Fair Recommendations for Online Barter Exchange Networks

Zeinab Abbassi
Columbia University
zeinab@cs.columbia.edu

Laks V.S. Lakshmanan
University of British Columbia
laks@cs.ubc.ca

Min Xie
University of British Columbia
minxie@cs.ubc.ca

ABSTRACT

Of late online social networks have become popular, with interest spanning various aspects including search, analysis/mining, and their potential use for item barter exchange markets. The idea is that users can leverage their social network for exchanging items they possess with other users. The problem of generating recommendations for item exchanges between users, consisting of *synchronous* exchange cycles has been investigated[2]. In this paper, we identify the shortcomings of the above exchange model and propose an *asynchronous* model that makes use of credit points. Rather than insist on exchanging items synchronously, we award points to users whenever they give items to other users, which can be redeemed later. Points and their redemption raise an issue of *fairness* which intuitively means users who contribute more should have a greater priority over others for receiving items they wish for. We focus on fairness maximization and prove that it is NP-hard and cannot be approximated within any factor in polynomial time unless P=NP. We then develop efficient heuristic algorithms, and experimentally demonstrate their effectiveness and scalability on both synthetic data and a real dataset from readitswapit.co.uk.

1. INTRODUCTION

The advent of online social networks has sparked considerable interest in several research problems such as link prediction, community detection, and network analytics. The specific problem of interest to us in this paper is *exchange markets* where users leverage their network for exchanging items with other users. E.g., Jen owns a DVD of *Life of Pi* and Mike owns a DVD of *Brave*. They have watched their DVD and are interested in watching the other movie. Rather than actually buy more DVDs, if they can find exchange partners, they can swap their DVDs and enjoy the movie for free. Finding exchange partners can be facilitated by an online social network provided in addition to users and their contacts, the items (e.g., movies) they have and items they wish for are registered and the system provides helpful suggestions based on available information. Indeed, there are several real instances of popular online exchange markets such as readswapit.co.uk[1], bookmooch.com, and swap.com. The first two enable exchanging of books between users while the last one enables exchange of different types of goods and services. In general, an item exchange could be more general than a swap and can take the form of users exchanging items in a cycle of length $k \geq 2$.

In a recent paper [2], the authors had proposed a model for online item exchange and motivated the problem of *recommending transactions* to users. More precisely, users register an *item list*, containing items they are willing to give other users and a *wish list*, containing items they would like to receive. A recommendation consists of a set of *give*s, telling each user which items in their item list they should give to which user and a set of *get*s, telling the user which items in their wish list they can receive from which user. In this model, the only condition under which an item gets transacted, i.e., changes hands, is when a set of users exchange items in a cycle, in a synchronous manner. Thus, the recommendations generated can maximize the number of items that change hands by maximizing the number of items covered via cycles. We refer to the above model for exchange markets as the *synchronous* model.

Unfortunately, the synchronous model suffers from the following limitations. First, a user cannot receive an item she wishes for unless and until she can find exchange partners with whom she can trade items. In a dynamic network where item and wish lists may get updated, it may take a long time before such exchange opportunities materialize. Second, given a window of time, the number of items transacted may be limited, because the number of items for which exchange opportunities exist based on the current set of item and wish lists may be limited. We validate these intuitions empirically in Section 4.

These observations motivate the question *can we reduce the wait times of users and increase the number of items transacted by doing away with synchronization, decoupling giving and receiving of items, and by linking them via credit points?* To study this question, we propose an *asynchronous* model for item exchange. In order to give an incentive for users to give items to others, we award credit points to item givers. Credit points can be redeemed by users against items they wish to receive. Furthermore, every user is started off with an initial credit so they can begin transacting. In order to develop a basis for awarding and redeeming points, we allow users to bid a price on the items they wish to receive and name an asking price on the items they are willing to give. These prices determine how credits get updated after each transaction. We assume credits are not allowed to get lower than a certain threshold (the threshold may be a very small negative number).

Given an instance of the asynchronous exchange markets model, we study the problem of generating recommendations of transactions for users. As in the synchronous model, we let a recommen-

[1]Some of our experiments are based on the data set from this system.

dation consist of a set of gives, saying what items a user should give to whom and a set of gets, saying what items the user should receive from whom. The set of gives and gets in a recommendation can be represented as triples $u \xrightarrow{i} v$, where u is asked to give item i to v (and v is asked to receive i from u). Such a triple is feasible iff i belongs to u's item list and v's wish list. A recommendation consisting of feasible triples is *consistent* provided no user is asked to give any item to more than one user and no user is asked to receive an item from more than one user. This corresponds to the assumption that each user owns and wants at most one copy of any item[2].

