed. It immediately becomes active. Given all the nodes in a source set U at iteration $\iota - 1$, if they are all active at iteration ι, a hyperedge $e \equiv \mathtt{U} \rightarrow \mathtt{v}$ has a chance to activate the inactive v with probability p_e. Each node (v, i) can be activated once, but it can try to activate other nodes multiple times, one for each incident hyperedges. For the seed selection problem that we target on, the total number of activated nodes represents the number of items adopted by customers (called *total adoptions* for the rest of this paper).

4 Social Item Maximization

Upon the proposed Social Item Graph (SIG), we now formulate a new seed selection problem, called *Social Item Maximization Problem* (SIMP), that selects a set of seed purchase actions to maximize potential sales or revenue in a marketing campaign. In Section 5, we will describe how to construct the SIG from purchase logs by a machine learning approach.

Definition 5. Given a seed number k, a list of targeted items I, and a social item graph $G_{SI}(V_{SI}, E_H)$, SIMP selects a set S of k seeds in V_{SI} such that $\alpha_{G_{SI}}(\mathtt{S})$, **the *total adoption* function of S**, is maximized.

Note that a seed in SIG represents the adoption/purchase action of a specific item by a particular customer. The total adoption function $\alpha_{G_{SI}}$ represents the total number of product items ($\in I$) purchased. By assigning prices to products and costs to the selected seeds, an extension of SIMP is to maximize the total revenue subtracted by the cost.

Here we first discuss the challenges in solving SIMP before introducing our algorithm. Note that, for the influence maximization problem based on the IC model, Kempe et al. propose a $1 - 1/e$ approximation algorithm [17], thanks to the submodularity in the problem. Unfortunately, the submodularity does not hold for the total adoption function $\alpha_{G_{SI}}(\mathtt{S})$ in SIMP. Specifically, if the function $\alpha_{G_{SI}}(\mathtt{S})$ satisfies the submodularity, for any node i and any two subsets of nodes $\mathtt{S}_1$ and $\mathtt{S}_2$ where $\mathtt{S}_1 \subseteq \mathtt{S}_2$, $\alpha_{G_{SI}}(\mathtt{S}_1 \bigcup \{\mathtt{i}\}) - \alpha_{G_{SI}}(\mathtt{S}_1) \geq \alpha_{G_{SI}}(\mathtt{S}_2 \bigcup \{\mathtt{i}\}) - \alpha_{G_{SI}}(\mathtt{S}_2)$ should hold. However, a counter example is illustrated below.

6. Notice that when $u_1 = u_2 = \cdots = u_m = v$, the hyperedge represents the item inference of item i. On the other hand, when $i_1 = i_2 = \cdots = i_m = i$, it becomes the social influence of user u on v.

7. Several variants of IC model have been proposed [5], [6]. However, they focus on modeling the diffusion process between users, such as aspect awareness [6], which is not suitable for social item graph since the topic is embedded in each SIG node.

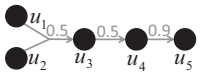


Fig. 3. A non-submodular example

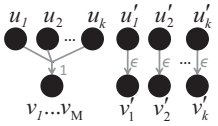


Fig. 4. An example of SIMP

(a) Original (b) Transformed

Fig. 5. An illustration of graph transformations

Example 1. Consider an SIMP instance with a customer and five items in Figure 3. Consider the case where $\mathtt{S}_1 = \{\mathtt{u}_4\}$, $\mathtt{S}_2 = \{\mathtt{u}_1, \mathtt{u}_4\}$, and $\mathtt{i}$ corresponds to node $\mathtt{u}_2$. For seed sets $\{\mathtt{u}_4\}$, $\{\mathtt{u}_2, \mathtt{u}_4\}$, $\{\mathtt{u}_1, \mathtt{u}_4\}$ and $\{\mathtt{u}_1, \mathtt{u}_2, \mathtt{u}_4\}$, $\alpha_{G_{SI}}(\{\mathtt{u}_4\}) = 1.9$, $\alpha_{G_{SI}}(\{\mathtt{u}_2, \mathtt{u}_4\}) = 2.9$, $\alpha_{G_{SI}}(\{\mathtt{u}_1, \mathtt{u}_4\}) = 2.9$, and $\alpha_{G_{SI}}(\{\mathtt{u}_1, \mathtt{u}_2, \mathtt{u}_4\}) = 4.4$. Thus, $\alpha_{G_{SI}}(\mathtt{S}_1 \bigcup \{\mathtt{u}_2\}) - \alpha_{G_{SI}}(\mathtt{S}_1) = 1 < 1.5 = \alpha_{G_{SI}}(\mathtt{S}_2 \bigcup \{\mathtt{u}_2\}) - \alpha_{G_{SI}}(\mathtt{S}_2)$. Hence, the submodularity does not hold.

Since the submodularity does not exist in SIMP, the $1 - 1/e$ approximation ratio of the greedy algorithm in [17] does not hold. Now, an interesting question is how large the ratio becomes. Example 2 shows an SIMP instance where the greedy algorithm performs poorly.

Example 2. Consider an example in Figure 4, where nodes $\mathtt{v}_1$, $\mathtt{v}_2$,...,$\mathtt{v}_M$ all have a hyperedge with the probability as 1 from the same k sources $\mathtt{u}_1$, $\mathtt{u}_2$,..., $\mathtt{u}_k$, and ϵ is an arbitrarily small edge probability $\epsilon > 0$. The greedy algorithm selects one node in each iteration, i.e., it selects $\mathtt{u}'_1$, $\mathtt{u}'_2$...$\mathtt{u}'_k$ as the seeds with a total adoption $k + k\epsilon$. However, the optimal solution actually selects $\mathtt{u}_1$, $\mathtt{u}_2$,..., $\mathtt{u}_k$ as the seeds and results in the total adoption $M + k$. Therefore, the approximation ratio of the greedy algorithm is at least $(M + k)/(k + k\epsilon)$, which is close to M/k for a large M, where M could approach $|V_{SI}|$ in the worst case.

One may argue that the above challenges in SIMP may be alleviated by transforming G_{SI} into a graph with only simple edges, as displayed in Figure 5, where the weight of every $\mathtt{u}_i \rightarrow \mathtt{v}$) with $\mathtt{u}_i \in \mathtt{U}$ can be set independently. However, if a source node $\mathtt{u}_m \in \mathtt{U}$ of $\mathtt{v}$ is difficult to activate, the probability for $\mathtt{v}$ to be activated approaches zero in Figure 5 (a) due to $\mathtt{u}_m$. However, in Figure 5 (b), the destination $\mathtt{v}$ is inclined to be activated by sources in $\mathtt{U}$, especially when $\mathtt{U}$ is sufficiently large. Thus, the idea of graph transformation does not work.

4.1 Hyperedge-Aware Greedy (HAG)

Here, we propose an algorithm for SIMP, *Hyperedge-Aware Greedy (HAG)*, with performance guarantee. The approximation ratio is proved in Section 4.3. A hyperedge requires all its sources activated first in order to activate the destination. Conventional single node greedy algorithms perform poorly because hyperedges are not considered. To address this important issue, we propose *Hyperedge-Aware Greedy (HAG)* to select multiple seeds in each iteration.

A naive algorithm for SIMP would examine $C_k^{|V_{SI}|}$ combinations to choose k seeds. In this paper, as all source nodes of a hyperedge need to be activated in order to activate its destination, an effective way is to consider only the combinations which include the source nodes of any hyperedge. We call the source nodes of a hyperedge as a *source combination*. Based on this idea, in each iteration, HAG includes the source combination leading to *the largest increment on total adoption divided by the number of new seeds added in this iteration*. Note that only the source combinations with no more than k sources are considered. The iteration continues until k seeds are selected. Note that HAG does not restrict the seeds to be the source nodes of hyperedges. Instead, the source node $\mathtt{u}$ of any simple edge $\mathtt{u} \rightarrow \mathtt{v}$ in SIG is also examined.

Complexity of HAG. To select k seeds, HAG takes at most k rounds. In each round, the source combinations of $|E_H|$ hyperedges are tried one by one, and the diffusion cost is c_{dif}, which will be analyzed in Section 4.2. Thus, the time complexity of HAG is $O(k \times |E_H| \times c_{dif})$.

4.2 Acceleration of Diffusion Computation

To estimate the total adoption for a seed set, it is necessary to perform Monte Carlo simulation based on the diffusion process described in Section 3.2 for many times. Finding the total adoption is very expensive, especially when a node $\mathtt{v}$ can be activated by a hyperedge with a large source set $\mathtt{U}$, which indicates that there also exist many other hyperedges with an arbitrary subset of $\mathtt{U}$ as the source set to activate $\mathtt{v}$. In other words, enormous hyperedges need to be examined for the diffusion on an SIG. It is essential to reduce the computational overhead. To address this issue, we propose a new index structure, called *SIG-index*, by exploiting FP-Tree [12] to pre-process source combinations in hyperedges in a compact form in order to facilitate efficient derivation of activation probabilities during the diffusion process.

The basic idea behind SIG-index is as follows. For each node $\mathtt{v}$ with the set of activated in-neighbors $N^a_{\mathtt{v},\iota}$ in iteration ι, if $\mathtt{v}$ has not been activated before ι, the diffusion process will try to activate $\mathtt{v}$ via every hyperedge $\mathtt{U} \rightarrow \mathtt{v}$ where the last source in $\mathtt{U}$ has been activated in iteration $\iota - 1$. To derive the activation probability of a node $\mathtt{v}$ from the weights of hyperedges associated with $\mathtt{v}$, we first define the activation probability as follows.

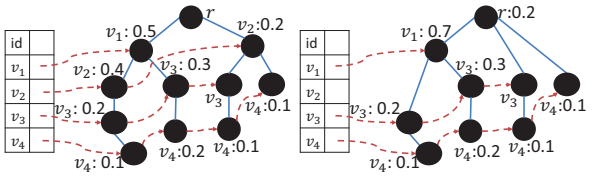
Definition 6. The activation probability of $\mathtt{v}$ at ι is

$$ap_{\mathtt{v},\iota} = 1 - \prod_{\mathtt{U} \rightarrow \mathtt{v} \in E_H, \mathtt{U} \subseteq N_{\mathtt{v},\iota-1}, \mathtt{U} \nsubseteq N_{\mathtt{v},\iota-2}} (1 - p_{\mathtt{U} \rightarrow \mathtt{v}}).$$

where $N_{\mathtt{v},\iota-1}$ and $N_{\mathtt{v},\iota-2}$ denote the set of active neighbors of $\mathtt{v}$ in iteration $\iota - 1$ and $\iota - 2$, respectively.

The operations on an SIG-index occur two phases: Index Creation Phase and Diffusion Processing Phase. As all hyperedges satisfying Definition 6 must be accessed, the SIG-index stores the hyperedge probabilities in Index Creation Phase. Later, the SIG-index is updated in Diffusion Processing Phase to derive the activation probability efficiently.

Index Creation Phase. For each hyperedge $\mathtt{U} \rightarrow \mathtt{v}$, we first regard each source combination $\mathtt{U} = \{\mathtt{v}_1, ...\mathtt{v}_{|\mathtt{U}|}\}$



(a) Initial (b) After v_2 is activated

Fig. 6. An illustration of SIG-index

as a transaction to build an FP-tree [12] by setting the minimum support as 1. As such, $v_1, ...v_{|U|}$ forms a path $r \rightarrow v_1 \rightarrow v_2... \rightarrow v_{|U|}$ from the root r in the FP-tree to node $v_{|U|}$ in U. Different from the FP-Tree, the SIG-index associates the probability of each hyperedge $U \rightarrow v$ with the last source node $v_{|U|}$ in U.[8] Initially, the probability associated with the root r is 0. Later the SIG-index is updated during the diffusion process. Example 3 illustrates the SIG-index created based on an SIG.

Example 3. Consider an SIG graph with five nodes, v_1-v_5, and nine hyperedges with their associated probabilities in parentheses: $\{v_1\} \rightarrow v_5$ (0.5), $\{v_1, v_2\} \rightarrow v_5$ (0.4), $\{v_1, v_2, v_3\} \rightarrow v_5$ (0.2), $\{v_1, v_2, v_3, v_4\} \rightarrow v_5$ (0.1), $\{v_1, v_3\} \rightarrow v_5$ (0.3), $\{v_1, v_3, v_4\} \rightarrow v_5$ (0.2), $\{v_2\} \rightarrow v_5$ (0.2), $\{v_2, v_3, v_4\} \rightarrow v_5$ (0.1), $\{v_2, v_4\} \rightarrow v_5$ (0.1). Figure 6 (a) shows the SIG-index initially created for node v_5.

