Efficient Revenue Maximization for Viral Marketing in Social Networks

Yuan Su[1](✉), Xi Zhang[1], Sihong Xie[2], Philip S. Yu[3,4], and Binxing Fang[1]

[1] Beijing University of Posts and Telecommunications, Beijing, China
{timsu,zhangx,fangbx}@bupt.edu.cn
[2] Lehigh University, Bethlehem, PA, USA
sxie@cse.lehigh.edu
[3] University of Illinois at Chicago, Chicago, IL, USA
psyu@cs.uic.edu
[4] Institute for Data Science, Tsinghua University, Beijing, China

Abstract. In social networks, the problem of revenue maximization aims at maximizing the overall revenue from the purchasing behaviors of users under the influence propagations. Previous studies use a number of simulations on influence cascades to obtain the maximum revenue. However, these simulation-based methods are time-consuming and can't be applied to large-scale networks. Instead, we propose calculation-based algorithms for revenue maximization, which gains the maximum revenue through fast approximate calculations within local acyclic graphs instead of the slow simulations across the global network. Furthermore, a max-Heap updating scheme is proposed to prune unnecessary calculations. These algorithms are designed for both the scenarios of unlimited and constrained commodity supply. Experiments on both the synthetic and real-world datasets demonstrate the efficiency and effectiveness of our proposals, that is, our algorithms run in orders of magnitude faster than the state-of-art baselines, and meanwhile, the maximum revenue achieved is nearly not affected.

Keywords: Revenue maximization · Social networks · Viral marketing

1 Introduction

Revenue maximization [1,10,14,20,23,25] is the problem of devising the marketing strategy to obtain the optimal revenue in social networks, by determining the price of a commodity and identifying a small set of influential vertices to give out commodities for free. In contrast to influence maximization [5–8,13,18,19], revenue maximization takes into account the impact of prices on adopting commodities, and quantifies the revenue obtained from the adoption after the information diffusion process.

In networks, one user's valuation towards a commodity often exhibits positive network externalities [2,4,12,17,21,23,24], i.e., a user's valuation will be positively influenced by other users who have already purchased the commodity.

Hoping to increase the future revenue, in networks, a seller can give out some commodities for free to some individuals to drive up the demands of the commodities. We study the seller's marketing strategy as follows: the seller selects a set S of consumers and gives each of them a commodity for free, and then sets a unified price λ for the other consumers to buy the commodity, such that the expected revenue is maximized. Different from the influence maximization problem, there exists a challenge in revenue maximization when optimizing the price λ. Specifically, lowering the price λ increases the number of purchasers but may not improve the total revenue, while raising the price loses potential purchases and may not improve the total revenue as well. Such a complex relationship is an inherent and challenging problem in revenue maximization.

We consider two scenarios of commodity supply, the unlimited supply and the constrained supply. The unlimited supply means that a commodity is with nearly zero marginal cost of manufacturing and the seller supply is unrestricted [23], e.g., the number of SIM cards released by a telecommunication company can be considered to be unlimited. The constrained supply suits that a seller has limited production or service capacity, e.g., the tickets of a cinema are restricted. For the scenario of unlimited supply, a randomized algorithm using a large number of simulations on influence cascades for revenue estimation is proposed in [23], which is time-consuming and can't be applied to large-scale networks. Moreover, it does not consider the "awareness" of a commodity in the dynamic process of commodity adoptions. In the dynamic process, a user won't be "aware" of a commodity until one of her in-degree neighbors adopted it. Thus, awareness should be the premise of the purchasing behavior and need to be incorporated into the dynamic process. For the scenario of constrained supply, a heuristic algorithm is proposed in [25], which, however, is only suitable to the situation that each user's inherent valuation of a commodity is known. This is a strong assumption that commonly does not hold in the real-world viral marketing, where a consumer's inherent estimation is usually unknown to the sellers.

To address the aforementioned problems, we consider the awareness of a commodity as the premise of a user's purchase, and generalize the revenue maximization problem by setting each user's inherent valuation as unknown. We then propose efficient and effective algorithms for both the scenarios of the unlimited supply and the constrained supply. Firstly, a fast method of approximate revenue calculation using local acyclic graphs is proposed to replace the time-consuming simulations on influence cascades. The intuition is to restrict the influence spreading to a vertex within its local acyclic graph, and the activation probability of this vertex can be calculated approximately in its local acyclic graph instead of the original large network. Based on this method, for the scenario of unlimited supply, we propose an efficient algorithm, i.e., the Calculation-based Randomized (CR) algorithm, in which the revenues are approximately calculated based on local acyclic graphs. For the scenario of constrained supply, we develop an efficient algorithm called the Calculation-based max-Heap updating Greedy (CHG) algorithm. The main idea of this algorithm contains two crucial parts: (1) fast approximate revenue calculation based on the local acyclic graphs;

(2) max-Heap updating scheme, which relies on the submodularity of the objective function of the revenue maximization problem. We conduct extensive experiments on both the synthetic and real-world networks. The results indicate that the proposed algorithms can achieve almost the same revenue while running in orders of magnitude faster than the baselines.

The contributions of this paper are summarized as below:

- We address the problem of revenue maximization for both the cases that the commodity supply is unlimited and constrained, and propose a fast and general method to approximately calculate the revenue.
- For unlimited commodity supply, an efficient algorithm CR is proposed based on the approximate revenue calculations; for constrained commodity supply, an efficient algorithm CHG is proposed based both on the approximate revenue calculations and max-Heap updating scheme.
- The results of experiments demonstrate the efficiency and effectiveness of CR algorithm and CHG algorithm, which can run orders of magnitude faster than the state-of-art baselines while gaining almost the same revenue.

2 Revenue Maximization Problem

In this section, we first describe the price-sensitive dynamic process in social networks, and then formally present the problem of revenue maximization.