The asking and bidding prices on items determine how the credits of users are updated after a transaction.

A recommendation is *maximal* if adding any feasible triple to it makes it inconsistent. The *net value gain* associated with a recommendation is 0 minus the algebraic sum of changes to the credits of all users. The intuition behind this definition is that the change to the credit of a giver represents the net *value loss* of that user.

We define a notion of fairness for recommendations, based on the intuition that users with a higher credit should have a greater priority for receiving items. One measure of fairness of a recommendation ρ is the maximum credit of any user once users perform transactions following ρ. For the same initial credits, if a recommendation ρ_i leads to a smaller maximum resulting credit than another recommendation ρ_j, we say ρ_i is *fairer* than ρ_j.

In this paper, we make the following contributions[3]:
(1) We propose a credit-based asynchronous model for item exchanges. We show empirically that it takes a long time for the synchronous model to catch up with a fraction of the transactions occurring in a given time slot in the asynchronous model (Section 4).
(2) We show that generating recommendations that maximize fairness is NP-hard and also that this problem is not approximable within any factor, in polynomial time, unless P=NP. We propose two heuristic algorithms for this purpose (Section 3).
(3) Finally, given that fairness maximization is not approximable, we empirically validate the quality and scalability of our heuristic algorithms. We report the results of an extensive set of experiments we conducted (Section 4).

1.1 Related Work

Two main bodies of work are relevant to us – recommender systems and P2P networks.

Recommender Systems: Extensive work has been done in the area of recommender systems [3]. A major distinction with our work is that traditional recommender systems recommend (passive) items while our system recommends (exchange) transactions. Furthermore, item recommendations made to different users are independent, while transactions recommended to users of a system as a whole must satisfy the conflict-free property [2]. In the current work, we consider the combination of transaction recommendations together with credit points, using which users can ask prices and place bids. This distinguishes our work from both the field of traditional recommender systems and [2]. In particular, the model we propose allows for asynchronous exchanges unlike[2]. In [10], the authors propose to recommend transactions for a pair of users by taking into consideration publicly agreed values of items. We note that usually in exchange markets buyers and sellers have different valuation of the same item, thus it is difficult to adapt [10] for settings where valuations of items vary. Furthermore, the transactions recommended in [10] is still based on current item lists of two users (thus synchronized), so unlike our credit-based model, user has to wait till there exists another user who can trasact with him/her.

P2P Networks: Newer peer to peer applications such as BitTorrent incorporate some kind of incentive mechanism to reward sharing peers: a built-in tit-for-tat mechanism. The intuition behind such a design is that it will encourage upload and lead to a *fair* and *efficient* allocation of bandwidth among the peers. In [11], the authors prove that the bandwidth allocation converges to a market equilibrium.

There have been several studies on fairness measures in resource allocation problem [8, 6, 1]. We have chosen the notion of max-min fairness [9]. There are several approximation algorithms known for this particular problem [4, 5]. However none of these results apply in our framework as we have shown that our fairness problem is inapproximable within any factor.

2. THE MODEL

We first define an exchange market, which will serve as a basis for the problems studied in this paper. An *exchange market* is a 6-tuple $\mathcal{E} = (U, I, S, W, A, B)$, where U is a set of users, I a set of items, S and W are functions specifying item lists and wish lists, and A, B are functions specifying asking prices and bids. More precisely, for each user u, $S(u) \subseteq I$ is the item list of u and $W(u) \subseteq I$ is the wish list of u. We will denote $S(u)$ and $W(u)$ as S_u and W_u for convenience. For each user u, item $i \in S_u$, A_{ui} denotes the asking price of i by u and for each item $i \in W_u$, B_{ui} denotes u's bid on i. The bid captures the number of credits a user is willing to spend for receiving an item in her wish list. The asking price captures the number of credits she would like to get in return for giving an item from her item list. Given an exchange market, an *exchange* (*transaction*) is a triple $u \xrightarrow{i} v$. This exchange is *feasible* provided $i \in S_u \cap W_v$. We use the terms exchange and transaction interchangeably.

An exchange market automatically induces a set $\mathcal{C}$ of all feasible transactions. A *recommendation* is a subset of feasible transactions $\rho \subset \mathcal{C}$ such that ρ is *consistent*, i.e., there do not exist two triples $u \xrightarrow{i} v$ and $u \xrightarrow{i} w$ in ρ: $v \neq w$, and there do not exist two triples $u \xrightarrow{i} v$ and $w \xrightarrow{i} v$ in ρ: $u \neq w$. That is, no user is asked to give the same item to two different users nor any user is promised the same item from two different users. We only consider consistent recommendations in what follows. The notion of a recommendation for a user, defined next, is convenient in our discussions.

