Sequential diversity maximization: An algorithmic approach

Anonymous Author(s)*

ABSTRACT

Diversification is a useful tool for exploring large collections of content items and is used to mitigate the problem of information overload. Diversification finds application in many different domains, including presenting search results of information-retrieval systems and selecting suggestions for recommender systems.

When selecting items from a large collection, it is essential to select not only diverse but also relevant items. Prior research addresses the interplay between relevance and diversity by combining the two measures into one objective, and devising methods to optimize the combined objective. Such models, however, assume a fixed number of items to be selected, and furthermore, they do not rank the selected items. The latter is a significant oversight given that users typically prioritize items that appear higher in their listings.

In this paper, we address these drawbacks by introducing the novel concept of *sequential diversity*. Our model requires to compute a ranking of the available items. It assumes that a user will examine the items in the ranking sequentially, and it incorporates the concept of a *continuation probability*, which is the likelihood of a user continuing to engage with subsequent items, based on the items' relevance. We then ask to maximize the *expected value of the diversity* of the items that the user will examine. We study this problem theoretically, we establish its hardness, and we present algorithms with constant approximation guarantees. Experimentally, we compare our methods against baselines and demonstrate the effectiveness of the proposed algorithms.

KEYWORDS

Diversification, ranking algorithms, approximation algorithms

ACM Reference Format:

Anonymous Author(s). 2018. Sequential diversity maximization: An algorithmic approach. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 17 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Diversification is used to mitigate the risk of information overload by presenting users with novel content. Diversification offers numerous benefits that enhance user experience, such as helping counter the filter-bubble effect, fostering a more inclusive and fair representation of different perspectives, and promoting a well-rounded understanding of topics.

When selecting varied items from a large collection, it is crucial to balance *diversity* and *relevance*. Common approaches to resolve the diversity vs. relevance trade-off combine the two measures into one objective and devise methods that optimize the combined objective. Ideas of this kind are explored, for instance, in the seminal works of Gollapudi and Sharma [15] and Borodin et al. [3].

Those works provide an algorithmic treatment to the diversification problem, which is particularly appealing, as it employs transparent definitions of relevance and diversity, and it leads to easily understood methods with approximation guarantees. On the other hand, existing approaches suffer from two main limitations. First, they fix the number of items to be selected, and second, they formulate the diversification task as a *set-selection problem* rather than a *ranking problem*, that is, they ignore the ordering among the selected items.

Both of these premises come in stark contrast with real-world practices, where users have no predisposition to the number of items that they examine. In fact, the number of items that users examine depends on the relevance of those items. Furthermore, the ranking of the selected items is very important, as it is more likely that users will examine items only at the top of a ranking.

It should be noted here that many diversification methods employ greedy strategies, and thus, a ranking of the items can be implicitly provided. However, such a ranking is not an integral part of the problem definition, and it is not designed to meet any desirable properties, it is only a by-product of the algorithmic solution.

In this paper, we address the above-mentioned limitations by introducing the novel concept of *sequential diversity*. Our idea is to incorporate a model describing how users examine the items presented to them. In particular, we assume that items are presented in order, and users examine the items sequentially. We define a *continuation probability*, which is the likelihood of a user continuing to examine the subsequent items, based on the relevance of the current item.

Thus, our model is stochastic, and the number of items examined by a user is a random variable depending on the ranking of items and their continuation probabilities (or relevance scores). Consequently, for an ordered sequence of items we define its *sequential diversity* to be the *expected value of diversity* of the items that the user will examine. We then seek to compute a ranking that maximizes sequential diversity. We refer to this problem as *sequential-diversity maximization*.

As a measure of diversity, we can use any off-the-shelf diversity function. We consider two such functions: pair-wise sum diversity [3] and coverage diversity [2, 25]. For coverage diversity we prove that the problem is *ordered-submodular* [19] and existing techniques can be adopted. For pair-wise sum diversity we need to develop novel techniques and approximation algorithms. The resulting *sequential sum-diversity maximization* problem, which we refer to as MaxSSD, is the main focus of this paper.

For the MaxSSD problem, we establish its hardness and we present algorithms with constant-factor approximation guarantees.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

Our techniques make interesting connections with a novel problem of *ordered-Hamiltonian path*. Our algorithms rely on the (intuitive) observation that, depending on the value of the continuation probabilities, the diversity score of our formulation is dominated by the top items in the ranking, and thus, we aim to optimize those items. We present two different algorithms, based on selecting the best-k items and on greedy matching, which offer different trade-offs of running time vs. approximation quality. In other words, the exact trade-off is determined by the value of k.

In summary, we make the following contributions.

- We introduce the novel concept of *sequential diversity*, which instils diversity into ranking settings, while modelling user behaviour and accounting for item relevance. We then define the problem of maximizing sequential diversity (MaxSSD).
- We instantiate our framework with two off-the-shelf diversity functions: *pair-wise sum diversity* [3] and *coverage diversity* [2, 25, 26]. For the latter problem, we show that it is *ordered submodular* [19] and standard techniques can be applied. For the former, we develop constant-factor approximation algorithms, leveraging an interesting connection with the *ordered-Hamiltonian path* problem.
- We compare experimentally our methods against baselines and demonstrate their effectiveness in finding high-quality solutions.

Due to space limitations, the proof of our theoretical results are presented in the Appendix.

2 RELATED WORK

We first discuss various manifestations of diversification in different problem scenarios, and then present variants of the travelling salesman problem that are closely related to our methodology.

Diversity maximization. The literature mainly focuses on two notions of diversity: coverage-based diversity and pairwise dissimilarity-based diversity. Coverage-based diversity maximization often models diversity using submodular terms. Agrawal et al. [1] introduce the concept of user coverage, where each user has a probability of querying items of a specific type, and each item has a set of probabilities of satisfying various query types. Their objective is to recommend a related and diverse set of results that maximizes the number of users finding at least one satisfactory document. Ashkan et al. [2] and Puthiya Parambath et al. [25] explore taxonomy coverage, defining diversity as the number of topics or users covered by selected item sets. All three works formulate their objectives as monotone submodular functions and consequently employ greedy algorithms for results with provable guarantees.

Our primary contribution lies in maximizing pairwise dissimilarity. Carbonell and Goldstein [4] introduce *marginal maximal relevance* (MMR), a convex combination of maximum pairwise distance and relevance of selected items. They generate a ranking of search results by greedily selecting the result that maximizes the marginal relevance. Borodin et al. [3] and Gollapudi and Sharma [15] investigate the notion of *max-sum diversity* (MSD) by incorporating a supermodular sum of pairwise distance terms into a submodular relevance function. Subsequent works on the MSD problem apply convex-programming [5] and local-search [6, 7] techniques for improved approximation results. Chen et al. [8] and Gan et al. [13] utilize the *determinantal point process* (DPP) model for result diversification by maintaining a kernel matrix whose off-diagonal entries measure the similarity between items.

The main limitation of the aforementioned studies is the assumption that users give equal consideration to all search results, and that the order of items does not matter. MMR outputs a ranking of selected items but does not consider the uncertainty that arises when users check an item. To better account for item order, Kleinberg et al. [19, 20] assume that the users' patience decays as they check the result list. They propose an *ordered-submodular* coverage function, and a greedy algorithm provides a 1/2 approximation. Unlike their work, we consider the sum of pairwise distance diversity, which is not ordered submodular, requiring to develop novel ideas.

Traveling salesman problem variants. Another topic related to our work is the maximum traveling-salesman problem (MaxTSP) and the time-dependent travelling salesman problem (TDTSP).

The MaxTSP problem aims to identify a tour of maximum weight from a non-negative weighted graph. The initial approximation algorithms for MaxTSP were proposed by Fisher et al. [12]. They establish that a Hamiltonian circuit built from a perfect two-matching is guaranteed to be at least 2/3 the weight of the optimal tour, inspiring the greedy-matching algorithms analyzed in our paper. The bound has been improved in subsequent work [17, 18, 21, 24], while the current best-known ratio is 4/5, achieved by Dudycz et al. [11].

The TDTSP problem focuses on finding a minimum-weighted tour, where edge weights are determined by the traversal time. Contemporary research in TDTSP primarily centres on exact algorithms, including mixed integer programming (MIP) [22], dynamic programming algorithms [23], and branch-and-cut algorithms [9]. Notably, in TDTSP, an edge weight remains independent of the nodes visited before traversing the respective edge, distinguishing it from our problem setting.

3 PRELIMINARIES

3.1 Notation and problem setting

We consider a finite set $U = \{1, \ldots, n\}$, representing n distinct items, e.g., videos or music tracks. Each item $i \in U$ is associated with a probability p_i, which we refer to as *continuation probability*. The value of p_i indicates the probability that, upon examining item i, the user will continue interacting with the system and will examine subsequent items. Conversely, $1 - p_i$ represents the probability that the user will terminate their session and will quit the system after examining item i.

We assume that p_i depends on the *relevance* of item i for user u, and thus, it is more precise to denote the continuation probability by $p_i(u)$. Furthermore, the continuation probability may depend on other context parameters. In our work, we fix such parameters and we assume that the continuation probability is provided as an input to our problem instance, thus, we simply denote it by p_i. Our methods extend easily to cases where the continuation probability depends on other parameters.

We consider scenarios where the items in U are presented to the user in an ordered sequence. Such an ordering of items could be the response to a user query, or a set of item recommendations presenting to the user. Note that, depending on the application,

only a subset of items may be selected to be presented to the user, however, this can be easily incorporated into our model.

Next, we assume that a user examines sequentially the sequence of presented items. At each step, after examining the i-th item, the user decides whether to continue interacting with the system and examine the next item, or to terminate their session and quit the system. Continuation, after examining item i, happens with probability p_i, as discussed above.