The social network. Given a commodity, a social network is represented as a weighted graph $G = (V, E)$, where V is the set of potential consumers, and E is the set of edges. For each vertex $v \in V$, her inherent valuation of the commodity $\chi_v \in [0, 1]$ is selected uniformly at random in $[0, 1]$, under the assumption that her inherent valuation is unknown. If the purchase by one vertex u will directly encourage the desire of another vertex v for the commodity, there exists an edge $e_{u,v} \in E$ from u to v, and the influence weight on edge $e_{u,v}$ is represented by $w_{u,v} > 0$. If there is no edge between vertices u and v, then $w_{u,v} = 0$. The non-negative and non-decreasing function $F : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ transforms the edge weight into the valuation increment. Given a set of vertices S which can directly influence the purchase decision of v, then v's valuation will be $\chi_v + F(\Sigma_{u \in S} w_{u,v})$.

The price-sensitive dynamic process. The activation of a vertex by a commodity has two important factors [16,20]: awareness and purchase. For the awareness, if one vertex is activated by the commodity, she makes her out-degree neighbors aware of the commodity, which means that these neighbors are exposed to the commodity and know about it. In other words, a vertex is aware of the commodity in condition that at least one of her in-degree neighbors is activated by the commodity. For the purchase, if some vertices are activated by the commodity in the network, they will exert influences on their out-degree neighbors, while the influence strength is denoted by the edge weights. Then the valuations of the out-degree neighbors towards the commodity will be incremented. A vertex who is aware of the commodity will purchase it if her valuation outweighs the price. A vertex is activated by a commodity if it is aware of the commodity and purchase that commodity.

The diffusion proceeds in discrete steps. Initially at step $t = 0$, all the vertices in seed set S_0 are activated, and the seller sets a fixed price $\lambda \in \Lambda$ for the commodity. Here Λ is the price set. At each step $t \geq 1$, for each vertex v, if some of v's in-degree neighbors are activated by the commodity, v becomes aware of the commodity, and then v's valuation will be updated by $\chi_v^t = \chi_v + F(\Sigma_{i \in S_{t-1}} w_{i,v})$, where $S_{t-1} \subseteq V$ is the set of vertices activated after time $t-1$. In this paper, we set $F(x) = x$ for convenience, following the same setting as in [23,25]. If $\chi_v^t \geq \lambda$, then v will purchase the commodity. This process continues and the dynamic process propagates until no more vertices are activated. If the commodity supply is constrained, when the activated number achieved is larger than the quantity Q of the commodity, the dynamic process terminates. For any vertex, if it has purchased the commodity, it will stay activated. Different from Linear Threshold Model [18] adopted in influence maximization problem where the commodity price is not considered, in revenue maximization problem, it is obvious that λ will affect the number of infected nodes: the higher the price, the fewer vertices will be activated; while the lower the price, the more vertices will be activated.

In our setting, a vertex is activated by the commodity in the propagation of social influence if and only if (1) the vertex is in the seed set, or (2) at least one of v's in-degree neighbors are activated by the commodity, i.e., v is aware of the commodity, and v's current valuation is no less than the price of the commodity.

In this paper, we set $\Sigma_u w_{u,v} \leq \lambda$, specifically, $\Sigma_u w_{u,v} \in [0, \alpha]$ and $\lambda \in [\alpha, 1]$ where $\alpha \in [0,\ 1]$. Please note that if all the in-degree neighbors of vertex v are activated, as $\chi_v \in [0, 1]$ is selected uniformly at random, v's valuation $\chi_v + \Sigma_u w_{u,v}$ ($u \in v$'s in-degree neighbors) will be selected in $[\Sigma_u w_{u,v}, 1 + \Sigma_u w_{u,v}]$ uniformly at random. Then the activation probability that $\chi_v + \Sigma_u w_{u,v} \geq \lambda$ is $1 - \lambda + \Sigma_u w_{u,v}$. This probability is exactly guaranteed in $[0,\ 1]$ as $\Sigma_u w_{u,v} \in [0, \alpha]$ and $\lambda \in [\alpha, 1]$. Besides, in the case that not all the in-degree neighbors are activated, we can get the same conclusion. These settings can make price-sensitive dynamic process have better probability characteristics for mathematical calculations.

Revenue maximization problem. The seller selects a set of vertices S as seeds and gives them the commodity for free. After the diffusion process, the number of activated vertices $\sigma(S, \lambda)$ is obtained, while the revenue comes from the number of activated vertices except the seeds, that is, $\sigma(S, \lambda) - |S|$. We define the revenue function as $\pi : 2^V \times \Lambda \rightarrow \mathbb{R}$. Then $\pi(S, \lambda) = \lambda(\sigma(S, \lambda) - |S|)$.

Formally, we define *Revenue Maximization Problem* as follows: given a commodity and a graph $G = (V, E)$, the problem is to determine the optimal price λ for this commodity and identify a seed set S to be the initial consumers, such that the expected revenue $\pi(S, \lambda)$ is maximized.

3 Approximate Revenue Calculation

In order to solve the revenue maximization problem efficiently, we provide a method for approximate revenue calculation based on local acyclic graph constructions, which is the basis for the proposed heuristic algorithms.

Activation Probability Calculation. In general directed graphs, there exists loops, and the expected social influence spread from the seed set is difficult to compute. But if the directed graphs are acyclic, the computation of the expected social influence can be conducted. For an acyclic graph G_A, given a seed set $S \subseteq V$ and the price λ, let $ap_\lambda(S, v)$ be the activation probability of vertex v.