DEFINITION 1. *Let ρ be a consistent recommendation. Let u be a user. Then the* recommendation for u *is* $\pi_u(\rho) = (G_u, R_u)$, *defined as* $G_u = \{i \mid \exists v : u \xrightarrow{i} v \in \rho\}$ *and* $R_u = \{i \mid \exists v : v \xrightarrow{i} u \in \rho\}$. *By the* recommendation set *of ρ, we mean the set of pairs* $\{\pi_u(\rho) \mid u \in U\}$ *and denote it* $\pi(\rho)$.

A recommendation set can also be specified independently of a recommendation, simply by listing for each user, the items she is supposed to give and items she is supposed to receive. Let $\mathcal{RS} = \{(G_u, R_u) \mid u \in U\}$ be a recommendation set specified in this way. Then a recommendation ρ *implements* this recommendation set iff $\forall u \in U$: $(G_u, R_u) = \pi_u(\rho)$.

Credit System: In our proposed system, a recommendation generation algorithm is run periodically and the set of potential feasible exchanges are discovered. Users involved in these potential exchanges are notified by the system. When a transaction is done the exchanged items are removed from the item lists and wish lists, and

[2]If a user owns/wants more than one copy of an item she can place it multiple times in her item/wish list

[3]A more comprehensive version of the paper with more contributions can be found at `http://www.cs.columbia.edu/~zeinab/FairRecs/fair-recs-fullversion.pdf`

the system is updated. Each user u collects credit for giving items, and can redeem credit points to receive some items. The credit system works as follows: when a user gives away an item their credit will be increased by the asking price for that particular item and when a user receives an item in realization of a wish, their credit decreased by the bid they have made on that item.

Initialization: We initialize the credit of each new user in the system to a small fixed value (as it is done in many similar systems based on credit points, like Yahoo! Answers which sets the initial credit to 100).

Updating Rules: Consider a transaction T at time t in which a user u gives an item i to a user v, i.e, item i is in item list of u and in wish list of v. We should update $C_u(t+1) = C_u(t) + A_{ui}(t)$ and $C_v(t+1) = C_v(t) - B_{vi}(t)$.

In order to *encourage early participation* in the market, we update the credit of users after each time step as follows: $C_u(t+1) = C_u(t) * (1+\delta)$ where δ is a small constant (interest rate), e.g., $\delta = 0.001$. This will encourage users to participate in the system as early as possible.

The bid B_{ui} and asking price A_{ui} model the value gain or value loss of users for receiving and giving the items, respectively.

Notice that the model introduced allows users to say, e.g., a copy of *Hugo* DVD has a value equal to the combined worth (for that user) of *The Help* and any one classical music CD, by placing the bids and asking prices appropriately.

3. FAIRNESS MAXIMIZATION

The problem of fairness maximization is defined as follows:

PROBLEM 1. *Given an exchange market $\mathcal{E} = (U, I, S, W, A, B)$, the goal is to find a maximal recommendation set (W', S') that minimizes the maximum credit.*

Indeed it is not difficult to show that the MAX-FAIR problem is NP-hard as it is harder than the well-known NP-complete problem of minimum makespan scheduling for unrelated machine scheduling [7]. However, in the following, we prove a stronger result that this problem is not even approximable within any factor.

THEOREM 1. *The MAX-FAIR problem is NP-complete and not approximable, in PTIME, within any factor $\alpha \geq 1$, unless P = NP. This hardness result holds even in the special case of the problem where for each item i all the bids and prices on the item lists and wish lists of all users are the same, i.e., for any two users u, v for which $i \in W_u$ and $i \in S_v$, we have $A_{ui} = B_{vi} = P_i$.*

PROOF. Suppose there is an α-approximation algorithm $\mathcal{A}$ for the MAX-FAIR problem for any $\alpha > 1$. That is, if the optimum solution for MAX-FAIR is v_{opt}, then algorithm $\mathcal{A}$ produces a solution v s.t. $v_{opt} \leq v \leq \alpha \times v_{opt}$. We will show that using such an algorithm, the NP-complete problem of subset sum can be solved in PTIME. Without loss of generality, we consider only instances of subset sum in which all weights are larger than or equal to 1. Clearly, the problem remains NP-complete under this restriction. Let $I = \{a_1, \ldots, a_n\}$ be such an instance. Let $B = \Sigma_{1 \leq i \leq n} a_i/2$. Create an instance J of MAX-FAIR with users $1, \ldots, n, n+1, n+2$ and items $1, \ldots, n$. User i has item list $S_i = \{i\}$ and wish list $W_i = \{\}$, $1 \leq i \leq n$ and user j has item list $S_j = \{\}$ and wish list $W_j = \{1, \ldots, n\}$, $n+1 \leq j \leq n+2$. The initial credit of user i is $-a_i$, $1 \leq i \leq n$ and of user j is $B + 1/\alpha$, $n+1 \leq j \leq n+2$. We have the following claim.

