Guangyi Zhang[†]

Nikolaj Tatti[‡]

Aristides Gionis[§]

Abstract

Maximizing submodular functions have been studied extensively for a wide range of subset-selection problems. However, much less attention has been given to the role of submodularity in sequence-selection and ranking problems. A recentlyintroduced framework, named maximum submodular ranking (MSR), tackles a family of ranking problems that arise naturally when resources are shared among multiple demands with different budgets. For example, the MSR framework can be used to rank web pages for multiple user intents. In this paper, we extend the MSR framework in the streaming setting. In particular, we consider two different streaming models and we propose practical approximation algorithms. In the first streaming model, called *function arriving*, we assume that submodular functions (demands) arrive continuously in a stream, while in the second model, called *item* arriving, we assume that items (resources) arrive continuously. Furthermore, we study the MSR problem with additional constraints on the output sequence, such as a matroid constraint that can ensure fair exposure among items from different groups. These extensions significantly broaden the range of problems that can be captured by the MSR framework. On the practical side, we develop several novel applications based on the MSR formulation, and empirically evaluate the performance of the proposed methods.

1 Introduction

Submodular set functions capture a "diminishing-returns" property that is present in many real-world phenomena [16]. Submodular functions are popular as they admit a rich toolbox of optimization techniques developed in the literature. Examples of submodular functions used in practical problems include document summarization [19], viral marketing in social networks [12], social welfare maximization [27], and many other. The majority of existing submodularity-based problem formulations are restricted to selecting a *subset of items*, and completely disregard the effect of *item order*. However, the order of items plays an important role in many applications. In this paper, we investigate a versatile approach to ranking items within the submodularity framework.

Creating a sequence of resources to be shared among multiple demands appears in a broad range of applications. For example, when ranking web pages in response to a user query we want to cater for multiple user intents; when creating a live stream of music content shared among a group we want to satisfy the tastes of all listeners; and when selecting advertising for a screen on public display we want it to be relevant for all passengers. Typically, each demand has an individual maximum budget, e.g., in the previous scenario, the budget models the number of web pages that a user is expected to browse. The main challenge in these problems is to find a (partial) ranking of resources that best satisfies multiple demands with different budgets.

More concretely, the maximum submodular ranking (MSR) formulation [28] deals with resources and demands. A resource is referred to as an item in a universe set V. Demands require resources, and the utility of a demand for a set of resources is characterized by a non-decreasing submodular set function $f: 2^V \to \mathbb{R}_+$. The budget of a demand is represented by a cardinality constraint, i.e., the maximum number of items its corresponding function is allowed to take. The objective is to find a (partial) sequence of items to maximize the total utility, i.e., the sum of function values. A formal problem definition is introduced in Section 3.

To capture a wider range of problems and increase the versatility of the framework, in this paper, we extend the MSR problem in two natural streaming models, which we name *function arriving* and *item arriving*. In the *function-arriving model*, we assume that demands arrive continuously in an online fashion and one is unaware of the type and/or volume of future demands. This model is common in practice; for example, new audience may join a live stream in any time. In the *item-arriving model*, we assume that items arrive in a stream, and we have to output a sequence of items in *one pass* and with limited memory after seeing all the items in the stream. In other words, we need to process each item immediately after its arrival. This model offers a

^{*}This research is supported by the Academy of Finland projects MALSOME (343045), AIDA (317085) and MLDB (325117), the ERC Advanced Grant REBOUND (834862), the EC H2020 RIA project SoBigData++ (871042), and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

[†]KTH Royal Institute of Technology. guaz@kth.se

[‡]HIIT, University of Helsinki. nikolaj.tatti@helsinki.fi

[§]KTH Royal Institute of Technology. argioni@kth.se

way to handle the MSR problem when items are arriving continuously, or are too many to be loaded into memory.

For both of the streaming models we consider we propose practical approximation algorithms. Besides, we study the MSR problem with additional constraints on the output sequence, such as a matroid constraint (see Section 3) that can ensure fair exposure among items from different groups.

On the practical side, we propose novel applications based on the MSR formulation. We highlight here an application on *progressively-diverse* personalized recommendation, while many other interesting ones are discussed in Section 6, including live streaming and catalogued viral marketing. A popular approach to personalized recommendation [22] is to select a succinct subset S of items that maximizes a weighted sum (with a trade-off parameter λ) of two submodular terms, relevance and diversity. Note that for a given value of the parameter λ , a subset S presents a fixed trade-off between relevance and diversity. On the other hand, a user's need for diversity may be better served in an adaptive manner. For example, a target user may appreciate a recommended list having the most relevant items at the beginning and becoming progressively diverse down the list. This requirement can be satisfied via an MSR formulation, by maximizing a weighted sum of multiple such functions, each with an increasing tradeoff value λ . See Section 3 for a formal definition.

Our contributions in this paper are summarized as follows.

- We study the maximum submodular ranking (MSR) problem in two different streaming models, and devise approximation algorithms for each model.
- In the function-arriving streaming model, we show that a simple greedy algorithm yields a 2 approximation for the MSR problem, if an item is allowed to be used multiple times. This approximation ratio is tight for the greedy algorithm. We also show that the problem is inapproximable if every item can be used at most once.
- We propose a novel reduction that maps the ranking problem into a constrained subset-selection problem subject to a bipartite-matching constraint. An immediate consequence is that there exist approximation algorithms for the MSR problem subject to a general *p*-matroid constraint on the output sequence. Another consequence is that we can obtain efficient approximation algorithms for the MSR problem in the item-arriving streaming model.
- We apply the enhanced MSR framework to several novel real-life applications, and empirically evaluate

the performance of the proposed algorithms.