Diffusion Processing Phase. The activation probability in an iteration is derived by traversing the initial SIG-index, which takes $O(|E_H|)$ time. However, a simulation may iterate a lot of times. To further accelerate the traversing process, we adjust the SIG-index for the activated nodes in each iteration. More specifically, after a node v^a is activated, accessing an hyperedge $U \rightarrow v$ with $v^a \in U$ becomes easier since the number remaining inactivated nodes in $U - \{v^a\}$ is reduced. Accordingly, SIG-index is modified by traversing every vertex labeled as v^a on the SIG-index in the following steps. 1) If v^a is associated with a probability p_a, it is crucial to aggregate the old activation probabilities p_a of v^a and p_p of its parent v^p, and update activation probability associated with v^p as $1-(1-p_a)(1-p_p)$, since the source combination needed for accessing the hyperedges associated with v^a and v^p becomes the same. The aggregation is also performed when v^p is r. 2) If v^a has any children c, the parent of c is changed to be v^p, which removes the processed v^a from the index. 3) After processing every node v^a in the SIG-index, we obtain the activation probability of v in the root r. After the probability is accessed for activating v, the probability of r is reset to 0 for next iteration.

Example 4. Consider an example with v_2 activated in an iteration. To update the SIG-index, each vertex v_2 in Figure 6 (a) is examined by traversing the linked list of v_2. First, the left vertex with label v_2 is examined. SIG-index reassigns the parent of v_2's child (labeled as v_3) to the vertex labeled as v_1, and aggregate the probability 0.4 on the v_2 and 0.5 on vertex v_1, since the hyperedge $\{v_1, v_2\} \rightarrow v_5$ can be accessed if the node v_1 is activated later. The probability of v_1 becomes $1-(1-p_{v_1})(1-p_{v_2}) = 0.7$. Then the right vertex with label v_2 is examined. The parent of its two children is reassigned to the root r. Also, the probability of itself (0.2) is aggregated with the root r, indicating that the activation probability of node v_5 in the next iteration is 0.2.

Complexity Analysis. For Index Creation Phase, the initial SIG-index for v is built by examining the hyperedges two times with $O(|E_H|)$ time. The number of vertices in SIG-index is at most $O(c_d|E_H|)$, where c_d is the number of source nodes in the largest hyperedge. During Diffusion Processing Phase, each vertex in SIG-index is examined only once through the node-links, and the parent of a vertex is changed at most $O(c_d)$ times. Thus, the overall time to complete a diffusion requires at most $O(c_d|E_H|)$ time.

4.3 Hardness Results

From the discussion earlier, it becomes obvious that SIMP is difficult. In the following, we will prove that SIMP is inapproximable with a non-constant ratio n^c for all $c < 1$, with a gap-introducing reduction from an NP-complete problem 3-SAT to SIMP, where n is the number of nodes in an SIG. Note that the theoretical result only shows that for any algorithm, there exists a problem instance of SIMP (i.e., a pair of an SIG graph and a seed number k) that the algorithm can not obtain a solution better than $1/n$ times the optimal solution. It does not imply that an algorithm always performs badly in every SIMP instance. Later in Section 9.3, we empirically show that the total adoption obtained by HAG is comparable to the optimal solution.

Lemma 1. For a positive integer q, there is a gap-introducing reduction from 3-SAT to SIMP, which transforms an n_{var}-variables expression ϕ to an SIMP instance with the SIG as $G_{SI}(V_{SI}, E_H)$ and the k as n_{var} such that
- if ϕ is satisfiable, $\alpha^*_{G_{SI}} \geq (m_{cla} + 3n_{var})^q$, and
- if ϕ is not satisfiable, $\alpha^*_{G_{SI}} < m_{cla} + 3n_{var}$,

where $\alpha^*_{G_{SI}}$ is the optimal solution of this instance, n_{var} is the number of Boolean variables, and m_{cla} is the number of clauses. Hence there is no $(m_{cla} + 3n_{var})^{q-1}$ approximation algorithm for SIMP unless P = NP.

Proof 1. Given a positive integer q, for an instance ϕ of 3-SAT with n_{var} Boolean variables $a_1, \ldots, a_{n_{var}}$ and m_{cla} clauses $C_1, \ldots, C_{m_{cla}}$, we construct an SIG G_{SI} with three node sets X, Y and Z as follows. 1) Each Boolean variable a_i corresponds to two nodes x_i, $\overline{x}_i$ in X and one node y_i in Y. 2) Each clause C_k corresponds to one node c_k in Y. 3) Z has $(|X| + |Y|)^q$ nodes. (Thus, G_{SI} has $(m_{cla} + 3n_{var})^q + m_{cla} + 3n_{var}$ nodes.) 4) For each y_j in Y, we add direct edges $x_j \rightarrow y_j$ and $\overline{x}_j \rightarrow y_j$. 5) For each c_k in Y, we add direct edges $\alpha \rightarrow c_k$, $\beta \rightarrow c_k$ and $\gamma \rightarrow c_k$, where α, β, γ are the nodes in X corresponding to the three literals in C_k. 6) We add a hyperedge $Y \rightarrow z_v$ from all for every $z_v \in Z$. The probability of every edge is set to 1. An example is illustrated in Figure 7.

We first prove that ϕ is satisfiable if and only if G_{SI} has a seed set S with n_{var} seeds and the total adoption of S contains Y. If ϕ is satisfiable, there exists a truth assignment T on Boolean variables $a_1, \ldots, a_{n_{var}}$ satisfying all clauses of ϕ. Let $S = \{x_i | T(a_i) = 1\} \cup \{\overline{x}_j | T(a_j) = 0\}$,

8. For ease of explanation, we assume the order of nodes in the SIG-index follows the ascending order of subscript.

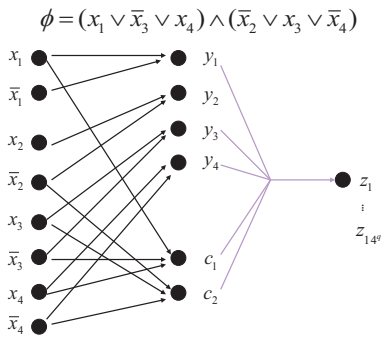


Fig. 7. An illustration instance built for 3-SAT

and S then has n_{var} nodes and the total adoption of S contains Y. On the other hand, if ϕ is not satisfiable, apparently there exists no seed set S with exactly one of $\mathtt{x}_i$ or $\overline{\mathtt{x}}_i$ selected for every i such that the total adoption of S contains Y. For other cases, 1) all seeds are placed in X, but there exists at least one i with both $\mathtt{x}_i$ and $\overline{\mathtt{x}}_i$ selected. In this case, there must exist some j such that none of $\mathtt{x}_j$ or $\overline{\mathtt{x}}_j$ are selcted (since the seed number is n_{var}), and thus Y is not covered by the total adoption of S. 2) A seed is placed in Y. In this case, the seed can be moved to an adjacent $\mathtt{x}_i$ without reducing the total adoption. Nevertheless, as explained above, there exists no seed set S with all seeds placed in X such that the total adoption of S contains Y, and thus the total adoption of any seed set with a seed placed in Y cannot cover Y, either. With above observations, if ϕ is not satisfiable, G_{SI} does not have a seed set S with n_{var} seeds such that the total adoption of S contains Y. Since the nodes of Z can be activated if and only if the total adoption of S contains Y if and only if ϕ is satisfiable, we have

- if ϕ is satisfiable, $\alpha^*_{G_{SI}} \geq (m_{cla} + 3n_{var})^q$, and
- if ϕ is not satisfiable, $\alpha^*_{G_{SI}} < m_{cla} + 3n_{var}$.

The lemma follows.

Theorem 1. For any $\epsilon > 0$, there is no $n^{1-\epsilon}$ approximation algorithm for SIMP, assuming P $\neq$ NP.

Proof 2. For any arbitrary $\epsilon > 0$, we set $q \geq \frac{2}{\epsilon}$. Then, by Lemma 1, there is no $(m_{cla} + 3n_{var})^{q-1}$ approximation algorithm for SIMP unless P = NP. Then $(m_{cla} + 3n_{var})^{q-1} \geq 2(m_{cla} + 3n_{var})^{q-2} \geq 2(m_{cla} + 3n_{var})^{q(1-\epsilon)} \geq (2(m_{cla} + 3n_{var})^q)^{1-\epsilon} \geq n^{1-\epsilon}$. Since ϵ is arbitrarily small, thus for any $\epsilon > 0$, there is no $n^{1-\epsilon}$ approximation algorithm for SIMP, assuming P $\neq$ NP. The theorem follows.

Theorem 2. HAG with SIG-index is n-approximated, where n is the number of nodes in SIG.

Proof 3. First, we prove that SIG-index obtains $ap_{\mathtt{v},\iota}$ correctly. Assume that there exists an incorrect $ap_{\mathtt{v},\iota}$, i.e., there exists an hyperedge $\mathtt{U} \rightarrow \mathtt{v}$ satisfying the conditions in Definition 6 (i.e., $\mathtt{U} \nsubseteq N_{\mathtt{v},\iota-2}$ and $\mathtt{U} \subseteq N_{\mathtt{v},\iota-1}$) but its probability is not aggregated to r in ι. However, the probability can not be aggregated before ι since $\mathtt{U} \nsubseteq N_{v,\iota-2}$ and it must be aggregated no later than ι since $\mathtt{U} \subseteq N_{\mathtt{v},\iota-1}$. There is a contradiction.

Proving that HAG with SIG-index is an n-approximation algorithm is simple. The upper bound of total adoption for the optimal algorithm is n, while the lower bound of the total adoption for HAG is 1 because at least one seed is selected. In other words, designing an approximation algorithm for SIMP is simple, but it is much more difficult to have the hardness result for SIMP, and we have proven that SIMP is inapproximable within $n^{1-\epsilon}$ for any arbitrarily small ϵ.

Corollary 1. HAG with SIG-index is n-approximated, where n is the number of nodes in SIG, because SIG-index only improves the efficiency.

5 Construction of SIG

To select seeds for SIMP, we need to construct the SIG from purchase logs and the social network. We first create possible hyperedges by scanning the purchase logs. Let τ be the timestamp of a given purchase $\mathtt{v} = (v, i)$. v's friends purchase and her own purchases that have happened within a given period before τ are considered as candidate source nodes to generate hyperedges to v. For each hyperedge e, the main task is then the estimation of its activation probability p_e. Since p_e is unknown, it is estimated by maximizing the likelihood function based on observations in the purchase logs. Note that learning the activation probability p_e for each hyperedge e faces three challenges.

C1. Unknown distribution of p_e. How to properly model p_e is critical.

C2. Unobserved activations. When v is activated at time τ, this event only implies that at least one hyperedge successfully activates v before τ. It remains unknown which hyperedge(s) actually triggers v, i.e., it may be caused by either the item inference or social influence or both. Therefore, we cannot simply employ the confidence of an association-rule as the corresponding hyperedge probability.

C3. Data sparsity. The number of activations for a user to buy an item is small, whereas the number of possible hyperedge combinations is large. Moreover, new items emerge every day in e-commerce websites, which incurs the notorious cold-start problem. Hence, a method to deal with the data sparsity issue is necessary to properly model an SIG.

To address these challenges, we exploit a statistical inference approach to identify those hyperedges and learn their weights. In the following, we first propose a model of the edge function (to address the first challenge) and then exploit the smoothed expectation and maximization (EMS) algorithm [21] to address the second and third challenges.

5.1 Modeling of Hyperedge Probability

To overcome the first challenge, one possible way is to model the number of success activations and the number of unsuccessful activations by the binomial distributions. As such, p_e is approximated by the ratio of the number of success activations and the number of total activation trials. However, the binomial distribution function is too complex for computing the maximum likelihood of a vast number of data. To handle big data, the previous study reported [14] that the binomial distribution (n, p) can be approximated

by the Poisson distribution $\lambda = np$ when the time duration is sufficiently large. According to the above study, it is assumed that the number of activations of a hyperedge e follows the Poisson distribution to handle the social influence and item inference jointly. The expected number of events equals to the intensity parameter λ. Moreover, we use an inhomogeneous Poisson process defined on the space of hyperedges to ensure that p_e varies with different e.

In the following, a hyperedge is of size n, if the cardinality of its source set $\mathtt{U}$ is n. We denote the intensity of the number of activation trials of the hyperedge e as $\lambda_T(e)$. Then the successful activations of hyperedge e follows another Poisson process where the intensity is denoted by $\lambda_A(e)$. Therefore, the hyperedge probability p_e can be derived by parameters $\lambda_A(e)$ and $\lambda_T(e)$, i.e., $p_e = \frac{\lambda_A(e)}{\lambda_T(e)}$.

