Implicit Learning for Explicit Discount Targeting in Online Social Networks

Srinivas Shakkottai, Lei Ying and Sankalp Sah

Abstract— Online Social networks are increasingly being seen as a means of obtaining awareness of user preferences. Such awareness could be used to target goods and services at them. We consider a general user model, wherein users could buy different numbers of goods at a marked and a discounted price. Our first objective is to learn which users would be interested in a particular good. Second, we would like to know how much to discount these users such that the entire demand is realized, but not so much that profits are decreased. We develop algorithms for multihop forwarding of discount coupons over an online social network, in which users forward such coupons to each other in return for a reward. Coupling this idea with the implicit learning associated with backpressure routing (originally developed for multihop wireless networks), we show how to realize optimal revenue. Using simulations, we illustrate its superior performance as compared to random coupon forwarding on different social network topologies. We then propose a simpler heuristic algorithm and using simulations, and show that its performance approaches that of backpressure routing.

I. INTRODUCTION

The past few years have seen the rapid and global emergence of online social networks as a medium for community interaction [1], [2]. Their success can be gauged by their meteoric adoption by a large populace, and the continued success of the medium requires a sound economic foundation for sustainable growth. The medium of choice to extract commercial value out of online social networks is the advertising and sale of goods and services by using the structure and nature of social interactions. Since an individual user’s preferences can be identified by his or her response to exogenous stimulation such as advertisements, an approach that is often used is to try to learn about user preferences through exploration. Such feedback obtained though the user responses could be used to offer incentives to purchase certain goods and services.

Consider goods and services that are consumed periodically (say on a weekly or monthly basis) such as movie tickets, car washes, fitness club visits and so on. Here, we could have a high displayed price that some consumers would be willing to pay. In order to extract maximum revenue, other consumers need to be subsidized to some extent by using discounts such as rebate coupons. In other words, discount coupons are used to create multiple tiers of prices for the same good or service. Two questions immediately arise, (i) which users should be given coupons?, and (ii) how many coupons should they be given? Further, the questions have to be answered in a system in which user preferences change over time.

Both questions are hard since the seller of the good is unlikely to be aware of the preferences of users, or possibly even of their existence. Even if the seller is aware of a user’s interest, he must not give too many or too few discounts – too many would reduce profits and too few would mean that the entire demand would not be realized. There are two classical methods of offering such incentives. The first is to flood communities of users in the hope that some of them would use the coupons. Here, the idea is to pre-identify communities that are not likely to buy the good without discounts. If identification is incorrect, either the users would not use the coupons, or the wrong set of users would be discounted. The second is to rely on self-identification of interested individuals. Here, the store gets the users to sign up for coupons, and then judiciously sends them coupons. Such a scheme would work only on users who do identify themselves to the store, and might not realize the entire demand.

Both the above solutions ignore the fact that users could belong to an online social network, and hence could obtain coupons by interacting with his or her friends. Thus, users could forward coupons from one to the next in a multihop fashion across the online social network. If a user is interested in the good that the coupon represents, he or she could use it. Otherwise the user could forward it onwards. Allowing for coupon forwarding implies that the two questions raised have to be modified slightly: (i) given that a user has a coupon that he does not want, which friend should he forward it to and why?, and (ii) what rate should coupons be injected into the system? Hence, we need to design a signaling scheme that incentivizes users to somehow learn the preferences of users in such a way that the profits of the store are maximized. An example of such a system in practice is mGinger [3] that acts as a multi-hop advertising and discount distribution system using SMS messages, with rewards being paid in a pyramidal fashion. The motivation for multi-hop coupon distribution is that since user preferences change with time, and new products are continuously introduced, it is impossible for any store to be aware of all its potential customers. Hence a system must learn user preferences, which then change after a while.

In this paper, we develop implicit distributed learning...
schemes based on ideas of backpressure [4] that has been used as a throughput optimal scheme for packet routing in multihop wireless networks [5], [6], [7], [8]. We assume that the capacity for consumption of a good $i$ by user $j$ can be divided naturally into two parts—one at at the marked (“high”) price $\bar{x}^{ih}_{ij}$, and one at the discounted (“low”) price $\bar{x}^{il}_{ij}$. We assume that these values are fixed for some duration of time, and so can be learned. The number of coupons given to the user must be carefully regulated; if it is larger than $\bar{x}^{il}_{ij}$, the store loses profits due to excessive discounting, while if it is less than $\bar{x}^{il}_{ij}$, the entire demand is not realized. We combine ideas of self-identification by users and directed flooding through backpressure to achieve an optimal solution that realizes the entire demand by injecting the right number of coupons, and maximizes profit by ensuring that the users receive coupons at the optimal rate.

