Ignorant vs. Anonymous Recommendations

Jara Uitto and Roger Wattenhofer

ETH Zürich

Abstract. We start with an unknown binary $n \times m$ matrix, where the entries correspond to the preferences of n users on m items. The goal is to find at least one item per user that the user likes, with as few queries as possible. Since there are matrices where any algorithm performs badly without any preliminary knowledge of the input matrix, we reveal an anonymized version of the input matrix to the algorithm in the beginning of the execution. The input matrix is anonymized by shuffling the rows according to a randomly chosen hidden permutation. We observe that this anonymous recommendation problem can be seen as an adaptive variant of the Min Sum Set Cover problem and show that the greedy solution for the original version of the problem provides a constant approximation for the adaptive version.

1 Introduction

Algorithmic research studies a variety of models that are beyond the traditional input-output paradigm, where the whole input is given to the algorithm. Examples of such unconventional models are distributed, streaming, and multiparty algorithms (where the input is distributed in space), or regret, stopping, and online algorithms (where the input is distributed in time). Not having all the input initially is a drawback, and one will generally not be able to produce the optimal result. Instead, the algorithm designer often compares the result of the restricted online/distributed algorithm with the result of the best offline/centralized algorithm by means of competitive analysis.

However, it turns out, this is sometimes not possible. In particular, if the hidden input contains more information than we can learn within the execution of the algorithm, we might be in trouble. This is generally an issue in the domain of recommendation and active learning algorithms. In this paper, we study the following example. We are given an unknown binary matrix, where the entries correspond to preferences of n users on m items, i.e., entry (i, j) corresponds to whether user i likes item j. Thus, row i in the matrix can be seen as the *taste* vector of user i. In each round, the algorithm is allowed to reveal one entry in the matrix, i.e., query one user about one specific preference. The goal is to find at least one 1-entry in each row with a minimum number of queries. We call this problem the *ignorant* recommendation problem, since initially, the algorithm knows nothing about the taste matrix, and only over time (hopefully) learns about the taste of the users.

In the domain of recommendation and active learning, competitive analysis is still waiting to make its outburst. The approach is often to assume certain properties about the tastes of people, e.g., the users are partitioned into a small number of classes with very similar taste or, more abstractly, that the underlying taste matrix features certain algebraic properties such as low rank. Competitive analysis seems to be out of reach, exactly because a ignorant algorithm cannot compete against an algorithm that knows everything about the taste of the users.

Since we do not want to change the ignorant recommendation problem, our only hope is to make the competition weaker. What is the strongest model for the adversary that allows reasonable (or non-trivial) results? In this paper, we propose an *anonymous* version of the problem. In the anonymized problem, the adversarial algorithm knows the whole taste matrix, but the users are anonymous, i.e., the rows of the taste matrix have been permuted arbitrarily. We call this the anonymous recommendation problem.

We build on two previous results: First, a result that studies a *oblivious* version of the problem, called Min Sum Set Cover (**mssc**) [5]. The input for **mssc** is a collection of elements and a set of subsets of these elements, similarly to the classical Set Cover problem. The output is a linear order of these sets where the ordinal of the set that first covers an element e induces a cost $f(e)$ for e. The goal is to minimize the sum $\sum_e f(e)$ of the costs of the elements.

The **mssc** problem is oblivious in the sense that the strategy of an algorithm solving the **mssc** problem is independent of the recommendation history. In other words, the algorithm chooses an ordering of the items in the beginning of the execution and each user is recommended items according to this ordering. Our setting on the other hand allows the algorithm to be *adaptive* and change the strategy after each recommendation. In the oblivious setting, it is known that the greedy algorithm is a 4-approximation [5]. The second previous result compares the ignorant problem to the oblivious problem [19]. We strengthen this result by showing that the bounds hold (asymptotically) even when comparing the ignorant problem to the anonymous problem, instead of the oblivious problem. The relations between the aforementioned problems are illustrated in Figure 1.

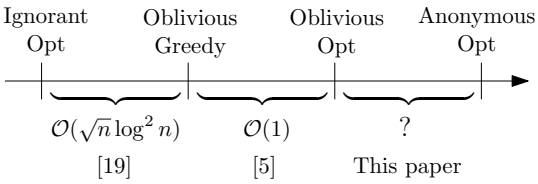


Fig. 1: We show that the greedy algorithm, and thus also the optimal algorithm, for the oblivious recommendation problem provide a constant approximation to the anonymous recommendation problem.

The core of this paper hence deals with the power of anonymous input, showing that a good solution for the oblivious problem also yields a good result

for the anonymous problem. In particular, we show that the greedy algorithm for **mssc** yields a constant approximation to the anonymous problem. In this sense, our problem is an anonymous and adaptive variant of the **mssc** problem.

2 Related Work

The classic result related to the Min Sum Set Cover problem is that the greedy algorithm provides a constant approximation. It was shown by Bar-Noy, Bellare, Halldórsson, Shachnai, and Tamir [5] that the greedy solution is a 4-approximation. Feige, Lovász, and Tetali gave a simpler proof for this result and showed that getting an approximation ratio of $4 - \epsilon$ for any $\epsilon > 0$ is NP-hard [9]. Our work extends their results from the oblivious and offline setting into an adaptive and online setting, where the algorithm is allowed to change its strategy during the execution but is not given full information in the beginning. Online variants of the **mssc** problem have been studied before for example by Munagala et al. [17], who showed that even if the elements contained in the sets are hidden from the algorithm, one can achieve an $\mathcal{O}(\log n)$-approximation.