Claim 1:.

I is a YES-instance of subset sum iff the optimal solution of J, i.e., the minimum possible value of the maximum credit of any user after a transaction is $1/\alpha$.

PROOF. Suppose I is a YES-instance. Then there is a partition of I into $I_1 \cup I_2$ such that $SUM(I_1) = SUM(I_2) = B$. Then each of the users $1, \ldots, n$ can give their item to user $n+1$ or $n+2$, depending on whether the user's index is in I_1 or I_2. It is easy to see after this set of transactions, the first n users have a resultant credit of 0 whereas the last two have a credit of $1/\alpha$ each. It is not hard to see that the minimum possible value of the maximum credit of a user after performing any set of transactions on the instance J cannot be less than $1/\alpha$.

Suppose the optimum solution to J is $1/\alpha$. Since each of the first n users can only give away their corresponding item, the maximum value of their resultant credit is 0. Thus, the optimum value $1/\alpha$ (which is positive) must be attained by one or both of the last two users, say $n+1$, without loss of generality. This is only possible if user $n+1$ received a set of items from the first n users whose total value equals B. This implies I must be a YES-instance. □

Now, given any instance I of subset sum, we can run our α-approximation algorithm $\mathcal{A}$ on the corresponding instance J of MAX-FAIR. The value v returned by $\mathcal{A}$ is $v_{opt} \leq v \leq \alpha \times v_{opt}$.

Claim 2:.

I is a YES-instance of subset sum iff $1/\alpha \leq v \leq 1$.

PROOF. If I is a YES-instance, $v_{opt} = 1/\alpha$, so clearly, $1/\alpha \leq v \leq 1/\alpha \times \alpha$. If I is a NO-instance, then clearly $v_{opt} > 1/\alpha$. Since all weights $a_i \geq 1$, we know $v_{opt} \geq \beta + 1/\alpha$, where $\beta \geq 1$. By the property of the approximation algorithm, we have $\beta + 1/\alpha \leq v \leq \alpha(\beta + 1/\alpha) = \alpha\beta + 1$. Since the two intervals $[1/\alpha, 1]$ and $[\beta + 1/\alpha, \alpha\beta + 1]$ are disjoint, the claim follows. □

It is thus clear that using algorithm $\mathcal{A}$, we can distinguish, in PTIME, between YES- and NO-instances of subset sum. The theorem follows.

In view of this hardness result, we propose two heuristic algorithms to solve MAX-FAIR; one is a simple and fast greedy algorithm, and the other is based on applying randomized rounding to a linear programming formulation of the problem.

3.1 Greedy Algorithm

A simple heuristic algorithm for the MAX-FAIR problem is to assign item exchanges among users one by one, and at each step find the one item transfer that minimizes the maximum credit after this item transfer. In other words, at each step, we examine all possible item transfers for items $i \in S_u \cap W_v$ for any two users u and v, and find the triple u, v, i such that after item transfer, the maximum credit would be the minimum possible. To do so, we sort the users in the decreasing order of their credit, and try transferring items from users at the beginning of the list. We keep doing this in a loop until no other exchange is possible. This will lead to a recommendation and the algorithm outputs the recommendation sets by projecting it on the various users. Since we continue finding exchanges until no more exchange can be found, the recommendation set is maximal as required. The corresponding pseudo-code for this algorithm is straightforward and is omitted.

3.2 An LP Rounding Heuristic

In this section, we develop an integer linear programming formulation of the MAX-FAIR problem as shown below. Recall that $\mathcal{C}$

denotes the set of all feasible exchanges for a given exchange market. For a triple $u \xrightarrow{i} v \in \mathcal{C}$, let $x_{uvi} = 1$ if we assign item i from u's item list to user v, and $x_{uvi} = 0$ otherwise. Also, we let variable t_u be a variable that corresponds to the credit of user u after the assignment of this step, and let variable t denote $\max_{u \in U} t_u$. Finally, let U be the set of users and I be set of items.