The rest of the paper is organized as follows. We discuss related work in Section 2. A formal problem definition is introduced in Section 3. We present approximation algorithms for the function-arriving MSR problem and the item-arriving MSR problem in Sections 4 and 5, respectively. Our empirical evaluation is conducted in Section 6, followed by concluding remarks in Section 7. Missing proofs and further experimental details are deferred to the supplementary materials. Our implementation is made publicly available.¹

2 Related work

Submodularity for sequences. The MSR problem was proposed by Zhang et al. [28], and it was later shown to be a special case of *ordered submodularity*, introduced by Kleinberg et al. [14]. In particular, both formulations avoid an unnatural *postfix monotonicity* property, which is required in prior formulations [1, 26, 29]. Postfix monotonicity requires a non-decreasing function value after prepending an arbitrary item at the front of a sequence. Additionally, the MSR problem can be seen as a dual problem to the *submodular ranking* problem [2], which aims to minimize the total cover time of all functions in the absence of individual budgets. No streaming extension has been known for the MSR problem.

Bipartite matching. Bipartite matching and its many variants have been extensively studied in the literature [21]. The offline weighted bipartite matching can be solved exactly, e.g., via a maximum-flow formulation, while the best-known approximation ratio for the online variant is achieved by Fahrbach et al. [6]. It is known that many popular variants can be treated as a special case of the *submodular social welfare* problem [17]. Similar to our reduction in Section 5, some assignment or scheduling problems can also be reduced to a subset-selection problem subjective to a bipartitematching constraint [24, 27].

Constrained submodular maximization. In the offline setting, for a *p*-matroid constraint, it is well-known that a greedy algorithm guarantees a *p*-approximation for a modular function and a (p+1)-approximation for a submodular function [10, 15]. Feldman et al. [7] achieve the best-known $(p + \epsilon)$ -approximation for submodular maximization under a *p*-exchange constraint, which is more general then a *p*-matroid. A lower bound of $\Omega(p/\ln p)$ is known for approximating *p*-dimensional matching, which is a special case of maximizing a modular function over a *p*-matroid [11].

¹https://github.com/Guangyi-Zhang/ subm-ranking-on-the-fly

In regards to the streaming setting, Levin and Wajc [18] offer a $3 + 2\sqrt{2} \approx 5.828$ approximation subject to a matching constraint. Under a *p*-matroid. a 4*p*-approximation is obtained by Chakrabarti and Kale [5], which is inspired by the modular variant in Badanidiyuru [3]. In terms of lower bounds, the analysis in Badanidiyuru [3] is shown to be optimal for any online algorithm (i.e., a streaming algorithm that maintains only a feasible solution at any moment). Besides, a 2.692approximation is impossible even subject to a bipartite matching constraint, and there exists evidence that the lower bound can be as high as 3-approximation [9]. For p-matroid constraint, a lower bound of p has been proven for any streaming algorithm with sub-linear memory; moreover, any logarithmic improvement over the best-known 4p-approximation requires memory super polynomial in p [9].

3 Problem definition

In this section we present the maximum submodular ranking (MSR) problem in two different streaming models. Afterwards, we introduce a formulation for progressivelydiverse personalized recommendation. Prior to that, we briefly review notions of submodularity and matroids.

Submodularity. Given a set V, a function $f: 2^V \to \mathbb{R}_+$ is called *submodular* if for any $X \subseteq Y \subseteq V$ and $v \in V \setminus Y$, it holds $f(v \mid Y) \leq f(v \mid X)$, where $f(v \mid Y) = f(Y + v) - f(Y)$ is the marginal gain of v with respect to set Y. A function f is called *modular* if $f(X)+f(Y) = f(X \cup Y)$ for any $X \cap Y = \emptyset$. A function f is called *non-decreasing* if for any $X \subseteq Y \subseteq V$, it holds $f(Y) \geq f(X)$. Without loss of generality, we can assume that function f is normalized, i.e., $f(\emptyset) = 0$.

Matroid. For a set V, a family of subsets $\mathcal{M} \subseteq 2^V$ is called a *matroid* if it satisfies the following two conditions: (1) downward closeness: if $X \subseteq Y$ and $Y \in \mathcal{M}$, then $X \in \mathcal{M}$; (2) augmentation: if $X, Y \in \mathcal{M}$ and |X| < |Y|, then $X + v \in \mathcal{M}$ for some $v \in Y \setminus X$. Two useful special cases are those of *uniform matroid* and *partition matroid*. The former is simply a k-cardinality constraint, i.e., $\mathcal{M} = \{S \subseteq V : |S| \leq k\}$, and the latter consists of multiple cardinality constraints, each placed on a disjoint subset G_{ℓ} of $V = \bigcup_{\ell} G_{\ell}$, i.e., $\mathcal{M} = \{S \subseteq V :$ $|S \cap G_{\ell}| \leq k_{\ell}$, for all $\ell\}$. Given p matroids $\{\mathcal{M}_j\}_{j \in [p]}$, their intersection is called a p-matroid.