The maximum likelihood estimation can be employed to derive $\lambda_T(e)$. Nevertheless, $\lambda_A(e)$ cannot be derived as explained in the second challenge. Therefore, we use the expectation maximization (EM) algorithm, which is an extension of maximum likelihood estimation containing latent variables to $\lambda_A(e)$ which is modeled as the latent variable. Based on the observed purchase logs, the E-step first derives the likelihood Q-function of the parameter p_e with $\lambda_A(e)$ as the latent variables. In this step, the purchase logs and p_e are given to find the probability function describing that all events on e in the logs occur according to p_e, whereas the probability function (i.e., Q-function) explores different possible values on latent variable $\lambda_A(e)$. Afterward, The M-step maximizes the Q-function and derives the new p_e for E-Step in the next iteration. These two steps are iterated until convergence.

With the employed Poisson distribution and EM algorithm, data sparsity remains an issue. Therefore, we further exploit a variant of EM algorithm, called *EMS algorithm* [21], to alleviate the sparsity problem by estimating the intensity of Poisson process using *similar hyperedges*. The parameter smoothing after each iteration is called S-Step, which is incorporated in EMS algorithm, in addition to the existing E-Step and M-Step.

5.2 Model Learning by EMS Algorithm

Let p_e and $\hat{p}_e$ denote the true probability and estimated probabilities for hyperedge e in the EMS algorithm, respectively, where $e = U \to v$. Let N_U and K_e denote the number of activations of source set U in the purchase logs and the number of successful activations on hyperedge e, respectively. The EM algorithm is exploited to find the maximum likelihood of p_e, while $\lambda_A(e)$ is the latent variable because K_e cannot be observed (i.e., only N_U can be observed). Therefore, E-Step derives the likelihood function for $\{p_e\}$ (i.e., the Q-function) as follows,

$$Q(p_e, \hat{p}_e^{(i-1)}) = E_{K_e}[\log P(K_e, N_U|p_e)|N_U, \hat{p}_e^{(i-1)}], \quad (1)$$

where $\hat{p}_e^{(i-1)}$ is the hyperedge probability derived in the previous iteration. Note that N_U and $p_e^{(i-1)}$ are given parameters in iteration i, whereas p_e is a variable in the Q-function, and K_e is a random variable governed by the distribution $P(K_e|N_U, p_e^{(i-1)})$. Since p_e is correlated to $\lambda_T(U)$ and $\lambda_A(e)$, we derive the likelihood $P(K_e, N_U|p_e)$ as follows.

$$\begin{aligned} & P(K_e, N_U|p_e) \\ = \; & P\left(\{K_e\}_{e\in E_H}, \{N_U\}_{U\subseteq V_{SI}}|\{p_e\}_{e\in E_H}, \{\lambda_T(U)\}_{U\subseteq V_{SI}}\right) \\ = \; & P\left(\{K_e\}_{e\in E_H}|\{p_e\}_{e\in E_H}, \{N_U, \lambda_T(U)\}_{U\subseteq V_{SI}}\right) \\ & \times P\left(\{N_U\}_{U\subseteq V_{SI}}|\{\lambda_T(U)\}_{U\subseteq V_{SI}}\right). \end{aligned}$$

It is assumed that $\{K_e\}$ is independent with $\{N_U\}$, and $Q(p_e, \hat{p}_e^{(i-1)})$ can be derived as follows:

$$\sum_{e\in E_H} \log P(K_e|N_U, p_e) + \log P(\{N_U\}_{U\subseteq V_{SI}}|\{\lambda_T(e)\}_{U\subseteq V_{SI}}).$$

Since only the first term contains the hidden K_e, only this term varies in different iterations of the EMS algorithm, because $\{N_U\}_{U\subseteq V_{SI}}$ in the second term always can be derived by finding the maximum likelihood as follows. Let $p_{U,k}$ denote the probability that the source set U exactly tries to activate the destination node k times, i.e., $p_{U,k} = P\{N_U = k\}$. The log-likelihood of λ_T is

$$\begin{aligned} & \sum_k p_{U,k} \ln\left(\frac{\lambda_T^k e^{-\lambda_T}}{k!}\right) = \sum_k p_{U,k}(-\lambda_T + k\ln\lambda_T - \ln k!) \\ = \; & -\lambda_T + (\ln\lambda_T)\sum_k k p_{U,k} - \sum_k p_{U,k}\ln k!. \end{aligned}$$

We acquire the maximum likelihood by finding the derivative with regard to λ_T:

$$-1 + \frac{1}{\lambda_T}\sum_k k p_{U,k} = 0. \quad (2)$$

Thus, the maximum log-likelihood estimation of $\lambda_T = \sum_k k p_{U,k}$, representing that the expected activation times (i.e., $\hat{\lambda}_T(e)$) is N_U. Let $\mathcal{A} = \{(\mathtt{v}, \tau)\}$ denote the action log set, where each log $(\mathtt{v}, \tau)$ represents that $\mathtt{v}$ is activated at time τ. N_U is calculated by scanning $\mathcal{A}$ and find the times that all the nodes in $\mathtt{U}$ are activated.

Afterward, we focus on the first term of $Q(p_e, \hat{p}_e^{(i-1)})$. Let $p_{e,k} = P\{K_e = k\}$ denote the probability that the hyperedge e exactly activates the destination node k times. In E-step, we first find the expectation for K_e as follows.

$$\begin{aligned} & \sum_{e\in E_H}\sum_{k=1,\cdots,N_U} p_{e,k}\log(k|N_U, p_e) \\ = \; & \sum_{k=1,\cdots,N_U} p_{e,k}\log P(k|N_U, p_e) \\ = \; & \sum_{k=1,\cdots,N_U} p_{e,k}\log\left(\binom{N_U}{k} {p_e}^k (1-p_e)^{N_U-k}\right) \\ = \; & \sum_{k=1,\cdots,N_U} p_{e,k}\left(\log\binom{N_U}{k} + k\log p_e + (N_U - k)\log(1-p_e)\right). \end{aligned}$$

Since $\sum_{k=1,\cdots,N_U} p_{e,k}k = E[K_e]$ and $\sum_{k=1,\cdots,N_U} p_{e,k} = 1$, the log-likelihood of the first term is further simplified as

$$\sum_{k=1,\cdots,N_U} p_{e,k}\log\binom{N_U}{k} + N_U\log(1-p_e) + E[K_e](\log p_e - \log(1-p_e)).$$

Afterward, M-step maximizes the Q-function by finding the derivative with regard to p_e:

$$\begin{aligned} & \frac{-N_U}{1-p_e} + E[K_e]\left(\frac{1}{p_e} + \frac{1}{1-p_e}\right) = 0 \\ & p_e = E[K_e]/N_U \end{aligned}$$

Therefore, the maximum likelihood estimator $\hat{p}_e$ is $\frac{E[K_e]}{N_U}$, $\hat{\lambda}_T(U)$ is N_U, and $\hat{\lambda}_A(e) = E[K_e]$.

The problem remaining is to take expectation of the latent variables $\{K_e\}$ in E-step. Let $\{w_{e,a}\}_{e\in E_H, a=(\mathtt{v},\tau)\in\mathcal{A}}$ be the conditional probability that $\mathtt{v}$ is activated by the source set U of e at τ given $\mathtt{v}$ is activated at τ, and let E_a denote the set of candidate hyperedges containing every possible e with its source set activated at time $\tau-1$, i.e., $E_a = \{(\mathtt{u}_1, \mathtt{u}_2, \cdots, \mathtt{u}_n) \to \mathtt{v}_i | \forall i = 1, \cdots, n, \mathtt{u}_i \in N(\mathtt{v}_i), (\mathtt{u}_i, \tau - 1) \in \mathcal{A}\}$. It's easy to show that given the estimation of the probability of hyperedges, $w_{e,a} = \frac{\hat{p}_e}{1-\prod_{e'\in E_a}(1-\hat{p}_{e'})}$, since $1 - \prod_{e\in E_a}(1-\hat{p}_{e'})$ is the probability for $\mathtt{v}$ to be activated by any hyperedge at time τ. The expectation of K_e is $\sum_{a\in A, e\in E_a\cap E_{H,n}} w_{e,a}$, i.e., the sum of expectation of each successful activation of $\mathtt{v}$ from hyperedge e, and $E_{H,n} = \{(\mathtt{u}_1, \mathtt{u}_2, \cdots, \mathtt{u}_n; v) \in E_H\}$ contains all size n hyperedges.

To address the data sparsity problem, we leverage information from similar hyperedges (described later). Therefore, our framework includes an additional step to smooth the results of M-Step. Kernel smoothing is employed in S-Step. In summary, we have the following steps:

E-Step:

$$E[K_e] = \sum_{a\in A, e\in E_a\cap E_{H,n}} w_{e,a},$$
$$w_{e,a} = \frac{\hat{p}_e}{1-\prod_{e'\in E_a}(1-\hat{p}_{e'})}.$$

M-Step:

$$p_e = \frac{\sum_{a\in A, e\in E_a\cap E_{H,n}} w_{e,a}}{N_U},$$
$$\lambda_A(e) = \sum_{a\in A, e\in E_a\cap E_{H,n}} w_{e,a},$$
$$\lambda_T(U) = N_U.$$

S-Step: To address the data sparsity problem, we leverage information from similar hyperedges (described later). Therefore, in addition to E-Step and M-Step, EMS includes **S-Step**, which smooths the results of M-Step. Kernel smoothing is employed in S-Step as follows:

$$\hat{\lambda}_A(e) = \sum_{a\in A, e'\in E_a\cap E_{H,n}} w_{e',a} L_h(F(e) - F(e'))$$
$$\hat{\lambda}_T(U) = \sum_{U\subseteq V_{SI}} N_U L_h(F(U) - F(U'))$$

where L_h is a kernel function with bandwidth h, and F is the mapping function of hyperedges, i.e., $F(e)$ maps a hyperedge e to a vector. The details of dimension reduction for calculating F to efficiently map hyperedges into Euclidean space are shown in [2] due to space constraint. If the hyperedges e and e' are similar, the distance of the vectors $F(e)$ and $F(e')$ is small. Moreover, a kernel function $L_h(x)$ is a positive function symmetric at zero which decreases when $|x|$ increases, and the bandwidth h controls the extent of auxiliary information taken from similar hyperedges.[9] Intuitively, kernel smoothing can identify the correlation of $\hat{p}_{e_1}$ with $e_1 = \mathtt{U}_1 \to \mathtt{v}_1$ and $\hat{p}_{e_2}$ with $e_2 = \mathtt{U}_2 \to \mathtt{v}_2$ for nearby $\mathtt{v}_1$ and $\mathtt{v}_2$ and similar $\mathtt{U}_1$ and $\mathtt{U}_2$.

9. A symmetric Gaussian kernel function is often used [13].

6 General Social Item Graph Model

Journal extension starts here.

In this section, we present the *General Social Item Graph* (GSIG) model and its diffusion process, to further capture the general item inference and social influence effects in viral marketing.

6.1 General Social Item Graph

In the basic SIG model, each hyperedge can have only a single destination node. Nevertheless, this setting can not appropriately represent all item inference and social influence behaviors. Consider the following two scenarios related to the motivating example.

(i) When Alice buys a DVD of "Star Wars", in addition to spreading her influence to Bob so that Bob purchases the DVD, it may turns out that Bob also needs a DVD player to watch it. One may argue that this can be represented by introducing two independent hyperedges: one from (Alice, DVD) to (Bob, DVD) and another from (Alice, DVD) to (Bob, DVD player). However, this is actually inappropriate since Bob has no intention to get the DVD player if he does not purchase the DVD.

(ii) Suppose Bob is a cinematics lecturer. If he adopts Star Wars for course material, all his students would likely purchase the DVD. This, again, can not be properly represented by separated hyperedges, since the relations are correlated. Motivated by the above, we first propose the general model as follows.

Definition 7. A *general social item graph* is denoted by $G_{GSI} = (V_{GSI}, E_G)$, where V_{GSI} is still a set of purchase actions, and E_G is a set of *multi-source multi-destination hyperedges* over V_{GSI}. A hyperedge $e \in E_G$ is of the following form:

$$\{(u_1, i_1), \cdots, (u_{n_s}, i_{n_s})\} \to \{(v_1, i'_1), \cdots, (v_{n_d}, i'_{n_d})\}$$

where each u_i is in the neighborhood of some v_j in G, i.e., $u_i \in N_G(v_j) = \{u | d(u, v_j) \le 1\}$ for some j.

Note that this general model contains both the conventional social influence edge (one source, one destination) and the basic SIG hyperedges (multiple source, one destination).

An issue here is learning the hyperedges and probabilities in practice. Due to data sparsity, it is often hard to detect or derive good estimations for hyperedges with a large size of source and/or destination nodes. Therefore, we may additionally introduce a limit on the size of n_s and n_d.

Definition 8. A *$(\bar{n_s}, \bar{n_d})$-degree-bounded general social item graph* is a general social item graph in which each hyperedge e has at most $\bar{n_s}$ source nodes and at most $\bar{n_d}$ destination nodes.