We then use optimization decomposition techniques in Section V to develop a coupon distribution scheme consisting of three entities: (i) a store at which goods may be purchased, (ii) users connected by an online social network and (iii) a coupon source (or sources). The behavior of these entities is as follows:

- A store sells goods $i$ at a marked price $p^i$, which it discounts to a price $q^i$ upon presentation of a coupon. The store assigns a “goodness value” to each user $j$ that makes a purchase from it. This value determines the probability with which neighbors of $j$ are rewarded for forwarding a coupon to $j$. However, all the other users (non-neighbors of $j$) that are involved in the forwarding path are guaranteed a reward. This artifice enables locality of information, as we show later. In other words, although the discount carried by each coupon is identical, the reward for forwarding coupons to each user $j$ is not. The store uses a simple up-down controller to determine the reward probability for forwarding, based on the number of goods purchased by user $j$.
- All users in the system maintain a count of the number of coupons of each kind that their neighbors possess via communication over the social network. Users also maintain a count of the number of unrewarded coupons associated with their neighbors by polling the relevant store. We refer to the sum of these two as the effective coupons. Coupons can be transferred among users in a multihop fashion, and users are incentivized to forward coupons in the direction of lowest effective pressure, i.e., to a neighbor who has the smallest number of effective coupons. This controller is similar in nature to a backpressure controller.
- Finally, the coupon source generates coupons of different kinds (corresponding to different goods), and sends them to users that have identified themselves as interested in receiving particular kinds of coupons.

The source chooses to send coupons using a threshold controller, which generates new coupons when the effective pressure is less than a certain threshold.

We prove that the system using this backpressure-based coupon distribution evolves with time to attain the maximum profits by ensuring that each potential consumer obtains exactly the right number of coupons. The system is distributed and each user only requires information associated with his or her neighbors. Thus, it succeeds in achieving light-weight learning framework, in which exploring for user capacity and exploiting existing capacity go hand-in-hand.

We then consider a simpler heuristic algorithm in Section VI, that is based on the delay in obtaining rewards. This delay-based algorithm does not require information exchange between users. At any time, users simply forward coupons to that neighbor for whom the delay experienced between forwarding a coupon and obtaining a reward for that coupon is the smallest. This algorithm inherently captures the idea of backpressure, although it is at a coarse level.

Our final scheme is even simpler, and consists of random coupon distribution. Here, each user randomly forwards coupons to its neighbors in the hope of finding correct paths. This system does not learn user preferences. We use this algorithm to test the efficacy of our other algorithms.

We simulate the distribution schemes in Section VII on different topologies to compare their performance. We show that the backpressure-based scheme achieves near-optimal revenue, while the delay-based scheme performs acceptably well. Further, both schemes significantly outperform the randomized scheme, thus making a strong case for backpressure based targeted coupon delivery in online social networks.

II. RELATED WORK

A typical learning problem is that of the bandit-problem [9], [10], [11], which has its historical origin in a coin-operated gambling machine that pays off according to the matching of symbols on wheels spun by a handle, also called a one-armed bandit. The multi-armed bandit is the situation confronted by a gambler who has a choice between $n$ one-armed bandits, and attempts to learn by experience which one to pull. Our approach to learning is quite different, and follows a more implicit method derived from communication networks.

The backpressure algorithm is a joint routing/scheduling for communication networks where the routing/scheduling decisions are dynamically made without requiring the information of the network topology and traffic arrivals [4]. The algorithm has been proved to stabilize any traffic that can be stabilized by any other routing/scheduling algorithm for different types of communication networks [12], [6], [5], [7], [8], [13]. The backpressure algorithm learns traffic and network information implicitly from queue-lengths. Based on the similar idea, we develop a coupon delivery scheme, where the users learn the coupon demands from coupon queues and dynamically distribute coupons to their neighbors based on the effective pressure. However, coupon delivery problem

Throughout this paper, we use the word reward to denote remuneration for forwarding coupons, and the word discount to denote remuneration for redeeming coupons (when purchasing goods) at a store.
III. SYSTEM MODEL

Network model: We consider an online social network structure as shown in Figure 1. Denote by \( \mathcal{N} \) the set of nodes and \( \mathcal{L} \) the set of links. There are three different nodes — coupon distributor, users, and store — in the network. The links represent social connections. A link from a coupon source to a user represents the idea that the user has registered with the source to receive its coupon periodically. A link from a user to a product means that the user buys that product periodically. The links between users are assumed to be bidirectional, and represent friendship between the connected users. In this paper, we assume that the coupon sources and the store are managed by the same entity. We use \( s \) to denote the store and \( d \) to denote the coupon distributor. We define \( \mathcal{S} \) to be the set of products and \( \mathcal{B}_i \) is the set of users who will buy product \( i \).