We first topologically sort [15] all the vertices into a linear order $\{v_1, v_2, ...v_n\}$, where n is the number of vertices, with the seeds at the beginning of this order, and then compute $ap_\lambda(S, v_i)$ $(1 \leq i \leq n)$ for all v_i following this order. The influence on a vertex v_i comes from the vertices in front of v_i in this topological order, but not comes from the vertices behind, with the reason that the graph G_A is acyclic. In other words, the valuation of v_i is influenced by the vertices in front of it in the order. Suppose the number of seed set S is k. The first k vertices in this topological order $\{v_1, v_2, ...v_k\}$ are the seeds, so that $ap_\lambda(S, v_i) = 1(i \leq k)$ as they have already been activated. For the vertex v_{k+1}, the influence of her valuation comes from the set $\{v_1, v_2, ...v_k\}$. If some of her in-degree neighbors are from the seed set, v_{k+1}'s valuation will increase to $\chi_{v_{k+1}} + \sum_{v_i \in S} w_{v_i, v_{k+1}}$. Because $\chi_{v_{k+1}} \in [0, 1]$ is selected uniformly at random, $\chi_{v_{k+1}} + \sum_{v_i \in S} w_{v_i, v_{k+1}}$ will be selected in $[\sum_{v_i \in S} w_{v_i, v_{k+1}}, 1 + \sum_{v_i \in S} w_{v_i, v_{k+1}}]$ uniformly at random. If $\chi_{v_{k+1}} + \sum_{v_i \in S} w_{i, v_{k+1}} \geq \lambda$, then v_{k+1} will adopt the commodity. Thus, the probability of v_{k+1}'s adoption is $ap_\lambda(S, v_{k+1}) = 1 - \lambda + \sum_{v_i \in S} w_{v_i, v_{k+1}}$. As $S = \{v_1, v_2, ...v_k\}$ and $ap_\lambda(S, v_i) = 1(i \leq k)$, then we got $ap_\lambda(S, v_{k+1}) = 1 - \lambda + \sum_{v_i \in \{v_1, v_2, ...v_k\}} ap_\lambda(S, v_i) w_{v_i, v_{k+1}}$. Similarly, for the vertex v_{k+2}, the influence on her valuation comes from $\{v_1, v_2, ...v_{k+1}\}$. Her valuation will become $\chi_{v_{k+2}} + \sum_{i \in \{v_1, v_2, ...v_{k+1}\}} ap_\lambda(S, v_i) w_{v_i, v_{k+2}}$. Then the probability of v_{k+2}'s adoption is $ap_\lambda(S, v_{k+2}) = 1 - \lambda + \sum_{v_i \in \{v_1, v_2, ...v_{k+1}\}} ap_\lambda(S, v_i) w_{v_i, v_{k+2}}$. For vertex $v_i(i > k + 2)$, the probability of adoption can be computed in the similar way. Suppose the set of vertices in front of vertex v_i in the topological order is C_{v_i}. In general, for each vertex $v_i(i > k)$, it can be derived that the activation probability is

$$ap_\lambda(S, v_i) = 1 - \lambda + \sum_{u \in C_{v_i}} ap_\lambda(S, u) w_{u, v_i} \tag{1}$$

The activation probability of vertex v_i is computed according to that of the vertices in front of v_i in the topological order. When computing the activation probability of v_i, all the activation probability of v_i's in-neighbors have been computed. Equation (1) shows that the activation probability of all the vertices can be calculated linear to the number of edges of an acyclic graph.

However, real-world social networks are not acyclic, so we cannot use Eq. (1) directly. To address this problem, we construct a local acyclic graph for every vertex v, and then use the activation probability of v in its local acyclic graph to approximate that in the original network. For each price λ, vertex v is associated

Algorithm 1. $LA_\lambda(v)$ construction

```
1: Initialize $LA_\lambda(v) \leftarrow v$, set $v$ newly added and $inf_{LA_\lambda(v)}(v, v) = 1$
2: for every added vertex $x \in LA_\lambda(v)$ do
3:     for every in-neighbor $u$ ($u \notin LA_\lambda(v)$) of $x$ do
4:         if $u$ is first check then
5:             $inf_{LA_\lambda(v)}(u, v) = w_{u,x} \times inf_{LA_\lambda(v)}(x, v)$
6:         else
7:             $inf_{LA_\lambda(v)}(u, v) += w_{u,x} \times inf_{LA_\lambda(v)}(x, v)$
8:         if $inf_{LA_\lambda(v)}(u, v) \geq \theta$ then
9:             Add $u$ into $LA_\lambda(v)$
10:            Add all $e_{u,y}$ ($y$ in $LA_\lambda(v)$) into $LA_\lambda(v)$
11: Output $LA_\lambda(v)$
```

with a local acyclic graph $LA_\lambda(v)$, which denotes a subgraph of G rooted at v. We need to find $LA_\lambda(v)$ that covers a significant portion of influence from other vertices to v and ignores the others. Given a seed set S, we assume that the influence from S to v is only propagated within $LA_\lambda(v)$.

Influence Calculation. The influence to v propagated from u in an acyclic graph G_A, defined as $inf_{G_A}(u, v)$, is v's incremental activation probability, in the case that u is activated relative to that u is not activated. We topologically sort [15] all the vertices that can reach v in G_A into a sequence, and then reverse this sequence into a new order. Initially, $inf_{G_A}(v, v) = 1$. Then for each node $u \neq v$ according to the order

$$inf_{G_A}(u, v) = \sum_{x \in V \setminus \{v\}} w_{u,x} inf_{G_A}(x, v) \tag{2}$$

In an acyclic graph G_A, supposing that the item price is λ and the seed set is S, the incremental activation probability of v contributed from u is $(1 - ap_\lambda(S, u))inf_{G_A}(u, v)$, which can be derived by repeatedly expanding $ap_\lambda(S, v)$ using Eq. (1), where $inf_{G_A}(u, v)$ is calculated according to Eq. (2).