Analogously to that in SIG, we also denote a GSIG hyperedge as $e \equiv \mathtt{U} \to \mathtt{V}$, where $\mathtt{U}$ is the set of source nodes and $\mathtt{V}$ is the set of destination nodes. The *activation probability* p_e now stands for the chance that *all destination nodes* in $\mathtt{V}$ are activated if all source nodes in $\mathtt{U}$ are activated.

TABLE 1
Comparison of precision, recall, and F1 for three models on Douban, Gowalla, Epinions

Dataset	**Douban**			**Gowalla**			**Epinions**		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
GT	0.420916	0.683275	0.520927	0.124253	0.435963	0.171214	0.142565	0.403301	0.189999
VAREM	0.448542	0.838615	0.584473	0.217694	0.579401	0.323537	0.172924	0.799560	0.247951
SIG	0.869348	0.614971	0.761101	0.553444	0.746408	0.646652	0.510118	0.775194	0.594529

The diffusion process in GSIG is similar to that in SIG. All nodes are all inactive initially, and the seed nodes immediately becomes active. Given all the nodes in a source set $\mathtt{U}$ at iteration $\iota - 1$, if they are all active at iteration ι, a hyperedge $e \equiv \mathtt{U} \to \mathtt{V}$ has a chance to activate the inactive nodes in $\mathtt{V}$ with probability p_e. Each node (v, i) can be activated once, but it can try to activate other nodes multiple times, one for each incident hyperedge.

6.2 General Social Item Maximization

Next, we formulate a more general seed selection problem that additionally considers the different *costs* of selecting seeds and *revenue* of purchase actions. Recall that a seed in GSIG represents an adoption/purchase action of a specific item by a particular customer. Therefore, instead of treating all seed selections and purchase actions equally, this problem, called *General Social Item Maximization Problem* (GSIMP), considers the cost and revenue of such actions.

Definition 9. Given a seed cost budget B, a list of targeted items I, a general social item graph $G_{GSI} = (V_{GSI}, E_G)$, the cost $c(v, i)$ and profit $p(v, i)$ for each purchase action $(v, i) \in G_{GSI}$, selects a set $\mathtt{S}$ of seeds so that (i) the total cost $\sum_{(v,i)\in \mathtt{S}} c(v, i)$ does not exceed the budget B, and (ii) the total revenue $\gamma_{G_{GSI}}(\mathtt{S}) = \sum_{(v,i)\text{is activated}} p(v, i)$ is maximized.

6.3 Hardness Results

In the following, we prove that GSIMP is inapproximable with a non-constant ratio n^c for all $c < 1$, even if the GSIG is $(\bar{n_s}, \bar{n_d})$-degree bounded for fixed constants $\bar{n_s}$ and $\bar{n_d}$. This will be proved with a gap-introducing reduction from the NP-complete Set Cover problem, where n is the number of nodes in a GSIG.

Lemma 2. For a positive integer q, there is a gap-introducing reduction from Set Cover to SIMP, which transforms an instance ϕ with n_e elements and m_s subsets to a GSIMP instance with the GSIG $G_{GSI} = (V_{GSI}, E_G)$ such that

- if ϕ is satisfiable, $\gamma^*_{G_{GSI}} \geq \tilde{n}^q$, and
- if ϕ is not satisfiable, $\gamma^*_{G_{GSI}} < \tilde{n}$,

where $\gamma^*_{G_{GSI}}$ is the optimal revenue in this instance, $\tilde{n}$ is the number of nodes in one part in the GSIG, where the total number of nodes in $n = \tilde{n} + \tilde{n}^q$. Hence there is no $\tilde{n}^{q-1}$ approximation algorithm for GSIMP unless P $=$ NP.

Proof 4. Given a positive integer q, for an instance ϕ of Set Cover decision problem with a universe of n_e elements $U = \{e_1, e_2, \cdots, e_{n_e}\}$ and a collection of m_s sets $S = \{m_1, m_2, \cdots, m_{m_s}\}$ whose union equals U, and an integer k,[10] we construct an GSIG G_{GSI} with three node sets $\mathtt{X}$, $\mathtt{Y}$ and $\mathtt{Z}$ as follows. 1) $\mathtt{Y}$ contains all nodes in a complete $\bar{n_s}$-ary tree, with each element e_i corresponds to one leaf $\mathtt{y}_i$ in $\mathtt{Y}$. For each group of children and parent nodes, we add a hyperedge with the source nodes being the set of children nodes, and the destination being the parent node. Therefore, almost all (with at most one exception at each level) hyperedges in this tree has $\bar{n_s}$ source nodes and one destination node, and $\mathtt{Y}$ has $1 + \bar{n_s} + \bar{n_s}^2 + \cdots + \bar{n_s}^{\lceil \log_{\bar{n_s}} n_e - 1 \rceil} + n_e = O(n_e)$ nodes. 2) For each set m_j, we add a corresponding node $\mathtt{x}_j$ to $\mathtt{X}$. Then, for each element $e_i \in m_j$, we construct an 1-source, 1-destination hyperedge from $\mathtt{x}_j$ to $\mathtt{y}_i$. Therefore, $\mathtt{X}$ has m_s nodes. 3) $\mathtt{Z}$ has $(|\mathtt{X}|+|\mathtt{Y}|)^q$ nodes. For each node $\mathtt{z}$ in $\mathtt{Z}$, we add an 1-source, 1-destination hyperedge from the root of the complete tree in $\mathtt{Y}$ to $\mathtt{z}$. 4) The probability of every edge is set to 1, and the profit and cost for every node is set to 1. 5) The budget B is set to k. An example is illustrated in Figure 8.

We first prove that ϕ is satisfiable if and only if G_{GSI} has a seed set $\mathtt{S}$ with the total adoption of $\mathtt{S}$ contains all nodes in $\mathtt{Y}$. If ϕ is satisfiable, there exists k sets $S' = \{m'_1, \ldots, m'_k\}$ covering all elements in U. Let $\mathtt{S} = \{\mathtt{x}_i | m_i \in S'\}$, it is straightforward to see that all n_e leaf nodes in $\mathtt{Y}$, which corresponds to all elements, will be activated, which will then activate all nodes in $\mathtt{Y}$. On the other hand, if ϕ is not satisfiable, there exists no seed set with at most k seeds that can activate all n_e leaf nodes. Moreover, observe that the nodes of $\mathtt{Z}$ can be activated if and only if the total adoption of $\mathtt{S}$ contains $\mathtt{Y}$ if and only if ϕ is satisfiable. We have

- if ϕ is satisfiable, $\gamma^*_{G_{GSI}} \geq (|\mathtt{X}| + |\mathtt{Y}|)^q$, and
- if ϕ is not satisfiable, $\gamma^*_{G_{GSI}} < |\mathtt{X}| + |\mathtt{Y}|$.

The lemma follows.

Theorem 3. For any $\epsilon > 0$, there is no $n^{1-\epsilon}$ approximation algorithm for GSIMP, assuming P $\neq$ NP.

Proof 5. For any arbitrary $\epsilon > 0$, we set $q \geq \frac{2}{\epsilon}$. Then, by Lemma 2, there is no $\tilde{n}^{q-1}$ approximation algorithm for SIMP unless P $=$ NP. Then $\tilde{n}^{q-1} \geq 2\tilde{n}^{q-2} \geq 2\tilde{n}^{q(1-\epsilon)} \geq (2\tilde{n}^q)^{1-\epsilon} \geq n^{1-\epsilon}$. Since ϵ is arbitrarily small, thus for any $\epsilon > 0$, there is no $n^{1-\epsilon}$ approximation algorithm for SIMP, assuming P $\neq$ NP. The theorem follows.

7 Advanced Algorithm for GSIMP

In this section, we turn our attention to solving GSIMP, by first describing how extend the HAG algorithm to solve the general problem. Next, we propose an advanced algorithm,

10. An implied assumption here is $n_e > k$, since the other cases are trivial in Set Cover.

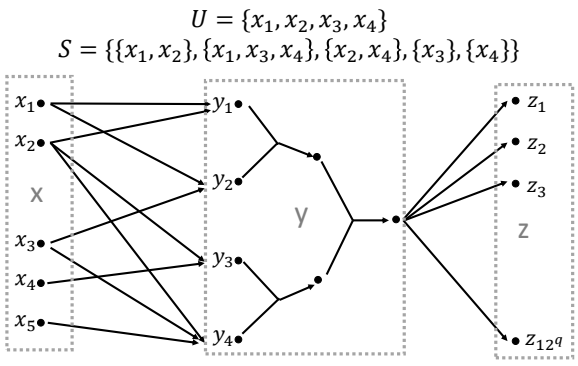


Fig. 8. An illustration instance built for Set Cover

Hyperedge-Aware Greedy with Substitution and Pruning (HAG-SP) that further enhances HAG so that both the solution quality and efficiency are improved.

To solve GSIMP, a straightforward approach is to modify the criteria in HAG, so that in each iteration, the algorithm picks the source combination leading to *the largest increment on total expected revenue divided by the total cost of new seeds added in this iteration*. The iteration continues until no source combination can be picked within the budget. Nevertheless, this approach is not efficient because there are way more hyperedges that need to be examined in GSIMP. Thus, examining all source combinations are significantly more computationally intensive in GSIMP. In addition, the SIG-index structure cannot directly extend to GSIMP since it does not capture calculation and updating of *the possibility that a bundle of destination nodes are activated together*. Therefore, it is desirable to design a more sophisticated algorithm to reduce the computational cost as well as devising an efficient alternative index structure that, like SIG-index in SIMP, accelerates the diffusion computation.

In this work, we first propose a number of pruning rules, namely *Inclusive Diffusion Pruning*, *Subtractive Diffusion Pruning*, *Exclusive Diffusion Pruning* and *Maximum Diffusion Pruning*, to accelerate the seed selection process by ignoring unpromising source combinations and avoiding to compute their effects on diffusion. Furthermore, we propose a multi-purposed hybrid structure, *GSIG-index Lattice*, which not only speeds up the diffusion computation in GSIG, but also tracks the inclusion relationships among the numerous source and destination combinations in GSIG. By exploiting these relationships, we propose an efficient strategy for examining the hyperedges, *Source Combination Ordering*, which can be effectively combined with the pruning strategies to enhance the efficiency of the seed selection process. Last but not least, we also propose a local search strategy, namely the *Seed Replacement Rule*, to tweak the selected seed set by making profitable modifications to the selected source combinations in each round. In summary, during the examination of source combinations in the HAG algorithm, we exploit the inclusion relationships among source and destination combinations to effectively prune redundant diffusion computation and efficiently find the desired seed set.

7.1 Pruning Unpromising Source Combinations

In each round of the seed selection process, HAG sequentially scans all possible $|E_H|$ source combinations. For each source combination, it invokes the costly diffusion computation procedure to calculate the incremental contribution on the revenue. This iterative process is unavoidable in traditional diffusion models where the seed candidates are independent nodes in the graph. However, the source combinations in GSIG are not completely independent units, because multiple hyperedges can share overlapping nodes, and a source combination can even be a subset of another. We leverage this phenomenon to propose the *Inclusive Diffusion Pruning* strategy. The idea is to discard a candidate set that, even with its maximum possible incremental revenue, cannot outperform the current best candidate.

Another phenomenon that contributes to the computational difficulty is that similar diffusion computation needs to be repeated in every round of the seed selection process. To resolve this, we further propose the *Subtractive Diffusion Pruning* and *Exclusive Diffusion Pruning*. strategies to save the computation effort. The idea of them is to reuse the diffusion results in previous rounds to filter unpromising candidates. Finally, we also observe that performing GSIG traversal beforehand can find the maximum possible contribution to the total revenue by any given node. Exploiting this property, we propose the *Maximum Diffusion Pruning* to filter very bad source combination candidates, which are bad candidates even if their value are extreme overestimated.

In the followings, we use $\mathtt{S}$ to denote the current solution set. Denote $\sigma_{\mathtt{S}}(\mathtt{U})$ to be the increment on total expected revenue when the source combination $\mathtt{U}$ is added to $\mathtt{S}$, and $c_{\mathtt{S}}(\mathtt{U}) = \sum_{(v,i)\in \mathtt{U}\setminus \mathtt{S}} c(v,i)$ being the total cost of doing so (note that some nodes in $\mathtt{U}$ might already belong in $\mathtt{S}$). The straightforward greedy approach adds the source combination $\mathtt{U}$ that maximizes $\frac{\sigma_{\mathtt{S}}(\mathtt{U})}{c_{\mathtt{S}}(\mathtt{U})}$. For simplicity, we use the term *Revenue-Cost Ratio* (RCR) to refer to this value.