Also other variations of **mssc** have been considered. As an example, Azar and Gamzu studied ranking problems, where the goal is to maintain an adaptive ranking, i.e., an ordering of the sets that can change over time, while learning in an active manner [4]. They provided an $\mathcal{O}(\log(1/\epsilon))$-approximation algorithm for ranking problems, where the cost functions have submodular valuations. Golovin and Krause [12] studied problems with submodular cost functions further and in particular, they considered them in an adaptive environment. Furthermore, **mssc** is not the only classic optimization problem studied in an active or an adaptive environment. There exists work on adaptive and active versions of, for example, the well-known Set Cover [10, 16], Knapsack [7], and Traveling Salesman [14] problems.

The input for the anonymous recommendation problem is a binary relation. Learning binary relations has been studied for example by Goldman et al [11]. They studied four different learning models: adversarial, random, a helpful teacher and similar to ours, a setting where the learner can choose which entry to look at. They studied upper and lower bounds on the number of mistakes the learner makes when predicting the entries in the input matrix. As a byproduct, they showed there are inputs where any algorithm can be forced to make $\Omega(k \cdot m)$ mistakes given a matrix with k different row types. In our setting, different row types correspond to users with different taste vectors and therefore, their bound immediately gives us a corresponding lower bound for any recommendation algorithm without any preliminary knowledge of the input, including **mssc** based solutions.

A common way to overcome such lower bounds is to perform a competitive analysis. However, an offline algorithm that sees the whole input for the ignorant recommendation problem can always solve the problem with 1 query per user. It was shown recently that if an online algorithm for the ignorant recommendation problem is compared to the optimal solution for **mssc**, the analysis becomes non-

trivial [19]. Our results extend this work by showing that the *quasi*-competitive ratio stays asymptotically the same if we compare the online solution to the optimal anonymous algorithm. We also improve the results from previous work by allowing the recommendation algorithms to choose the sequence according to which the users are picked. In the previous work, the sequence was chosen uniformly at random. Another way to relax the competitive analysis is to give the online algorithm more power. For example the list update and bin packing problems have been studied under more powerful online algorithms [1, 13].

The task we are considering can also be seen as a relaxed version of learning the identities of the users, that is, we wish to classify the unknown users into groups according to their preferences. Since the users are determined by their preferences, this can further be seen as finding a matching between the users and the preferences. The matching has to be perfect, i.e., in the end every user has to be matched to a unique preference. A similar setting was studied in economics, where the basic idea is that each buyer and seller have a hidden valuation on the goods that they are buying or selling and the valuations are learned during the execution. Then the goal is to find a perfect matching between a set of buyers and a set of sellers, where an edge in the matching indicates a purchase between the corresponding agents [6, 15].

The main motivation for our work comes from the world of recommendations and the main interpretation of our variation of **mssc** is an online recommendation problem. Models of recommendation systems close to ours were studied by Drineas et al. [8], Awerbuch et al. [3] and Nisgav and Patt-Shamir [18], where the recommendation system wishes to find users that have interests in common or good items for users. One of their real life examples are social networks. In their works, the users are assumed to have preferences in common, whereas we study an arbitrary feasible input. With an arbitrary input, Alon et al. [2] showed that one can learn the whole preference matrix with minimal error in polylogarithmic time in a distributed setting.

3 Model

The input for the anonymous recommendation problem is a pair (U, V) consisting of a set of users $U = \{u_1, \ldots, u_n\}$ and a set of preference vectors $V = \{v_1, \ldots, v_n\}$ of length m, where preference vector $v \in V$ corresponds to the (binary) preferences of some user $u \in U$ on m items. Each user u_i is assigned exactly one preference vector v_j according to a hidden bijective mapping $\pi : U \to V$. By identifying the users with the preference vectors, π is a permutation of the users. The permutation π is chosen uniformly at random from the set of all possible permutations.

The execution of a recommendation algorithm works in rounds. In each round, a recommendation algorithm first picks a user $u \in U$ and then recommends some item b to this user. Recommending item b to user u is equivalent to checking whether user u likes b or not, i.e., the corresponding entry is revealed

to the algorithm immediately after the recommendation. A recommendation algorithm is allowed to pick the user and the item at random.

We say that user u is *satisfied* after she has been recommended an item that she likes. The goal is to satisfy all users which corresponds to finding at least one 1-entry from each preference vector. The algorithm terminates when all users are satisfied. The runtime of a recommendation algorithm is measured as the expected number of queries. In other words, the runtime corresponds to the number of rounds until all users are satisfied. Therefore, the trivial upper and lower bound for the runtime are $n \cdot m$ and n, respectively, since it takes $n \cdot m$ queries to learn every element of every preference vector and n queries to learn one entry from each preference vector.

We assume that for any user u, there is always at least one item that she likes but we do not make any further assumptions on the input. Let OPT be the optimal recommendation algorithm for the anonymous recommendation problem. We measure the quality of a recommendation algorithm A by its approximation ratio, i.e., the maximum ratio between the expected number of queries by A and by OPT for any input I.

An important concept throughout the paper is the *popularity* of an item. The popularity of an item corresponds to the number of users that like it.

Definition 1. *Let b be an item. The popularity $|b|$ of item b is the number of users that like this item, i.e., $|b| = |\{v \in V \mid v(b) = 1\}|$.*

4 Anonymous Recommendations

The main goal of this paper is to show that from an asymptotic perspective, the anonymous and the oblivious recommendation problems are equally hard. Recall that in the oblivious setting, the algorithm sees a probability distribution D over the set of possible preference vectors for the users and must fix an ordering O of the items before the first query. Then, each user is recommended items according to O until she is satisfied. Otherwise, the oblivious model is similar to the anonymous model. To achieve our goal, we first observe that solving the oblivious recommendation problem takes at least as much time as solving the anonymous recommendation problem for any instance of preferences selected according to D. Clearly, an anonymous algorithm that chooses the best fixed ordering of books is at least as fast as any oblivious algorithm for this instance.