We denote by $\sigma(V)$ the set of all sequences formed by items in V. Given a sequence $\pi \in \sigma(V)$, we write π_i for the *i*-th item in π , and $\pi + v$ for the new sequence obtained by appending item v to π . The length of a sequence π is denoted as $|\pi|$. The set of items in π is denoted by $V(\pi) \subseteq V$. A sequence π being a subsequence of another sequence π' is denoted by $\pi \preceq \pi'$. Given an interval w = [s, e], where s, e are integers, we write $\pi[w] = \pi[s : e] = \{\pi_s, \dots, \pi_e\}$. More generally, given a subset of items $R \subseteq V$, we write $\pi[R] = \{\pi_i \mid i \in R\}$.

We are now ready to define the MSR problem [28] and its streaming variants.

PROBLEM 3.1. (MAX-SUBMODULAR RANKING (MSR)) Given a set V of n items, a collection of m nondecreasing submodular functions $F = \{f_i\}_{i \in [m]}$, each associated with an integer k_i , the objective is to find a sequence solving

(3.1)
$$\arg \max_{\pi \in \sigma(V)} \sum_{f_i \in F} f_i(\pi[1:k_i]).$$

If an additional *p*-matroid constraint $\mathcal{M} \subseteq 2^V$ is imposed on items in a feasible sequence π , i.e., $V(\pi) \in \mathcal{M}$, we refer to the problem as MSR*p*. Such a *p*-matroid constraint is useful, for example, to avoid overrepresentation of some group of items in the returned sequence. If the functions in *F* are modular, we refer to the problem as *maximum modular ranking* (MMR).

In the function-arriving streaming model, we observe a set of new functions F_t at time step t, and the objective is to produce a sequence π in real time, that is, to decide irrevocably one item in π at each time step. Note that items that are placed in previous item steps cannot be used anymore. We assume that we observe the new functions F_t before deciding the t-th item at step t, and a function will stay active in subsequent steps after its arrival until it exhausts its budget. It is also possible not to place any item at a step (by introducing dummy items in V). More formally, the MSR problem in the function-arriving model is defined as follows.

PROBLEM 3.2. (FUNCTION-ARRIVING MSR (MSR-F)) Given a set V of n items, and a collection of nondecreasing submodular functions F_t that arrive at the beginning of step t, with arrival time $\tau(f) = t$ and integers k(f) for each $f \in F_t$, the objective is to find a sequence solving

3.2)
$$\arg \max_{\pi \in \sigma(V)} \sum_{t} \sum_{f \in F_t} f(\pi[t:k(f)]),$$

by irrevocably deciding the t-th item π_t at step t.

In contrast, in the item-arriving streaming model, we have full information about the functions that are used. Actually, we further allow each function f_i to "reserve" arbitrary k_i slots $R_i \subseteq [n]$ in a sequence, instead of merely the first k_i slots $[k_i]$ in MSR. When given a sequence, function f_i receives items only from slots R_i . For example, when deciding showtimes in a cinema, a user (f_i) may only be available during weekends or at specific time of a day. The goal of the item-arriving MSR problem is to produce a sequence π after processing all arriving items in one pass and using "small" memory size. In other words, items that are discarded from the memory cannot be used later. If there is a slot in the sequence where no function is available, one is allowed to not place any item. Formally, the MSR problem in the item-arriving streaming model is defined as follows.

PROBLEM 3.3. (ITEM-ARRIVING MSR (MSR-I))

Given a collection of non-decreasing submodular functions $F = \{f_i\}_{i \in [m]}$, each associated with k_i available slots specified by $R_i \subseteq [n]$, and items in V that arrive in a stream, the objective is to find a sequence in

(3.3)
$$\arg \max_{\pi \in \sigma(V)} \sum_{f_i \in F} f_i(\pi[R_i]).$$

In addition, the number of items one can store at any moment depends only on $\{k_i\}$ instead of n.

In the offline setting where all items are in place, we call this variant MSR with availability (MSR-A) problem. Note that when $R_i = [k_i] = \{1, \ldots, k_i\}$, the MSR-A problem becomes equivalent to the original MSR problem. We also note that when there is a single function in F, MSR-I generalizes the problem of streaming submodular maximization for which no streaming algorithm with sublinear memory in n has an approximation ratio better than 2 [8].

Progressively-diverse personalized recommendation. A popular approach to personalized recommendation [22] is to select a succinct subset S of items that maximizes a submodular function of the form

$$f_{\lambda,k}(S) = (1-\lambda) \sum_{v \in S} \operatorname{rel}(v) + \frac{\lambda k}{|V|} \sum_{u \in V} \max_{v \in S} \operatorname{sim}(u, v),$$
(3.4) such that $|S| \leq k$.

Here $\operatorname{rel}(v)$ measures the relevance of an item v to the target user, and $\operatorname{sim}(u, v)$ the similarity between two items u, v. The second term represents one specific notion of diversity (also known as representativeness or global coverage), i.e., for every non-selected item $u \in V$, there exists some item $v \in S$ that is similar enough to u. To create a recommended list that adaptively serves a user's need for diversity, one could maximize a weighted sum of multiple functions $\{f_{\lambda,k}\}$ with increasing trade-off value $\lambda \in [0, 1]$ and cardinality k, so that the later suffix of the list will be dominated by functions with larger λ .