$$\min t \text{ subject to:}$$

$$t \geq t_u \qquad \forall u \in U \tag{1}$$

$$t_u = \sum_{i \in S_u} \sum_{v:u \xrightarrow{i} v \in \mathcal{C}} x_{uvi} A_{ui} - \sum_{i \in W_u} \sum_{v:v \xrightarrow{i} u \in \mathcal{C}} x_{vui} B_{ui} + C_u \qquad \forall u \in U \tag{2}$$

$$\sum_{v:u \xrightarrow{i} v \in \mathcal{C}} x_{uvi} \leq 1 \quad \forall u \in U, i \in S_u \tag{3}$$

$$\sum_{v:v \xrightarrow{i} u \in \mathcal{C}} x_{vui} \leq 1 \quad \forall u \in U, i \in W_u \tag{4}$$

$$x_{uvi} + \sum_{v':v' \xrightarrow{i} v \in \mathcal{C}, v' \neq u} x_{v'vi} + \sum_{v':u \xrightarrow{i} v' \in \mathcal{C}, v' \neq v} x_{uv'i} \geq 1 \qquad \forall u \xrightarrow{i} v \in \mathcal{C} \tag{5}$$

$$x_{uvi} \in \{0, 1\} \qquad \forall u \xrightarrow{i} v \in \mathcal{C} \tag{6}$$

The first inequality indicates that t is the maximum credit of users. The second inequality quantifies the credit of user u after the assignment. The third inequality models the constraint that each item i in the item list of user u can be assigned to somebody at most once, and the fourth inequality models the fact that each item i in the with list of user u can be received by user u. Finally, the fifth inequality models the maximality of the solution, i.e, the fact that if user u does not give an item $i \in W_v$ to user v, then either user u gives i to somebody else, or user v gets it from somebody else.

Next, we give an intuitive description of the algorithm. Consider an LP relaxation of the above integer linear program in which we relax each variable $0 \leq x_{uvi} \leq 1$ for each $u \xrightarrow{i} v \in \mathcal{C}$. Consider a solution x^* to this LP. In the LP relaxation, we can interpret the variable x_{uvi} to be the extent to which user u gives item i to user v. In other words, we use the value x^*_{uvi} as the probability of transferring item i from u to v. In order to find a feasible solution, we need to perform a rounding algorithm. In fact, we perform a natural randomized rounding procedure: we set $\overline{x}_{uvi}$ to 1 with a probability proportional to x^*_{uvi} (i.e, $\frac{x^*_{uv'i}}{Y_{ui}}$ in the algorithm below). After performing this natural rounding procedure, for one user v, there might be several users u for which $\overline{x}_{uvi} = 1$. In this case, v receives item i from several users which is not valid. To obtain a feasible solution, we have to choose only one user u' among users u for which $\overline{x}_{uvi} = 1$, and keep $\overline{x}_{u'vi} = 1$, and make the rest of variables $\overline{x}_{uvi}$ zero. We iterate the above process until we find a maximal solution (i.e, no other item exchange is possible for solution x^* and as a result, the solution $\overline{x}$ did not change in the last step of the algorithm. Here is a formal description of the above idea:

1. Initialize $\overline{x} = 0$.
2. Solve the LP relaxation and find an optimal solution x^*.
3. For each user u and item $i \in S_u$ such that $\sum_v \overline{x}_{uvi} = 0$,
   (a) Let $Z_{vi} = \sum_u \overline{x}_{uvi}$.
   (b) Let $Y_{ui} = \sum_{v:Z_{vi}=0} x^*_{uvi}$.
   (c) If $Y_{ui} > 0$ then among all users v for which $x^*_{uvi} > 0$ and $\overline{Z}_{vi} = 0$,
      i. Choose one user v' with probability $\frac{x^*_{uv'i}}{Y_{ui}}$
      ii. Set $\overline{x}_{uv'i} = 1$, and $S_u = S_u \backslash \{i\}$.
4. For each item i and each v such that $i \in W_v$,
   (a) Let set T_v be the set of users u for which $\overline{x}_{uvi}$ became one in this round.
   (b) If $|T_v| = 1$, then for each $v' \in T_v$, $x^*_{v'vi} = 0$, and $W_v = W_v \backslash \{i\}$ (finalize transfer $v' \xrightarrow{i} v$),
   (c) else
      i. Choose one user $v' \in T_v$ with prob $\frac{1}{|T_v|}$.
      ii. For each $u \neq v'$ and $u \in T_v$, set $\overline{x}_{uvi} = 0$, and $S_u = S_u \cup \{i\}$.
      iii. For v', let $x^*_{v'vi} = 0$, and $W_v = W_v \backslash \{i\}$ (finalize transfer $v' \xrightarrow{i} v$).
5. If any new item transfer was assigned in the last round, go back to 3 to start a new round, otherwise output $\overline{x}$ and terminate.