Then we show that the greedy algorithm for the oblivious recommendation problem is a 20-approximation to the anonymous recommendation problem. We follow the general ideas of the analysis of the greedy algorithm for **mssc** by Feige et al. [9], where they show that the greedy algorithm provides a 4-approximation. The fundamental difference between our analysis and theirs comes from bounding from below the number of recommendations needed to satisfy a given set $U' \subseteq U$ of users. In the oblivious setting, it is easy to get a lower bound on the number of recommendations needed per user. Given the most popular item b^* among users in U', $\Omega(|U'|^2/|b^*|)$ recommendations are needed, since one item

can satisfy at most $|b^*|$ users, and each item is recommended to all unsatisfied users. In the adaptive setting, this is not necessarily the case.

We first give a lower bound on the amount of queries that are needed to satisfy any group of users as a function of the best item within this group. In essence, we show that from an asymptotic and amortized perspective, any adaptive algorithm also needs $\Omega(|U'|^2/|b^*|)$ rounds to satisfy all users in U'. Then, in Section 6, we utilize our lower bound and get that the greedy algorithm for **mssc** yields a constant approximation to the anonymous recommendation problem.

4.1 Consistency Graph

We identify the users with their preference vectors, which indicates that each user $u \in U$ corresponds to an (initially) unknown binary preference vector of length m. Therefore, each user can be seen as a preference vector that denotes the information we have gained about user u. We also identify the items with their corresponding indices, i.e., for an item b that has been recommended to u, $u(b)$ denotes the entry in the preference vector of user u that corresponds to whether u likes b or not. We call $u \in U$ and $v \in V$ *consistent*, if $u(i) = v(i)$ for every revealed entry $u(i)$.

Let OPT be the optimal anonymous recommendation algorithm. We model the state of an execution of OPT as a bipartite graph $G = (U \cup V, E)$, where $(u, v) \in E$ iff $u \in U$ and $v \in V$ are consistent. We refer to G as the *consistency* graph. The purpose of the consistency graph is to model the uncertainty that OPT has on the preferences of the users. To simplify our analysis, we provide OPT with the following advantage. Whenever OPT recommends user u an item b that u likes, we identify $u \in U$ with $v \in V$ in permutation π, i.e., OPT learns that $\pi(u) = v$. We note that this advantage can only improve the runtime of OPT, i.e., if we prove a lower bound for the performance of this "stronger" version of OPT, the same bound immediately holds for the optimal anonymous recommendation algorithm.

Now since the connection is revealed after finding a 1-entry and thus, the complete preference vector of u is learned, nothing further can be learned by recommending u more items. Therefore, we can simply ignore $u \in U$ and $\pi(u) \in V$ for the rest of the execution. Thus, upon recommending user u an item that she likes, we simply remove u from U and the corresponding preference vector $\pi(u)$ from V. The modification also implies that the termination condition, i.e., all users being satisfied, is equivalent to the sets U and V becoming empty.

The construction of G is illustrated in Figure 2. We emphasize that the graph G changes over time and we denote the state of G in round $r \geq 0$ by $G_r = (U_r \cup V_r, E_r)$, where U_r and V_r are the users and their preference vectors remaining in round r, respectively, and E_r is the set of edges between consistent nodes in round r. Notice that $G_0 = (U_0 \cup V_0, E_0)$ is a complete bipartite graph. We omit the index from the consistency graph whenever the actual round number is irrelevant. In addition, we note that even if the identity of a certain user u is clear (see user u_3 in Figure 2 for an example), the edges connected to u and $\pi(u)$

are only removed from G when the corresponding users and preference vectors become inconsistent.

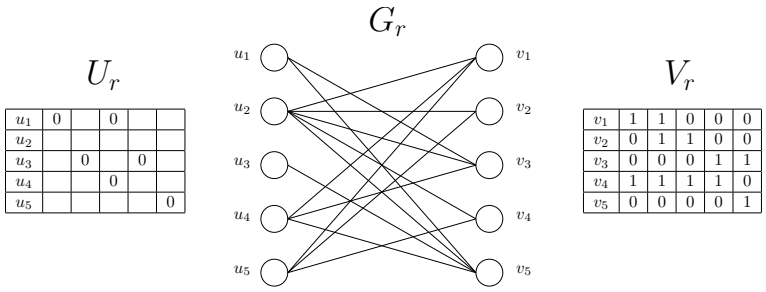


Fig. 2: A matrix representation of the unknown and the known entries of an input are given on the left and right, respectively. The consistency graph constructed based on V_r and the state of U_r is denoted by G_r. Nodes u_i and v_j are connected if and only if the corresponding rows are consistent.

5 Learning the Preferences

The goal of this section is to quantify the amount of knowledge OPT can gain per round. The intuition behind modeling the anonymous recommendation problem as a bipartite graph is that the number of remaining edges correlates with the amount of uncertainty OPT has. In other words, by querying the preferences of the users OPT can exclude inconsistent edges in G. When a 0-entry is discovered by recommending item b to user u, at most $|b|$ preference vectors can become inconsistent with u since there are $|b|$ vectors $v \in V$ such that $v(b) = 1$. On the other hand, when discovering a 1-entry, up to $2|U|$ edges might get removed due to removing u, $\pi(u)$ and all edges adjacent to them from G. Note that the consistency graph is simply a representation of the knowledge OPT has about the preference vectors, i.e., excluding the edges from the consistency graph only happens implicitly according to the revealed entries.