Local Acyclic Graph Construction. According to Eq. (2), the influence $inf_{LA_\lambda(v)}(u, v)$ from u to v can be calculated in constructing an acyclic graph $LA_\lambda(v)$. Algorithm 1 shows how to construct the local acyclic graph of each vertex v and calculate the influences from other vertices to v. Let $\theta \in [0, 1]$ be a threshold, and we need to make sure that $inf_{LA_\lambda(v)}(u, v) \geq \theta$ for each vertex u in $LA_\lambda(v)$. The intuition behind is that we need to find $LA_\lambda(v)$ that covers a significant portion of influences from other vertices to v and ignores the vertices that have only little influence to v. Initially vertex v is added into $LA_\lambda(v)$. Then in each iteration, for every newly added vertex $x \in LA_\lambda(v)$, for every in-neighbor u of x, we calculate $inf_{LA_\lambda(v)}(u, v)$ and select every vertex u satisfying $inf_{LA_\lambda(v)}(u, v) \geq \theta$, and then add these vertices into $LA_\lambda(v)$ together with all the corresponding edges. The process continues until there is a vertex u with $inf_{LA_\lambda(v)}(u, v) < \theta$. The runtime of constructing each $LA_\lambda(v)$ is linear to $|E|$.

Total Revenue Calculation. For each price λ, vertex v is associated with a local acyclic graph $LA_\lambda(v)$. We use the activation probability of v in its local acyclic graph to approximate that in the original network. Given a seed set S, we assume that the influence from S to v is only propagated within $LA_\lambda(v)$. The activated number $\sigma(S,\lambda)$ is $\sum_v ap_\lambda(S,v)$, then the expected revenue $\pi(S,\lambda)$ is

$$\pi(S,\lambda) = \lambda(\sum_{v\in V} ap_\lambda(S,v) - |S|) \tag{3}$$

Marginal Revenue Calculation. Suppose the current seed set is S, if a vertex u is selected as a seed, the incremental influence that u imposes on the activation probability of v, i.e., $ap_\lambda(S\cup\{u\},v)-ap_\lambda(S,v)$, is $(1-ap_\lambda(S,u))inf_{LA_\lambda(v)}(u,v)$. We define the incremental marginal revenue from u as $\pi(u|S,\lambda)$. Applying Eq. (3),

Algorithm 2. CR Algorithm

```
1: for every price $\lambda$ do
2:     for each vertex $v \in V$ do
3:         $LA_\lambda(v)$ construction
4:     Initialize $X_0 \leftarrow \emptyset$, $Y_0 \leftarrow V$, $n \leftarrow |V|$
5:     for $i = 1$ to $n$ do
6:         $a_i \leftarrow \pi(u_i|X_{i-1},\lambda)$ according to Eq. (4)
7:         $b_i \leftarrow \pi(u_i^-|Y_{i-1},\lambda)$ according to Eq. (5)
8:         $a_i' \leftarrow max\{a_i,0\}$, $b_i' \leftarrow max\{b_i,0\}$
9:         with probability $a_i'/(a_i'+b_i')$ do:
10:            $X_i \leftarrow X_{i-1}\cup\{u_i\}$, $Y_i \leftarrow Y_{i-1}$
11:        else (with probability $b_i'/(a_i'+b_i')$) do:
12:            $X_i \leftarrow X_{i-1}$, $Y_i \leftarrow Y_{i-1}\backslash\{u_i\}$
13:    $S \leftarrow X_n$ (or equivalently $Y_n$)
14:    Calculate $\pi(S,\lambda)$ according to Eq. (3)
15: Output $\lambda$ and $S$ with the maximum $\pi(S,\lambda)$
16: * If $a_i' = b_i' = 0$, we assume $a_i'/(a_i'+b_i') = 1$
```

Algorithm 3. CG Algorithm

```
1: for every price $\lambda$ do
2:     for each vertex $v \in V$ do
3:         $LA_\lambda(v)$ construction
4:     Initialize $S_0 \leftarrow \emptyset$
5:     for $i = 1$ to $Q$ do
6:         if $(\sigma(S_{i-1},\lambda) = \sum_v ap_\lambda(S_{i-1},v)) \geq Q$ then
7:             break
8:         for each vertex $v \in V\backslash S_{i-1}$ do
9:             Calculate $\pi(v|S_{i-1},\lambda)$ according to Eq. (4)
10:        $u = argmax_{v\in V\backslash S_{i-1}}\pi(v|S_{i-1},\lambda)$
11:        if $\pi(u|S_{i-1},\lambda) \leq 0$ then
12:            break
13:        $S_i \leftarrow S_{i-1}\cup\{u\}$
14:    For price $\lambda$, the seed set is $S_i$
15:    Calculate $\pi(S_i,\lambda)$ according to Eq. (3)
16: Output $\lambda$ and $S_i$ with the maximum $\pi(S_i,\lambda)$
```

$$\pi(u|S,\lambda) = \pi(S\cup\{u\},\lambda) - \pi(S,\lambda) = \lambda(-1 + (1-ap_\lambda(S,u))\sum_{v\in V} inf_{LA_\lambda(v)}(u,v)) \tag{4}$$

Similarly, if $u \in S$ is removed from S, the decreased marginal revenue is

$$\pi(u^-|S,\lambda) = \lambda(1 + (ap_\lambda(S\backslash\{u\},u) - 1)\sum_{v\in V} inf_{LA_\lambda(v)}(u,v)) \tag{5}$$

The local acyclic graph constructions as well as the total and marginal revenue calculations are the basis for the proposed heuristic algorithms.

4 The Proposed Algorithms

We propose two efficient algorithms: (1) the Calculation-based Randomized (CR) algorithm for unlimited commodity supply; (2) the Calculation-based max-Heap updating Greedy (CHG) algorithm for constrained commodity supply.

4.1 CR Algorithm

For unlimited commodity supply, instead of time-consuming simulations in [23], we calculate the approximate marginal and total revenue.

Algorithm 2 shows CR. In the main loop of lines 1–14, we pick a price in each round and identify the corresponding seed set. Lines 2–3 are the preparation phase for each price, in which each $LA_\lambda(v)$ is generated, and $inf_{LA_\lambda(v)}(u,v)$ for all $u \in LA_\lambda(v)$ are gained as well. Lines 5–12 identify whether a vertex is selected as a seed. Instead of using a large number of simulations to get the marginal revenue, line 6 calculates the incremental marginal revenue from vertex u_i when the seed set is X_{i-1}, and line 7 calculates the decreased marginal revenue from vertex u_i when the seed set is Y_{i-1}. Lines 9–12 give the probability of vertex u_i being selected as a seed or the probability being not selected as a seed. Line 13 gives the seed set at the predetermined price, and line 14 calculates the total revenue of this seed set and the predetermined price. Finally, in line 15, we choose the price and the corresponding seed set achieving the maximum revenue. The time complexity of CR is $O(|\Lambda||V||E|)$.