In more detail, let a *permutation* $\pi : U \to U$ define the ordering of the items in U; item $\pi(i)$ is assigned at the i-th position. Let $A_{(\pi)} = (\pi(i))_{i=1}^{n}$ denote the ordered sequence according to π; in the rest part of the paper we omit the subscript π and we write $A = A_{(\pi)}$. Let $A_k = (\pi(i))_{i=1}^{k}$ be the k-prefix of A, i.e., the subsequence of A consisting of the first k items. Equivalently, we write $A_k \sqsubseteq A$.

Given an ordered sequence of items A and a k-prefix A_k, let $\Pr(A_k)$ be the probability that the user examines exactly the set of items A_k before quitting. Let $\mathcal{D}(\cdot)$ be a *diversity measure* on the set of items A_k examined by the user. As we will discuss shortly, we aim to maximize the diversity score $\mathcal{D}(A_k)$ on the set of items A_k.

3.2 Problem definition

This paper focuses on the problem of *maximizing* sequential diversity. We first introduce the *sequential-diversity* objective.

Definition 1 (Sequential diversity ($\mathcal{S}$)). *Let $U = \{1, \ldots, n\}$ be a finite set of n distinct items, and $p_1, \ldots, p_n$ be continuation probabilities assigned to each item. Let $A = (\pi(i))_{i=1}^{n}$ be an ordered sequence of U according to some permutation π. Let $\mathcal{D}(A_k)$ be a diversity measure of the items in A_k. The* sequential diversity *of A, denoted by $\mathcal{S}(A)$, is defined as the expectation of the diversity of the items that the user examines before quitting, i.e., $\mathcal{S}(A) = \mathbb{E}_{A_k \sqsubseteq A}[\mathcal{D}(A_k)]$.*

The expectation in Definition 1 is taken over the event that the user examines exactly k items in the list A. The probability of such an event can be computed using the item continuation probabilities.

In the rest of the paper we consider the diversity function $\mathcal{D}(\cdot)$ to be the *pair-wise sum diversity*. Specifically, we consider a distance function $d : U \times U \to \mathbb{R}_{\geq 0}$ between any pair of items in U. We then set $\mathcal{D}(A_k) = \sum_{i,j \in A_k} d(i, j)$, i.e., the diversity is defined by summing all pair-wise distances.

Accordingly, we define the *sequential sum diversity* objective.

Definition 2 (Sequential sum diversity ($\mathcal{S}_+$)). *The* sequential sum diversity *of a sequence A, denoted by $\mathcal{S}_+(A)$, is defined as in Definition 1, with the diversity function $\mathcal{D}(\cdot)$ taken as the pair-wise sum diversity, i.e.,*

$$\mathcal{S}_+(A) = \mathbb{E}_{A_k \sqsubseteq A}[\mathcal{D}(A_k)] = \mathbb{E}_{A_k \sqsubseteq A}\left[\sum_{i,j \in A_k} d(i, j)\right]. \tag{3.1}$$

Next, we define the problem of finding an ordering of items in U that maximizes the sequential sum diversity objective. We refer to this problem as MaxSSD.

Problem 1 (Maximizing sequential sum diversity (MaxSSD)). *Given a finite set $U = \{1, \ldots, n\}$ of n distinct items and associated continuation probabilities $p_1, \ldots, p_n$, find an ordering A^* of the items in U that maximizes the sequential sum diversity objective, i.e.,*

$$A^* = \arg\max_{A = \pi(U)} \mathcal{S}_+(A). \tag{3.2}$$

Problem 1 can also be defined using other notions of diversity. One commonly-used diversity notion is based on *coverage* [2, 25]. Here we assume that each item $i \in U$ is associated with a set of attributes $X(i)$, which is a subset of a ground set of attributes $\mathcal{X}$. For example, if U represents a set of news articles, the attributes $\mathcal{X}$ could be the set of all possible topics, and $X(i) \subseteq \mathcal{X}$ is the subset of topics associated with news article i. The *coverage diversity* of a sequence A_k is then defined as the union of all attributes of the items in A_k, i.e., $\mathcal{D}(A_k) = \bigcup_{i \in A_k} X(i)$.

Analogously to Problem 1, we can then define the problem of *maximizing the sequential coverage diversity* (MaxSCD, Problem 2).

Definition 3 (Sequential coverage diversity ($\mathcal{S}_c$)). *The* sequential coverage diversity *of a sequence A, denoted by $\mathcal{S}_c(A)$, is defined as in Definition 1, with the diversity function $\mathcal{D}(\cdot)$ taken as the coverage diversity, i.e.,*

$$\mathcal{S}_c(A) = \mathbb{E}_{A_k \sqsubseteq A}[\mathcal{D}(A_k)] = \mathbb{E}_{A_k \sqsubseteq A}\left[\bigcup_{i \in A_k} X(i)\right]. \tag{3.3}$$

Problem 2 (Maximizing sequential coverage diversity (MaxSCD)). *Given a finite set $U = \{1, \ldots, n\}$ of n distinct items and associated probabilities $p_1, \ldots, p_n$, find an ordering A^* of the items in U that maximizes the sequential coverage diversity objective, i.e.,*

$$A^* = \arg\max_{A = \pi(U)} \mathcal{S}_c(A). \tag{3.4}$$

Note that the items in U can be viewed as a *complete weighted graph*. This is because each pair of items can be seen as an edge, and we can use the distance function $d(\cdot, \cdot)$ to assign weights on all edges. We make heavy use of this observation as we employ *graph-theoretic ideas*, such as *Hamiltonian paths*, *graph matchings*, etc.

3.3 Reformulation and complexity

We start our analysis with a simple observation that gives us a simplified formulation of our objective.

Observation 1. *As the probabilities assigned to all items are independent, the probability that the user quits directly without examining the first item is $\Pr(A_0) = 1 - p_{\pi(1)}$; the probability that the user quits after examining the first k items is $\Pr(A_k) = \prod_{i=1}^{k} p_{\pi(i)}(1 - p_{\pi(k+1)})$, if $k \leq n - 1$; and the probability that the user examines all items is $\Pr(A_n) = \prod_{i=1}^{n} p_{\pi(i)}$.*

Based on Observation 1, we propose a simplified formulation of Equation (3.1).

Lemma 1. *The sequential sum diversity objective in Equation (3.1) can be equivalently formulated as*

$$\mathcal{S}_+(A) = \sum_{i=1}^{n-1} p_{A_{i+1}} d(\pi(i+1), A_i), \text{ where}$$

$$p_{A_{i+1}} = \prod_{t=1}^{i+1} p_{\pi(t)} \text{ and } d(\pi(i+1), A_i) = \sum_{t=1}^{i} d(\pi(i+1), \pi(t)).$$

Complexity. We can show that the diversity problems that we consider in this paper are **NP**-hard.

Theorem 1. *Problem MaxSSD is* **NP***-hard, even when all p_i have the same value.*

4 SEQUENTIAL COVERAGE DIVERSITY

In this section, we show that the problem of maximizing sequential coverage diversity is ordered submodular [19].

Definition 4 (Ordered submodularity [19]). *A sequence function f is ordered-submodular if for all sequence A and B, the following property holds for all elements s and $\bar{s}$:*

$$f(A||s) - f(A) \geq f(A||s||B) - f(A||\bar{s}||B) \tag{4.1}$$

where $||$ denotes the concatenation of sequences and elements.

Ordered submodularity is introduced by Kleinberg et al. [19], who extend the concepts of monotonicity and submodularity from set functions to sequence functions.

Observation 2. *$\mathcal{S}_c(\cdot)$ is an ordered submodular function.*

As proved by Kleinberg et al. [19], a simple greedy algorithm provides a $1/2$ approximation for an ordered submodular function, and thus, for the MaxSCD problem. On the other hand, sequential sum diversity is not ordered submodular.

Observation 3. *$\mathcal{S}_+(\cdot)$ is not an ordered submodular function.*

In the remainder of our paper, we concentrate on the technically more intriguing MaxSSD problem. Nonetheless, it is important to highlight that our definition of the MaxSCD problem incorporates considerations for user uncertainty and item ordering into coverage diversity maximization. This constitutes a novel contribution to recommender systems and information-retrieval applications.

5 REDUCTION TO ORDERED HAMILTONIAN PATH

In the previous section, we showed that the MaxSSD problem is **NP**-hard. This result is derived via a reduction from the well-known *clique problem* [14], suggesting that a direct solution to our problem is challenging. Furthermore, we observed that MaxSSD is not ordered submodular, thus, ruling out the use of the common *greedy* strategy for achieving a constant-factor approximation.

In this section, we introduce a new problem, which we call *maximum ordered Hamiltonian path* problem (MaxOHP). We will devise a reduction from our problem MaxSSD to MaxOHP, incurring only a constant-ratio approximation loss. This reduction enables us to concentrate on solving an "easier" problem for which we propose approximation algorithms in the latter sections.

First, we define the *ordered Hamiltonian-path* objective.

Definition 5 (Ordered Hamiltonian path ($\mathcal{H}$)). *We are given a finite set $U = \{1, \ldots, n\}$ of n distinct items, a distance function $d(\cdot,\cdot)$, and probabilities $\{p_1, \ldots, p_n\}$ assigned to each item $i \in U$. Let $A = (\pi(i))_{i=1}^{n}$ be the ordered sequence of items of U according to an order π. The ordered Hamiltonian path objective, denoted by $\mathcal{H}$, is defined as*

$$\mathcal{H}(A) = \sum_{i=1}^{n-1} W_{A_i} d(\pi(i), \pi(i+1)), \tag{5.1}$$

where $W_{A_i} = \sum_{j=i+1}^{n} p_{A_j}$.

It is important to note that the *ordered* Hamiltonian-path problem differs significantly from the classic setting [16]. In the classic Hamiltonian-path problem, edge weights are predetermined and do not depend on the *order* in which an edge is placed. In contrast, in the ordered Hamiltonian-path problem, the edge weights $d(\pi(i), \pi(i+1))$ are multiplied by a coefficient w_i, which depends on the order π as well as on the probabilities of all the items in U that come before $\pi(i)$. As a result, the weight of traversing a pair of items (u_i, u_j) depends on the distance $d(u_i, u_j)$ as well as on the order in which the pair is traversed.