We employ an amortized scheme, where we pay in advance for edges that get removed by discovering 1-entries. Consider the case where a 0-entry is revealed from $u(b)$. Now all the edges $(u, v) \in E$, where $u(b) \neq v(b)$, are removed. For every edge (u, v) removed this way, we give both node u and node v two units of money that can be used later when their corresponding connections are revealed. Since at most $|b|$ edges are removed, we pay at most $2|b| + 2|b| = 4|b|$ units of money in total in a round where OPT discovers a 0-entry.

As an example, consider the graph illustrated in Figure 2 and assume that a 0 is revealed from $u_4(5)$. Now u_4 becomes inconsistent with v_3 and v_5, and the corresponding edges are removed. Upon removing these edges, we give two units of money to v_3, two units of money to v_5 and four units of money to u_4.

5.1 Finding a 1-entry

Now we look at the case of discovering a 1-entry. In the following, we consider the consistency graph G_r for an arbitrary round r but omit the index when it is irrelevant for the proofs. Consider user $u \in U$ and let $\pi(u) = v$. Upon discovering the 1-entry from user u, we reveal that $\pi(u) = v$ and all the edges adjacent to u and v are removed. We divide the analysis into two cases. Consider first the case where $|\Gamma(u)| + |\Gamma(v)| \leq 4|U|/3$, where $\Gamma(u)$ denotes the exclusive neighborhood of u. The exclusive neighborhood of node $u \in U$ in graph $G = (U \cup V, E)$ contains all the nodes in the 1-hop neighborhood of u except node u itself, i.e., $\Gamma(u) = \{v \in V \mid (u, v) \in E\}$ and analogously for $v \in V$.

Note that since satisfied users are removed from U, $G = (U \cup V, E)$ is a complete bipartite graph if there are no revealed 0-entries. Therefore, any edge $(u, v) \notin E$, where $u \in U = U_r, v \in V = V_r$, was removed by revealing a 0-entry. Given that $|\Gamma(u)| + |\Gamma(v)| \leq 4|U|/3$, we know that at least $2|U| - 4|U|/3$ of the edges adjacent to u or v were removed by revealing 0-entries. We pay two units of money to either u or v, for every edge removed from the set of edges adjacent to nodes in $\Gamma(u) \cup \Gamma(v)$ and therefore, the combined money that the nodes have is at least $2(2|U| - 4|U|/3) = 4|U|/3$. Therefore, the money "pays" for all edges that are removed due to revealing the connection between u and v.

We use the rest of this section to study the second case, that considers the case where the sum of degrees of nodes u and v is high, i.e., larger than $4|U|/3$. The aim is to show that it is unlikely that v is the preference vector of u, since there are many consistent nodes with u and v and thus, there has to be considerably more valid permutations π', where $\pi'(u) \neq v$, than permutations, where $\pi'(u) = v$. This in turn implies that a randomly chosen permutation is likely not to have u connected to v.

We call a matching σ *compatible* with an edge $e = (u, v)$ if $(u, v) \in \sigma$ and *incompatible* with e otherwise. In the following lemma, we bound the number of perfect matchings that are compatible with a given edge e in G. Note that every perfect matching in G corresponds to some permutation of the users.

Lemma 1. *Assume that $|\Gamma(u)| + |\Gamma(v)| > 4|U|/3$ for some nodes u and v in G. Let h be the total number of perfect matchings in G. Then there are at most $3h/|U|$ perfect matchings that are compatible with (u, v).*

Proof. Let σ be a perfect matching that is compatible with (u, v) in G. Let

$$U' = \{u' \in \Gamma(v) \mid (u', v') \in \sigma \text{ and } v' \in \Gamma(u)\} \setminus \{u\} ,$$

$|\Gamma(u)| = k$ and let $\Gamma_\sigma(\Gamma(u)) = \{u' \in U \mid (u', v') \in \sigma \text{ and } v' \in \Gamma(u)\}$ be the set of nodes matched to $\Gamma(u)$ by σ. See Figure 3 for an illustration. Since σ is a matching, we get that $k = |\Gamma(u)| = |\Gamma_\sigma(\Gamma(u))|$. Also, we know that $|\Gamma(v)| + |\Gamma(u)| > 4|U|/3$ and therefore $|\Gamma(v)| \geq 4|U|/3 - k + 1$.

By taking a closer look at the definition of U', we see that $U' = \Gamma(v) \cap \Gamma_\sigma(\Gamma(u)) \setminus \{u\}$ and by re-writing, we get that $U' = \Gamma(v) \setminus (U \setminus \Gamma_\sigma(\Gamma(u))) \setminus \{u\}$.

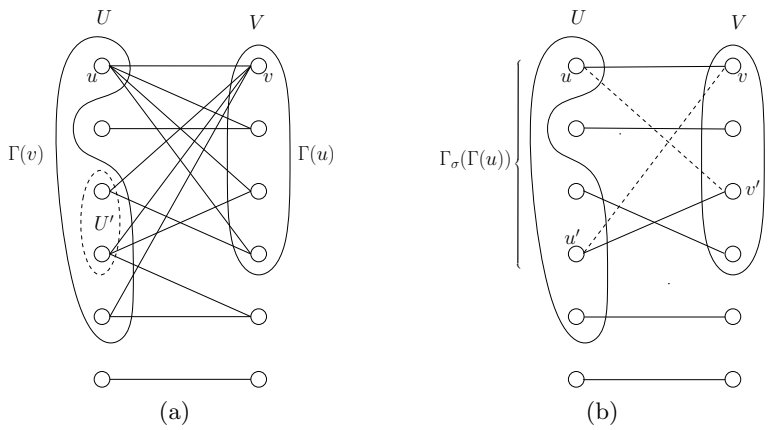


Fig. 3: The consistency graph is illustrated on the left. On the right, we show a perfect matching compatible with (u, v) with the solid lines. For every node $u' \in U'$, we have a valid perfect matching that is incompatible with (u, v) if we use edges (u, v') and (u', v) instead of (u, v) and (u', v').