Inclusive Diffusion Pruning. Consider two source combinations $\mathtt{U}_A$ and $\mathtt{U}_B$ where $\mathtt{U}_B \subseteq \mathtt{U}_A$. In such cases, the increment on total revenue achieved by $\mathtt{U}_A$ (the superset) is actually an upper bound on that of $\mathtt{U}_B$ (the subset). Suppose that

$$\frac{\sigma_{\mathtt{S}}(\mathtt{U}_A)}{c_{\mathtt{S}}(\mathtt{U}_B)} \leq \frac{\sigma_{\mathtt{S}}(\mathtt{U}^*)}{c_{\mathtt{S}}(\mathtt{U}^*)},$$

where $\mathtt{U}^*$ is the current best candidate. Since the actual RCR of $\mathtt{U}_B$ is at most the LHS of the equation, the greedy process is impossible to select $\mathtt{U}_B$. Therefore, it is unnecessary to compute the diffusion of adding $\mathtt{U}_B$ to the current solution.

Running Example. The GSIG shown in Figure 9 is given to the algorithm. Consider the first round of the seed selection algorithm. Suppose that the current best source combination is $\{C\}$, which has an RCR value of $\frac{3+0.8\cdot 12}{2} = 6.3$, and has already been examined. Next, the algorithm examines the source combination set $\{A, B, C\}$ and calculates its expected revenue, which is $3 + 3 + 2 + 0.5 \cdot 8 + 1 \cdot 12 = 24$. Therefore, its RCR value is $\frac{24}{3+2+3} = 3.0$, which is inferior to $\{C\}$. Nevertheless, upon examining the source combination set $\{A, B\}$, instead of calculating its expected revenue, the algorithm checks the Inclusive Diffusion Pruning criterion

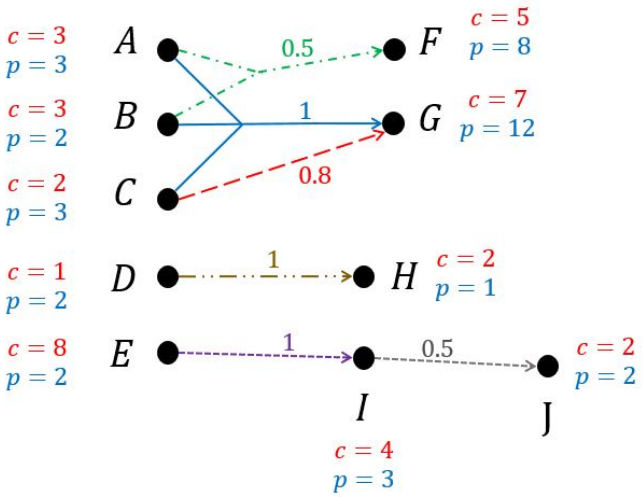


Fig. 9. Running Example of the Pruning Strategies (temporary).

and sees that $\frac{\sigma(\{A,B\})}{c(\{A,B\})} \leq \frac{\sigma(\{A,B,C\})}{c(\{A,B\})} = \frac{24}{3+2} = 4.8 < 6.3 = \frac{\sigma(\{C\})}{c(\{C\})}$. Therefore, $\{A, B\}$ cannot be the best source combination set in this round. The algorithm then omits calculating its exact expected revenue.

Subtractive Diffusion Pruning. Consider the r-th round in the greedy algorithm. Let $\mathtt{U}_i$ denote the source combination selected in the i-th round, and $\mathtt{S}_{r-1} = \bigcup_{i=1}^{r-2} \mathtt{U}_i$ be the solution set at the beginning of the last round (i.e., round $r-1$). Consider two source combinations $\mathtt{U}_A$ and $\mathtt{U}_B$, where $\mathtt{U}_A \cup \mathtt{U}_{r-1} \subseteq \mathtt{U}_B$. Here, in the last round, the benefit of adding $\mathtt{U}_B$ is at least that of adding both $\mathtt{U}_A$ and $\mathtt{U}_{r-1}$ to the solution. While the action of adding both combinations seems simultaneous, it can be carried out sequentially without affecting the outcome, e.g. first adding $\mathtt{U}_{r-1}$, then adding $\mathtt{U}_A$. Thus, we have the following relation

$$\sigma_{\mathtt{S}_{r-1}}(\mathtt{U}_{r-1}) + \sigma_{\mathtt{S}}(\mathtt{U}_A) \leq \sigma_{\mathtt{S}_{r-1}}(\mathtt{U}_B),$$

or equivalently,

$$\sigma_{\mathtt{S}}(\mathtt{U}_A) \leq \sigma_{\mathtt{S}_{r-1}}(\mathtt{U}_B) - \sigma_{\mathtt{S}_{r-1}}(\mathtt{U}_{r-1}).$$

Therefore, if

$$\frac{\sigma_{\mathtt{S}_{r-1}}(\mathtt{U}_B) - \sigma_{\mathtt{S}_{r-1}}(\mathtt{U}_{r-1})}{c_{\mathtt{S}}(\mathtt{U}_A)} \leq \frac{\sigma_{\mathtt{S}}(\mathtt{U}^*)}{c_{\mathtt{S}}(\mathtt{U}^*)},$$

the greedy process is impossible to select $\mathtt{U}_A$, and we can discard $\mathtt{U}_A$ in this round.

Argue that additional pruning does not work?

Running Example. Still, consider the example in Figure 9. Suppose that source combination $\{C\}$ is selected in the first round. Therefore, $\mathtt{S} = \{C\}$. In the second round, the algorithm first examines the source combination $\{D\}$, which has an RCR value of $\frac{2+1}{1} = 3.0$. Next, the algorithm needs to evaluate the source combination $\{A, B\}$. Since that $\{A, B\} \cup \{C\} \subseteq \{A, B, C\}$, the algorithm finds that $\frac{\sigma_{(\mathtt{S})}(\{A,B\})}{c_{(\mathtt{S})}(\{A,B\})} \leq \frac{\sigma(\{A,B,C\})-\sigma(\{C\})}{c_{(\mathtt{S})}(\{A,B\})} = \frac{24-12.6}{3+2} = 2.28 < 3.0$. Therefore, $\{A, B\}$ is still not the best source combination set in this round. The algorithm then omits calculating its exact expected revenue again.

Maximum Diffusion Pruning. By performing a graph traversal starting from $\mathtt{u}$, the traversal process can find the *maximum possible influence* of $\mathtt{u}$, simply by aggregating the profits among the traversed nodes. This aggregated value is a natural upper bound of possible incremental profit of adding $\mathtt{u}$ to the seed set. Moreover, this upper bound is additive since we are already overestimating the profit by not considering other source nodes in the hyperedges. Therefore, we can obtain an upper bound of possible incremental profit for a whole node set by simply aggregating the single-node values.

Formally, let $\sigma_{max}(\mathtt{u})$ denote the maximum possible influence of $\mathtt{u}$ found in the traversal phase. Given any source combination $\mathtt{U}$, we can obtain the upper bound of possible incremental profit $\sigma_{max}(\mathtt{U}) = \sum_{\mathtt{u}\in\mathtt{U}} \sigma_{max}(\mathtt{u})$. Therefore, if the RCR value estimated this way is still inferior, i.e.,

$$\frac{\sigma_{max}(\mathtt{U})}{c_{\mathtt{S}}(\mathtt{U})} \leq \frac{\sigma_{\mathtt{S}}(\mathtt{U}^*)}{c_{\mathtt{S}}(\mathtt{U}^*)},$$

then $\mathtt{U}$ is impossible to be selected, and thus can be pruned.

Running Example. We still consider the third round of the seed selection process in the example in Figure 9. With the seed set being $\mathtt{S} = \{C, D\}$, the best candidate source combination is now $\{A, B\}$ with an RCR value of 2.28, the algorithm then examines the source combination $\{E\}$. In the previous traversal, it is found that the nodes $\{E, I, J\}$ are reachable from $\{E\}$. Hence, the maximum possible incremental profit of $\{E\}$ is $2 + 3 + 2 = 7$. Since $\frac{\sigma_{(\mathtt{S})}(\{E\})}{c_{(\mathtt{S})}(\{E\})} \leq \frac{\sigma_{max}(\{E\})}{c_{(\mathtt{S})}(\{E\})} = \frac{7}{8} = 0.875 < 2.28$, $\{E\}$ is impossible to be the best source combination. The algorithm then discards $\{E\}$ without computing the exact expected profit (which is in fact smaller than 7).

7.2 Indexing the Node Combinations

While the aforementioned pruning strategies effectively reduces the need for computing diffusion, checking the pruning criteria can be costly. The Inclusive and Subtractive Diffusion Prunings all incur one-by-one scan over all source combinations, resulting in a $O(|E_H|)$-time complexity. The Maximum Diffusion Pruning condition performs a GSIG traversal for each node, which is $O(N \cdot |E_H|)$.

Moreover, the power of these strategies are significantly impacted by the source combination examining order. Intuitively, the more promising source combinations should be examined earlier so that the reference value (RHS of all the pruning criteria) is higher, which implies more source combinations being pruned. On the other hand, larger source combinations (those who contain more nodes) should also be examined earlier, since the Inclusive Diffusion Pruning strategy lies on the intermediate result of examining large source combinations.

To address the above difficulties, we propose an effective index structure, namely the *GSIG-index Lattice*, to facilitate quick queries of the set relationships between source combinations. The GSIG-index Lattice pre-examines the GSIG and leverages the *lattice structure* in the field of frequent itemset mining, where each source combination corresponds to a vertex in the lattice. The idea is to exploit the special structure of the lattice, so that for each type of pruning strategies, only a small part of the lattice needs to be examined. Instead

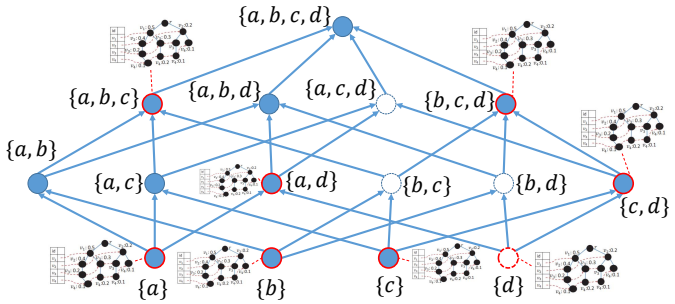
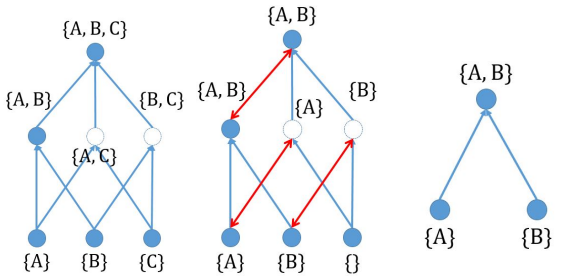


Fig. 10. An illustrative example of the GSIG-index Lattice structure



(a) Initial (b) After $\{C\}$ is selected (c) After merging

Fig. 11. Part of GSIG-index lattice of the running example

of performing exhaustive check on the prerequisites (the set relationship between source combinations), the GSIG-index Lattice helps to filter the candidates, while the algorithm can focus on the comparison on RCR value in the online computing process. With the help of this index structure, we also design the *Lattice Ordering* strategy as a guideline on searching the source combinations, so that the algorithm can make the best of the pruning strategies.

GSIG-index Lattice. Figure 10 presents an illustrative example. Each vertex in the GSIG-index Lattice structure corresponds to a *node combination* that represents a group of at most $\max\{\bar{n_s}, \bar{n_d}\}$ nodes. Recall that $\bar{n_s}$ and $\bar{n_d}$ are the maximum possible number of nodes in the source and destination set of a GSIG hyperedge. Therefore, all source/destination combinations of the hyperedges are represented by the lattice vertices.

Direct links between the lattice vertices correspond to the inclusion relationships between the node combination sets, with larger sets appearing near the top of the lattice. A lattice vertex can be a source combination of some hyperedge (illustrated as solid points), a destination combination of some hyperedge (framed points), both (the framed solid points), or neither (the hollow points). Moreover, each destination combination node is linked with its specified SIG-index to accelerate the diffusion process as in Section 4.2.

Accelerate the algorithm using the GSIG-index Lattice. The hybrid GSIG-index Lattice speeds up multiple aspects of the seed selection algorithm as well as the diffusion computation. First, in each round of the seed selection process, the source combinations are examined according to the *Lattice Ordering* strategy on the lattice (detailed later). Next, given the current examined source combination $\mathtt{U}$, the algorithm traverses the lattice upwards from $\mathtt{U}$ to retrieve all source combinations that include $\mathtt{U}$ to check the Inclusive Diffusion Pruning criteria. Similarly, given the source combination $\mathtt{U}_{r-1}$ selected in the last round, the algorithm traverses the lattice upwards from $\mathtt{U}_{r-1}$ to retrieve all its supersets. As all supersets of $\mathtt{U}$ and $\mathtt{U}_{r-1}$ are obtained, the possible candidates of $\mathtt{U}_B$ in the Subtractive Diffusion Pruning are exactly those source combinations in the intersection of the above two sets.