4 Function-arriving MSR

In this section we discuss two scenarios for the functionarriving MSR (MSR-F) problem, depending on whether

Algorithm 1: Greedy algorithm for Functionarriving MSR (MSR-F)

1 Initialize an empty sequence π					
$2 \ F \leftarrow \emptyset$					
3 for $t = 1,$ do					
4	$F \leftarrow F \cup F_t$ // receive functions F_t				
5	$A \leftarrow \{f \in F : k(f) \ge t\}$ // active				
	functions at the t -th step				
6	$v^* \leftarrow \arg\max_{v \in V} \sum_{f \in A} f(v \mid \pi[\tau(f) : t-1])$				
7	$ \ \ \ \ \ \ \ \ \ \ \ \ \ $				
s return π					

items in V can be used at most once, or more than one time. We show that the problem is inapproximable in the former case, and we present a 2-approximation algorithm for the latter case.

To start off our analysis, if the output sequence is constrained to not contain duplicate items, it can be shown that the MSR-F problem is inapproximable, even when all functions are modular. This result, stated below, follows from the inapproximability of the *onlineselection problem*, which aims to select the maximum of an adversarial sequence with no recall [13].

THEOREM 4.1. If items can be used at most once, the MMR-F problem generalizes the online selection problem. Thus, no randomized algorithm guarantees an o(n)-approximation for the MMR-F problem.

Given the inapproximability of MMR-F for the case that the output sequence should not contain duplicates, we proceed to study the problem when items in V can be used multiple times. This assumption is reasonable in many application, for example, a song can be added many times in a playlist — and notice here that unnecessary duplicates are discouraged implicitly as their marginal gain is zero with respect to functions that have included these items already.

When duplicates are allowed in the output sequence, we prove that a simple greedy algorithm returns a solution with a 2-approximation guarantee. The greedy, which is displayed as Algorithm 1, selects the most beneficial item with respect to the current set of "active" functions at each step. A function f is called *active* if it has not exhausted its item budget k(f) up to that point.

THEOREM 4.2. If items in V can be used multiple times in the output sequence, Algorithm 1 yields a 2approximation solution for the MSR-F problem.

The approximation guarantee of the greedy can be shown to be tight, as the lower bound provided by Zhang et al. [28] applies to our case, as well. In particular, since the MSR-F problem generalizes the MSR problem, by letting all functions to arrive at the beginning, we obtain the following result.

REMARK 4.1. (ZHANG ET AL. [28]) If an item can be used multiple times in the output sequence, the 2approximation solution obtained by Algorithm 1 is tight for the MSR-F problem.

In the rest of this section, we prove Theorem 4.2, and the proof of Theorem 4.1 is deferred to Section A.1.

Proof. [Proof of Theorem 4.2] We write $A_t = \{f \in \bigcup_{t' \leq t} F_{t'} : k(f) \geq t\}$ for the set of active functions at step t. We denote by π the sequence produced by Algorithm 1 with objective value ALG, and by π^* the optimal sequence with objective value OPT.

By the greedy selection criterion, we know that for any arbitrary item $v \in V$, it holds that (4.5)

$$\sum_{f \in A_t}^{(4,J)} f(\pi_t \mid \pi[\tau(f) : t-1]) \ge \sum_{f \in A_t} f(v \mid \pi[\tau(f) : t-1]).$$

To simplify the notation, let us write $\pi[f]$ to mean $\pi[\tau(f):k(f)]$.

It is easy to see that ALG is equal to the sum over t of the left-hand side in Equation (4.5), implying

$$\begin{aligned} \text{ALG} &= \sum_{t} \sum_{f \in A_{t}} f(\pi_{t} \mid \pi[\tau(f) : t - 1]) \\ &\stackrel{(a)}{\geq} \sum_{t} \sum_{f \in A_{t}} f(\pi_{t}^{*} \mid \pi[\tau(f) : t - 1]) \\ &= \sum_{f \in F} \sum_{t = \tau(f)}^{k(f)} f(\pi_{t}^{*} \mid \pi[\tau(f) : t - 1]) \\ &\stackrel{(b)}{\geq} \sum_{f \in F} \sum_{t = \tau(f)}^{k(f)} f(\pi_{t}^{*} \mid \pi[f]) \\ &\stackrel{(c)}{\geq} \sum_{f \in F} f(\pi^{*}[f] \mid \pi[f]) \\ &= \sum_{f \in F} f(\pi^{*}[f] \cup \pi^{*}[f]) - f(\pi[f]) \\ &= \sum_{f \in F} f(\pi^{*}[f]) - f(\pi[f]) = \text{OPT} - \text{ALG}, \end{aligned}$$

where inequality (a) is by Eq. (4.5), and inequalities (b) and (c) are due to submodularity.

5 Item-arriving MSR

In this section, we present the reduction that turns the ranking problem into a subset selection problem subject to a bipartite matching constraint, and its rich consequences. A summary of approximation ratios in different settings is displayed in Table 1, using algorithms provided in the citations.

THEOREM 5.1. For any integer $p \ge 1$, the MSR-Ap problem is an instance of maximizing a non-decreasing submodular function subject to a (p+1)-matroid.

As an immediate consequence of Theorem 5.1, the item-arriving MSR (MSR-I) can be solved by streaming algorithms for constrained submodular maximization.

COROLLARY 5.1. For any integer $p \ge 1$, the MSR-Ip problem is an instance of maximizing a non-decreasing submodular function subject to a (p+1)-matroid in one pass while remembering $\mathcal{O}(\sum_i k_i)$ items at any moment.