We consider a synchronized slotted-time system. We define \( \mu_j \) to be the coupon transmission capacity of node \( j \), which is the maximum number of coupons user \( j \) can send out in one time-step (this could be a constraint such as the number of SMS messages that a user is willing to send per unit time). We also impose the constraint that a user can buy a discounted product only if a coupon is presented.

Two-capacity model for user demands: We assume that users naturally have a maximum number of goods that they would buy at the marked price, \( \hat{x}^i_j \), and number of additional goods that users would buy at the discounted price, \( \hat{x}^d_j \). Note that either of these quantities could be zero. We further define \( b_j^i = \hat{x}^i_j + \hat{x}^d_j \). These values can be thought of as the capacities associated with a user. We consider two different time scales in this paper. The small time scale \( t \) is the one in which purchases are made and coupons are delivered. The large time-scale, consisting of \( T \) small time slots, is the buying interval after which the customers start afresh. Since the users have the incentive to buy a product with a low price, we assume that the \( \hat{x}^i_j \), associated with the high price goods could be used to buy low priced goods as well. Specifically, during each large time scale, if the user were given no more than \( b_j^i T \) coupons, he would use them all and buy \( b_j^i T \) goods.

Note that if a user were given more than \( \hat{x}^i_j T \) coupons, the store would not extract the maximum extractable revenue. If he were given less than \( \hat{x}^i_j T \) coupons, he would not buy enough discounted goods, which reduces the profit of the store as well. A store that is unaware of these two capacities needs to probe customers in order to find their true potential, and neither supply too few or too many coupons. In what follows, we present a distributed solution that automatically explores for and attains the capacity of users, thus achieving the profit maximizing solution.

IV. PROFIT MAXIMIZATION

Consider the profit made by the store. We say a coupon is valid if it is eventually used to purchase a product. We denote by \( y_{(m,n)}^i \) the average number of valid type-\( i \) coupons sent from user \( m \) to user \( n \) in a time slot. The profit the store extracts from user \( j \) is

\[
q^i_j y_{(j,s)}^i + p^i_j \min \left\{ \hat{x}^i_j, b_j^i - y_{(j,s)}^i \right\}.
\]

Thus, the maximum profit the store can extract is defined by the following optimization problem:

\[
\text{OPT 1:} \quad \max \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{B}_i} \left( q^i_j y_{(j,s)}^i + p^i_j \min \left\{ \hat{x}^i_j, b_j^i - y_{(j,s)}^i \right\} \right) \tag{1}
\]

subject to:

\[
\sum_{i \in \mathcal{S}} \sum_{j : (m,j) \in \mathcal{L}, j \neq s} y_{(m,j)}^i \leq \mu_m, \forall m \in \mathcal{N} \tag{2}
\]

\[
\sum_{j : (m,j) \in \mathcal{L}} y_{(m,j)}^i = \sum_{n : (n,m) \in \mathcal{L}} y_{(n,m)}^i, \forall m \in \mathcal{N} \tag{3}
\]

\[
y_{(m,j)}^i = 0 \text{ if } m \in \mathcal{B}_i \text{ and } j \neq s \tag{4}
\]

where (2) is the capacity constraint, which indicates node \( m \) cannot send more than \( \mu_m \) coupons in a time-slot, (3) is the flow-conservation constraint for the coupons, and (4) indicates that user \( j \) will not forward type-\( i \) coupons to his/her neighbors if he/she uses type-\( i \) coupons.

To extract the maximum revenue, we need to distribute coupons to the users. There are two difficulties in distributing the right number of coupons to the users:

1. The optimal \( (\hat{x}^i_j, \hat{x}^d_j) \) is unknown at the store, and needs to be identified.
2. Since all users interested in a product may not be registered to directly receive coupons, they might need to receive such coupons via the social network. The store cannot directly control the number of coupons sent to such users.

To tackle these two difficulties, we develop a two time-scale coupon distribution scheme in the next section.
V. COUPON DISTRIBUTION

In this section, we develop an implicit distributed learning scheme based on the idea of backpressure routing/scheduling [4]. Our algorithm consists of two control loops that operate at the small time scale and the large time scale. The purpose of the control loops is as follows:

1) **Choice of Coupon Forwarding Reward Rate:** At the large time scale, each store must adapt the target rate $\gamma_j^i$ for the next buying interval using the information gathered about the customers' preferences over the past intervals. In our algorithm, $\gamma_j^i$ is set to $x^d_j$. As discussed in Section III, if $\gamma_j^i$ is set too low, customer $j$ may not purchase all the goods that he potentially could, and if $\gamma_j^i$ is too high, customer $j$ may be being discounted excessively and the store is not extracting the maximum extractable revenue.