Example 1. *Let $U = \{u_1, u_2, u_3\}$ and take $p_i = p$, for all $i \in U$. Consider the ordered sequences $A = \{u_1, u_2, u_3\}$ and $A' = \{u_3, u_1, u_2\}$. The coefficient of $d(u_1, u_2)$ for $\mathcal{H}(A)$ is $p^2 + p^3$; the coefficient of $d(u_1, u_2)$ for $\mathcal{H}(A')$ is p^3.*

Next, we formally define the problem of maximizing the ordered Hamiltonian path (MaxOHP).

Problem 3 (MaxOHP). *We are given a finite set $U = \{1, \ldots, n\}$ of n distinct items, a distance function $d(\cdot,\cdot)$, and probabilities $\{p_1, \ldots, p_n\}$ assigned to each item $i \in U$. The goal is to find an order A^* of the items in U so as to maximize the $\mathcal{H}$ objective, that is,*

$$A^* = \arg\max_{A=\pi(U)} \mathcal{H}(A) = \arg\max_{A=\pi(U)} \sum_{i=1}^{n-1} W_{A_i} d(\pi(i), \pi(i+1)),$$

where $W_{A_i} = \sum_{j=i+1}^{n} p_{A_j}$.

The following lemma facilitates our exposition.

Lemma 2. *$\mathcal{H}(A)$ can be equivalently formulated as*

$$\mathcal{H}(A) = \sum_{i=1}^{n-1} p_{A_{i+1}} d_L(A_{i+1}),$$

where $d_L(A_k) \coloneqq \sum_{t=1}^{k-1} d(\pi(t), \pi(t+1))$.

Next, we state the main result of this section.

Theorem 2. *Let $p_i \in [a, b]$, with $0 < a < b < 1$. A solution that is an α-approximation for the MaxOHP problem is a $\frac{2b(1-a)}{a(1-b)}\alpha$-approximation for the MaxSSD problem.*

Corollary 1. *Consider the special case where $p_i = p$ for all $i \in U$. A solution that is an α-approximation for the MaxOHP problem is a 2α-approximation for the MaxSSD problem.*

Proof of Theorem 2. For the proof of Theorem 2 we use the following auxiliary results.

Lemma 3. *For any ordered sequence A of the items in set U it holds*

$$2\mathcal{S}_+(A) \geq \mathcal{H}(A).$$

The following lemma holds due to the optimality of $A^* = \arg\max \mathcal{S}_+(A)$. Namely, let $\{\pi(i), \pi(i+1)\}$ be any pair of items swapped in A^*, and denote $A' = (\ldots, \pi(i+1), \pi(i), \ldots)$. It follows that $\mathcal{S}_+(A^*) \geq \mathcal{S}_+(A')$.

Lemma 4. *For an optimal solution $A^* = \arg\max \mathcal{S}_+(A)$ to the MaxSSD problem, it holds*

$$d(\pi(i+1), A_i^*) \leq \frac{1 - p_{\pi(i+1)}}{p_{\pi(i+1)}} \sum_{j=1}^{i} \frac{p_{\pi(j+1)}}{1 - p_{\pi(j+1)}} d(\pi(j), \pi(j+1)),$$

for any $1 \leq i \leq n-1$, and where $d(\pi(i+1), A_i^*)$ is defined as in Lemma 1.

Finally, the following corollary can be obtained by applying Lemma 4 and observing that if $p_i \in [a, b]$, it follows that $\frac{1-p_i}{p_i} \leq \frac{1-a}{a}$, and $\frac{p_i}{1-p_i} \leq \frac{b}{1-b}$.

Corollary 2. *For an optimal solution $A^* = \arg\max \mathcal{S}_+(A)$ to the MaxSSD problem, and assuming that $p_i \in [a, b]$, for all $i \in U$, it holds that $\mathcal{S}_+(A^*) \leq \frac{b(1-a)}{a(1-b)}\mathcal{H}(A^*)$.*

We are ready to prove Theorem 2.

Proof. Let A^* be an optimal solution for the MaxSSD problem, and A^o be an optimal solution for the MaxOHP problem. Let A^α be an α-approximation solution for MaxOHP. It holds

$$\mathcal{S}_+(A^*) \overset{(\dagger)}{\leq} \frac{b(1-a)}{a(1-b)}\mathcal{H}(A^*) \leq \frac{b(1-a)}{a(1-b)}\mathcal{H}(A^o)$$
$$\leq \alpha \cdot \frac{b(1-a)}{a(1-b)}\mathcal{H}(A^\alpha) \overset{(\ddagger)}{\leq} 2\alpha \cdot \frac{b(1-a)}{a(1-b)}\mathcal{S}_+(A^\alpha),$$

where inequality (†) follows from Corollary 2, and (‡) from Lemma 3. □

6 UNIFORM CONTINUATION PROBABILITIES

In this section, we address the case that all the continuation probabilities p_i are equal and we devise approximation algorithms for the MaxSSD problem for this special case. Importantly, we consider a general setting, where the probability p is not simply a constant, but it can be a function of n, which may asymptotically go to 0 or to 1. Note that we assume $p \neq 0$ and $p \neq 1$, as these corner cases lead to trivial solutions. We consider three different regimes for the values of p, and we design novel algorithms that offer constant-factor approximations for the MaxSSD problem in all three regimes.

As hinted in the previous section we work with the MaxOHP problem, which then leads to a solution for the MaxSSD problem.

Recall that for an order A, the ordered Hamiltonian-path objective is defined by $\mathcal{H}(A) = \sum_{i=1}^{n-1} W_{A_i} d(\pi(i), \pi(i+1))$, where $W_{A_i} = \sum_{j=i+1}^{n} p_{A_j}$. In the uniform probability setting, since $p_i = p$ for all i, the coefficient W_{A_i} is determined by its position in the order, rather than the actual items being ranked. In this case, we have

$$W_{A_i} = \sum_{j=i+1}^{n} p^j = \frac{p^{i+1} - p^{n+1}}{1-p}. \tag{6.1}$$

6.1 The best-k items algorithm

We first consider the case that is the most likely to happen in practice. In this case, we assume that p is smaller than a constant, which in turn is strictly smaller than 1. In particular, this occurs when p is equal to a constant that is smaller than 1. In this case, the value of the coefficient W_{A_i} is dominated by $\frac{p^{i+1}}{1-p}$ and W_{A_i} *decreases exponentially*. Hence, the value of the objective $\mathcal{H}$ is dominated by the top-k items in the order A.

To get a better intuition for this regime of values of p, consider the extreme scenario that p is very small, i.e., close to 0. Then it is very likely that a user will quit at an early stage of the process, after examining only a very small number of items. Hence, only the first few items of the ranking matter, and it is beneficial to place at the top of the ranking the items that maximizes diversity the most, as those will contribute exclusively to the objective of an ordered Hamiltonian path.

Our algorithm leverages this insight but works for the general case that p is bounded away from 1. We call this algorithm *best-k items*, and refer to it by BkI. Pseudocode is shown in Algorithm 1.

Algorithm 1: Best-k items (BkI) algorithm, for uniform p

Input: Finite set $U = \{1, \ldots, n\}$, integer $k \leq n-1$, probability p
Output: An ordered sequence of U.
1 $A_k \leftarrow$ Solution of Equation (6.2)
2 $A_n \leftarrow$ Extend A_k arbitrarily to a Hamiltonian path if $k < n$.
3 **return** A_n.

From Equation (6.1), as n goes to infinity, the expression for W_{A_i} converges to $\frac{p^{i+1}}{1-p}$, indicating exponential decrease with respect to i. Thus, the objective is dominated by the contribution of the first few items in A.

The best-k items (BkI) algorithm works as follows:

For any integer k, let $\hat{\mathcal{H}}(A_k) = \sum_{i=1}^{k-1} \frac{p^{i+1}}{1-p} d(\pi(i), \pi(i+1))$ denote the contribution of A_k to $\mathcal{H}(A)$. The BkI algorithm seeks the optimal k-item sequence, denoted as BI_k, which maximizes $\hat{\mathcal{H}}(A_k)$; in other words,

$$\mathrm{BI}_k = \underset{A_k \sqsubseteq \pi(U)}{\arg\max} \sum_{t=1}^{k-1} \frac{p^{t+1}}{1-p} d(\pi(t), \pi(t+1)). \tag{6.2}$$

The BkI algorithm initially finds the order BI_k of best-k items, and then it extends that order to include all other items in U. An arbitrary ordering of the rest of the items can be used to extend BI_k, as the particular ordering does not influence the approximation ratio of the algorithm. In our pseudocode, we present a greedy heuristic to extend BI_k.

For the performance of the BkI algorithm, we can prove the following result.

Theorem 3. *Assume that $p_i = p$, for all $i \in U$. Moreover, assume that there is a constant positive number $\epsilon > 0$ such that $p < 1 - \epsilon$. For a fixed value of k, where $2 \leq k \leq n$, the BkI algorithm yields a $\left(\frac{1}{1-p^{k-1}}(1 + \Theta(p^n))\right)$-approximation for the MaxOHP problem.*

Concurring with our intuition, Theorem 3 establishes that optimizing the dominating part, $\hat{\mathcal{H}}(A_k)$, provides a good approximation for the MaxOHP problem.

Furthermore, Theorem 3 captures nicely the inherent approximation-quality vs. efficiency trade-off that results from the selection of k: the larger the value of k, the more expensive the algorithm BkI becomes to run, but the better the approximation ratio it yields.

Observe that searching for the best k items to form BI_k requires $O(\binom{n}{k})$ time. When k is not a constant, i.e., when $k = o(n)$, the execution of Algorithm 1 becomes infeasible. This situation, however, can be handled by the greedy-matching algorithm, presented below, albeit with a constant-factor loss in the approximation ratio.