From the equations above, it follows that

$$|U'| \geq |\Gamma(v)| - (|U| - k) - 1 \geq \frac{4|U|}{3} - k + 1 - (|U| - k) - 1 = \frac{|U|}{3} \ .$$

For each node $u' \in U'$, we have a perfect matching $\sigma_{u'}$ that is incompatible with (u, v) in G, where (u, v) and $(u', v') \in \sigma$ are replaced by (u, v') and (u', v). In addition, the incompatible perfect matching $\sigma_{u'}$ is different for every $u' \in U'$, since $(u', v) \notin \sigma_z$ for any $u' \neq z \in U'$. Therefore, we have at least $|U|/3$ perfect matchings incompatible with (u, v) for every perfect matching that is compatible with (u, v). Note that no matchings are counted twice. The claim follows. □

In the beginning of the execution, the probability of user $u \in U$ to be matched to vector $v \in V$ is simply $1/n$. When revealing the unknown entries, these probabilities change. The next step is to bound the probability of user $u \in U$ to be matched to vector $v \in V$ given the state of the consistency graph G. We identify the randomly chosen permutation π with a perfect matching σ_π where $(u, v) \in \sigma_\pi$ iff $\pi(u) = v$. Since the permutation π was chosen uniformly at random, any valid permutation, i.e., a permutation that does not contradict the revealed entries, is equally likely to be σ_π. Therefore, the probability that an edge (u, v) is in matching σ_π corresponds to the ratio of perfect matchings in G that are compatible with (u, v) and the number of all perfect matchings in G. We denote the event that edge (u, v) is in σ_π by $A(u, v)$ and the probability of $A(u, v)$ given G by $\mathbb{P}[A(u, v) \mid G]$.

Lemma 2. *Let $G = (U \cup V, E)$ be the consistency graph. For any nodes $u \in U$ and $v \in \Gamma(u)$, such that $|\Gamma(u)| + |\Gamma(v)| > 4|U|/3$, $\mathbb{P}[A(u, v) \mid G] \leq \frac{4}{|\Gamma(u)|+|\Gamma(v)|}$.*

Proof. Since the permutation π is chosen uniformly at random, σ_π is equally likely to be any of the possible perfect matchings in G. Therefore, the likelihood of u being matched to $v \in V$ is the number of perfect matchings that are compatible with (u, v) divided by the number of all possible perfect matchings. By Lemma 1, the number of perfect matchings that are compatible with (u, v) is at most $3h/|U|$, where h is the total number of perfect matchings in G. Therefore,

$$\mathbb{P}[A(u,v) \mid G] \leq \frac{3h}{|U|} \cdot \frac{1}{h} = \frac{3}{|U|} \leq \frac{3}{\frac{3}{4}(|\Gamma(u)| + |\Gamma(v)|)} = \frac{4}{|\Gamma(u)| + |\Gamma(v)|} \ . \quad \square$$

5.2 Progress

Now, we define the *progress* $c(u, b, r)$ for recommending item b to user u in round $r \geq 0$. Informally, the idea of the progress value is to employ the money paid during the execution to bound the expected number of edges removed per round.

Consider any round r and let $w_r(z)$ denote the *wealth* of node $z \in U_r \cup V_r$, where wealth refers to the amount of money z has in round r. In the case of revealing a 0-entry, the progress indicates the number of removed edges and the money that is paid to the nodes adjacent to the removed edges. When finding a 1-entry and revealing the connection between u and $\pi(u)$, the progress indicates the number of removed edges minus the money already paid to u and $\pi(u)$. Let $\Gamma_r(u) = \{v \in V_r \mid (u, v) \in E_r\}$ and $\Gamma_r^b(u)$ denote the neighbors of u that like item b, i.e., $\Gamma_r^b(u) = \{v \in V_r \mid (u, v) \in E_r \wedge v(b) = 1\}$. Then, for entry $u(b)$ revealed in round r, the progress is given by

$$c(u,b,r) = \begin{cases} \sum_{v \in \Gamma_r^b(u)} 5 & \text{if } u(b) = 0 \\ |\Gamma_r(u)| + |\Gamma_r(\pi(u))| - (w_r(u) + w_r(\pi(u))) & \text{if } u(b) = 1 \ . \end{cases}$$

An illustration of the wealth and progress concepts is given in Figure 4. In the example given in Figure 4, revealing entry $u_2(1) = 1$, denoted by x, has a progress value of $|\Gamma(u_2)| + |\Gamma(v_4)| - (w(u_2) + w(v_4)) = 5 + 2 - 0 - 6 = 1$ and revealing entry $u_4(5) = 0$, denoted by y, yields a progress of $\sum_{v_3, v_5} 5 = 10$.

Next, we show that the total progress value counted from the first round up to any round r is never smaller than the number of edges removed from G within the first r rounds. We denote an execution of an algorithm until round r by $\mathcal{E}_r = (u^1, b^1), (u^2, b^2), \ldots, (u^{r-1}, b^{r-1})$, where u^i corresponds to the user selected in round $i < r$ and similarly for item b^i.