4.2 CHG Algorithm

For the scenario of constrained commodity supply, we propose CG and CHG. A vertex with greater influence should be regarded as more important, so that greedily selecting the individuals with greatest influence as seeds can lead to a feasible solution. To make the algorithm more efficient, in CG and CHG, we approximately calculate the revenue within the local acyclic graphs. Compared with CG, CHG uses a max-Heap updating scheme to reduce unnecessary calculations to achieve better efficiency.

Algorithm 3 shows CG. In the main loop of lines 1–15, we select a price λ in each round and identify the seed set at λ. Lines 2–3 are the preparation phase for each price, in which each $LA_\lambda(v)$ is generated, and $inf_{LA_\lambda(v)}(u,v)$ for all $u \in LA_\lambda(v)$ are obtained as well. In the seed selecting process for price λ, we iteratively select a seed with the maximum marginal revenue (lines 5–13), where the marginal revenue can be calculated as Eq. 4. The iteration process can be terminated in two cases: (1) the number of the activated vertices exceeds the quantity of the commodity Q (lines 6–7); (2) the examined seed's marginal revenue is negative (lines 11–12). The seed set at each price λ is obtained in line 14. Finally, we choose the price and the corresponding seed set achieving the maximum revenue (line 16). In CG, the marginal revenue from one vertex (line 9), the activated number of a seed set (line 6), the total revenue by setting a price and a seed set (line 15) are all calculated based on the local acyclic graphs. The time complexity of CG is $O(Q|\Lambda||V||E|)$.

However, there is still a limitation of CG. To identify the seed with the maximum marginal revenue, given the current seed set S_{i-1}, we have to enumerate all candidate vertices to obtain the one with the maximum marginal revenue. Actually, for some of the vertices, the computations for their marginal revenue

Algorithm 4. CHG Algorithm

```
1: for every price $\lambda$ do
2:     for each vertex $v \in V$ do
3:         $LA_\lambda(v)$ construction
4:     Initialize $S_0 \leftarrow \emptyset$, and initialize a hash map $M_\lambda$
5:     Build max-Heap $H_\lambda$ with $\pi(\{v\}|\emptyset, \lambda)$ of all $v \in V$
6:     for $i = 1$ to $Q$ do
7:         if $(\sigma(S_{i-1}.\lambda) = \sum_v ap_\lambda(S_{i-1}, v)) \geq Q$ then
8:             break
9:         Clear hash map $M_\lambda$
10:        while the $i$'th seed not obtained do
11:            $v = H_\lambda.pop()$
12:            if $v \notin M_\lambda$ (unchecked) then
13:                Calculate $\pi(\{v\}|S_{i-1}, \lambda)$ based on Eq. (4)
14:                Add $\langle v, \pi(\{v\}|S_{i-1}, \lambda)\rangle$ into $H_\lambda$
15:                Put $\langle v, calculated\rangle$ into $M_\lambda$ (set checked)
16:                continue
17:            if $v \in M_\lambda$ (checked) then
18:                $S_i \leftarrow S_{i-1} \cup \{v\}$
19:        if $\pi(\{v\}|S_{i-1}, \lambda) \leq 0$ then
20:            Remove $v$ from $S_i$
21:            break
22:     For price $\lambda$, the seed set is $S_i$
23:     Calculate $\pi(S_i, \lambda)$ according to Eq. (3)
24: Output $\lambda$ and $S_i$ with the maximum $\pi(S_i, \lambda)$
```

can be avoided. The intuition behind is that revenue functions follow the submodular property, i.e., the marginal revenue from a vertex decreases as the seed set grows. It has been proved that the revenue function $\pi(S, \lambda)$ is non-negative submodular [3,9,23] if the commodity supply is unlimited, and it is obvious that the property also holds when the commodity supply is constrained, for the quantity constraint will not break this property. Then we propose the CHG algorithm using a max-Heap updating scheme to prune unnecessary calculations.

In CHG algorithm, the process to identify seeds at price λ is as follows. Firstly we calculate the marginal revenue $\pi(v|\emptyset, \lambda)$ for each vertex $v \in V$, and build a max-Heap H_λ with the initial marginal revenue $\pi(v|\emptyset, \lambda)$ of every vertex. Obviously, the top vertex on the max-Heap is the first seed selected for price λ, denoted by u. We pop u, add u into S_0, adjust the heap, and set all the vertices in the max-Heap unchecked. Then we keep updating this heap to select the rest seeds. Specifically, in the following iterations, we pop the top vertex v in the max-Heap, and deal with v in two cases: (1) if v's marginal revenue $\pi(v|S_{i-1}, \lambda)$ is unchecked, we calculate $\pi(v|S_{i-1}, \lambda)$, add $\langle v, \pi(v|S_{i-1}, \lambda)\rangle$ into the max-Heap H_λ, and set it checked; (2) if v's marginal revenue is already checked, v will be chosen as a seed, and thus the current iteration will be terminated and all the vertices in the max-Heap is set unchecked. A hash map M_λ is used to maintain whether a vertex is checked or not. According to the submodular property, if the marginal revenues of some vertices obtained in previous iterations is less than the marginal revenue from another vertex in the current iteration, it's not possible for those vertices to be selected as seeds in the current iteration. Consequently, unnecessary computations can be pruned. Algorithm 4 shows CHG. The marginal revenue from one vertex (line 13), the activated number of a seed set (line 7), and the total revenue by setting a price and a seed set (line 23) are all calculated based on the local acyclic graphs. In the seed selecting process for price λ, we first build a max-Heap (line 5), and then iteratively select a seed with the maximum marginal revenue from the heap (lines 6–21). The time complexity of CHG algorithm is $O(Q|\Lambda||V||E|)$. With regard to the effectiveness, the maximum revenue obtained by CHG is the same as that of CG.