2) **Coupon Routing at Target Rate:** At the small time scale a store arbitrarily assigns to a rate of coupon delivery $\gamma_j^i$ to each product $i$ and each customer $j$ that purchases goods from it. The purpose of this control loop is to ensure that the customer would indeed receive discount coupons at this target rate. Mathematically, we will guarantee that the coupon distribution algorithm solves the following optimization problem:

**OPT 2:**

$$\max \sum_{i \in S} q^i \left( \sum_{j \in B_i} y^i_{(j,s)} \right)$$

s.t.

$$\sum_{j \in S \min \sum_{j \in L, j \neq s} y^i_{(m,j)} \leq \mu_m, \forall m \in N}$$

$$\sum_{j : (m,j) \in L} y^i_{(m,j)} = \sum_{n : (n,m) \in L} y^i_{(n,m)} \forall m \in N}$$

$$y^i_{(j,s)} \leq \gamma_j^i \forall j, i$$

$$y^i_{(m,j)} = 0 \text{ if } m \in B_i \text{ and } j \neq s$$

To show the correctness of the proposed algorithm, we first need the following straightforward lemma.

**Lemma 1:** Given that $\gamma_j^i = \tilde{x}^d_j$ for all $i$ and $j$, OPT 1 is equivalent to OPT 2.

**Proof:** The proof is skipped due to space constraints and may be found in [14].

Next, we develop a distributed coupon routing algorithm that solves OPT 2.

A. Small time scale control: backpressure coupon routing

We first introduce the coupon management scheme which consists of three parts: (i) each user maintains a per-product queue, and monitors the lengths of the queues; (ii) store rewards the neighbors that forwarded type $i$ coupons used by each customer $j$ at a rate $\gamma_j^i$, and monitors the number of unrewarded coupons at each customer; (iii) coupon distributor $i$ monitors the number of coupons she has not yet sent out, and generates additional coupons based on this value.

**A1: Coupon Management:**

1) Per-product queues are maintained at each user, and the number of type $i$ coupons user $j$ has at a finer time-step $t$ is denoted by $Q^i_j[t]$. Thus, the dynamics of $Q^i_j[t]$ is as follows: If $j \not\in B_i$, then

$$Q^i_j[t+1] = \left(Q^i_j[t] + \sum_{m : (m,j) \in L} y^i_{(m,j)}[t] - \sum_{n : (n,j) \in L} y^i_{(n,j)}[t] \right)^{+}$$

otherwise

$$Q^i_j[t+1] = Q^i_j[t] + \sum_{m : (m,j) \in L} y^i_{(m,j)}[t] - y^i_{(j,s)}[t],$$

where

$$y^i_{(j,s)}[t] = \min \left\{ Q^i_j[t] + \sum_{m : (m,j) \in L} y^i_{(m,j)}[t], \left( b_j^i T - \sum_{\tau=0}^{t-1} y^i_{(j,s)}[\tau] \right)^{+} \right\},$$

i.e., user $j$ will use up all available coupons unless she has already bought enough ($b_j^i T$) products.

2) Store maintains a queue for unrewarded coupons corresponding to each product $i$ and its customers $j$. We may think of these as virtual coupons that are used to maintain a pressure on $j$’s neighbors. Note that it is only the neighbors of $j$ that are not rewarded for forwarding these coupons, the rest of the users involved in forwarding coupons would be guaranteed a reward (and, of course, $j$ has already redeemed these coupons for a discount). This measure ensures that pressure against forwarding coupons to a particular customer is maintained adjacent to the customer, and not at arbitrary queues in the network. Denote by $\hat{Q}^i_j[t]$ the number of such unrewarded coupons corresponding to customer $j$. This is essentially a “virtual queue” that will be used to enforce constraint (8).

$$\hat{Q}^i_j[t+1] = \left( \hat{Q}^i_j[t] + y^i_{(j,s)}[t] - \gamma_j^i \right)^{+},$$

where $\gamma_j^i$ is the coupon forwarding reward rate for neighbors of customer $j$.

3) Coupon distributor $d$ maintains a separate queue for each type of coupons that have not been sent out. The length of the queue is denoted by $\hat{Q}^i_d[t]$ for each product $i$. We have

$$\hat{Q}^i_d[t+1] = \left( \hat{Q}^i_d[t] + \Theta^i[t] - \sum_{j : (d,j) \in L} y^i_{(d,j)}[t] \right)^{+},$$

where $\Theta^i[t]$ is the number of new type $i$ coupons generated by coupon distributor $i$ at time $t$. 

\[2\text{Recall that these are coupons that have been redeemed for a discount by } j \text{, but the neighbors of } j \text{ who forwarded these coupons have not been rewarded.}\]
(4) We also assume that when user $j$ receives a type $i$ coupon such that $j \notin B_i$, user $j$ will insert her identity and the coupon queue length $Q^i_j[t]$ in the coupon before sending the coupon to her neighbor. This information allows the store to reconstruct path and reward the coupon relays based on $Q^i_j[t]$.