Moreover, when functions are modular, we can obtain a stronger result.

COROLLARY 5.2. For any integer $p \ge 1$, the MMR-Ap problem is an instance of maximizing a modular function subject to a (p+1)-matroid. In particular, the MMR-A problem is an instance of maximum weighted bipartite matching.

Proof. [Proof of Theorem 5.1] The main idea of the reduction is to create an extended universe set V', i.e.,

5.6)
$$V' = \{(v,t) : v \in V, t \in [n]\}$$

which can be seen as the edges in a complete bipartite between items L = V and ranks R = [n]. Let us define $V(S') = \{v : (v,t) \in S'\}$ to be the projection of S'onto V. Let us also write $X_v = \{(v,t) : t \in [n]\}$ and $Y_t = \{(v,t) : v \in V\}.$

Suppose we are given a *p*-matroid $\mathcal{M} \subseteq 2^V$ over *V*. Define $\mathcal{M}' = \mathcal{A} \cap \mathcal{B} \cap \mathcal{C}$, where

$$\mathcal{A} = \{ S' \subseteq V' : V(S') \in \mathcal{M} \}, \\ \mathcal{B} = \{ S' \subseteq V' : |S' \cap X_v| \le 1, \text{ for all } v \in V \}, \\ \mathcal{C} = \{ S' \subseteq V' : |S' \cap Y_t| \le 1, \text{ for all } t \in [n] \}.$$

Here, \mathcal{B} forces that an item appears only once and \mathcal{C} forces that only one item appears at time t. Consequently, a feasible sequence π can be written as a subset of V' that satisfy \mathcal{M}' .

On the other hand, a feasible subset $S' \subseteq V'$ can be transformed to a sequence by ordering items in V(S')according to their associated ranks in S'. If S' consists of non-consecutive ranks, one can insert dummy items (or shifting items forward in MSR) to obtain a sequence, with no decrease in the objective function.

We claim that \mathcal{M}' is a (p+1)-matroid. We will prove the claim by arguing that $\mathcal{A} \cap \mathcal{B}$ is a *p*-matroid

Table 1: Summary of approximation ratios for MSR-A and MSR-I

	unconstrained	<i>p</i> -matroid
MMR-A	exact	p+1 (Korte and Hausmann [15])
MSR-A	$2 + \epsilon$ (Feldman et al. [7])	$p + 1 + \epsilon$ (Feldman et al. [7])
MMR-I	1/0.5086 (Fahrbach et al. [6])	$2(p+1+\sqrt{(p+1)p})-1$ (Badanidiyuru [3])
MSR-I	5.828 (Levin and Wajc [18])	4(p+1) (Chakrabarti and Kale [5])

(by Lemma A.1 in Section A.2). Then $\mathcal{A} \cap \mathcal{B} \cap \mathcal{C}$ is a (p+1)-matroid because \mathcal{C} is a matroid.

What is left is to show that the objective function for MSR-A (Equation 3.3) is non-decreasing and submodular with respect to the new universe set V'. Showing non-decreasing is obvious, so we only elaborate on submodularity. Recall that $R_i \subset [n]$ consists of available slots for function f_i . Let us define $R'_i = \{(v,t) : v \in V, t \in R_i\}$. The objective function can be now written as $g(S) = \sum_{f_i \in F} f_i(V(S \cap R'_i))$. For any subset $S \subseteq W \subseteq V'$, the marginal gain $g(\cdot | S)$ of including an item (v,t) into S is

$$g((v,t) \mid S) = \sum_{f_i \in F: t \in R_i} f_i(v \mid V(S \cap R'_i))$$

$$\geq \sum_{f_i \in F: t \in R_i} f_i(v \mid V(W \cap R'_i)) = g((v,t) \mid W),$$

since $V(S \cap R'_i) \subseteq V(W \cap R'_i)$ and submodularity of each f_i . Hence, the MSR-Ap problem can be cast as an instance of non-decreasing submodular maximization under a (p+1)-matroid.

6 Experiments

We evaluate our methods on novel use cases that motivate the MSR-F and MSR-I problems. For each use case, we simulate a concrete task using real-life data, and empirically evaluate the performance of the proposed algorithms. A summary of the datasets can be found in Table 2. An examination on the running time is deferred to Section A.4. Our implementation has been made publicly available.²

6.1 Function-arriving MSR We present two use cases for the MSR-F problem, live streaming and catalogued viral marketing, and we evaluate our methods on relevant datasets.

Algorithms. The proposed greedy algorithm in Algorithm 1 is termed Greedy. Other baselines include Random, which picks a random item at every step, and TopK, which selects the top-k items repetitively — in

 Table 2: Datasets statistics

Dataset	n = V	m = F
Music [4]	$61 \ 415$	10000
Github social network [25]	37 700	100
Sogou web pages [20]	725	$1 \ 017$
Twitter words [23]	10000	8

our use case this corresponds to playing the most popular songs in a loop. Note that TopK is *omniscient* as it requires information about the item popularity in advance.

Live streaming. One increasingly popular application on the internet is live streaming, where a live streamer performs various shows continuously, while the audience may join or leave any time. More concretely, we consider music live streaming, where the live streamer plays songs continuously. To simulate this application, we use the Million Song Dataset [4], consisting of triples representing a *user*, *song*, and *play count*. We assume that a user *likes* a song if it is played more than once. We define *user utility* to be fractional coverage of the liked songs, which is a submodular function. We set a random budget for each user between 1 to a maximum-budget parameter, i.e., how many songs the user will listen to. Besides, every user is given a random arrival time over a long horizon. Our goal is decide a sequence of songs in real time that maximizes total user utility.