5 Evaluation

In this section, we conduct experiments on both synthetic networks and real-world networks to demonstrate the efficiency and effectiveness of the proposed algorithms CR and CHG. All the algorithms are implemented in JAVA, and performed on Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40 GHz.

5.1 Experimental Setup

Datasets. We do experiments in synthetic networks and real-world networks. Synthetic networks are power-law graphs generated by NetworkX [1] with various sizes, which will be introduced in details in Sect. 5.2. We use four real-world networks: (a) Residence Hall Friendship (RHF) network [11] with 217 vertices and 2,672 edges; (b) Ego-Facebook (EgoFB) network [22] with 4,039 vertices and 88,234 edges; (c) NetHEPT network [5] with 15,233 vertices and 58,891 edges; and (d) NetPHY network [5] with 37,154 vertices and 231,584 edges.

For each network, we need to generate the weights on all edges with a random method, that is, the weight of each edge is firstly generated uniformly at random in the range $[0, \alpha]$. To meet $\Sigma_u w_{u,v} \leq \alpha$, the weights of all in-neighbor edges of a vertex v are divided by the number of v's in-neighbor edges. Here we set $\alpha = 0.5$ in the experiment. For each network, we repeated this process for 10 times, and the results (including the running time, the maximum revenue and the corresponding price) for evaluations are the mean value of the results obtained each time. When constructing acyclic graphs, the threshold θ is set to 0.01. In the experiments, we set the prices of a commodity as a set Λ of input parameters [0.50, 0.52, 0.54, ..., 0.70], and $\lambda \in \Lambda$.

Baselines. We compare our algorithms with three baselines described as follows.

Randomized Algorithm. It is the algorithm proposed in [23], and we compare CR with it for unlimited supply of commodities.

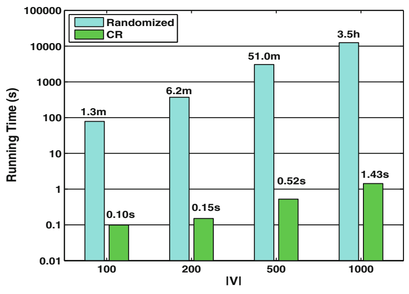
Greedy Algorithm. It is the variant of CG, in which the marginal and the total revenue are gained by simulations but bot by calculations. It is compared with CG and CHG for constrained supply of commodities.

PRUB+IF Algorithm. It is proposed in [25] for constrained supply of commodities, which is compared with CG and CHG. Please note that this algorithm

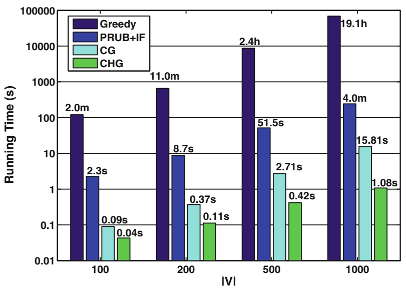
[1] https://networkx.github.io/.

only suites the cases that each user's inherent valuation is known by the seller, which is a strong assumption that does not hold in real-world scenarios. To address this issue, we generalize this algorithm by setting each user's inherent valuation as random variables. Specifically, in the process of obtaining the upper bound of the maximum revenue, each user's valuation is set as the average of 1,000 random values in the range [0, 1].

All the baselines and proposed algorithms are run in the price-sensitive dynamics process described in Sect. 2. For the baselines, 2000 simulations are executed to estimate the revenue for each candidate price and seed set. In each simulation, the valuation of each user is generated at random. This number of simulations is chosen to make the estimates accurate while considering the poor efficiency of the baselines. As the running time of the baselines are linear to the number of simulations, if our proposed algorithms are much faster than the baselines with 2000 simulations, then we can conclude that they would be more efficient than the baselines conducting a larger number of simulations. For the evaluations of effectiveness, in each algorithm of CR, CG, CHG and the baselines,



(a)



(b)

Fig. 1. The running time (in log scale) on synthetic networks for: (a) unlimited commodity supply; (b) constrained commodity supply where $Q/|V| = 60\%$.

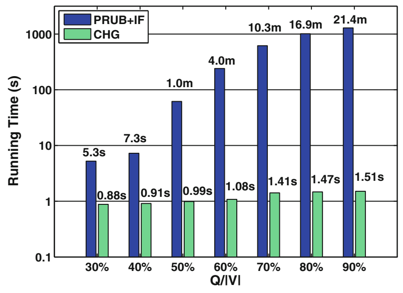


Fig. 2. The running time (in log scale) in a network (1,000 vertices, 3,000 edges) as the quantity of the commodity varies.

Table 1. The running time of CR and CHG ($Q/|V| = 60\%$) on a network (500 vertices) when the density varies.

$\lvert E\rvert$	1500	3000	4500	6000	7500
CR	0.52 s	0.62 s	0.65 s	1.03 s	1.26 s
CHG	0.42 s	0.55 s	0.88 s	1.30 s	1.49 s

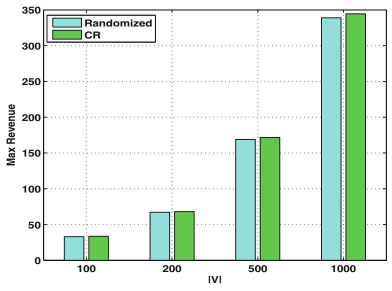
after selecting λ and S, we use the average revenue by 2000 simulations as the result to compare. To make fair comparisons, we use the same settings in the baselines as our work.