6.2 The greedy-matching algorithm

In this section, we investigate the case where p is large, namely, $\lim_{n\to\infty} p = 1$, however, it is $\lim_{n\to\infty} p^n = 0$. Intuitively this means that the user will quit before examining all the items. To simplify our exposition, we assume that $p = 1 - \frac{1}{t_n}$, where t_n is a increasing function of n.

Algorithm 2: Greedy matching (GM) algorithm

Input: Finite set U, probability p, diversity function d.
Output: An ordered sequence of U according to π, $A = \pi(U)$.

$k = \lfloor \frac{|U|}{2} \rfloor$; $M \leftarrow$ Empty list.
$E \leftarrow$ all possible pairs (u, v) in decreasing order of $d(u, v)$.
for $(u, v) \in E$ **do**
  add (u, v) to M if it forms a matching.
Re-index nodes in M such that $M = [(u_1, v_1), (u_3, v_3) \cdots, (u_{2k-1}, v_{2k-1})]$.
if $|U| = 2k + 1$ **then**
  Let $v_{2k+1} \in U \setminus M$, $\pi(2k + 1) \leftarrow v_{2k+1}$.
for $i = k; i > 1; i = i - 1$ **do**
  if $i = k$ *and* $|U| = 2k$ **then**
    $\pi(2k - 1) \leftarrow v_{2k-1}$, $\pi(2k) \leftarrow u_{2k-1}$;
    continue.
  if $d(v_{2i-1}, \pi(2i + 1)) \geq d(u_{2i-1}, \pi(2i + 1))$ **then**
    $\pi(2i - 1) \leftarrow v_{2i-1}$, $\pi(2i) \leftarrow u_{2i-1}$.
  else
    $\pi(2i - 1) \leftarrow u_{2i-1}$, $\pi(2i) \leftarrow v_{2i-1}$.
return A.

As searching for the best-k items becomes computationally expensive when k is large, especially when k is not a constant, we introduce a polynomial-time *greedy-matching* (GM) algorithm.

The high-level concept behind GM is to employ a size-$(k/2)$ matching consisting of k items to approximate the best k items, with only a constant factor loss in the approximation ratio. In essence, the GM algorithm serves as an efficient alternative to the BkI algorithm, trading computational complexity for the quality of approximation. The approximation guarantee for GM (Theorem 4) applies to all values of p, although it is weaker compared to the one provided by BkI.

The pseudocode for the GM algorithm is provided in Algorithm 2. Let $G = (U, E)$ be the complete graph constructed from U with edge weights $w(i, j) = d(i, j)$.

The GM algorithm starts by sorting the edges by decreasing weight, and then adds edges to list M in this order as long as M forms a matching. The size of M, denoted by k, is thus equal to $\lfloor \frac{n}{2} \rfloor$. We denote M as $M = [(u_{2j-1}, v_{2j-1})]_{j=1}^{k}$.

The next step is to extend M to a Hamiltonian path. We construct the Hamiltonian path to ensure that: (1) (u_{2j-1}, v_{2j-1}) is the $(2j-1)$-th edge in the path, and (2) The $2j$-th edge weight is at least $1/2$ of the $(2j - 1)$-th edge. To achieve (2), we build the Hamiltonian path from the last node to the first node. We fix the last node of the path and switch the order of u_{2j-1} and v_{2j-1} if the inequality in (2) is not satisfied. It is not difficult to verify that property (2) can always be satisfied due to the triangle inequality.

Let π be the node order of the Hamiltonian path obtained from the GM algorithm. Then the above-mentioned construction naturally leads to the following two properties:

(1) $d((\pi(2i - 1), \pi(2i))) \geq d((\pi(2i + 1), \pi(2i + 2)))$, for $i < k$;
(2) $d((\pi(2i), \pi(2i + 1))) \geq \frac{1}{2} d((\pi(2i - 1), \pi(2i)))$, for $i \leq k$.

For the approximation quality of GM, the following holds.

Theorem 4. *Assume that $p_i = p$, for any $i \in U$. Let p be a function of n such that $\lim_{n\to\infty} p = 1$ and $\lim_{n\to\infty} p^n = 0$. Moreover, assume that $p = 1 - \frac{1}{t_n}$, where t_n is a increasing function of n and $t_n = o(n)$. The algorithm GM yields a $\left(\frac{16e^2}{3(e-1)} + O\left(\frac{1}{t_n}\right)\right)$-approximation for the MaxOHP problem.*

6.3 The case of an arbitrary ranking

Finally, we consider the case that p is asymptotically close to 1. In particular, we assume that $\lim_{n\to\infty} p = 1$ and $\lim_{n\to\infty} p^n = c$, where c is a non-zero constant. For this special case, we notice that the order of items no longer matters — even an arbitrary order will provide a constant-factor approximation to our problem. Intuitively, this is because for such a large value of p the user will examine all the items of U before quitting. Thus, we can apply Lemma 1, and derive a constant-factor approximation algorithm for problem MaxSSD directly, without considering MaxOHP.

Theorem 5. *Assume that $p_i = p$, for all $i \in U$. Moreover, assume that $\lim_{n\to\infty} p = 1$ and $p^n = c - \frac{1}{\Theta(n)}$, where c is a constant. Then any ordering of the items of U yields a $\left(\frac{1}{c} + \frac{1}{\Theta(n)}\right)$-approximation for the MaxSSD problem.*

7 NON-UNIFORM CONTINUATION PROBABILITIES

In this section, we adapt the BkI algorithm, discussed in Section 6, to the case when the continuation probabilities p_i are non-uniform. We assume $p_i \in [a, b]$, for all $i \in U$, where $0 < a \leq b < 1$.

Similar to the uniform case, we define

$$\hat{\mathcal{H}}(A_k) = \sum_{i=1}^{k-1} W_{A_i} d(\pi(i), \pi(i + 1)),$$

aiming to find the best k items maximizing $\hat{\mathcal{H}}(A_k)$. However, this becomes impractical due to the dependency of $W_{A_i} = \sum_{j=i+1}^{n} p_{A_j}$ on order π, necessitating an exhaustive search over all permutations of U to identify the optimal k items. To address this, we adapt W_{A_i} to $W'_{A_i} := \sum_{j=i+1}^{k} p_{A_j}$ to remove its dependency on π, and define $\hat{\mathcal{H}}(A_k) = \sum_{i=1}^{k-1} W'_{A_i} d(\pi(i), \pi(i + 1))$.

Consequently, we redefine BI_k as follows:

$$\mathrm{BI}_k = \arg\max_{A_k \sqsubseteq \pi(U)} \sum_{i=1}^{k-1} W'_{A_i} d(\pi(t), \pi(t + 1)). \tag{7.1}$$

We can show that the adjusted BkI algorithm achieves the following approximation guarantee.

Theorem 6. *Assume $p_i \in [a, b]$, for all $i \in U$, for $0 \leq a \leq b < 1$. Then, Algorithm BkI with BI_k defined as in Equation (7.1) yields a $\frac{a^2 + kb^k}{a^2(1-b-2b^{k-1})}$-approximation guarantee for the MaxOHP problem for any constant integer $2 \leq k \leq n$.*

8 EXPERIMENTAL EVALUATION

8.1 Datasets

We evaluate our methods using five public datasets, each containing genre or category information for individual items, which allows

Table 1: Dataset statistics. $\mathcal{U}$ denotes the users, $\mathcal{I}$ denotes the items, and $\text{avg}(d)$ denotes the average item-item distance.

	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{I}\rvert$	#{Ratings}	$\text{avg}(d)$
Coat	290	300	6960	0.73
Netflix-Prize	4999	1112	557176	0.83
Movielens	6040	3706	1000208	0.83
Yahoo-R2	21181	3000	963296	0.26
KuaiRec-2.0	1411	3327	4676570	0.91

us to calculate inter-item distances. Furthermore, each dataset comprises user-item rating data, or data equivalent to ratings that enable the computation of user-item relevance scores. Table 1 provides a summary of the data statistics.

Coat:[1] Coat ratings in the range $[1, 5]$, accompanied by categorical features, such as gender, jacket type, and color.

Netflix-Prize:[2] Movie ratings in the range $[1, 5]$. Genre information is retrieved from a separate dataset.[3] From the original dataset, we randomly sample 5 000 movies and exclude users with fewer than 20 interactions.

Movielens:[4] Movie ratings in the range $[1, 5]$ with movie genres.

Yahoo-R2:[5] Song ratings in the range $[1, 5]$ with song genres. From the original dataset, we randomly sample 3 000 songs and exclude users with fewer than 20 interactions.

KuaiRec-2.0:[6] The dataset comprises recommendation logs from a video-sharing mobile app, featuring users' watch ratios for videos and corresponding video categories. To get video ratings, we interpolate watch ratios from the interval $[0, 2]$ to $[1, 5]$, where a watch ratio of 0 indicates "never watched" and 2 indicates "watched twice."

8.2 Baselines

We assess the performance of the proposed BkI algorithm by comparing it against several baseline methods.

Random: This simple baseline randomly shuffles all items into a sequence for each user.

Diversity-weighted utility maximization (DUM) [2]: DUM seeks to find a permutation π that maximizes a diversity-weighted relevance objective

$$\text{DUM} = \arg\max_{A \in \pi(U)} \sum_{i=1}^{|U|} \left(f(A_k) - f(A_{k-1})\right) w(\pi(k)),$$

where $f(\cdot)$ is a diversity function and $w(\cdot)$ is the item relevance.