Catalogued viral marketing. For viral-marketing applications in social networks, the goal is to identify a small set of seed nodes who can influence many other users. For popular diffusion models, the number of influenced nodes is a submodular function of the seed set [12]. Here, we introduce a generalization of this classic result to cope with multiple marketing demands simultaneously, where each demand is only interested in reaching a specific group of users. For example, an advertiser may want to influence female users, while another may want to influence users near a specific city. Each demand provides a number of product samples to seed nodes, with the hope to advertise their products by word of mouth.

²https://github.com/Guangyi-Zhang/ subm-ranking-on-the-fly



(b) Catalogued viral marketing in Github

Figure 1: MSR-F. Each demand is given a random arrival time, and a random budget between 1 to a parameter of maximum budget. (a) Utility of a user demand is $f(S) = |S \cap S_{\text{like}}| / |S_{\text{like}}|$, where S_{like} is the set of songs the user likes. (b) Utility of a marketing demand is $f(S) = |N(S) \cap S_{\text{g}}| / |S_{\text{g}}|$, where N(S) is neighborhood of S and S_{g} is the target group of nodes in the network.

As a use case for our experimental evaluation, we consider a platform designed to help with these marketing demands, and assume that a package of samples of different products can be sent to one seed node at each time step. As the marketing demands arrive in real time, we aim to catalogue unfinished demands by identifying a seed node that is beneficial to all of them. Note that samples from one demand should be distributed as soon as possible after its arrival to the outgoing packages. To simulate this use case, we use the GitHub social network [25]. We consider 100 demands, each targeting a random subset of the network, together with a random number of samples from 1 to a maximumbudget parameter, and a random arrival time. Our goal is to maximize the total utility of demands by sending a package to one carefully chosen seed node at each time step.

Results. The result of the simulation is shown in Figure 1, in which each data point represents an average



(a) Web page ranking in Sogou



(b) Finding synonyms to "Trump" in Twitter

Figure 2: MSR-I. (a) Utility of a user intent is $f(S) = |S \cap S_{rel}|/|S_{rel}|$, where S_{rel} is the set of relevant pages. Each intent is given a random budget between 1 to a parameter of maximum budget. (b) Given a sequence, its relevance and diversity (Equation 3.4) is measured at every prefix.

of three runs, each with a different random seed. For the music-streaming task (Figure 1(a)), the Greedy algorithm outperforms all other baselines by a large margin. This suggests that for users with diverse preferences in songs, an algorithm like Greedy, which can adapt to the need of current active users, is required for good performance. In the catalogued viral-marketing task (Figure 1(b)), the Greedy algorithm continues to achieve the best performance. However, several baselines from TopK come closer to Greedy as the demand budget increases. This signifies the existence of a group of influencers who can collectively reach the most users in the Github network.

6.2 Item-arriving MSR-I. Next we will present experiments for MSR-I as well as for progressively-diverse personalized recommendation.

Algorithms. We adopt the state-of-the-art streaming algorithm in Chakrabarti and Kale [5] as our algorithm, which greedily assigns each arriving item to one of ranks in the sequence whenever possible, starting from rank 1. We call this algorithm Exchange (Exc). If a rank is occupied by some previous item, Exc replaces

it if the current item is twice more valuable than the existing item. Other baselines include Random, which produces a random sequence, and Top, which orders items by non-increasing singleton utility. We also include an offline baseline, Omniscient Greedy (Greedy-O) [28], which serves as a tighter estimate for the optimal value. Greedy-O sequentially selects greedily an item with respect to the current set of active functions.

Multiple intents re-ranking. In the absence of explicit user intent for a given query, a search engine needs to take into account all possible intents when providing a list of returned web pages. Each intent is only relevant to a subset of web pages. We represent the utility of an intent by the fractional coverage of relevant pages browsed before running out of patience. The goal is to produce a list of web pages to maximize the utility over all intents.

In this use case, we extract user intents from the SogouQ click log dataset [20]. We consider all queries related to "movie," and for each such query we collect all users who issued the query. We also collect the pages they clicked. We treat each user as an intent, and pages they clicked as the set of relevant pages. For each intent, we generate a random number from 1 to a maximumbudget parameter as user "patience," i.e., the number of pages that the user will browse.

Progressively-diverse personalized recommendation. Next we consider the recommendation task introduced in Section 3. To simulate a concrete task, we consider the task of finding representative synonyms with respect to a given keyword. We choose "Trump" as our keyword. We use the pre-trained word embedding Glove over the Twitter corpus [23]. Similarity or relevance between any two words is measured by the cosine similarity minus 0.5 due to dense vectors. The top 10 000 relevant words form our candidate set V, among which 100 random words are chosen as bases to compute the diversity term (Equation 3.4). Given a maximum length (k = 40) of the recommended list, we create multiple functions $\{f_i\}$ (Equation 3.4) for $i \leq 8$, where the *i*-th function is associated with a weight $(1/2)^i$, a trade-off value $\lambda = (i-1)/k$ and a budget of 5*i*. Thus, function f_i is dominant for the *i*-th length-5 subsequence, and f_i with a large i favors increasingly diverse items.