5.2 Results on Synthetic Networks

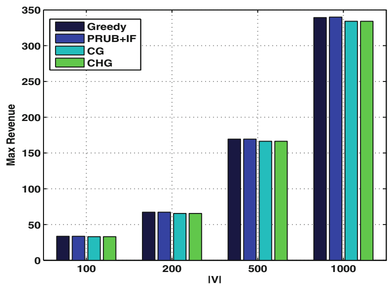
We generate different scales of synthetic power-law graphs with 100, 200, 500 and 1,000 vertices, each with 300, 600, 1,500 and 3,000 edges respectively to evaluate the algorithms. Please note that as the baseline algorithms are not scalable, we can only use small datasets to conduct the comparisons. Later we will evaluate our methods on large-scale real-world networks to show the scalability.

Figure 1(a) shows the running time in log scale for the Randomized algorithm and CR algorithm, which demonstrates that, when the quantity of the commodities is unlimited, the Randomized algorithm runs much slower than CR. For example, when $|V|$ is 1000, CR achieves a speedup of 3 orders of magnitude against the Randomized algorithm. Note that as the runtime of the Randomized algorithm is linear to the number of simulations, even if the number of simulations in the Randomized algorithm decreases from 2,000 to 20, CR can still achieve a speedup of 1 order of magnitude. It can also be observed that with the increase of the scale of a network, the runtime of CR grows slower than that of the Randomized algorithms, which shows that CR has better scalability. Figure 1(b) shows the running time in log scale for Greedy, PRUB+IF, CG and CHG algorithm, which demonstrates that CG and CHG runs much faster than the baselines for constrained commodity supply. When $|V|$ =1000, compared to Greedy and PRUB+IF, CHG achieves speedups of 4 orders of magnitude and 2 orders of magnitude respectively. CHG still runs faster than Greedy and PRUB+IF even if the number of simulations decreases from 2,000 to 200. It can also be observed that CHG runs faster than CG, with the reason that the scheme of max-Heap updating adopted in CHG can help to prune a significant portion of the calculations. Figure 2 shows the running time in log scale of PRUB+IF and CHG on a network with 1,000 vertices and 3,000 edges, as the quantity of the commodities varies. As the quantity becomes larger, the running time of PRUB+IF increases much more than that of CHG, which also demonstrates the efficiency of CHG. In Table 1 we do sensitivity analysis when the density of the graph varies. It can be observed that the running time of CR and CHG are almost linear to the number of edges when the number of the vertices is fixed.

Figure 3 shows the maximum revenues obtained by different algorithms. It can be observed that the results of CR can consistently match the results of the Randomized algorithm for the scenario of unlimited supply, and the results of CHG and other baselines also match each other very well for the scenario of constrained supply. Thus it can be summarized that, compared to the baselines, the maximum revenues obtained by CR and CHG are almost not impacted, with a much better efficiency.



(a)



(b)

Fig. 3. The maximum revenues on synthetic networks for: (a) unlimited commodity supply; (b) constrained commodity supply where $Q/|V| = 60\%$.

Table 2. The running time on real-world networks.

Datasets	RHF	EgoFB	NetHEPT	NetPHY
Unlimited commodity supply				
Randomized	1359.8 s	N/A	N/A	N/A
CR	0.7 s	31.7 s	161.3 s	2790.3 s
Constrained commodity supply				
Greedy	584.3 s	N/A	N/A	N/A
PRUB+IF	35.5 s	18123.9 s	N/A	N/A
CG	0.6 s	363.1 s	43633.9 s	N/A
CHG	0.5 s	24.3 s	363.3 s	13640.2 s

Table 3. The maximum revenues on real-world networks.

Datasets	RHF	EgoFB	NetHEPT	NetPHY
Unlimited commodity supply				
Randomized	71.8	N/A	N/A	N/A
CR	71.5	1392.2	4636.2	11557.5
Constrained commodity supply				
Greedy	70.6	N/A	N/A	N/A
PRUB+IF	70.6	1376.3	N/A	N/A
CG	70.2	1296.5	4631.6	N/A
CHG	70.2	1296.5	4631.6	11448.3

5.3 Results on Real-World Networks

In this section, we conduct experiments on real-world networks. Please note that in these experiments, if the running time of an algorithm exceeds 24 h, we just stop running this algorithm and discard its results of the running time and the maximum revenue, and mark them as "N/A" in the corresponding table.

Table 2 shows the runtime of the baselines and the proposed algorithms. For unlimited commodity supply, it can be observed that CR algorithm runs much faster than the Randomized algorithm. For the residence hall friendship dataset, CR achieves a speedup of 4 orders of magnitude against the Randomized algorithm. As the runtime of the Randomized algorithm is linear to the number of simulations, even if the number of simulations in the Randomized algorithm decreases from 2,000 to 20, CR can still achieve a speedup of 2 orders of magnitude. For the other three datasets which have larger size than the residence hall friendship dataset, the Randomized algorithm runs over 24 h, but CR is quite efficient. For the dataset NetPHY that has the largest size, CR runs less than an hour. For the scenario of constrained commodity supply, it is shown that CG and CHG runs much faster than the baselines. For the residence hall friendship dataset, compared to Greedy and PRUB+IF, CHG achieves speedups of

Table 4. The prices at achieving the maximum revenue on Ego-Facebook.

CR	0.52						
$Q/\|V\|$	30%	40%	50%	60%	70%	80%	90%
CHG	0.70	0.70	0.62	0.54	0.52	0.52	0.52

3 orders of magnitude and 2 orders of magnitude respectively. CHG still runs faster than Greedy and PRUB+IF even if the number of simulations decreases from 2,000 to 200. For datasets of NetHEPT and NetPHY, Greedy and PRUB+IF run over 24 h, while CHG runs quite fast. It can also be observed that CHG runs faster than CG, with the reason that the scheme of max-Heap updating adopted in CHG works well, which can help to prune a significant portion of the calculations. According to Table 2, it can be concluded that the baselines are constrained in efficiency, while the proposed algorithms show good scalability and would be feasible solutions for real-world scenarios.