4. EXPERIMENTAL EVALUATION

Goals. The goals of our experiments are as follows:
(1) To examine the quality of the results found by our heuristic algorithms for MAX-FAIR, a problem we showed to be hard to approximate within any factor.
(2) To measure the advantages of the asynchronous model over the synchronous model.
(3) To study the scalability of the heuristic algorithms and also the impact of the parameters used in the experiments on the performance of the algorithms.

All experiments are done on a Windows machine with a 2.4 GHz CPU, 8GB of Memory, and 320GB of SCSI hard disk.

Dataset. We perform our experiments on both synthetic datasets and a real dataset adapted from a real exchange market application.
Real Data: We have obtained a recent snapshot of the data from ReadItSwapIt.co.uk. ReadItSwapIt, a popular online book exchange website located in United Kingdom, allows book lovers to swap used books with each other. In this dataset, there are more than 50,000 users, 270,000 active used books in users' item lists and 170,000 books in users' wish lists.

In Figure 1, we show some properties of the ReadItSwapIt dataset. From Figure 1 (a) and (b), it is clear that the number of times that a book occurs in an item list and in a wish list both follows a power-law distribution. And in Figure 1 (d), we have shown that size of wish list also follows the power-law distribution. Although the size of item list as shown in Figure 1 (c) may not exactly follow a power-law distribution, we can still infer from the figure that the number of item lists having a specific size has an exponential decay behavior as the size grows. These observations validate our intuition on how to generate synthetic dataset, discussed below.

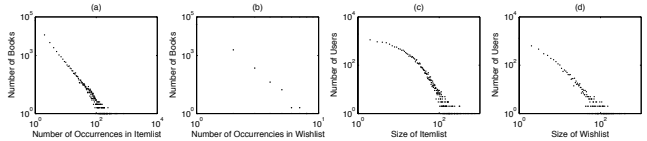

Figure 1: Distribution of: (a) occurrences of books in item list, (b) occurrences of books in wish list, (c) size of item list, (d) size of wish list.

Synthetic Data: The intuition behind the data generation is that the popularity of items in wish lists and item lists follows some

power law distributions, i.e., there are many items which are wished for or provided by a small number of people, and there is a small number of items which are provided by or wished for by many people. To achieve this distribution, first, we generate some power law distributions with a given power as the parameter. We actually examined four different powers of 0.25, 0.5, 0.75 and 1. We use one of these power law distributions for the popularity of the items, i.e., the number of people who own an item. We also generate a set of item list sizes based on some other power law vectors (this is justified by the fact that the size of wish lists and item lists should also follow some power law distribution). We also generate a vector of wish list sizes in direct correlation with the vector of item list sizes. The intuition here is that if a user provides more items, then probably she has larger wish lists as well. This intuition need not be true in all cases, so we add some noise to this process with a small probability. The initial credit score for users for experiments is set to 100. The asking price and bid values are also set as uniform random numbers between 5 and 50.

Synchronous vs. Asynchronous.

Here, we explore the advantage of the asynchronous model w.r.t. both the number of transactions completed over a given time slot and the *catchup time* for synchronous model under a very general setting. We define the catchup time to be the number of time slots required for the synchronous model to achieve a fraction of the transactions occurring in one time slot in the asynchronous model.

Our simulation of the runtime environment of an exchange market is as follows: at each fixed time slot, we generate several sets of items, whose arrival pattern follows Gaussian distribution. [4] The number of transactions completed in the synchronous and asynchronous model are calculated using the above analysis result. Here in this experiment, the number of users is set to 500 and 600, and for the Gaussian distribution, we have experimented with the environment under the setting of $\mu = 20$, $\sigma = 10$ and $\mu = 10$, $\sigma = 10$.

In Figure 2, under various settings, we show the ratio of (the upper bound on the) number of transactions completed in the synchronous model over the asynchronous model at different time slots. It can be observed from the figure that the ratio decreases as the number of users increases, and when μ of the Gaussian distribution increases, which means the number of items generated at each time slot increases, the ratio will increase.

To better examine the "time latency" of the synchronous model, we calculate the catchup time, the number of time slots needed to catch up with a certain percentage (e.g., 5% to 50%) of transactions completed in the asynchronous model (see Figure 3). Similar to the ratio of transactions completed at different time slots shown in Figure 2, we can also observe from Figure 3 that the catchup time increases as the number of users increases, and the catchup time decreases as the μ of Gaussian distribution increases. And we can easily observe that under the best setting, it takes more than 50 time slots for the synchronous model to catch up with 50% of the total transactions done under the asynchronous model when $\lambda = 1$ (where λ is the number of times an item is put in wish lists, $\lambda \geq 1$) , and even when $\lambda = 3$, to catch up with the same amount, it takes more than 10 time slots.