Maximal marginal relevance (MMR) [4]: MMR optimizes iteratively the function defined in the equation below, where Q denotes a user query, U is the set of items, S is the set of items already selected, and sim_1 and sim_2 are similarity functions.

$$\text{MMR} = \arg\max_{i \in U \setminus S} \left[\lambda\, sim_1(i, Q) - (1 - \lambda) \max_{j \in S} sim_2(i, j)\right].$$

Max-sum diversification (MSD) [3]: MSD aims to maximize the equation below consisting of a submodular relevance term and a supermodular sum-of-distances diversity term.

$$\text{MSD} = \arg\max_{S \in U} f(S) + \lambda \sum_{u,v \in S} d(u, v), \quad \text{such that } |S| \le k.$$

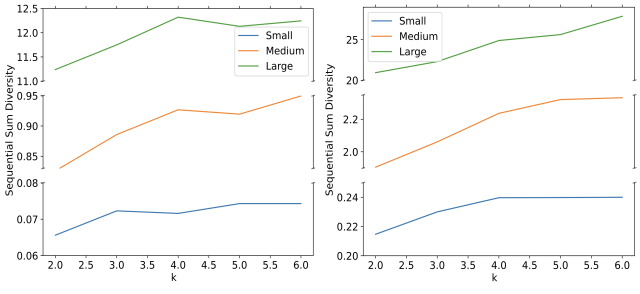
In our experiments, we set $f(S) = \sum_{i \in S} p_i$. We set $k = |U|$ and run the greedy algorithm from Borodin et al. [3] to obtain a ranking.

Determinantal point process (DPP) [8]: DPP iteratively selects the item i according to the following criterion

$$i = \arg\max_{j \in U \setminus S} \left[\log\det(\mathbf{L}_{S \cup \{j\}}) - \log\det(\mathbf{L}_S)\right],$$

where S is the already selected item set, and $\mathbf{L}_S$ is a kernel matrix.

BkI-H: In addition to the baselines, we experiment with a heuristic variant of the BkI algorithm, where we use a two-step strategy to find the best-k items. Initially, a candidate set is identified through a filtering process, and the search for the best-k items is confined to this reduced candidate set, This heuristic optimizes the efficiency of BkI, while maintaining its effectiveness in identifying top-ranked items. Details on the filtering process are provided below.



(a) *Coat* **(b) *Synthetic***

Figure 1: The performance of BkI algorithm with different k.

8.3 Experimental setting

Continuation probability: We first perform matrix factorization to complete the user-rating and item-feature matrices. We then interpolate the ratings from the interval $[1, 5]$ into three probability regimes: small ($[0.1, 0.3]$), medium ($[0.4, 0.6]$), and large ($[0.7, 0.9]$).

Relevance: We use the continuation probability as user-item relevance across all baseline methods.

Distance function: Given the categorical nature of the data, we use the Jaccard distance as the item-item distance function.

Parameter k: For the BkI algorithm, experiments are conducted with $k = 2$, due to prohibitive time complexity for larger k. For BkI-H, experiments are conducted with $k = 2, 3, 4$.

Filtering process: The BkI-H heuristic employs two filtering steps to select candidate item sets. The first filtering selects the top η percent of items with the highest relevance. The second filtering selects the top η percent of item pairs with maximum distance. The intersection of these two filtered results constitutes the final candidate set. The parameter η varies for each dataset: 0.1 for Coat, 0.01 for Movielens and KuaiRec-2.0, 0.015 for Yahoo-R2, and 0.03 for Netflix-Prize, with the aim of maintaining a compact candidate set.

The code used in our experiments is publicly available.[7]

[1] https://www.cs.cornell.edu/~schnabts/mnar/
[2] https://www.kaggle.com/datasets/rishitjavia/netflix-movie-rating-dataset
[3] https://github.com/tommasocarraro/netflix-prize-with-genres
[4] https://grouplens.org/datasets/movielens/1m/
[5] https://webscope.sandbox.yahoo.com/catalog.php?datatype=i&did=67
[6] https://KuaiRec-2.0.com/
[7] https://anonymous.4open.science/r/Sequential-Diversity-Maximization-DCD3

Table 2: Sequential sum diversity with item continuation probability mapped to $[0.1, 0.3]$

Expectation	Random	MSD	MMR	DPP	DUM	BkI (k=2)	BkI-H (k=2)	BkI-H (k=3)	BkI-H (k=4)
Coat	0.024	0.072	0.077	0.103	0.103	**0.109**	0.080	0.078	0.076
Netflix-Prize	0.063	0.159	0.160	0.158	0.158	**0.161**	**0.161**	0.155	0.155
Movielens	0.048	**0.182**	**0.182**	0.180	0.180	**0.182**	**0.182**	0.171	0.170
Yahoo-R2	0.018	**0.147**	**0.147**	0.145	0.145	**0.147**	0.143	0.125	0.123
KuaiRec-2.0	0.020	0.045	0.048	0.041	0.041	**0.056**	0.051	0.048	0.048

Table 3: Sequential sum diversity with item continuation probability mapped to $[0.4, 0.6]$

Expectation	Random	MSD	MMR	DPP	DUM	BkI (k=2)	BkI-H (k=2)	BkI-H (k=3)	BkI-H (k=4)
Coat	0.470	0.936	0.946	1.260	1.264	**1.290**	0.952	0.874	0.874
Netflix-Prize	0.935	0.891	1.788	1.901	1.900	**1.915**	1.914	1.877	1.863
Movielens	0.876	2.206	2.205	2.189	2.188	2.211	**2.212**	2.128	2.098
Yahoo-R2	0.240	1.788	1.788	1.757	1.757	**1.790**	1.731	1.592	1.560
KuaiRec-2.0	0.353	0.705	0.689	0.561	0.561	**0.743**	0.706	0.676	0.644

Table 4: Sequential sum diversity with item continuation probability mapped to $[0.7, 0.9]$

Expectation	Random	MSD	MMR	DPP	DUM	BkI (k=2)	BkI-H (k=2)	BkI-H (k=3)	BkI-H (k=4)
Coat	7.018	17.613	17.730	21.453	21.513	**23.013**	16.916	16.776	16.770
Netflix-Prize	16.348	39.946	**47.964**	42.149	42.138	46.361	46.361	46.438	46.540
Movielens	14.196	64.434	68.618	61.107	61.114	70.560	**70.577**	70.457	70.391
Yahoo-R2	3.453	38.406	39.016	37.101	37.096	**41.552**	40.263	40.528	40.568
KuaiRec-2.0	5.544	9.851	**14.008**	7.942	7.953	10.431	10.407	10.467	10.550

8.4 Results and discussion

Performance of the BkI algorithm: To evaluate the performance of the BkI algorithm, we run it with $k = 2$ on the five datasets and compare its results with all baselines. We extend the best best 2 items greedily to a sequence for each user. To analyze the influence of the continuation probability on performance, we run all algorithms in the three probability regimes. We present the results in Tables 2, 3, and 4. The reported metric is the average sequential sum diversity value for all users in each dataset. The best results are highlighted in bold and the second best are underlined.

We observe that our methods achieve excellent results. The BkI and BkI-H algorithms outperform all baselines when the continuation probability is within the ranges $[0.1, 0.3]$ and $[0.4, 0.6]$. In the probability range $[0.7, 0.9]$, BkI performs the best on two datasets, and BkI-H is the second best on the remaining three datasets.

We also observe that the performance of BkI-H tends to decrease as the value of k increases across the majority of datasets and probability regimes. This phenomenon can be attributed to the fact that BkI-H selectively chooses the best-k items from a small candidate set. Consequently, the observed performance decline does not refute our theoretical results. The inherent nature of BkI-H, focusing on a restricted candidate set, contributes to this trend.

Analysis of parameter k: To assess the performance of BkI for different values of k, we run it on two small datasets. The first dataset, coat, consists of 50 randomly sampled users and 50 items. The second dataset is a synthetic dataset with 100 users and 50 items, where the items are placed in a two-dimensional Euclidean space with both x- and y-axes ranging from $[0, 1]$. The item continuation probabilities are uniformly randomly distributed in the small range $[0.1, 0.3]$, medium range $[0.4, 0.6]$, and large range $[0.7, 0.9]$.

For each user in the two datasets, we run the BkI algorithm to select the best-k items and arbitrarily extend it to a sequence that contains all items. Figures 1a and 1b present the sequential sum diversity increases as k increases, confirming our theoretical results. It is worth noting that for the BkI algorithm, the way we extend the best-k items to a sequence does not influence its theoretical performance guarantee, but it can make a practical difference, which explains the minor fluctuations observed as k increases.

Run time: All experiments are conducted on a dual-socket server, each socket housing an Intel Xeon Gold 6326 CPU running at 2.90GHz, with a total of 128 cores and 256 threads. The run time of our methods and all baselines are shown in Table 5 in the Appendix.

9 CONCLUSION

In this paper, we introduce the novel concept of *sequential diversity*, aiming to instil diversity into rankings while considering the relevance of items and modelling user behaviour. The proposed framework is designed to find rankings that consist of items that are both diverse and relevant. The diversification task we define gives rise to a novel computational problem, for which we establish a connection with the ordered Hamiltonian-path problem and design approximation algorithms with provable guarantees. Our algorithms offer trade-offs of efficiency vs. approximation quality.

The proposed framework opens many exciting directions for future work. First, it will be interesting to devise more efficient combinatorial algorithms without losing their approximation guarantees. Additionally, it will be valuable to incorporate user models of higher complexity into the framework, in order to capture more nuanced user behaviours. Last, it will be interesting to study the proposed framework with a user study on a real-world system.

REFERENCES

[1] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. 2009. Diversifying search results. In *Proceedings of the second ACM international conference on web search and data mining*. 5–14.

[2] Azin Ashkan, Branislav Kveton, Shlomo Berkovsky, and Zheng Wen. 2015. Optimal Greedy Diversity for Recommendation.. In *IJCAI*, Vol. 15. 1742–1748.

[3] Allan Borodin, Hyun Chul Lee, and Yuli Ye. 2012. Max-sum diversification, monotone submodular functions and dynamic updates. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*. 155–166.

[4] Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. 335–336.