Results. The results are shown in Figure 2. Every data point represents an average of three runs, each with a different random seed. For the task of web page ranking (Figure 2(a)), all algorithms except for Random perform almost equally well. This implies a heavy overlap in relevant pages among different user intents. For the task of finding diverse synonyms (Figure 2(b)), the ranking in terms of the objective is Greedy-O \approx Exc > Top > Random (7.139 \approx 7.136 > 6.652 > 3.527). We

demonstrate two components of the objective, relevance and diversity, separately in Figure 2(b). Note that the Random algorithm is a classic method in finding diverse representatives, while the Top algorithm is optimal if the objective degenerates into a single modular term of relevance. The word list returned by the Exc algorithm is indeed increasingly the most diverse, while it also finds relevant synonyms at the beginning. The actual word list returned by Exc is presented in Section A.3.

7 Conclusions

In this paper, we study extensions of the MSR problem in two streaming models. In the first demand-arriving model, we show that a greedy algorithm guarantees 2approximation, if items can be reused. In the second item-arriving model, we discover a reduction that turns the ranking problem into a constrained subset-selection problem, and inherit approximation guarantees from standard submodular maximization. The reduction further allows us to approximate the MSR problem subjective to additional p-matroid constraints. Finally, we describe several novel applications for the MSR problem, and examine empirical performance of the proposed algorithms.

One limitation of the MSR formulation is that individual budgets are not always known in some applications. Another limitation is that it may be computationally costly if both the number of demands and items are large, especially for the item-arriving MSR problem.

With respect to ethical considerations of the work, our algorithm is a general submodularity-based framework for ranking, which is not dedicated to a specific application. We cannot identify strong negative societal concerns. Many of the broader machine-learning issues, such as misuse of technology, biases in data, effects of automation in the society, and so on, are relevant to this work, as well, but in no greater degree than the whole machine-learning field.

References

- S. Alaei, A. Makhdoumi, and A. Malekian. Maximizing sequence-submodular functions and its application to online advertising. *Management Science*, 2021.
- [2] Y. Azar and I. Gamzu. Ranking with submodular valuations. In Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms, pages 1070–1079. SIAM, 2011.
- [3] A. Badanidiyuru. Buyback problem-approximate matroid intersection with cancellation costs. In International Colloquium on Automata, Languages,

and Programming, pages 379–390. Springer, 2011.

- [4] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *Proceedings* of the 12th International Conference on Music Information Retrieval (ISMIR 2011), 2011.
- [5] A. Chakrabarti and S. Kale. Submodular maximization meets streaming: Matchings, matroids, and more. *Mathematical Programming*, 154(1):225– 247, 2015.
- [6] M. Fahrbach, Z. Huang, R. Tao, and M. Zadimoghaddam. Edge-weighted online bipartite matching. In 2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS), pages 412–423. IEEE, 2020.
- M. Feldman, J. S. Naor, R. Schwartz, and J. Ward. Improved approximations for k-exchange systems. In *European Symposium on Algorithms*, pages 784– 798. Springer, 2011.
- [8] M. Feldman, A. Norouzi-Fard, O. Svensson, and R. Zenklusen. The one-way communication complexity of submodular maximization with applications to streaming and robustness. In *Proceedings of the* 52nd Annual ACM SIGACT Symposium on Theory of Computing, pages 1363–1374, 2020.
- [9] M. Feldman, A. Norouzi-Fard, O. Svensson, and R. Zenklusen. Submodular maximization subject to matroid intersection on the fly. arXiv preprint arXiv:2204.05154, 2022.
- [10] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions—II. In *Polyhedral combinatorics*, pages 73–87. Springer, 1978.
- [11] E. Hazan, S. Safra, and O. Schwartz. On the complexity of approximating k-set packing. *computational complexity*, 15(1):20–39, 2006.
- [12] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. *Theory OF Computing*, 11(4):105–147, 2015.
- [13] T. Kesselheim. Lecture notes in yao's principle and the secretary problem, 2016. URL https://www. mpi-inf.mpg.de/fileadmin/inf/d1/teaching/ summer16/random/yaosprinciple.pdf.
- [14] J. Kleinberg, E. Ryu, and É. Tardos. Ordered submodularity and its applications to diversifying recommendations. arXiv preprint arXiv:2203.00233, 2022.
- [15] B. Korte and D. Hausmann. An analysis of the greedy heuristic for independence systems. In Annals of Discrete Mathematics, volume 2, pages 65–74. Elsevier, 1978.
- [16] A. Krause and D. Golovin. Submodular function maximization. *Tractability*, 3:71–104, 2014.