Lemma 3. *For any round r and execution $\mathcal{E}_r$, it holds that $\sum_{i=0}^{r-1} c(u^i, b^i, i) \geq |E_0| - |E_r|$.*

Proof. Let I_0 denote the set of indices $i < r$ such that $u^i(b^i) = 0$ and I_1 analogously for indices such that $u^i(b^i) = 1$. Let W_r denote the amount of money paid until round r, i.e.,

$$W_r = \sum_{i \in I_0} \sum_{v \in \Gamma_i^{b^i}(u^i)} 4 \ .$$

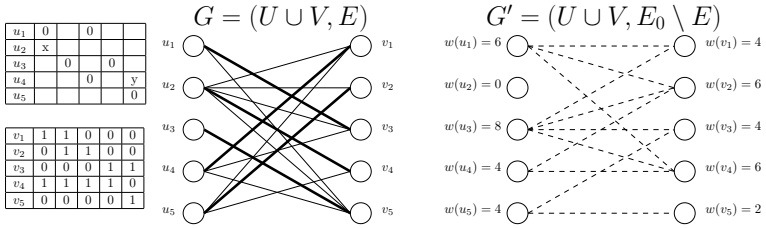


Fig. 4: In graph G' the dashed lines indicate the edges removed from the consistency graph during the execution of an algorithm. The wealth of each node is illustrated next to the corresponding node. The bold lines denote the underlying permutation π that connects users in U with their preference vectors in V.

Now, we can write

$$\sum_{i \in I_0} c(u^i, b^i, i) = W_r + \sum_{i \in I_0} \sum_{v \in \Gamma_i^{b^i}(u^i)} 1 = W_r + \sum_{i \in I_0} |\Gamma_i^{b^i}(u^i)| \ ,$$

where $|\Gamma_i^{b^i}(u^i)|$ equals to the number of edges removed from the consistency graph in round i, i.e., $|\Gamma_i^{b^i}(u^i)| = |E_i| - |E_{i+1}|$.

Furthermore, we can bound the progress in rounds in I_1 by

$$\begin{aligned} \sum_{i \in I_1} c(u^i, b^i, i) &= \sum_{i \in I_1} |\Gamma_i(u^i)| + |\Gamma_i(\pi(u^i))| - (w_i(u^i) + w_i(\pi(u^i))) \\ &\geq \left(\sum_{i \in I_1} |\Gamma_i(u^i)| + |\Gamma_i(\pi(u^i)| \right) - W_r \ , \end{aligned}$$

where $|\Gamma_i(u^i)| + |\Gamma_i(\pi(u^i))| = |E_i| - |E_{i+1}|$. Combining the sums from above we get that

$$\begin{aligned} \sum_{i \in I_0 \cup I_1} c(u^i, b^i, i) &\geq W_r + \sum_{i \in I_0} |\Gamma_i^{b^i}(u)| + \left(\sum_{i \in I_1} |\Gamma_i(u^i)| + |\Gamma_i(\pi(u^i)| \right) - W_r \\ &= \sum_{i \in I_0} |\Gamma_i^{b^i}(u)| + \left(\sum_{i \in I_1} |\Gamma_i(u^i)| + |\Gamma_i(\pi(u^i)| \right) \\ &= \sum_{i=0}^{r-1} |E_i| - |E_{i+1}| = |E_0| - |E_r| \ . \end{aligned}$$

□

The last thing we need to show is an upper bound on the expected progress per round for any round r. For ease of notation, we omit the round index for the rest of the section. In addition, we identify $c(u, b)$ with a random variable that equals to the progress gained by revealing entry (u, b).

Lemma 4. *Let b^* be the most popular item among the set U of unsatisfied users. Then for any user $u \in U$ and item b, $\mathbb{E}[c(u,b)] \leq 5|b^*|$.*

Proof. Consider any user $u \in U$ and any item b. We partition the event space into three disjoint parts according to the outcome of querying user u for item b and show that for each part, $\mathbb{E}[c(u,b)] \leq 5|b^*|$. First, consider the case where a 0-entry is revealed. By definition, we get $\mathbb{E}[c(u,b) \mid u(b) = 0] \leq 5|b^*|$. Furthermore,

$$\mathbb{E}\big[c(u,b) \mid (u(b) = 1) \wedge \big(|\Gamma(u)| + |\Gamma(\pi(u))| \leq 4|U|/3\big)\big] \leq 0 \ ,$$

since $w(u) + w(\pi(u)) \geq 4|U|/3$ given that $|\Gamma(u)| + |\Gamma(\pi(u))| \leq 4|U|/3$.

Let us then consider the third part of the event space, where $u(b) = 1$ and $|\Gamma(u)| + |\Gamma(\pi(u))| > 4|U|/3$ and let us denote this event by B. Let

$$E' = \{(u,v) \in E \mid |\Gamma(u)| + |\Gamma(\pi(u))| > 4|U|/3)]\}$$

and $\hat{\Gamma}(u) = \{v \in \Gamma(u) \mid (u,v) \in E' \wedge v(b) = u(b) = 1\}$. Then, by Lemma 2,

$$\begin{aligned}\mathbb{E}[c(u,b) \mid B] &\leq \sum_{v \in \hat{\Gamma}(u)} (|\Gamma(u)| + |\Gamma(v)|) \cdot \mathbb{P}[A(u,v) \mid G] \\ &\leq \sum_{v \in \hat{\Gamma}(u)} (|\Gamma(u)| + |\Gamma(v)|) \frac{4}{|\Gamma(u)| + |\Gamma(v)|} = \sum_{v \in \hat{\Gamma}(u)} 4 \leq 4|b^*| \ ,\end{aligned}$$

where $|b^*| \geq |b| \geq |\hat{\Gamma}(u)|$, since b^* is the most popular item. Since the three aforementioned parts span the whole probability space, $\mathbb{E}[c(u,b)]$ is bounded by the maximum of the three cases and thus, the claim follows. □

Theorem 1. *Let $R \subseteq U_0$ be a set of users and b^* the most popular item among these users. Any algorithm requires at least $|R|^2/(5|b^*|)$ queries to users in R in expectation to satisfy all users in R.*

Proof. Consider only users in R and let $v_1, \ldots, v_{|R|} = V_R \subseteq V$ be the corresponding preference vectors. Since each user $u \in R$ initially has $|R|$ edges connected to users in R, there are $|R|^2$ edges in total.