After the pruning strategies filter away unnecessary source combinations, the diffusion of the remaining candidate sets need to be calculated. The activation probability of each destination combination is maintained by the additional SIG-index structures linked to it, accelerating the computation. Finally, when some node combination is selected by the greedy algorithm and added to the seed set, the corresponding elements are deleted from the node combinations. The lattice then updates itself by merging the identical vertices (node combinations), further simplifies its structure.

Running Example. Figure 11 shows a part of GSIG-index lattice of the running example in Figure 9. Not every node combination set is shown for simplicity. Figure 11 (a) is the original state of the relevant part. After the source combination $\{C\}$ is chosen in the first round, the selected node C is removed from all relevant node combinations, resulting in the state shown by (b). Since the removal of C creates multiple duplicated node combination nodes, as shown in red edges in (b), the GSIG-index then merges the identical nodes, resulting in the state shown by (c).

Lattice Ordering. As mentioned earlier, it is desirable to examine the more promising source combinations, i.e. those with higher RCR values, earlier to provide a high reference RCR early in the pruning process. Also it is desirable to examine the large source combination sets earlier to faciliate the Inclusive Diffusion Pruning. To effectively combine these two ideas, we propose the Lattice Ordering for examining the source combinations as follows:

- In each round, the source combinations with top-k RCR values in the last round is examined first. Note that the best candidate is already selected; hence it suffices to examine the next $k-1$ sets. This step is omitted in the first round.
- Next, the algorithm examines the remaining source combination sets in top-down order w.r.t the lattice structure. In other words, larger sets are always examined first than its subsets, unless the subset is highly-promising and appeared in top-k in the last round. This is achieved by a BFS process with the links between sets are treated in reversed direction.
- Each time the diffusion of a source combination is measured, it will be propagated downwards to its subsets as an upper bound of diffusion. Only the tightest upper bound (i.e., the lowest value) is maintained.
- Upon examination, the upper bound value is used in the Inclusive Diffusion Pruning criterion.

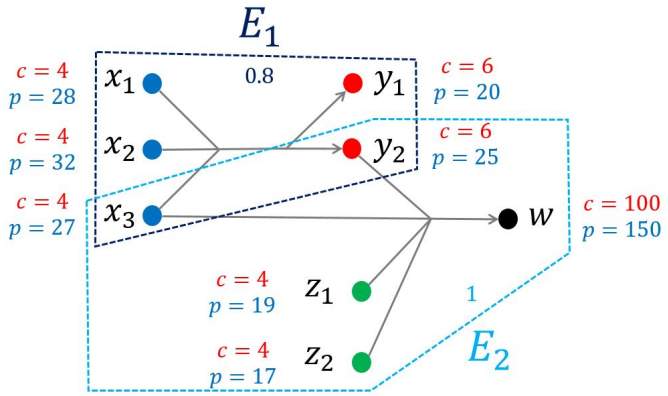


Fig. 12. A motivating example for the seed replacement rule. (Temporary)

Running Example. We still consider the running example in Figure 9. In the first round, there are no top-k RCR values in the previous round, so the first step is omitted. The Lattice Ordering strategy examines the source combinations by the following order: $\{A, B, C\}$, $\{A, B\}$, $\{A\}$, $\{B\}$, $\{C\}$, $\{D\}$, $\{E\}$, $\{I\}$, $\{J\}$. Let $k = 2$. That is, the top-2 promising source combinations will be checked first in the next round. Since the best source combination $\{C\}$ is already chosen, the algorithm only examines the second best source combination, which is $\{D\}$ in the second round. After $\{D\}$ is examined, the remaining source combination sets are checked in the same order as in the first round, which leads to $\{A, B\}$, $\{E\}$ and $\{I\}$ being pruned by different pruning strategies.

7.3 Seed Replacement

While the above pruning and ordering strategies can significantly reduce the computational effort of the seed selection process, there still exist some instances that the greedy process does not perform well. Figure 12 shows one of these instances.

Motivation Example. In Figure 12, there exist two hyperedges E_1 and E_2. Hyperedge E_1 emerges from source nodes $\mathtt{S}_1 = \{\mathtt{x}_1, \mathtt{x}_2, \mathtt{x}_3\}$ to destination nodes $\mathtt{D}_1 = \{\mathtt{y}_1, \mathtt{y}_2\}$ with activation probability 0.8, and hyperedge E_2 is from source nodes $\mathtt{S}_2 = \{\mathtt{x}_3, \mathtt{y}_2, \mathtt{z}_1, \mathtt{z}_2\}$ to destination node $\mathtt{D}_2 = \{\mathtt{w}\}$ with activation probability 1. The cost and profit values for each node is shown besides the node. The total budget is set at 20.

Consider the first round of the GSIG algorithm. The source combination set $\mathtt{S}_1$ has an RCR value of $\frac{28+32+27+20+25}{4+4+4} = 11$. The source combination set $\mathtt{S}_2$ has an RCR value of $\frac{27+19+17+25+150}{4+6+4+4} = 13.22$. Also, the highest RCR value of a singleton node is $\mathtt{x}_2$'s 8. Therefore, the greedy process selects $\mathtt{S}_2$, which leaves the remaining budget to be 2. Since the minimum cost of any node is 4, no further node can be activated, and the final total revenue is 238. However, it is intuitive that the best strategy in this instance is to select all the blue and green nodes, i.e., $\{\mathtt{x}_1, \mathtt{x}_2, \mathtt{x}_3, \mathtt{z}_1, \mathtt{z}_2\}$. In this case, with a probability of 0.8, the whole network is activated. This assignment has an expected revenue of $(28 + 32 + 27 + 19 + 17) + 0.8 \times (20 + 25 + 150) = 279$. However, since the best assignment is not a simple source combination set, the GSIG algorithm does not examine this possibility, thus does not find the optimal solution.

In the above example, a key observation that the simple GSIG algorithm misses is that activating $\{\mathtt{x}_1, \mathtt{x}_2, \mathtt{x}_3\}$ *almost guarantees* the activation of $\mathtt{y}_2$, since the hyperedge activation probability is close to 1. Therefore, in suitable conditions, it may be profitable to substitute the former set for the latter, as the example shows. Motivated by this observation, we propose the *Seed Replacement Rule*, which is a local search strategy that strengthens the GSIG algorithm by mining the hidden relation between node sets and selecting the seed nodes more intelligently.

The idea of the Seed Replacement Rule is as follows. We examine the network in a pre-evaluation phase, where the "almost-guaranteed activation" relations are identified and indexed. Later, in the online seed selection phase, after the source combination set $\mathtt{U}^*$ is selected, the algorithm further examines the possible ways to substitute a subset of $\mathtt{U}^*$, so that the incremental contribution of the selected set is improved. Consider the case that activating a node set $\mathtt{X}$ almost guarantees activating another node set $\mathtt{Y}$. When the greedy process selects the best source combination set $\mathtt{U}^*$, which is from a single hyperedge, the algorithm will examine the possibility to replace $\mathtt{U}^* \cup \mathtt{Y}$ by $\mathtt{X} \setminus \mathtt{U}^*$. In doing so, it needs to evaluate three things: 1) the original expected incremental profit: it will be slightly reduced since the "almost-guaranteed activation" does not completely guarantee the activation of the relevant nodes. 2) the difference in incremental profit: the substitution gains the profit on $\mathtt{X} \setminus \mathtt{U}^*$ but possibly also loses the profit on $\mathtt{U}^* \cup \mathtt{Y}$. 3) the side profit: sometimes, the substitution leads to more nodes being activated, as the node $\mathtt{y}_1$ in the example.

More explicitly, we outline the computation for comparing the total profit with or without the substitution. Let $\sigma_{\mathtt{S}}(\mathtt{U}^*)$ be the expected incremental profit of adding $\mathtt{U}^*$ to the current solution $\mathtt{S}$. Also let p be the activation probability of the "almost-guaranteed activation" relation. We are interested in the incremental profit if $\mathtt{U}^* \cup \mathtt{Y}$ is replaced by $\mathtt{X} \setminus \mathtt{U}^*$, which can be calculated as follows:

- With probability $1 - p$, the replaced part, $\mathtt{U}^* \cup \mathtt{Y}$, is not activated by the "almost-guaranteed activation". In this case, the worst scenario does not activate any node other than the selected nodes. Hence, the worst incremental profit is only the profit on the (modified) node set, which is $\sigma_0 = \sum_{\mathtt{u} \in \mathtt{U}^* \cup \mathtt{X} \setminus \mathtt{Y}} p(\mathtt{u})$.
- With probability p, the original source combination $\mathtt{U}^*$ is completely activated. In this case, the incremental profit consists of three possible parts:
  - The original incremental profit $\sigma_1 = \sigma_{\mathtt{S}}(\mathtt{U}^*)$.
  - The additional profit on the manually activated nodes: $\sigma_2 = \sum_{\mathtt{u} \in \mathtt{X} \setminus \mathtt{U}^*} p(\mathtt{u})$.
  - The side profit of activating $\mathtt{X} \setminus \mathtt{U}^*$, which is $\sigma_3 = \sum_{\mathtt{u} \in \mathtt{Y} \setminus \mathtt{U}^* | \nexists \mathtt{v} \in \mathtt{U}^* \setminus \mathtt{X} \setminus \mathtt{Y} \text{ s.t. } \mathtt{v} \hookrightarrow \mathtt{u}} p(\mathtt{u})$.

Therefore, the incremental profit after substitution is at least $(1 - p) \cdot \sigma_0 + p \cdot (\sigma_1 + \sigma_2 + \sigma_3)$, and the algorithm

examines whether the following inequality is true:

$$\frac{(1-p)\cdot\sigma_0 + p\cdot(\sigma_1+\sigma_2+\sigma_3)}{\sum_{\mathtt{u}\in\mathtt{U}^*\cup\mathtt{X}\setminus\mathtt{Y}} c(\mathtt{u})} > \frac{\sigma_{\mathtt{S}}(\mathtt{U}^*)}{c_{\mathtt{S}}(\mathtt{U}^*)}$$

If the inequality holds, the algorithm replaces $\mathtt{U}^*$ by $\mathtt{U}^* \cup \mathtt{X} \setminus \mathtt{Y}$.

Example Revisited. In 12, upon selecting E_2 with expected incremental profit of 238, the algorithm considers that activating $\{\mathtt{x}_1, \mathtt{x}_2, \mathtt{x}_3\}$ has a probability of 0.8 to further activate $\{\mathtt{y}_1, \mathtt{y}_2\}$. It examines the possibility to actually select the modified node set $\mathtt{U}' = \{\mathtt{x}_1, \mathtt{x}_2, \mathtt{x}_3, \mathtt{z}_1, \mathtt{z}_2\}$.

Here, the profit of $\mathtt{U}'$ is guaranteed even in the worst case. Thus $\sigma_0 = 28{+}32{+}27{+}19{+}17 = 123$. In the good case, the first part of the profit is just the original profit, which is 238. The second part comes from $\mathtt{x}_1$ and $\mathtt{x}_2$, which are not in the original seeds. Hence, $\sigma_2 = 28 + 32 = 60$. Finally, the third part of the profit is from $\mathtt{y}_1$, which would not be activated by the original selected nodes. Therefore $\sigma_3 = 20$. Finally, the total incremental profit is $(1-0.8)\cdot 123 + 0.8\cdot(238+60+20) = 279$, which, not coincidentally, equals the previous result. Since the RCR of this assignment is higher than the original configuration (i.e., $\frac{279}{20} = 13.95 > 13.22$), this modification is indeed profitable and will be carried out by the algorithm.

8 Multi-Destination Hyperedge Construction

To extend the single-destination hyperedges to multi-destination hyperedges, the learning process of the SIG model should be modified due to the tremendous number of activation combinations. Under the settings of the single-destination hyperedges, when $\mathtt{v}$ is activated at time τ, this event implies that at least one hyperedge successfully activates $\mathtt{v}$ before τ. In contrast, when the number of destination nodes is more than 1, the number of combinations that can trigger the activation becomes large. For example, given the destination set $\mathtt{V} = \{\mathtt{v}_1, \mathtt{v}_2, \mathtt{v}_3\}$ and source set $\mathtt{U}$, when nodes $\mathtt{v}_1$, $\mathtt{v}_2$, and $\mathtt{v}_3$ are both activated in a short period, the number of possible combinations that can trigger $\mathtt{V}$ is 5, i.e., $\mathtt{U}$ activates $\mathtt{v}_1$, $\mathtt{v}_2$ and $\mathtt{v}_3$ separately, $\mathtt{U}$ activates two nodes and then one node, or $\mathtt{U}$ activates $\mathtt{V}$.