In our system the store need to reward coupon forwarding in order to motivate users to forward coupons to their friends. The efficiency of a coupon distribution scheme is determined by: (i) the incentive scheme that the store use, and (ii) the users’ decisions under the incentive scheme. Next, we propose a coupon rewarding scheme, under which a rational user will distribute the coupons according to a backpressure policy. The optimality of the coupon distribution scheme will be proved in Theorem 2.

A2: Reward Scheme for Coupon Forwarding

Store rewards the users involved in forwarding each used type $i$ coupon with a total of $\alpha^i$ dollars. We assume that $\alpha^i$ is fixed and is such that the store still makes a profit, i.e., we do not optimize over $\alpha^i$. Consider a specific coupon associated with product $i$, and assume $\mathcal{R}$ is the path (consisting of the sequence of transmissions used to distribute the coupon) over which the coupon was transferred. Then node $m$ gets a reward

$$\left(Q^i_m - Q^i_{m:(m,n) \in \mathcal{R}}\right) + \frac{\alpha^i}{\sum_{l \in \mathcal{L}} \left(Q^i_{s(l)} - Q^i_{r(l)}\right)},$$

where $l$ is a link on path $\mathcal{R}$, $s(l)$ is the sender, and $r(l)$ is the receiver. Note that this queue length information is inserted by the users before they forward the coupons to their neighbors. Furthermore, note that the amount of reward user $m$ obtains is proportional to the queue difference. The idea is to motivate user $m$ to send her coupon to a neighbor who has the least number of coupons and hence is most likely the one who needs the coupon. Under this scheme, the user has the motivation to follow the backpressure-like coupon distribution scheme.

A3: User Behavior:

(1) First, if node $j$ is interested in buying product $i$, then user $j$ uses all available type $i$ coupons up to her purchasing limit $b^i_j$. Thus, at finer time-step $t$, user $j$ purchases $y^i_{(j,s)}[t]$ products with coupons such that

$$y^i_{(j,s)}[t] = \min \left\{ Q^i_j[t] + \sum_{m:(m,j) \in \mathcal{L}} y^i_{(m,j)}[t], \left( b^i_j T - \sum_{\tau=0}^t y^i_{(j,s)}[\tau] \right) \right\},$$

We assume that user $j$ never forwards type-$i$ coupons to her neighbors if user $j$ buys product $i$.

(2) If user $j$ is not a customer buying product $i$, then user $j$ needs to distribute type $i$ coupons to her neighbors. We assume that at the beginning of finer time-step $t$, user $j$ requests $Q^i_m[t]$ if user $m$ is her neighbor, and also polls the store to obtain $\tilde{Q}^i_m[t]$. Since the amount of coupon forwarding reward is determined by the queue difference as described in (10), user $j$ selects a coupon type $i^*$ and neighbor $m^*$ such that

$$(i^*, m^*) \in \arg \max_{(j,m) \in \mathcal{L}} \left( Q^i_j[t] + Q^i_{m^*}[t] - Q^i_m[t] - \tilde{Q}^i_m[t] \right),$$

and transfers $\min \{\mu_j, Q^i_{m^*}[t]\}$ of type $i^*$ coupons to node $m^*$.

Note that $\tilde{Q}^i_m[t]$ is the number of coupons used by user $m$ but not been rewarded yet, so $Q^i_m[t] = 0$ if user $m$ is not a customer buying product $i$. A store maintains this unrewarded coupon queue to prevent a customer receiving too many coupons. When user $j$ uses too many coupons, the unrewarded coupon queue becomes large. After a neighbor of user $j$ finds a large $Q^i_j[t]$, the neighbor realizes that user $j$ has received too many coupons and the store might not reward him for forwarding coupons to user $j$. Then the neighbor will stop forwarding more coupons to user $j$.

A4: Coupon Generation Scheme: The coupon distributor needs to decide the number of coupons to inject into the network. We assume that coupon distributor generates $\mu_d$ type-$i$ coupons when $Q^i_d[t] \leq Q_T q^i$, and zero type-$i$ coupon otherwise. Here, $Q_T$ is a constant threshold value. In other words, $\Theta^i[t] = \mu_d$ if $Q^i_d[t] \leq Q_T q^i$, and $\Theta^i[t] = 0$ otherwise.