Recall that satisfied users are removed from the consistency graph. Thus, when all users in R are satisfied, the set of edges in the consistency graph is empty. By Lemma 4 and by linearity of expectation at least $|R|^2/6|b^*|$ queries are needed before the progress value is greater than $|R|^2$ in expectation. By Lemma 3, the progress value gives an upper bound on the number of edges removed. Therefore, the number of queries needed before all users are satisfied is at least $\frac{|R|^2}{5|b^*|}$. □

6 The Greedy mssc Algorithm

The goal of this section is to show that the greedy algorithm for the **mssc** problem provides an $\mathcal{O}(1)$-approximation for the anonymous recommendation problem. In particular, the goal is to establish the following theorem.

Algorithm 1 Greedy **mssc** algorithm

```
while |U| > 0 do
  Choose the most popular item b among users in U.
  for all u ∈ U do
    Recommend b to u.
  end for
end while
```

Theorem 2. *The greedy* **mssc** *algorithm provides a* 20*-approximation for the anonymous recommendation problem.*

Our proof follows the steps of the 4-approximation proof by Feige et al. [9] The crucial difference between their proof and ours is that in the oblivious case of **mssc**, where each user is recommended items according to the same fixed order, it is clear that every algorithm requires $\Omega(|R|^2/|b|)$ rounds to to satisfy all users in $R \subseteq U_0$ given that b is the most popular item among users in U. In our case, we use Theorem 1 to provide a similar observation in terms of expectation. For the sake of completeness, we dedicate this section to give a detailed proof that the existing tools used to prove the 4-approximation of the greedy **mssc** algorithm can be used for our purposes with a few modifications. The pseudo-code for the greedy algorithm is given in Algorithm 1.

We refer to the iterations of the while loop of the greedy algorithm as *steps* and label them by the positive integers. Note that each step of Algorithm 1 consists of $|U|$ many rounds according to our model, where U is the set of unsatisfied users in the beginning of the step. Let X_i be the set of users satisfied in step i and let R_i be the set of unsatisfied users prior to step i. The cost of the greedy algorithm is given by $\sum_i i|X_i| = \sum_i |R_i|$.

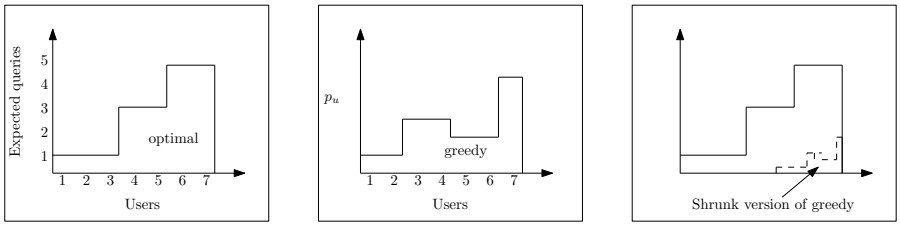
We define the price of user $u \in X_i$ to be $p_u = |R_i|/|X_i|$. Then we set

$$\textbf{price} = \sum_{u \in U} p_u = \sum_i \sum_{u \in X_i} p_u = \sum_i |X_i| \frac{|R_i|}{|X_i|} = \sum_i |R_i| \ ,$$

which shows that **price** is equal to the cost of the greedy algorithm.

We model the solutions for both greedy algorithm and an optimal algorithm OPT for the anonymous recommendation problem by the following diagrams. Consider first the greedy algorithm. There are $|U_0|$ columns, one for each user in the input. The users are ordered from left to right by the order in which they are satisfied by the greedy algorithm. The height of each column is the price p_u for the corresponding user u. The area under the histogram equals therefore to **price**.

Similarly, we have a diagram for OPT. Again, there is a column for every user. In the case of OPT however, the height of each column corresponds to the expected number of queries made to the corresponding user. Again, the area of the diagram of the optimal algorithm is equal to the expected cost of OPT. The columns are ordered in an ascending order by the height of each column. The diagrams are illustrated in Figure 5.



(a) (b) (c)

Fig. 5: The diagram corresponding to the solution of the optimal algorithm is given on the left. The height of each column in the diagram of the optimal solution corresponds to the expected number of queries to the corresponding user. The diagram of the greedy algorithm is shown in the middle. In this diagram, the height of each column is the price of the corresponding user. The shrinking and aligning of the diagram of the greedy algorithm into the diagram of the optimal algorithm is shown on the right.

As the next step, we show that the area of the greedy diagram is at most 20 times the size of the optimal diagram. To show this, we shrink the diagram of the greedy algorithm by shrinking the height of each column by a factor of 10 and the width by 2. Then, we align the shrunk version of the diagram to the right of the other diagram corresponding to the solution of OPT. In other words, the diagram of the greedy algorithm now occupies the space of columns of OPT from $|U_0|/2+1$ up to $|U_0|$ (where $|U_0|$ is assumed w.l.o.g. to be even).