[5] Alfonso Cevallos, Friedrich Eisenbrand, and Rico Zenklusen. 2015. Max-sum diversity via convex programming. *arXiv preprint arXiv:1511.07077* (2015).

[6] Alfonso Cevallos, Friedrich Eisenbrand, and Rico Zenklusen. 2017. Local search for max-sum diversification. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 130–142.

[7] Alfonso Cevallos, Friedrich Eisenbrand, and Rico Zenklusen. 2019. An improved analysis of local search for max-sum diversification. *Mathematics of Operations Research* 44, 4 (2019), 1494–1509.

[8] Laming Chen, Guoxin Zhang, and Eric Zhou. 2018. Fast greedy map inference for determinantal point process to improve recommendation diversity. *Advances in Neural Information Processing Systems* 31 (2018).

[9] Jean-François Cordeau, Gianpaolo Ghiani, and Emanuela Guerriero. 2014. Analysis and branch-and-cut algorithm for the time-dependent travelling salesman problem. *Transportation science* 48, 1 (2014), 46–58.

[10] Zdravko Cvetkovski. 2012. *Inequalities: theorems, techniques and selected problems*. Springer Science & Business Media.

[11] Szymon Dudycz, Jan Marcinkowski, Katarzyna Paluch, and Bartosz Rybicki. 2017. A 4/5-approximation algorithm for the maximum traveling salesman problem. In *International Conference on Integer Programming and Combinatorial Optimization*. Springer, 173–185.

[12] Marshall L Fisher, George L Nemhauser, and Laurence A Wolsey. 1979. An analysis of approximations for finding a maximum weight Hamiltonian circuit. *Operations Research* 27, 4 (1979), 799–809.

[13] Lu Gan, Diana Nurbakova, Léa Laporte, and Sylvie Calabretto. 2020. Enhancing recommendation diversity using determinantal point processes on knowledge graphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2001–2004.

[14] Michael R Garey and David S Johnson. 1979. Computers and intractability. *A Guide to the* (1979).

[15] Sreenivas Gollapudi and Aneesh Sharma. 2009. An axiomatic approach for result diversification. In *Proceedings of the 18th international conference on World wide web*. 381–390.

[16] Yuri Gurevich and Saharon Shelah. 1987. Expected computation time for Hamiltonian path problem. *SIAM J. Comput.* 16, 3 (1987), 486–502.

[17] Refael Hassin and Shlomi Rubinstein. 1998. An approximation algorithm for the maximum traveling salesman problem. *Inform. Process. Lett.* 67, 3 (1998), 125–130.

[18] Refael Hassin, Shlomi Rubinstein, and Arie Tamir. 1997. Approximation algorithms for maximum dispersion. *Operations research letters* 21, 3 (1997), 133–137.

[19] Jon Kleinberg, Emily Ryu, and Éva Tardos. 2022. Ordered submodularity and its applications to diversifying recommendations. *arXiv preprint arXiv:2203.00233* (2022).

[20] Jon Kleinberg, Emily Ryu, and Éva Tardos. 2023. Calibrated Recommendations for Users with Decaying Attention. *arXiv preprint arXiv:2302.03239* (2023).

[21] S Rao Kosaraju, James K Park, and Clifford Stein. 1994. Long tours and short superstrings. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*. IEEE, 166–177.

[22] Chryssi Malandraki and Mark S Daskin. 1992. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation science* 26, 3 (1992), 185–200.

[23] Chryssi Malandraki and Robert B Dial. 1996. A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem. *European Journal of Operational Research* 90, 1 (1996), 45–55.

[24] Jérôme Monnot, Vangelis Th Paschos, and Sophie Toulouse. 2003. Approximation algorithms for the traveling salesman problem. *Mathematical methods of operations research* 56 (2003), 387–405.

[25] Shameem A Puthiya Parambath, Nicolas Usunier, and Yves Grandvalet. 2016. A coverage-based approach to recommendation diversity on similarity graph. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 15–22.

[26] ChengXiang Zhai, William W Cohen, and John Lafferty. 2015. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *Acm sigir forum*, Vol. 49. ACM New York, NY, USA, 2–9.

A OMITTED PROOFS FROM SECTION 5

Lemma 1. *The sequential sum diversity objective in Equation (3.1) can be equivalently formulated as*

$$\mathcal{S}_+(A) = \sum_{i=1}^{n-1} p_{A_{i+1}} d(\pi(i+1), A_i), \text{ where}$$

$$p_{A_{i+1}} = \prod_{t=1}^{i+1} p_{\pi(t)} \text{ and } d(\pi(i+1), A_i) = \sum_{t=1}^{i} d(\pi(i+1), \pi(t)).$$

Proof. We start with demonstrating that the sequential sum-diversity objective can be precisely reformulated via the expression $2\sum_{i=2}^{n}\sum_{j=1}^{i-1}\prod_{t=1}^{i}\Pr_{\pi(t)} d(\pi(i), \pi(j))$. Next, the formulation can be simplified using the definition of p_{A_i} and $d(\pi(i), A_{i-1})$.

We write the definition of sequential sum-diversity objective, and proceed by merging the terms according to our observations.

$$\begin{aligned}
&\sum_{k=1}^{n} \Pr(A_k) \sum_{\pi(i),\pi(j)\in A_k} d(\pi(i),\pi(j)) \\
&\overset{(a)}{=} \sum_{k=1}^{n} \Pr(A_k) \sum_{i,j\le k} d(\pi(i),\pi(j)) \\
&\overset{(b)}{=} \sum_{k=1}^{n-1} \Big(\prod_{t=1}^{k} p_{\pi(t)} - \prod_{t=1}^{k+1} p_{\pi(t)}\Big) \sum_{i,j\le k} d(\pi(i),\pi(j)) \\
&+ \prod_{t=1}^{n} p_{\pi(t)} \sum_{i,j\le n} d(\pi(i),\pi(j)) \\
&= \sum_{k=1}^{n-1}\prod_{t=1}^{k+1} p_{\pi(t)} \Big(-\sum_{i,j\le k} d(\pi(i),\pi(j)) + \sum_{i,j\le k+1} d(\pi(i),\pi(j))\Big) \\
&= \sum_{k=1}^{n-1}\prod_{t=1}^{k+1} p_{\pi(t)} \Big(\sum_{i=k+1,j\le k} d(\pi(i),\pi(j)) + \sum_{j=k+1,i\le k} d(\pi(i),\pi(j))\Big) \\
&= 2\sum_{k=1}^{n-1}\prod_{t=1}^{k+1} p_{\pi(t)} \Big(\sum_{i=k+1,j\le k} d(\pi(i),\pi(j))\Big) \\
&= 2\sum_{k=2}^{n}\prod_{t=1}^{k} p_{\pi(t)} \Big(\sum_{i=k,j\le k-1} d(\pi(i),\pi(j))\Big) \\
&= 2\sum_{i=2}^{n}\sum_{j=1}^{i-1}\prod_{t=1}^{i} p_{\pi(t)} d(\pi(i),\pi(j)).
\end{aligned} \tag{A.1}$$

Notice that equality (a) holds by the definition of $\pi(i)$: indeed, $\pi(i)$ indicates the item that is placed at the i-th position. Equality (b) holds by substituting the expression of $\Pr_\pi(A_k)$ from the observation 1. For the following equations, we re-arrange the simplify the formulas. □

For the proof of Observation 2 and Observation 3, we use two lemma from Kleinberg et al. [19].

Lemma 5 (Kleinberg et al. [19]). *If f and g are ordered-submodular, then $\alpha f + \beta g$ is also ordered-submodular for any $\alpha, \beta \ge 0$.*

Lemma 6 (Kleinberg et al. [19]). *Suppose h is a monotone submodular set function, Then the function f constructed by evaluating h on the set of the first t elements of S, that is,*

$$f(S) = \begin{cases} h(S) & \text{if } |S| \le t \\ h(S_t) & \text{if } |S| > t \end{cases}$$

is ordered-submodular.

Observation 2. *$\mathcal{S}_c(\cdot)$ is an ordered submodular function.*

Proof. By the definition of the $\mathcal{S}_c$ objective we have:

$$\mathcal{S}_c(A) = \mathbb{E}_{A_k \sqsubseteq A}[\mathcal{D}(A_k)] = \sum_{k=1}^{n} \Pr(A_k)\mathcal{D}(A_k), \tag{A.2}$$

where $\mathcal{D}(A_k)$ is a monotone submodular function. Let $f_k(A) = \mathcal{D}(A_k)$ so we can rewrite $\mathcal{S}_c(A)$ as $\mathcal{S}_c(A) = \sum_{k=1}^{n} p_{A_k} f_k(A)$. Since $f_k(A)$ is a monotone submodular function evaluated on the first k items, by Lemma 6, $f_k(A)$ is an ordered submodular function. Since $\Pr(A_k) \ge 0$ for all k, by Lemma 5, $\mathcal{S}_c(\cdot)$ is ordered-submodular. □

Observation 3. *$\mathcal{S}_+(\cdot)$ is not an ordered submodular function.*

Proof. To show $\mathcal{S}_+(\cdot)$ is not ordered submodular, we only need one counter example. Let A and B be two sequences that satisfy $p_A > 0$ and $p_B > 0$. Let s be an item such that $p_s > 0$ and $d(s, B) > 0$. Also, let $\bar{s}$ be an item such that $p_{\bar{s}} = 0$. It follows that

$$\begin{aligned}
\mathcal{S}_+(A||s||B) - \mathcal{S}_+(A||\bar{s}||B) &= \mathcal{S}_+(A||s||B) - \mathcal{S}_+(A) \\
&> \mathcal{S}_+(A||s) - \mathcal{S}_+(A).
\end{aligned}$$

This example violates the definition of ordered submodularity, which proves that $\mathcal{S}_+(\cdot)$ is not an ordered submodular function. □

Theorem 1. *Problem MaxSSD is* NP*-hard, even when all p_i have the same value.*

Proof. We provide a reduction from the decision version of the maximum clique problem to the decision version of the MaxSSD problem. Define CLIQUE $= (G, V, k)$, with $|V| = n$ to describe instances of the maximum clique problem, where we want to decide whether there exists a clique of size k in G. Also define an instance of the MaxSSD problem by (U, d, p, θ), where U is the item set, p is a uniform continuation probability, and $d(\cdot,\cdot)$ is a metric distance function, and we want to decide whether there exist an ordering π of U such that $\mathcal{S}_+(\pi(U)) > \theta$.