Table 3 shows the maximum revenues. It can be observed that the results of CR match those of the Randomized algorithm, and the results of CHG and other baselines also match each other very well. Thus it can be summarized that, compared with the baselines, CR and CHG can successfully obtain almost the same maximum revenue with superiority in efficiency.

We also study the chosen price at the maximum revenue from CR and CHG. It is shown in Table 4 that in Ego-Facebook network, for unlimited commodity supply, the selected price is a lower value within the range, 0.52 in our case, which reveals a good strategy should focus on the number of activated vertices when the differences among the candidate prices are not large. For constrained commodity supply, when the quantity of the commodity is small, the selected price should be high. The possible reason is that both the higher and lower price can result in a significant number of activated vertices relative to the small quantity of the commodities, and thus the higher price can bring in more revenue. We can also see that the chosen price decreases as the amount of the commodity grows larger. The possible reason is that as the quantity of the commodity grows, the higher price can no longer make the number of the activated vertices approach to the quantity of the commodity. As a result, we have to lower the price to increase the number of the activated vertices. It is consistent with the marketing strategies in real-world applications.

6 Conclusion

In this paper, two efficient revenue maximization algorithms CR and CHG are proposed for both the scenarios of unlimited and constrained commodity supply respectively. In these algorithms, local acyclic graphs are constructed which help to gain the revenue through fast calculation. Furthermore, a max-Heap updating scheme is proposed to prune unnecessary calculations in CHG. Experiments on both the synthetic networks and real-world networks demonstrate the efficiency and effectiveness of the proposed algorithms.

Acknowledgments. This work is supported in part by the State Key Development Program of Basic Research of China (No. 2013CB329605), the National Key Research and Development Program of China (No. 2016QY03D0605), the Natural Science Foundation of China (No. 61300014, 61372191, 61672313), NSF through grants IIS-1526499, and CNS-1626432.

References

1. Akhlaghpour, H., Ghodsi, M., Haghpanah, N., Mirrokni, V.S., Mahini, H., Nikzad, A.: Optimal iterative pricing over social networks (Extended abstract). In: Saberi, A. (ed.) WINE 2010. LNCS, vol. 6484, pp. 415–423. Springer, Heidelberg (2010). doi:10.1007/978-3-642-17572-5_34
2. Bensaid, B., Lesne, J.P.: Dynamic monopoly pricing with network externalities. Int. J. Ind. Organ. **14**(6), 837–855 (1996)
3. Buchbinder, N., Feldman, M., Naor, J., Schwartz, R.: A tight linear time (1/2)-approximation for unconstrained submodular maximization. SIAM J. Comput. **44**(5), 1384–1402 (2015)
4. Cabral, L.M.B., Salant, D.J., Woroch, G.A.: Monopoly pricing with network externalities. Int. J. Ind. Organ. **17**(2), 199–214 (1999)
5. Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: KDD, pp. 199–208 (2009)
6. Chen, W., Yuan, Y., Zhang, L.: Scalable influence maximization in social networks under the linear threshold model. In: ICDM, pp. 88–97 (2010)
7. Cohen, E., Delling, D., Pajor, T., Werneck, R.F.: Sketch-based influence maximization and computation: Scaling up with guarantees. In: CIKM, pp. 629–638 (2014)
8. Domingos, P., Richardson, M.: Mining the network value of customers. In: KDD, pp. 57–66 (2001)
9. Feige, U., Mirrokni, V.S., Vondrak, J.: Maximizing non-monotone submodular functions. SIAM J. Comput. **40**(4), 1133–1153 (2011)
10. Fotakis, D., Siminelakis, P.: On the efficiency of influence-and-exploit strategies for revenue maximization under positive externalities. Theor. Comput. Sci. **539**, 68–86 (2014)
11. Freeman, L., Webster, C., Kirke, D.: Exploring social structure using dynamic three-dimensional color images. Soc. Netw. **20**(2), 109–118 (1998)
12. Fromlet, H.: Predictability of financial crises: lessons from Sweden for other countries. Bus. Econ. **47**(4), 262–272 (2012)
13. Goyal, A., Lu, W., Lakshmanan, L.V.S.: SIMPATH: an efficient algorithm for influence maximization under the linear threshold model. In: ICDM, pp. 211–220 (2011)
14. Hartline, J., Mirrokni, V., Sundararajan, M.: Optimal marketing strategies over social networks. In: WWW, pp. 189–198 (2008)
15. Kahn, A.B.: Topological sorting of large networks. Commun. ACM **5**(11), 558–562 (1962)
16. Kalish, S.: A new product adoption model with price, advertising, and uncertainty. Manag. Sci. **31**(12), 1569–1585 (1985)

17. Katz, M., Shapiro, C.: Network externalities, competition, and compatibility. Am. Econ. Rev. **75**(3), 424–440 (1985)
18. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: KDD, pp. 137–146 (2003)
19. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N.: Cost-effective outbreak detection in networks. In: KDD, pp. 420–429 (2007)
20. Lu, W., Lakshmanan, L.V.S.: Profit maximization over social networks. In: ICDM, pp. 479–488 (2012)
21. Mason, R.: Network externalities and the coase conjecture. Eur. Econ. Rev. **44**(10), 1981–1992 (2000)
22. McAuley, J., Leskovec, J.: Learning to discover social circles in ego networks. In: NIPS, pp. 539–547 (2012)
23. Mirrokni, V.S., Roch, S., Sundararajan, M.: On fixed-price marketing for goods with positive network externalities. In: Goldberg, P.W. (ed.) WINE 2012. LNCS, vol. 7695, pp. 532–538. Springer, Heidelberg (2012). doi:10.1007/978-3-642-35311-6_43
24. Sundararajan, A.: Local network effects and complex network structure. BE J. Theor. Econ. **7**(1) (2007)
25. Teng, Y., Tai, C., Yu, P.S., Chen, M.: An effective marketing strategy for revenue maximization with a quantity constraint. In: KDD, pp. 1175–1184 (2015)