We have observed from the ReadItSwapIt dataset (Figure 1(b)) that the number of times an item occurs in users' wish lists follows the power law distribution, and more than 90% of all items are wished by just one user, implying in practice the average value of λ may be closer to 1. From Figure 2 and Figure 3, we observe that the asynchronous model has a great advantage over the synchronous model w.r.t. both ratio of number of transactions completed and catchup time, thus clearly establishing an empirical motivation for studying the asynchronous model.

[4]Other distributions are possible; note that this is orthogonal to the item and wish lists following power law.

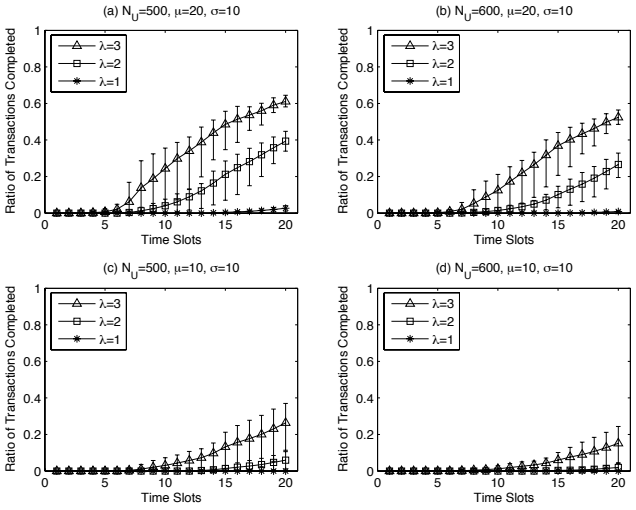

Figure 2: Comparison of asynchronous model and synchronous model on ratio of transactions completed

Fairness

Since MAX-FAIR is inapproximable, there is a need to evaluate the quality (i.e., accuracy) of our heuristic algorithms. We implemented the integer linear program (ILP) formulation of MAX-FAIR using GNU Linear Program Kit (GLPK) and the LP-rounding and greedy algorithms using MATLAB. The experiments were run on a computer with a 2.40GHz Intel Core 2 Duo CPU and 3 GB of RAM under Windows Vista. We ran our algorithms on three different data sets with 100, 500 and 1000 users. The data were generated with four different values for $\alpha = 0.25, 0.5, 0.75$ and 1, where α is the skew factor of the power law.

We solved the ILP for two small datasets including 10 users, 50 items and 20 users, 100 items. ILP gives us the optimal solution however its running time is exponential. We compare the results of the three algorithms, depicted in Figure 4. Notice that the performance (maximum credit value) achieved by LP-rounding is very close to the optimal value of ILP. Greedy's performance is a little worse, however it is within a factor of 2.3 of the optimum.

Figure 5 shows the results of the LP, LP-rounding and the greedy heuristic for datasets of size 100, 500, and 1000 respectively. The total number of items in each case is 5 times the number of users. The numbers indicate the maximum credit, for various values of α for different algorithms. The LP relaxation discussed in the previous sections gives a polynomial-time computable lower bound on the optimal value of the fairness optimization problem. In fact, we use this lower bound to evaluate the performance (i.e., quality) of our heuristic algorithms. As can be seen, the LP-rounding has a better performance than greedy. Also, we can observe that as α increases (in most cases), the algorithms' performance also improves.

Scalability

So far, our focus has been on the quality of the results returned by LP-rounding and greedy. To evaluate scalability, we note that in practice, the numbers of users and items may run in tens to hundreds of thousands. Since the size of the LP that needs to be solved is correspondingly huge, we restrict the scalability evaluation to the greedy algorithm, since the free LP solver we have access to cannot scale up to that size.

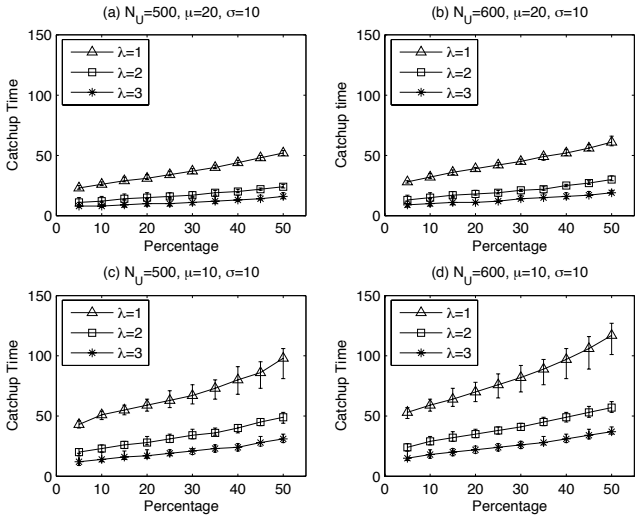

Figure 3: Comparison of asynchronous model and synchronous model on catchup time.