Given an instance of CLIQUE, we construct an instance of MaxSSD in the following way: we set $U = V$, and we assign $d(u, v) = 2$ if there is an edge between nodes u and v in G, otherwise we set $d(u, v) = 1 + \epsilon$, where ϵ is any constant smaller than 1. We set $p = \frac{1-\epsilon}{2n^2}$ and $\theta = 2\sum_{i=1}^{k-1} i \times \left(\frac{1-\epsilon}{2n^2}\right)^{i+1}$. One can easily verify that this transformation can be done in polynomial time, and that $d(\cdot,\cdot)$ is a metric.

If there is a clique of size $k \le n$ in G, then for the corresponding item set $U = V$, we can construct an ordered sequence $A = (\pi(i))_{i=1}^{n}$ by assigning A_k to be the k nodes in U that corresponds to the clique in G, and setting in an arbitrary way the rest of the sequence order.

Let $\mathcal{L}(A)$ be a lower bound of $\mathcal{S}_+(A)$. One can verify it is also a lower bound for the optimal objective value of the MaxSSD problem. We can define $\mathcal{L}(A)$ as follows:

$$\begin{aligned}\mathcal{L}(A) &= \sum_{i=1}^{k-1} W_{A_i} d(\pi(i+1), A_i) \\ &= 2(p^2 + 2p^3 + 3p^4 + \cdots + (k-2)p^{k-1} + (k-1)p^k) \\ &= 2\sum_{i=1}^{k-1} i \times p^{i+1}.\end{aligned}$$

Furthermore, it holds that $\mathcal{L}(A) = \theta$.

On the other hand, if there does not exist a size k clique in G, then the densest possible graph structure of G contains $T = \lfloor \frac{n}{k-1} \rfloor$ size-$(k-1)$ cliques, and one clique of size $L = n - (k-1)T$. A node in a size-$(k-1)$ clique can connect to at most $k-2$ nodes in another size-$(k-1)$ clique, thus the maximum number of edges between any two size-$(k-1)$ cliques is $(k-1)(k-2)$. Similarly, the number of edges between a size-L clique and a size-$(k-1)$ clique is $(k-2)L$. Given this graph structure, we can construct an ordered sequence $\tilde{A} = (\tilde{\pi}(i))_{i=1}^n$, where sequence $[\tilde{\pi}((j-1)(k-1)+1), \cdots, \tilde{\pi}((k-1)\times j)]$ corresponds to the j-th size-$(k-1)$ clique of G, with $j \in [1, T]$, and the remaining sequence corresponds to the size-L clique of G.

Let $\mathcal{U}(\tilde{A}) = \mathcal{H}(\tilde{A})$. One can verify that $\mathcal{U}(\tilde{A})$ is an upper bound of the optimal objective value of the corresponding MaxSSD problem. It holds that

$$\begin{aligned}\mathcal{U}(\tilde{A}) &= \sum_{i=1}^{n-1} W_{\tilde{A}_i} d(\tilde{\pi}(i+1), \tilde{A}_i) \\ &\overset{(a)}{=} 2(p^2 + 2p^3 + \cdots + (k-2)p^{k-1}) \\ &+ \sum_{i=1}^{T-1}\sum_{j=0}^{k-2} p^{j+1+i(k-1)}(2[i(k-2)+j] + i(1+\epsilon)) \\ &+ \sum_{j=0}^{L-1} p^{T(k-1)+j+1}(2[T(k-2)+j] + T(1+\epsilon)) \\ &\overset{(b)}{<} \mathcal{L}(A) + (\epsilon-1)p^k + 2n(p^{k+1} + p^{k+2} + \cdots + p^n),\end{aligned} \tag{A.3}$$

where (a) holds because of Lemma 1 and (b) holds because for $j \in [k+1, n]$, the j-th node contributes at most $2np^j$ to $\mathcal{U}(\tilde{A})$.

Furthermore, since

$$\begin{aligned}\mathcal{L}(A) - \mathcal{U}(\tilde{A}) &= \theta - \mathcal{U}(\tilde{A}) \\ &> (1-\epsilon)p^k - 2n(p^{k+1} + p^{k+2} + \cdots + p^n) \\ &= (1-\epsilon)p^k - 2np(p^k + p^{k+1} + \cdots + p^{n-1}) \\ &> (1-\epsilon)p^k - 2np(n-k)p^k \\ &= p^k(1 - \epsilon - 2np(n-k)) = \frac{p^k(1-\epsilon)k}{n} > 0\end{aligned}$$

holds for all k, we conclude that a yes instance of the CLIQUE problem indicates a yes instance of the MaxSSD problem, and a no instance indicates a no instance of the MaxSSD problem.

This finishes our reduction from the CLIQUE problem to the MaxSSD problem and we conclude that the MaxSSD problem is **NP**-hard. □

Lemma 2. *$\mathcal{H}(A)$ can be equivalently formulated as*

$$\mathcal{H}(A) = \sum_{i=1}^{n-1} p_{A_{i+1}} d_L(A_{i+1}),$$

where $d_L(A_k) \coloneqq \sum_{t=1}^{k-1} d(\pi(t), \pi(t+1))$.

Proof. We simply re-arrange the formulas.

$$\begin{aligned}\mathcal{H}(A) &= \sum_{i=1}^{n-1} w_i d(\pi(i), \pi(i+1)) \\ &= \sum_{i=1}^{n-1}\sum_{k=i+1}^{n} p_{A_k} d(\pi(i), \pi(i+1)) \\ &= p_{A_2} d(\pi(1), \pi(2)) + p_{A_3}(d(\pi(1), \pi(2)) + d(\pi(2), \pi(3))) \\ &+ \cdots + p(A)\sum_{t=1}^{n-1} d(\pi(t), \pi(t+1)) \\ &= p_{A_2} d_L(A_2) + p_{A_3} d_L(A_3) + \cdots + p(A) d_L(A) \\ &= \sum_{i=1}^{n-1} p_{A_{i+1}} d_L(A_{i+1}).\end{aligned} \tag{A.4}$$

□

Lemma 3. *For any ordered sequence A of the items in set U it holds*

$$2\mathcal{S}_+(A) \geq \mathcal{H}(A).$$

Proof. First, we show that if $2d(\pi(i+1), A_i) \geq d_L(A_{i+1})$ holds for any $i \in [n-1]$, then our lemma is proven:

$$\begin{aligned}2\mathcal{S}(A) &= 2\sum_{i=1}^{n-1} p_{A_{i+1}} d(\pi(i+1), A_i) \\ &= \sum_{i=1}^{n-1} p_{A_{i+1}} 2d(\pi(i+1), A_i) \\ &\geq \sum_{i=1}^{n-1} d_L(A_{i+1}) \\ &= \mathcal{H}(A).\end{aligned}$$

Next, we show that $2d(\pi(i+1), A_i) \geq d_L(A_{i+1})$ holds for any $1 \leq i \leq n-1$:

$$\begin{aligned}d_L(A_{i+1}) &= \sum_{j=1}^{i} d(\pi(j), \pi(j+1)) \\ &\overset{(a)}{\leq} \sum_{j=1}^{i} [d(\pi(i+1), \pi(j)) + d(\pi(i+1), \pi(j+1))] \\ &= \sum_{j=1}^{i} d(\pi(i+1), \pi(j)) + \sum_{j=1}^{i} d(\pi(i+1), \pi(j+1)) \\ &= d(\pi(i+1), A_i) + d(\pi(i+1), A_{i+1}) - d(\pi(i+1), \pi(1)) \\ &\overset{(b)}{=} d(\pi(i+1), A_i) + d(\pi(i+1), A_i) - d(\pi(i+1), \pi(1)) \\ &< 2d(\pi(i+1), A_i),\end{aligned}$$

where (a) holds by triangle inequality, and (b) holds since $d(\pi(i+1), \pi(i+1)) = 0$. □

Lemma 4. *For an optimal solution $A^* = \arg\max \mathcal{S}_+(A)$ to the MAX-SSD problem, it holds*

$$d(\pi(i+1), A_i^*) \leq \frac{1-p_{\pi(i+1)}}{p_{\pi(i+1)}} \sum_{j=1}^{i} \frac{p_{\pi(j+1)}}{1-p_{\pi(j+1)}} d(\pi(j), \pi(j+1)),$$

for any $1 \leq i \leq n-1$, and where $d(\pi(i+1), A_i^*)$ is defined as in Lemma 1.