Consider any point q' in the diagram of the greedy algorithm. Let u be the user that corresponds to this column and let i be the step when this user is satisfied. Thus, the height of point q' is at most $|R_i|/|X_i|$ and the distance to the right hand boundary is $|R_i|$. The shrinking maps q' to another point q, where the height h of q satisfies $h \leq |R_i|/(10|X_i|)$ and the distance to the right boundary r satisfies $r \leq |R_i|/2$.

We now show that q lies within the histogram of OPT. To prove this, we need to show that there is at least one user in the first $|R_i|/2$ users who is queried at least $|R_i|/(10|X_i|)$ times. Consider now only the first $|R_i|/2$ users in R_i and denote these users by R_i^f. By Theorem 1, it takes at least

$$\frac{|R_i^f|^2}{5|X_i|} = \frac{\left(\frac{|R_i|}{2}\right)^2}{5|X_i|} = \frac{|R_i|^2}{20|X_i|}$$

rounds in expectation to satisfy all of these users. Therefore, there is at least one user $u \in R_i^f$ that is queried at least

$$\frac{\frac{|R_i|^2}{20|X_i|}}{|R_i^f|} = \frac{\frac{|R_i|^2}{20|X_i|}}{\frac{|R_i|}{2}} = \frac{|R_i|}{10|X_i|}$$

times in expectation. Since the users are ordered according to the number of queries, every user $v \in R_i \setminus R_i^f$ is queried at least $|R_i|/(10|X_i|)$ times. Thus q indeed lies within the histogram of the optimal algorithm, yielding Theorem 2.

References

1. Susanne Albers. A Competitive Analysis of the List Update Problem with Lookahead. *Theoretical Computer Science*, 197:95–109, 1998.
2. Noga Alon, Baruch Awerbuch, Yossi Azar, and Boaz Patt-Shamir. Tell Me Who I Am: an Interactive Recommendation System. In *Proceedings of the 18th Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 261–279, 2006.
3. Baruch Awerbuch, Boaz Patt-Shamir, David Peleg, and Mark R. Tuttle. Improved Recommendation Systems. In *Proceedings of the 16th Symposium on Discrete Algorithms (SODA)*, pages 1174–1183, 2005.
4. Yossi Azar and Iftah Gamzu. Ranking with Submodular Valuations. In *Proceedings of the 22nd Symposium on Discrete Algorithms (SODA)*, pages 1070–1079, 2011.
5. Amotz Bar-Noy, Mihir Bellare, Magnús M. Halldórsson, Hadas Shachnai, and Tami Tamir. On Chromatic Sums and Distributed Resource Allocation. *Information and Computation*, 140(2):183–202, 1998.
6. Sushil Bikhchandani, Sven de Vries, James Schummer, and Rakesh Vohra. An Ascending Vickrey Auction for Selling Bases of a Matroid. *Operations Research*, 59(2):400–413, 2011.
7. Brian Dean, Michel Goemans, and Jan Vondrák. Approximating the Stochastic Knapsack Problem: The Benefit of Adaptivity. *Mathematics of Operations Research*, 33:945–964, 2008.
8. Petros Drineas, Iordanis Kerenidis, and Prabhakar Raghavan. Competitive Recommendation Systems. In *Proceedings of the 34th Symposium on Theory of Computing (STOC)*, pages 82–90, 2002.
9. Uriel Feige, László Lovász, and Prasad Tetali. Approximating Min Sum Set Cover. *Algorithmica*, 40:219–234, 2004.
10. Michel Goemans and Jan Vondrák. Stochastic Covering and Adaptivity. In *Proceedings of the 7th Latin American Conference on Theoretical Informatics (LATIN)*, pages 532–543, 2006.
11. Sally A. Goldman, Robert E. Schapire, and Ronald L. Rivest. Learning Binary Relations and Total Orders. *SIAM Journal of Computing*, 20(3):245 – 271, 1993.
12. Daniel Golovin and Andreas Krause. Adaptive Submodularity: Theory and Applications in Active Learning and Stochastic Optimization. *Journal of Artificial Intelligence Research (JAIR)*, 42:427–486, 2011.
13. Edward Grove. Online Bin Packing with Lookahead. In *Proceedings of the 6th Symposium on Discrete Algorithms (SODA)*, pages 430–436, 1995.
14. Anupam Gupta, Viswanath Nagarajan, and R. Ravi. Approximation Algorithms for Optimal Decision Trees and Adaptive TSP Problems. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 690–701. Springer-Verlag, 2010.
15. Rachel Kranton and Deborah Minehart. A Theory of Buyer-Seller Networks. *American Economic Review*, 91:485–508, 2001.
16. Zhen Liu, Srinivasan Parthasarathy, Anand Ranganathan, and Hao Yang. Near-Optimal Algorithms for Shared Filter Evaluation in Data Stream Systems. In

Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pages 133–146, 2008.

17. Kamesh Munagala, Shivnath Babu, Rajeev Motwani, and Jennifer Widom. The Pipelined Set Cover Problem. In *Proceedings of the 10th International Conference on Database Theory (ICDT)*, pages 83–98, 2005.
18. Aviv Nisgav and Boaz Patt-Shamir. Finding Similar Users in Social Networks: Extended Abstract. In *Proceedings of the 21st Annual Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2009.
19. Jara Uitto and Roger Wattenhofer. On Competitive Recommendations. In *Proceedings of the 24th International Conference on Algorithmic Learning Theory (ALT)*, pages 83–97., 2013. Invited to a special issue of Theoretical Computer Science.