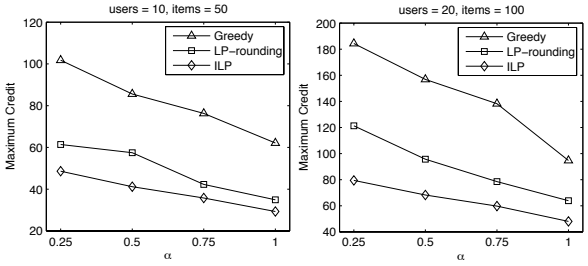

Figure 4: Performance of the heuristics compared to the optimal solution given by ILP on various datasets

Our scalability experiments were conducted on the real data set – ReadItSwapIt, ReadItSwapIt is a synchronous exchange system so its dataset does not contain pricing and credit information, so we synthetically generated asking price, bidding price and initial credits, using a procedure similar to that for our synthetic data generation.

The run times correspond to the greedy heuristic algorithm in case of MAX-FAIR (see Figure 6). As can be seen from this figure, the Greedy algorithms scales well for large data sets.

5. FINAL REMARKS

In a recent paper [2], the problem of exchange markets for items in an online social network of users was studied, where exchanges are required to be synchronous. We pointed out the shortcomings of this model. In this paper, we proposed an asynchronous system using virtual credit points. We empirically demonstrated that both w.r.t. number of transactions done in a time slot and w.r.t. catchup time, the asynchronous model significantly outperforms the synchronous one. Introducing virtual points to the system raises the issue of fairness and motivates the problem of fairness maximization (MAX-FAIR). We showed that MAX-FAIR is NP-hard and is not approximable within any factor, unless P = NP. We proposed a simple greedy algorithm and a LP rounding heuristic for this problem and evaluated their quality empirically. Our experiments show both heuristics have a reasonable quality. While LP-rounding has better quality, greedy is much more efficient and scalable.

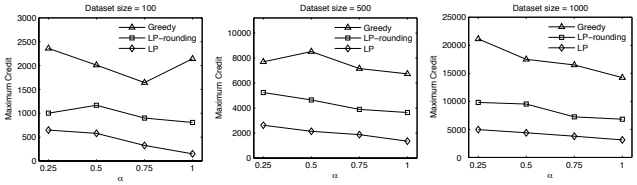

Figure 5: Performance of different algorithms on datasets of different sizes

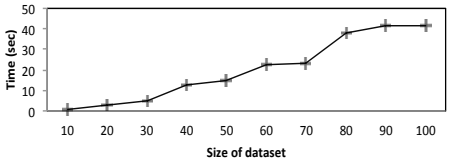

Figure 6: Scalability of the Greedy Algorithm

Clearly, in exchange markets, reputation of users may affect the preference of users for transacting with other users. Inferring reputation and incorporating it in the context of recommendations is an important direction worthy of research. And further research is needed to devise strategies for suggesting "optimal" asking and bidding prices.

6. REFERENCES

[1] S. P. A Goel, A Meyerson. Combining fairness with throughput: Online routing with multiple objectives. *Journal of Computer and System Sciences*, 63(1):62 – 79, 2001.

[2] Z. Abbassi and L. Lakshmanan. On efficient recommendations for online exchange markets. In *25th International Conference on Data Engineering (ICDE)*, 2009.

[3] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[4] A. Asadpour and A. Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. In *STOC*, pages 114–121, 2007.

[5] N. Bansal and M. Sviridenko. The santa claus problem. In *STOC*, pages 31–40, 2006.

[6] É. T. J Kleinberg, Y Rabani. Fairness in routing and load balancing. In *Foundations of Computer Science (FOCS)*, page 568, 1999.

[7] D. B. S. J. Lenstra and E. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46(3):259–271, 1990.

[8] A. Kumar and J. Kleinberg. Fairness measures for resource allocation. In *Foundations of Computer Science*, pages 75–85, November 2000.

[9] E. M. R. J. Lipton, E. Markakis and A. Saberi. On approximately fair allocations of indivisible goods. In *ACM Conference on Electronic Commerce*, pages 125–131, 2004.

[10] Z. Su, A. K. H. Tung, and Z. Zhang. Supporting top-k item exchange recommendations in large online communities. In *EDBT*, pages 97–108, 2012.

[11] F. Wu and L. Zhang. Proportional response dynamics leads to market equilibrium. In *STOC*, pages 354 – 363, 2007.