The large number of possible hyperedge combinations not only increases the computation complexity but also results in severe data sparsity problem, i.e., the number of activations for a user to buy an item is much smaller than the possible combinations. To tackle this challenge, we propose a joint user-item mixture model and dimension reduction technique to make the SIG model workable for hyperedges with multi-destination nodes. In the following, we first formulate the EMS for hyperedges with multi-destination nodes and then present the user-item mixture model and dimension reduction to deal with the data sparsity issue.

8.1 Modeling of Hyperedge Probability with Multi-Destination Hyperedges

To model the hyperedges with multi-destination nodes, we first modify the EMS as follows. Let $E_{\mathtt{V}}$ denote the set of candidate hyperedges containing every possible e linked to destination set $\mathtt{V} = \{\mathtt{v}_1, \mathtt{v}_2, \cdots, \mathtt{v}_m\}$ with its source set activated at time $\tau-1$, i.e., $E_{\mathtt{V}} = \{(\mathtt{u}_1, \mathtt{u}_2, \cdots, \mathtt{u}_i, \cdots, \mathtt{u}_n) \to (\mathtt{v}_1, \mathtt{v}_2, \cdots, \mathtt{v}_m) | \forall i = 1, \cdots, n, \mathtt{u}_i \in N(\mathtt{V}), (\mathtt{u}_i, \tau-1) \in \mathcal{A}\}$. Let $\mathbb{W}$ denote the possible combinations of subsets comprising $\mathtt{V}$, i.e., $\mathbb{W} = \{\{W_1, W_2, \cdots, W_j\} | W_1 \cup W_2 \cup \cdots \cup W_j = \mathtt{V}, W_x \cap W_y = \emptyset, \forall x \neq y, W_x^{(1)} < W_y^{(1)}, \forall x < y\}$, where $W_j \subseteq \mathtt{V}$ denotes the j-th ordered subset of $\mathbb{W}$ and $W_j^{(i)}$ denote the i-th element of W_j.

Therefore, given the estimation of the probability of hyperedges, the probability for $\mathtt{v}$ to be activated by any hyperedge combinations at time τ, $P_{\mathtt{v}}$ is obtained as follows.

$$\sum_{w=\{W_1,W_2,\cdots,W_j\}} (1 - \prod_{e_1'\in E_{W_1}} (1-\hat{p}_{e'})) \cdots (1 - \prod_{e_1'\in E_{W_j}} (1-\hat{p}_{e'})).$$

Therefore,

$$w_{e,a} = \frac{\hat{p}_e}{P_{\mathtt{v}}}. \tag{3}$$

The expectation of K_e is $\sum_{a\in A, e\in E_a} w_{e,a}$, i.e., the sum of expectation of each successful activation of $\mathtt{v}$ from hyperedge e. In summary, we have the following steps.

E-Step:

$$E[K_e] = \sum_{a\in A, e\in E_a} w_{e,a}, \tag{4}$$

where $w_{e,a} = \hat{p}_e / \sum_{w=\{W_1,W_2,\cdots,W_j\}} (1 - \prod_{e_1'\in E_{W_1}} (1 - \hat{p}_{e'})) \times \cdots \times (1 - \prod_{e_1'\in E_{W_j}} (1-\hat{p}_{e'}))$.

M-Step:

$$p_e = \frac{\sum_{a\in A, e\in E_a} w_{e,a}}{N_U},$$
$$\lambda_A(e) = \sum_{a\in A, e\in E_a} w_{e,a},$$
$$\lambda_T(U) = N_U.$$

S-Step: To address the data sparsity problem, we can leverage information from similar hyperedges. Specifically, in addition to E-Step and M-Step, EMS includes **S-Step**, which smooths the results of M-Step. Kernel smoothing is employed in S-Step as follows:

$$\hat{\lambda}_A(e) = \sum_{a\in A, e'\in E_a\cap E_{H,n}} w_{e',a} L_h(F(e) - F(e'))$$
$$\hat{\lambda}_T(U) = \sum_{U\subseteq V_{SI}} N_U L_h(F(U) - F(U'))$$

where L_h is a kernel function with bandwidth h, and F is the mapping function of hyperedges, i.e., $F(e)$ maps a hyperedge e to a vector. The details of dimension reduction for calculating F to efficiently map hyperedges into Euclidean space are shown in [2] due to space constraint. If the hyperedges e and e' are similar, the distance of the vectors $F(e)$ and $F(e')$ is small. Moreover, a kernel function $L_h(x)$ is a positive function symmetric at zero which decreases when $|x|$ increases, and the bandwidth h controls the extent of auxiliary information taken from similar hyperedges.[11] Intuitively, kernel smoothing can identify the correlation of $\hat{p}_{e_1}$ with $e_1 = \mathtt{U}_1 \to \mathtt{v}_1$ and $\hat{p}_{e_2}$ with $e_2 = \mathtt{U}_2 \to \mathtt{v}_2$ for nearby $\mathtt{v}_1$ and $\mathtt{v}_2$ and similar $\mathtt{U}_1$ and $\mathtt{U}_2$.

To facilitate the computation of smoothing function in EMS algorithm, we exploit a dimension reduction technique

11. A symmetric Gaussian kernel function is often used [13].

[25], [28] to map a graph into Euclidean space as a set of vectors. Specifically, given N users, let $Z \in \mathbb{R}^{N\times N}$ denote the projection matrix, where z_i is the i-th row of Z and is the projection of vertex v_i. We derive Z as follows:

$$Z = \arg\min_{Z^T I Z = c} z^T L z,$$
$$L = D - W, D_{ii} = \sum_{i\neq j} W_{ij} \forall i,$$

where D is a diagonal matrix, I is the identity matrix, and L is the Laplacian matrix of distance matrix $W \in \mathbb{R}^{N\times N}$. This optimization problem attempts to preserve the distance between nodes. However, the constraint $Z^T I Z = c$ restricts that only c columns of Z can be non-zero vector. Therefore, the objective function reduces the dimension of z_i from N to c while maintaining the local structure as much as possible.

Suppose we solve this generalized eigenproblem for first c solutions and the i-th component of j-th eigenvector z_j is denoted as $z_{j,i}$. The projection of i-th node v_i has a K-dimensional representation $f(v_i) = (z_{1,i}, z_{2,i}, \cdots, , z_{K,i})$. In our paper, we employ one of the most widely used nonlinear dimension reduction technique, ISOMAP [25] only. The distance matrix $W = -HSH/2$, where $H = I - 1/N \overrightarrow{1}\overrightarrow{1}^T$ (I is the identity matrix and $\overrightarrow{1}$ is the vector of all ones) and S_{ij} is distance between two nodes in the graph. For our SIG, we first project the users whose graph is G, and then the items whose graph is set to be the complete graph. Other method sharing the above optimization formulation can be used as a substitute without much effort.

By employing the above graph embedding approaches, we project the graph into Euclidean space while preserving the distance between the nodes locally. Therefore, the customers who are socially near and the similar commodity items can be extracted efficiently. Therefore, after the dimension reduction procedure, each node has a K-dimensional representation. Therefore, each hyperedge e of size n comprising of n source nodes and 1 destination node can be mapped to a vector on the space $\mathbb{R}^{(n+1)K}$.

8.2 Shared group weighting

The number of possible candidate hyperedges significantly increases when supporting the hyperedges with multi-destination nodes. Although the s-step in EMS alleviate the data sparsity issue, the s-step can only share the $w_{e,a}$ among the hyperedges with the same size. To reduce the computation complexity, one of the possible ways is to restrict the number of source and destination nodes for a hyperedge. However, the data sparsity problem is still severe since there are too many parameters require being inferred with relatively small number of purchase events by individuals. Moreover, it is promising that the proposed model can derive higher-level information such as different types of behavior patterns of all customers and the proportion of customers with a certain type of behavior for some items. As such, to address the severe data sparsity problem, we propose a parameter sharing method by assuming the latent groups among hyperedges.

Since the activation probability of each hyperedge is modeled by the parameter of Poisson distribution, i.e., λ_A and λ_T, a mixture model of EMS is then applied to partition hyperedges based on the purchase behavior patterns characterized by the intensity parameter λ. In other words, some of hyperedges may share the same activation λ. For example, the frequencies that apple fans buy the new iPhone and influence their friends and Audi fans buy the new car and influence their friends can be modeled as two groups. On the other hand, the frequencies that people buy fried chickens and influence their friends and people buy ham burgers and influence their friends can be grouped as one.

Therefore, given a transaction data set with customers and products, we assume that there are L latent groups among hyperedges, where each latent group shares the same $\lambda_{A,l}$. Let $\zeta_e = \{\zeta_{e1}, \zeta_{e2}, \cdots, \zeta_{e1}, \cdots, \zeta_{eL}\}$ denote the probability vector that hyperedge e belonging to groups 1 to L, where $\sum_{l=1}^{L} \zeta_{el} = 1$. Moreover, let $\Lambda_A = \{\lambda_{A1}, \lambda_{A2} \cdots, \lambda_{A1}, \cdots, \lambda_{AL}\}$ denote the Poisson intensity vector of each latent group. The new task is to 1) discover the K latent groups, 2) find ζ_e for each hyperedge e, and 3) estimate λ_{Al} and ζ_{el} for calculating the activation probability of hyperedges. The log likelihood of the observation is rewritten with the expectation of K_e as follows.

$$\sum_{e\in E_H} \log \sum_{l=1}^{L} \log P(K_e|N_U, \Lambda_{Al}, \zeta_{el}) P(\zeta_{el})$$

Therefore, in **E-step**, in the first iteration, we initialize ζ_e for each hyperedge e randomly. From the second iteration, we use the estimation of Λ_A from previous iteration to infer new ζ_e. The posterior probability of hyperedge e in the latent group l, i.e.,$P(\zeta_{el} = 1|K_e, N_U, \Lambda_A) =$, can be derived as follows.

$$\frac{P(K_e|\zeta_e = l, N_U, \Lambda_{Al}) P(\zeta_{el} = 1)}{\sum_{l=1}^{L} P(K_e|\zeta_{el} = 1, N_U, \Lambda_{Al}) P(\zeta_{el} = 1)} \quad (5)$$

Moreover, given the expectation of K_e, the likelihood is calculated as

$$P(K_e|\zeta_{el} = 1, \Lambda_{Al}) = \frac{e^{-\Lambda_{Al}} - (\Lambda_{Al})_e^K}{K_e!}.$$

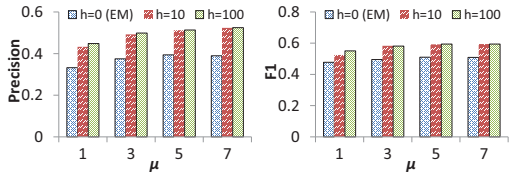
As such, the posterior probability of $P(\zeta_{el} = 1|K_e, N_U, \Lambda_A)$ in Eq. 5 can be regarded as the soft membership of the hyperedge to the latent group. Moreover, $E[K_e]$ is updated by Eq. 4. After calculating the **E-step**, since $Q(p_e|\hat{p}_e^{i-1})$ is in a quadratic form, determining the maximizing values of p_e. Also, $Lambda_A$, $Lambda_T$, and ζ can be maximized independently since they all appear in separate linear terms. Therefore, we start **M-step** by the maximizing $Q(p_e|\hat{p}_e^{i-1})$ with regards to $Lambda_A$, $Lambda_T$, and ζ. For the latent group l, we use $\sum_e \zeta_{el} E[K_e] / \sum_e \zeta_{el}$ to estimate the value of p_e by maximizing the likelihood of Λ_{Al}.

9 Evaluation

We conduct comprehensive experiments to evaluate the proposed SIG model, learning framework and seed selection algorithms. In Section 9.1, we discuss the data preparation for our evaluation. In Section 9.2, we compare the predictive power of the SIG model against two baseline models: i) independent cascade model learned by implementing Variance Regularized EM Algorithm (VAREM) [18] and ii) the generalized threshold (GT) model learned by [11].[12] In

12. http://people.cs.ubc.ca/~welu/downloads.html

addition, we evaluate the learning framework based on the proposed EM and EMS algorithms. Next, in Section 9.3, we evaluate the proposed HAG algorithm for SIMP in comparison to a number of baseline strategies, including random, single node selection, social, and item approaches. Finally, in Section 9.4, we evaluate alternative approaches for diffusion processing, which is essential and critical for HAG, based on SIG-index, Monte Carlo simulations and sorting enhancement.