In the following theorem, we analyze the performance of the backpressure coupon routing, and prove that

**Theorem 2:** Assume that $\gamma^i_j \leq b^i_j$ for all $i$ and $j$. Under the coupon management, coupon rewarding and generating scheme, and user behavior defined above, we have

$$\lim_{Q_T \to \infty} \lim_{T \to \infty} \frac{\sum_{t=1}^{T-1} \Theta^i[t]}{T} = \left( \sum_{j \in B_i} \tilde{y}^i_{(j,s)} \right),$$

and

$$\lim_{Q_T \to \infty} \lim_{T \to \infty} \frac{\sum_{t=1}^{T-1} y^i_{(j,s)}[t]}{T} = \tilde{y}^i_{(j,s)},$$

where $\tilde{y}$ is the optimal solution of OPT 2.

**Proof:** We skip the proof details due to space constraints. Interested readers can find the details in [14].

The algorithm is similar to that proposed in [5]. Note that although the theorem is an asymptotic result, the algorithm itself works for any value of $T$. A finite value of $T$ may result in a sub-optimal solution. In our simulations, we choose $T = 300$ and the final coupon allocation is very close to the optimal.

B. Large time scale control: Coupon rate selection

We assume that the algorithm for coupon delivery at the small time scale converges quickly to the target rate, and now consider how to choose this target rate. Recall that our means of implementing coupon delivery at rate $\gamma^i_j$ is to reward the
neighbors of a customer \( j \) for forwarding coupons to \( j \) at rate \( \gamma_j \). In this section all dynamics take place at the large time. Thus, we have the sequence of target rates \( \gamma_j^0[0], \ldots, \gamma_j^k[k-1], \gamma_j^k[k+1], \ldots \), and the large time scale algorithm needs to guarantee that \( \lim_{k \to \infty} \gamma_j^k[k] = \hat{x}_j^m \).

Denote by \( z_{ij}^h[k] \) and \( z_{ij}^z[k] \) the number of product \( i \) that user \( j \) buys from store in the interval \( [k, k+1] \) at the marked price and the discounted price, respectively. Let the total number of goods purchased in the interval \( [k, k+1] \) be denoted \( z_{ij}^h[k] = z_{ij}^h[k] + z_{ij}^z[k] \). Further, denote the difference in purchases made by user \( j \) over intervals \( [k, k+1] \) and \( [k-1, k] \) by \( \Delta z_{ij}^h[k] = z_{ij}^h[k] - z_{ij}^h[k-1] \) corresponding to a difference in the coupon delivery rate \( \Delta \gamma_j^h[k] = \gamma_j^h[k] - \gamma_j^h[k-1] \). We first intuitively understand the four possibilities associated with \( \Delta \gamma_j^h[k], \Delta z_{ij}^h[k] \) (assuming that \( \Delta \gamma_j^h[k] \) is small):

- \( \Delta \gamma_j^h[k] < 0 \) and \( \Delta z_{ij}^h[k] = 0 \) : This implies that \( \gamma_j^h[k] \geq \hat{x}_j^m \) and the user is receiving more coupons than he can use. We need to ensure \( \gamma_j^h[k + 1] < \gamma_j^h[k] \).
- \( \Delta \gamma_j^h[k] < 0 \) and \( \Delta z_{ij}^h[k] < 0 \) : This implies that \( \gamma_j^h[k] < \hat{x}_j^m \) and the user is not receiving enough coupons to realize the maximum possible number of purchases. We need to ensure \( \gamma_j^h[k + 1] > \gamma_j^h[k] \).
- \( \Delta \gamma_j^h[k] > 0 \) and \( \Delta z_{ij}^h[k] = 0 \) : This implies that \( \gamma_j^h[k] \geq \hat{x}_j^m \) and the user is receiving more coupons than he can use. We need to ensure \( \gamma_j^h[k + 1] < \gamma_j^h[k] \).
- \( \Delta \gamma_j^h[k] > 0 \) and \( \Delta z_{ij}^h[k] > 0 \) : This implies that \( \gamma_j^h[k] < \hat{x}_j^m \) and the user is not receiving enough coupons to realize the maximum possible number of purchases. We need to ensure \( \gamma_j^h[k + 1] > \gamma_j^h[k] \).

Note that an increase in the coupon rate cannot cause a decrease in the number of purchases. A simple controller that takes into account all the four possible conditions is

\[
\gamma_j^h[k + 1] = (\gamma_j^h[k] + \delta) \chi_{\{\Delta \gamma_j^h[k] \Delta z_{ij}^h[k] > 0\}} + (\gamma_j^h[k] - \delta) \chi_{\{\Delta \gamma_j^h[k] \Delta z_{ij}^h[k] = 0\}}.
\]

Here, \( \delta > 0 \) is a constant small amount by which we increase or decrease \( \gamma_j^h \). We can now easily prove that the controller converges to within \( \delta / 2 \) of the desired value of \( \hat{x}_j^m \).