- [17] B. Lehmann, D. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 55(2):270–296, 2006.
- [18] R. Levin and D. Wajc. Streaming submodular matching meets the primal-dual method. In Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 1914–1933. SIAM, 2021.
- [19] H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *Proceedings* of the 49th annual meeting of the association for computational linguistics: human language technologies, pages 510–520, 2011.
- [20] Y. Liu, J. Miao, M. Zhang, S. Ma, and L. Ru. How do users describe their information need: Query recommendation based on snippet click model. *Expert systems with applications*, 38(11): 13847–13856, 2011.
- [21] A. Mehta. Online matching and ad allocation. Foundations and Trends® in Theoretical Computer Science, 8(4):265-368, 2013.
- [22] S. Mitrović, I. Bogunovic, A. Norouzi-Fard, J. Tarnawski, and V. Cevher. Streaming robust submodular maximization: A partitioned thresholding approach. arXiv preprint arXiv:1711.02598, 2017.
- [23] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pages 1532–1543, 2014.
- [24] M. L. Pinedo. Scheduling, volume 29. Springer, 2012.
- [25] B. Rozemberczki, C. Allen, and R. Sarkar. Multiscale attributed node embedding, 2019.
- [26] M. Streeter and D. Golovin. An online algorithm for maximizing submodular functions. In *Proceedings* of the 21st International Conference on Neural Information Processing Systems, pages 1577–1584, 2008.
- [27] J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In Proceedings of the fortieth annual ACM symposium on Theory of computing, pages 67–74, 2008.
- [28] G. Zhang, N. Tatti, and A. Gionis. Ranking with submodular functions on a budget. *Data Mining* and Knowledge Discovery, 36(3):1197–1218, 2022.
- [29] Z. Zhang, E. K. Chong, A. Pezeshki, W. Moran, and S. D. Howard. Submodularity and optimality of fusion rules in balanced binary relay trees. In 2012 IEEE 51st IEEE Conference on Decision and Control (CDC), pages 3802–3807. IEEE, 2012.

A Appendix

A.1 Proof of Theorem 4.1

Proof. [Proof of Theorem 4.1] For the online-selection problem, the input is a sequence of n integer numbers a_1, \ldots, a_n that are revealed one after another. The problem asks to select exactly one number immediately after it is revealed, and the goal is to maximize the value of the selected number. It is well-known that this problem does not admit a o(n) approximation [13].

We observe that the online-selection problem is a special case of MMR-F. To see this, consider the following instance: let the universe set V consisting of an item v and n-1 other dummy items. Every arriving modular function f has an identical form, with f(v) = a and f(u) = 0 for any item $u \neq v$. Besides, every function f is associated with a budget of k(f) = 1. At the t-th step, a function as defined above with $f(v) = a_t$ arrives. It is easy to see that the optimal objective value of MMR-F is equal to the largest number a_t . Then, deciding the rank of item v in the output sequence for the MMR-F problem is equivalent to deciding which number in $\{a_t\}$ to select for the online-selection problem. Hence, MMR-F generalizes the online-selection problem.

A.2 Missing proof in Theorem 5.1

LEMMA A.1. $\mathcal{A} \cap \mathcal{B}$ is a p-matroid.

Proof. Since \mathcal{M} is a *p*-matroid over V, then we can write $\mathcal{M} = \mathcal{M}_1 \cap \cdots \cap \mathcal{M}_p$ where \mathcal{M}_i a matroid. For each matroid \mathcal{M}_i , we construct another system over V',

$$\mathcal{M}'_{i} = \{ S' \subseteq V' : V(S') \in \mathcal{M}_{i} \} \cap \mathcal{B} \\ = \{ S' \subseteq V' : V(S') \in \mathcal{M}_{i}, |S' \cap X_{v}| \le 1, \text{ for all } v \in V \}$$

Notice that $\mathcal{A} = \bigcap_i \{ S' \subseteq V' : V(S') \in \mathcal{M}_i \}$, and $\mathcal{A} \cap \mathcal{B} = \bigcap_i \mathcal{M}'_i$. We prove that $\mathcal{A} \cap \mathcal{B}$ is a *p*-matroid by verifying that each \mathcal{M}'_i is a matroid.

To show that \mathcal{M}'_i is a matroid, downward closeness is obvious, and we only verify augmentation. Given any $T, U \in \mathcal{M}'_i$ such that |T| < |U|, we have that $|V(T)| = |T| \le |V(U)| = |U|$. Since $V(T), V(U) \in \mathcal{M}_i$, there exists $v \in V(U)$ such that $v+V(T) \in \mathcal{M}_i$. Suppose $(v,t) \in U$ for that particular v, and it is obvious that $(v,t) + T \in \mathcal{M}'_i$. Hence, \mathcal{M}'_i is a matroid, implying that $\mathcal{A} \cap \mathcal{B} = \bigcap_i \mathcal{M}'_i$ is a p-matroid. \Box

A.3 Synonyms to "Trump" in Twitter trump banks warren clinton gates

newman buffett founder reagan carson ceo appoints butcher duffy carlson lowe travis costello joins airbnb company tesla sanford krause dunlap cassidy does shipbuilding shooter hired rwanda asml hartman barb grandfather rig exchanging lowes varela lamontagne



Figure 3: Running time

A.4 Running time To examine the running time of the proposed algorithms, we generate synthetic data with an increasing number of items or demands. More specific, we either fix the number of demands (100) while increase the number of items, or fix the number of items (1000) while increase the number of demands. Each demand is represented by a coverage function of a random subset of size 100, and is assigned a random budget between 1 to 100. For the MSR-F model (Exc), a random arrival time is given to each demand.

The results are displayed in Figure 3. Running time of both Exc and Greedy algorithms increases linearly in the number of demands. As to an increasing number of items, running time of both appears to be sub-linear instead of linear, which is due to the fact that many items fail to hit any demand subsets.

A.5 Additional experimental details All experiments were carried out on a server equipped with 24 processors of AMD Opteron(tm) Processor 6172 (2.1 GHz), 62GB RAM, running Linux 2.6.32-754.35.1.el6.x86_64. We use Python 3.8.5.