(a) Precision (b) F1-Score

Fig. 13. Comparisons of precision and F1 in various μ and h on *Epinions*

9.1 Data Preparation

Here, we conduct comprehensive experiments using three real datasets to evaluate the proposed ideas and algorithms. The first dataset comes from Douban [1], a social networking website allowing users to share music and books with friends. Dataset *Douban* contains $5,520,243$ users and $86,343,003$ friendship links, together with $7,545,432$ (user, music) and $14,050,265$ (user, bookmark) pairs, representing the music noted and the bookmarks noted by each user, respectively. We treat those (user, music) and (user, bookmark) pairs as purchase actions. In addition to *Douban*, we adopt two public datasets, i.e., *Gowalla* and *Epinions*. Dataset *Gowalla* contains $196,591$ users, $950,327$ links, and $6,442,890$ check-ins [8]. Dataset *Epinions* contains $22,166$ users, $335,813$ links, 27 categories of items, and $922,267$ ratings with timestamp [22]. Notice that we do not have data directly reflecting item inferences in online stores, so we use the purchase logs for learning and evaluations. The experiments are implemented on an HP DL580 server with 4 Intel Xeon E7-4870 2.4 GHz CPUs and 1 TB RAM.

We split all three datasets into 5-fold, choose one subsample as training data, and test the models on the remaining subsamples. Specifically, we ignore the cases when the user and her friends did not buy anything. Finally, to evaluate the effectiveness of the proposed SIG model (and the learning approaches), we obtain the purchase actions in the following cases as the ground truth: 1) item inference - a user buys some items within a short period of time, and 2) social influence - a user buys an item after at least one of her friends bought the item. The considered periods of item inference and social influence are set differently according to [15] and [29], respectively. It is worth noting that only the hyperedges with the probability larger than a threshold parameter θ are considered. We empirically tune θ to obtain the default setting based on optimal F1-Score. Similarly, the threshold parameter θ for the GT model is obtained empirically. The reported precision, recall, and F1 are the average of these tests. Since both SIGs and the independent cascade model require successive data, we split the datasets into continuous subsamples.

9.2 Model Evaluation

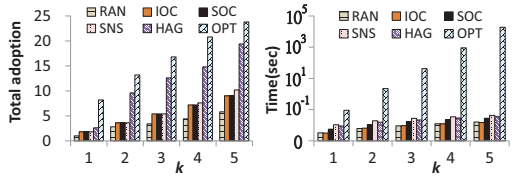
Tables 1 present the precision, recall, and F1 of SIG, VAREM and GT on *Douban*, *Gowalla*, and *Epinions*. All three models predict most accurately on *Douban* due to the large sample size. The SIG model significantly outperforms the other two models on all three datasets, because it takes into account both effects of social influence and item inference, while the baseline models only consider the social influence. The difference of F1 score between SIG and baselines is more significant on *Douban*, because it contains more items. Thus, item influence plays a more important role. Also, when the user size increases, SIG is able to extract more social influence information leading to better performance than the baselines. The offline training time is 1.68, 1.28, and 4.05 hours on *Epinions*, *Gowalla*, *Douban*, respectively.

To evaluate the approaches adopted to learn the activation probabilities of hyperedges for construction of SIG, Fig. 13 compares the precision and F1 of EMS and EM algorithms on Epinions (results on other datasets are consistent and thus not shown due to space limitation). Note that EM is a special case of EMS (with the smoothing parameter $h = 0$, i.e., no similar hyperedge used for smoothing). EMS outperforms EM on both precision and F1-score in all settings of μ (the maximum size of hyperedges) and h tested. Moreover, the precision and F1-score both increases with h as a larger h overcomes data sparsity significantly. As μ increases, more combinations of social influence and item inference can be captured. Therefore, the experiments show that a higher μ improves F1-score without degrades the precision. It manifests that the learned hyperedges are effective for predicting triggered purchases.

9.3 Algorithm Effectiveness and Efficiency

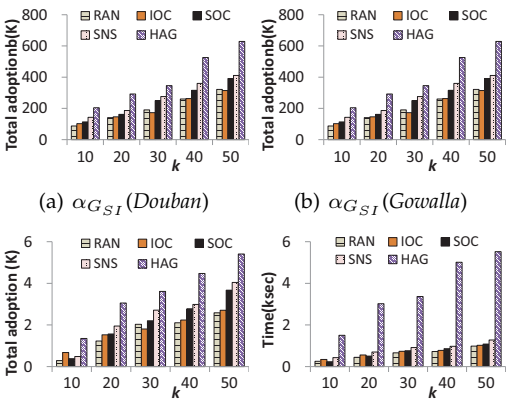
We evaluate HAG proposed for SIMP, by selecting top 10 items as the marketing items to measure their total adoption, in comparison with a number of baselines: 1) *Random approach (RAN)*. It randomly selects k nodes as seeds. Note that the reported values are the average of 50 random seed sets. 2) *Single node selection approach (SNS)*. It selects a node with the largest increment of the total adoption in each iteration, until k seeds are selected, which is widely employed in conventional seed selection problem [6], [7], [17]. 3) *Social approach (SOC)*. It only considers the social influence in selecting the k seeds. The hyperedges with nodes from different products are eliminated in the seed selection process, but they are restored for calculation of the final total adoption. 4) *Item approach (IOC)*. The seed set is the same as HAG, but the prediction is based on item inference only. For each seed set selected by the above approaches, the diffusion process is simulated 300 times. We report the average in-degree of nodes learned from the three datasets in the following: Douban is 39.56; Gowalla is 9.90; Epinions is 14.04. In this section, we evaluate HAG by varying the number of seeds (i.e., k) using two metrics: 1) total adoption, and 2) running time.

To understand the effectiveness, we first compared all those approaches with the optimal solution (denoted as OPT) in a small subgraph sampled, *Sample*, from the SIG of *Douban* with 50 nodes and 58 hyperedges. Figures 14



(a) *Sample* (b) Time

Fig. 14. Total adopting and running time of *Sample* in various k



(a) $\alpha_{G_{SI}}$(*Douban*) (b) $\alpha_{G_{SI}}$(*Gowalla*)

(c) $\alpha_{G_{SI}}$(*Epinions*) (d) Time (*Douban*)

Fig. 15. Total adoption $\alpha_{G_{SI}}$ and running time in various k

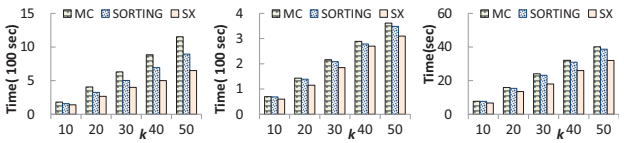
(a) displays the total adoption obtained by different approaches. As shown, HAG performs much better than the baselines and achieves comparable total adoption with OPT (the difference decreases with increased k). Note that OPT is not scalable as shown in Figures 14 (b) since it needs to examine all combination with k nodes. Also, OPT takes more than 1 day for selecting 6 seeds in *Sample*. Thus, for the rest of experiments, we exclude OPT.

Figures 15 (a)-(c) compare the total adoptions of different approaches in the SIG learned from real networks. They all grow as k increases, since a larger k increases the chance for seeds to influence others to adopt items. Figure 15 (a)-(c) manifest that HAG outperforms all the other baselines for any k in SIG model. Among them, SOC fails to find good solutions since item inference is not examined during seed selection. IOC performs poorly without considering social influence. SNS only includes one seed at a time without considering the combination of nodes that may activate many other nodes via hyperedges.

Figure 15 (d) reports the running time of those approaches. Note that the trends upon *Gowalla* and *Epinions* are similar with *Douban*. Thus we only report the running time of *Douban* due to the space constraint. Taking the source combinations into account, HAG examines source combinations of hyperedges in E_H and obtains a better solution by spending more time since the number of hyperedges is often much higher than the number of nodes.

9.4 Online Diffusion Processing

Diffusion processing is an essential operation in HAG. We evaluate the efficiency of diffusion processing based on SIG-index (denoted as SX), in terms of the running time, in comparison with that based on the original Monte Carlo simulation (denoted as MC) and the sorting enhancement (denoted as SORTING), which accesses the hyperedges in descending order of their weights. Figure 16 plots the running time of SX, SORTING, and MC under various k using the *Douban*, *Gowalla*, and *Epinions*. For each k, the average running times of 50 randomly selected seed sets for SX, SORTING, and MC, are reported. The diffusion process is simulated 300 times for each seed set. As Figure 16 depicts, the running time for all the three approaches grows as k increases, because a larger number of seeds tends to increase the chance for other nodes to be activated. Thus, it needs more time to diffuse. Notice that SX takes much less time than SORTING and MC, because SX avoids accessing hyperedges with no source nodes newly activated while calculating the activation probability. Moreover, the SIG-index is updated dynamically according to the activated nodes in diffusion process. Also note that the improvement by MC over SORTING in *Douban* is more significant than that in *Gowalla* and *Epinions*, because the average in-degree of nodes is much larger in *Douban*. Thus, activating a destination at an early stage can effectively avoid processing many hyperedges later.



(a) Douban (b) Gowalla (c) Epinions

Fig. 16. Running time of different simulation methods

10 Conclusion

In this paper, we argue that existing techniques for item inference recommendation and seed selection need to jointly take social influence and item inference into consideration. We propose Social Item Graph (SIG) for capturing purchase actions and predicting potential purchase actions. We propose an effective machine learning approach to construct an SIG from purchase action logs and learn hyperedge weights. We also develop efficient algorithms to solve the new and challenging Social Item Maximization Problem (SIMP) that effectively select seeds for marketing. Experimental results demonstrate the superiority of the SIG model over existing models and the effectiveness and efficiency of the proposed algorithms for processing SIMP. We also plan to further accelerate the diffusion process by indexing additional information on SIG-index.

Appendix A
Proof of the First Zonklar Equation

Appendix one text goes here.

Appendix B

Appendix two text goes here.

Acknowledgments

The authors would like to thank...

REFERENCES

[1] The Douban dataset download. http://arbor.ee.ntu.edu.tw/~hhshuai/Douban.tar.bz2.
[2] When social influence meets association rule learning. In *CoRR (1502.07439)*, 2016.
[3] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, 1993.
[4] R. M. Bond, C. J. Fariss, J. J. Jones, A. D. Kramer, C. Marlow, J. E. Settle, and J. H. Fowler. A 61-million-person experiment in social influence and political mobilization. *Nature*, 2012.
[5] S. Bourigault, S. Lamprier, and P. Gallinari. Representation learning for information diffusion through social networks: an embedded cascade model. In *WSDM*, 2016.
[6] S. Chen, J. Fan, G. Li, J. Feng, K. Tan, and J. Tang. Online topic-aware influence maximization. In *VLDB*, 2015.
[7] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD*, 2010.
[8] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In *KDD*, 2011.
[9] P. Domingos and M. Richardson. Mining the network value of customers. In *KDD*, 2001.
[10] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan. Learning influence probabilities in social networks. In *WSDM*, 2010.
[11] A. Goyal, W. Lu, and L. V. S. Lakshmanan. SIMPATH: An efficient algorithm for influence maximization under the linear threshold model. In *ICDM*, 2011.
[12] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 3rd Edition, 2011.
[13] N. L. Hjort and M. C. Jones. Locally parametric nonparametric density estimation. In *The Annals of Statistics*, 1996.
[14] R. V. Hogg and E. A. Tanis. *Probability and statistical inference*. Prentice Hall, 7th Edition, 2005.
[15] R. Jones and K. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *CIKM*, 2008.
[16] R. J. B. Jr. Efficiently mining long patterns from databases. In *SIGMOD*, 1998.
[17] D. Kempe, J. M. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003.
[18] H. Li, T. Cao, and Z. Li. Learning the information diffusion probabilities by using variance regularized em algorithm. In *ASONAM*, 2014.
[19] J. Nail. The consumer advertising backlash, 2004. Forrester Research and Intelliseek Market Research Report.
[20] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *ICDT*, 1999.
[21] B. W. Silverman, M. C. Jones, J. D. Wilson, and D. W. Nychka. A smoothed em approach to indirect estimation problems, with particular, reference to stereology and emission tomography. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1990.
[22] J. Tang, H. Gao, H. Liu, and A. D. Sarma. etrust: Understanding trust evolution in an online world. In *KDD*, 2012.
[23] J. Tang, S. Wu, and J. Sun. Confluence: conformity influence in large social networks. In *KDD*, 2013.
[24] Y. Tang, X. Xiao, and Y. Shi. Influence maximization: near-optimal time complexity meets practical efficiency. In *SIGMOD*, 2014.
[25] J. B. Tenenbaum, V. d. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 2000.
[26] H. Vahabi, I. K. F. Gullo, and M. Halkidi. Difrec: A social-diffusion-aware recommender system. In *CIKM*, 2015.
[27] Z. Wen and C.-Y. Lin. On the quality of inferring interests from social neighbors. In *KDD*, 2010.
[28] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin. Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.
[29] J. Yang and J. Leskovec. Modeling information diffusion in implicit networks. In *ICDM*, 2010.

PLACE PHOTO HERE

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.