**Theorem 3**: Under the time scale separation assumption, using the controller (13) we have

\[
\lim_{k \to \infty} |\gamma_j^h[k] - \hat{x}_j^m| \leq \delta / 2 \quad \forall i \in S, \ j \in R.
\]

**Proof**: We skip the proof details due to space constraints. Interested readers can find the details in [14].

Note that when \( \delta \) is smaller and the algorithm starts with a small \( \gamma_j^0 \), we can guarantee that \( \gamma_j^k[k] \leq b_j^m \) for all \( k \). Combining Lemma 1, Theorem 2 and Theorem 3, we conclude that the number of coupons consumed under the two time-scale algorithm converges to the optimal solution to OPT 1.

VI. DELAY-BASED COUPON FORWARDING

Suppose that the store rewards relays only after a coupon has been used to make a purchase. The insight that we obtain from the optimality of backpressure is the following:

- If coupons are not used on a particular path, queues build up. This would cause the average delay in being rewarded to all relays on the path to increase.
- If a higher rate of coupons than that set by the store are transferred along a path, the store does not reward the relays for some fraction of coupons and virtual coupons build up. Again, this would mean that the average delay in being rewarded to all relays on the path would increase.

The observation immediately suggests that perhaps a simpler algorithm would be to replace the backpressure-based user control of Section V A3 with a much simpler scheme. Users need only keep track of the average delay experienced in obtaining rewards when they forward coupons to each of their neighbors. They choose to forward coupons to that neighbor who has the lowest such delay. The scheme is intuitively incentive compatible, since users might want to obtain rewards as soon as possible. Thus, we may replace the reward scheme of Section V A2 with an equal reward for all users in the path.

However, a few further additions are required to construct a workable heuristic algorithm. The first addition stems from the fact that under backpressure, if a user finds that all her neighbors have larger effective queue lengths than herself, she does not forward coupons to any of them. The equivalent in the delay based regime would be to simply choose a threshold value of delay (e.g., \( D_T \)), and refuse to forward coupons to any neighbor that yields a delay larger than this threshold.

The second addition is that while keeping track of delays, even small differences in delays could result in a particular user being ignored entirely. Hence, instead of a hard comparison between the delays of different options, we soften the comparison. For example, if neighbors 1 and 2 of a node yield delays \( d_1 \) and \( d_2 \), we consider both as equally lucrative if \( |d_1 - d_2| \leq D_T \), where \( D_T \) is a constant delay threshold. Our expectation is that this simplified algorithm would perform almost as well as the backpressure-scheme.

Based on the observations above, we propose the following delay-based coupon forwarding to replace the user control of Section V A3 for all coupons in which user \( i \) is not interested.

**Delay-based coupon forwarding**: Consider product \( j \) that user \( i \) is not interested. Denote by \( D^m_i(t) \) the average delay experienced in obtaining rewards when user \( j \) forwards type \( i \) coupons to neighbor \( m \). User \( i \) keeps track of \( D^m_i(t) \) for all neighbors. At time step \( t \), user \( j \) first selects type \( i^* \) coupon such that

\[
i^* \in \arg \min_i \min_{m:(j,m) \in L} D^m_i(t)
\]

and a subset of neighbors associated with type \( i^* \) coupon

\[
\mathcal{K}^*_j = \left\{ m : |D^m_i(t) - \min_{(j,m) \in L} D^m_i(t)| \leq D_T \right\}.
\]

Then user \( j \) sends

\[
\min_{\mathcal{K}^*_j} \{ Q^*_{j,m}(t), \mu_j(t) \}
\]
number of type \( i^* \) coupons to each of the neighbors in \( K^* \).

Remark: Backpressure based user control requires a user to obtain the lengths of coupon queues from her/his neighbors and from the store. Delay-based coupon forwarding, on the other hand, does not require any information exchange among the users. Each user makes decisions based on her/his own information history, which results in a much smaller communication overhead as compared to backpressure based user control. Further, unlike backpressure, the reward given to every user in the path of a coupon can be identical.

VII. SIMULATION RESULTS

We simulate our coupon distribution system on different network topologies to study the validity of our schemes. For the sake of comparison, we also create a third coupon distribution system in which coupons are forwarded by relays randomly to their neighbors. This would indeed be the case if multihop coupon distribution were allowed without incentives for forwarding in any particular direction. Intuitively, such a distribution scheme should over-distribute coupons, if multihop coupon distribution were allowed without incentives, to every user in the path of a coupon can be identical.

A. Simple Tree Topology

A simple tree topology is illustrated in Figure 3. There is a single coupon source, two relays, six leaf nodes (customers), and one store. Relays may choose one of their neighbors to forward coupons to at each time instant. Each customer \( j \) has a different value of \( \hat{x}^j \) and \( \hat{x}^h_j \). At each time instant \( t \), users utilize all the coupons that they possess if the cumulative number of purchases made is less than \((\hat{x}^j + \hat{x}^h_j)T\). Once this is done, they purchase a random number of additional goods at the marked price, as long as it is rational to do so (i.e., either \( \sum_{\tau=0}^{t} x^j_{\tau} \leq \hat{x}^j T \) and \( \sum_{\tau=0}^{t-1} x^h_{\tau+j} \leq \hat{x}^h j T \), or \( \sum_{\tau=0}^{t} x^j_{\tau} \leq \hat{x}^j T \) + \( \sum_{\tau=0}^{t-1} x^h_{\tau+j} \leq \hat{x}^h j T \) and \( \sum_{\tau=0}^{t-1} x^h_{\tau+j} \leq \hat{x}^h j T \)). Users repeat this process until the end of the small time period \( T = 300 \). At the last instant \( t = T - 1 \), if \( \sum_{\tau=0}^{T-2} x^h_{\tau+j} \leq \hat{x}^h j T \), user \( j \) purchases \( \hat{x}^h j T - \sum_{\tau=0}^{T-2} x^h_{\tau+j} \) goods.

We first verify that the small time scale dynamics of backpressure is able to distribute the correct number of coupons to any user \( j \). The capacities of all the relay links are set to 300 coupons per unit time. We illustrate the trajectory of purchases made by user 3 who has \( \hat{x}^j_3 = 50 \) and \( \hat{x}^h_3 = 60 \) over a time interval \( T = 300 \) units in Figure 4. For purposes of illustration, we assume that \( \gamma_3 = \hat{x}^j_3 = 50 \). In other words, we set the reward rate for coupon forwarding by neighbors of user 3 exactly equal to the average rate at which the user 3 should be given coupons in order to extract maximum revenue. We see in Figure 4 that the backpressure scheme indeed gives the right number (and rate) of coupons to the user, ensuring that \( x^j_3[T] = 50 \) and \( x^h_3[T] = 60 \).

![Fig. 3. Simple tree topology.](image)

Fig. 3. Simple tree topology.

![Fig. 4. An example trajectory for user 3 over the small time scale. The solid line indicates purchases made at the marked price, while the dashed line indicates discounted purchases. The user has \( \hat{x}^j_3 = 50 \) and \( \hat{x}^h_3 = 60 \). In this example we have set (for illustration) \( \gamma_3 = \hat{x}^j_3 = 50 \), i.e., the reward rate for coupon forwarding is known exactly, and we see that the user receives exactly the right number of coupons.](image)

We now simulate all three schemes (small and large time scales) and the results are as follows. Figure 2(a) shows the fractional error in high and low price purchases made by user 3, as compared to \( \hat{x}^j_3 \) and \( \hat{x}^h_3 \) for the delay-based scheme. We notice that there is a significant error. We plot the same quantities when we use the backpressure-scheme in Figure 2(b). The scheme quickly converges, causing the errors to be small. Finally, we plot the same for the delay-based scheme in Figure 2(c). For this scheme, we chose the cut-off threshold to be \( T/8 \) and the acceptable delay difference to be 15%. Its error is between the other schemes.

![Fig. 5. Trajectory of revenue obtained by the store using different schemes.](image)

Finally, we plot the total revenue obtained by the store for the three different schemes, and compare them to the
maximum possible revenue in Figure 5. The upper bound is the value \( \sum_j p \hat{x}^h_j + q \hat{x}^l_j \), which is the maximum extractable revenue. We see that the randomized algorithm does significantly worse than backpressure as well as delay-based schemes. Even accounting for the fact that a constant part of the revenue would have to be used to incentivize the scheme, the performance improvement is still significant, although the delay-based scheme performs worse than backpressure.

B. Power Law Topology

We now consider a power-law topology that might bear a closer resemblance to real-world social networks. The graph consists of 100 nodes, and is constructed using preferential-attachment [15] with each entering node connecting to two others. Nodes are connected to the coupon source, are relays, and are customers with probabilities 0.2, 0.7 and 0.1, respectively. Customers have arbitrary spending capacities. We show the upper bound and the performance of the three schemes in Figure 6. Backpressure clearly performs the best, followed by the delay-based and random schemes. The results indicate that using such coupon distribution schemes could significantly increase the revenue obtained.

VIII. CONCLUSION

We developed distributed schemes for targeted coupon delivery using online social networks. The objective was to create a two-tier price structure for maximum revenue extraction by selective discounting. We designed systems that allow users to obtain coupons from their neighbors, and incentivize these neighbors by rewarding them for coupon forwarding. We proved how backpressure ideas could be used to achieve an optimal solution, and also how to use it to obtain a simpler (albeit less efficient) scheme. Future work includes dealing with potential malicious users, and a testbed implementation.

REFERENCES