Proof. Let $A^* = (\pi(i))_{i=1}^n$ be an optimal sequence, and let us denote $A' = (\ldots, \pi(i+1), \pi(i), \ldots)$ to be the sequence after swapping node pair $\{\pi(i), \pi(i+1)\}$. Then,

$$\begin{aligned}\mathcal{S}_+(A') = &\sum_{j=1}^{i-2} p_{A_{j+1}^*} d(\pi(j+1), A_j^*) + p_{\pi(i+1)} p_{A_{i-1}^*} d(\pi(i+1), A_{i-1}^*) \\ &+ p_{\pi(i)} p_{\pi(i+1)} p_{A_{i-1}^*} d(\pi(i), A_{i-1}^* \cup \pi(i+1)) \\ &+ \sum_{j=i+1}^{n-1} p_{A_{j+1}^*} d(\pi(j+1), A_j^*),\end{aligned}$$

and

$$\begin{aligned}\mathcal{S}_+(A^*) = &\sum_{j=1}^{i-2} p_{A_{j+1}^*} d(\pi(j+1), A_j^*) + p_{\pi(i)} p_{A_{i-1}^*} d(\pi(i), A_{i-1}^*) \\ &+ p_{\pi(i)} p_{\pi(i+1)} p_{A_{i-1}^*} d(\pi(i+1), A_{i-1}^* \cup \pi(i)) \\ &+ \sum_{j=i+1}^{n-1} p_{A_{j+1}^*} d(\pi(j+1), A_j^*).\end{aligned}$$

Since $A^* = \arg\max \mathcal{S}_+(A)$ is an optimal sequence, it must hold that $\mathcal{S}_+(A^*) - \mathcal{S}_+(A') \geq 0$, that is,

$$\begin{aligned}\mathcal{S}_+(A^*) - \mathcal{S}_+(A') = &\, p_{\pi(i)} p_{A_{i-1}^*} d(\pi(i), A_{i-1}^*) \\ &- p_{\pi(i+1)} p_{A_{i-1}^*} d(\pi(i+1), A_{i-1}^*) \\ &+ p_{\pi(i)} p_{\pi(i+1)} p_{A_{i-1}^*} d(\pi(i+1), A_{i-1}^*) \\ &- p_{\pi(i)} p_{\pi(i+1)} p_{A_{i-1}^*} d(\pi(i), A_{i-1}^*) \geq 0.\end{aligned}$$

This is equivalent to

$$\begin{aligned}&p_{\pi(i)} p_{A_{i-1}^*} (1-p_{\pi(i+1)}) d(\pi(i), A_{i-1}^*) \\ &\geq p_{\pi(i+1)} p_{A_{i-1}^*} (1-p_{\pi(i)}) d(\pi(i+1), A_{i-1}^*).\end{aligned} \tag{A.5}$$

Since $p_{A_{i-1}^*} \geq 0$, we can cancel that term from both sides of Equation (A.5). We then divide both sides by $(1-p_{\pi(i)}) \times (1-p_{\pi(i+1)})$ and obtain

$$\frac{p_{\pi(i)}}{1-p_{\pi(i)}} d(\pi(i), A_{i-1}^*) \geq \frac{p_{\pi(i+1)}}{1-p_{\pi(i+1)}} d(\pi(i+1), A_{i-1}^*). \tag{A.6}$$

By taking the telescope sum of Equation (A.6) over i from $i=2$ to $i=j$, we get

$$\sum_{i=2}^{j} \frac{p_{\pi(i)}}{1-p_{\pi(i)}} d(\pi(i), A_{i-1}^*) \geq \frac{p_{\pi(j+1)}}{1-p_{\pi(j+1)}} d(\pi(j+1), A_{j-1}^*). \tag{A.7}$$

By adding $\frac{p_{\pi(j+1)}}{1-p_{\pi(j+1)}} d(\pi(j+1), A_j^*)$ on both sides of Equation (A.7), we get

$$\sum_{i=2}^{j+1} \frac{p_{\pi(j)}}{1-p_{\pi(i)}} d(\pi(i), A_{i-1}^*) \geq \frac{p_{\pi(j+1)}}{1-p_{\pi(j+1)}} d(\pi(j+1), A_j^*). \tag{A.8}$$

After moving $\frac{p_{\pi(j+1)}}{1-p_{\pi(j+1)}}$ to the left hand side of Equation (A.8) and exchanging the indexes of i and j, we get

$$d(\pi(i+1), A_i^*) \leq \frac{1-p_{\pi(i+1)}}{p_{\pi(i+1)}} \sum_{j=1}^{i} \frac{p_{\pi(j+1)}}{1-p_{\pi(j+1)}} d(\pi(j), A_{j+1}^*),$$

which completes the proof. □

Corollary 2. *For an optimal solution $A^* = \arg\max \mathcal{S}_+(A)$ to the MAX-SSD problem, and assuming that $p_i \in [a,b]$, for all $i \in U$, it holds that $\mathcal{S}_+(A^*) \leq \frac{b(1-a)}{a(1-b)} \mathcal{H}(A^*)$.*

Proof. Let $A^* = (\pi(i))_{i=1}^n$ denote an optimal sequence. To proof this corollary it suffices to show that $d(\pi(i+1), A_i^*) \leq \frac{b(1-a)}{a(1-b)} d_L(A_{i+1}^*)$, for all $i \in [n-1]$.

Since we assume that $p_i \in [a,b]$, for all $i \in [n]$, we get

$$\begin{aligned}&\frac{1-p_{\pi(i+1)}}{p_{\pi(i+1)}} \sum_{j=1}^{i} \frac{p_{\pi(j+1)}}{1-p_{\pi(j+1)}} d(\pi(j), \pi(j+1)) \\ &\leq \frac{b(1-a)}{a(1-b)} d_L(A_{i+1}^*).\end{aligned} \tag{A.9}$$

Substituting Equation (A.9) into Lemma 4 leads to

$$\begin{aligned}d(\pi(i+1), A_i^*) &\leq \frac{1-p_{\pi(i+1)}}{p_{\pi(i+1)}} \sum_{j=1}^{i} \frac{p_{\pi(j+1)}}{1-p_{\pi(j+1)}} d(\pi(j), \pi(j+1)) \\ &\leq \frac{b(1-a)}{a(1-b)} d_L(A_{i+1}^*),\end{aligned}$$

which completes the proof. □

B ADDITIONAL PRELIMINARIES

In this section, we list a few useful lemmas that we are going to use repeatedly in the following sections. Since they are simple algebraic statements, we state them directly, without proof.

Lemma 7. *Let s_n be the sum of the the first n terms of a geometric series, namely $s_n = \sum_{i=0}^{n-1} ar^i$, where a is a constant, and r is any number such $0 < r < 1$. Then, $s_n = a\frac{1-r^n}{1-r}$.*

Lemma 8. *When r is a constant, $\lim_{n\to\infty} s_n = \frac{a}{1-r}$. In other words, when $n \to \infty$, it is $s_n = \frac{a}{1-r} - \Theta(r^n)$.*

Lemma 9 (Theorem 4.3, Chebyshev's inequality [10])**.** *Let $a_1 \leq \ldots \leq a_n$ and $b_1 \leq \ldots b_n$ be real numbers. Then, it holds*

$$\left(\sum_{i=1}^{n} a_i\right)\left(\sum_{i=1}^{n} b_i\right) \leq n \sum_{i=1}^{n} a_i b_i,$$

and equality holds when $a_1 = \ldots = a_n$ or $b_1 = \ldots = b_n$.

Lemma 10. $\left(1-\frac{1}{n}\right)^n = \frac{1}{e} - \Theta\left(\frac{1}{n}\right)$.

C OMITTED PROOFS FROM SECTION 6

Theorem 3. *Assume that $p_i = p$, for all $i \in U$. Moreover, assume that there is a constant positive number $\epsilon > 0$ such that $p < 1 - \epsilon$. For a fixed value of k, where $2 \le k \le n$, the BkI algorithm yields a $\left(\frac{1}{1-p^{k-1}}(1 + \Theta(p^n))\right)$-approximation for the MaxOHP problem.*

Proof. We start by introducing some additional notation.

We let π be the ordering of U obtained from Algorithm 1, and A be the corresponding sequence. Notice that

$$\mathcal{H}(A) = \sum_{i=1}^{n-1} W_{A_i} d(\pi(i), \pi(i+1)).$$

We let π^o be the optimal ordering for the MaxOHP problem, and A^o be the corresponding optimal sequence. Hence,

$$\mathcal{H}(A^o) = \sum_{i=1}^{n-1} W_{A^o_i} d(\pi^o(i), \pi^o(i+1)).$$

Since $p_i = p$, for all $i \in U$, we notice that $p_{A_j} = p_{A^o_j} = p^j$. Hence, $W_{A_i} = W_{A^o_i} = \sum_{j=i+1}^{n} p^j$. We notice that the coefficient W_{A_i} is completely determined by its position, the actual items being ranked does not change its value, or to be specific,

$$\sum_{j=i+1}^{n} p^j \overset{(a)}{=} p^{i+1}\frac{1-p^{n-i}}{1-p} = \frac{p^{i+1}}{1-p} - \frac{p^{n+1}}{1-p},$$

where equality (a) holds by applying Lemma 7.

Observe that when p is strictly smaller than 1, both W_{A_i} and $W_{A^o_i}$ are dominated by the term $\frac{p^{i+1}}{1-p}$, as $\lim_{n\to\infty} \sum_{j=i+1}^{n} p^j = \frac{p^{i+1}}{1-p}$. For a rigorous calculation, we can write that $\sum_{j=i+1}^{n} p^j = \frac{p^{i+1}}{1-p} - \Theta(p^n)$.

Recall that the BkI algorithm finds the optimal k-item sequence that maximizes the following equation:

$$A_k = \underset{A_k \sqsubseteq \pi(U)}{\arg\max} \sum_{i=1}^{k-1} \frac{p^{i+1}}{1-p} d(\pi(i), \pi(i+1)). \tag{C.1}$$

For the convenience in our calculations, we let

$$\ell_\pi = \sum_{i=1}^{k-1} \frac{p^{i+1}}{1-p} d(\pi(i), \pi(i+1)).$$

We then simplify $\mathcal{H}(A)$ as follows:

$$\begin{aligned}
\mathcal{H}(A) &= \sum_{i=1}^{n-1} \frac{p^{i+1}}{1-p} d(\pi(i), \pi(i+1)) - \sum_{i=1}^{n-1} \frac{p^{n+1}}{1-p} d(\pi(i), \pi(i+1)) \\
&\ge \sum_{i=1}^{k-1} \frac{p^{i+1}}{1-p} d(\pi(i), \pi(i+1)) - \sum_{i=1}^{k-1} \frac{p^{n+1}}{1-p} d(\pi(i), \pi(i+1)) \\
&= \ell_\pi - \frac{p^{n+1}}{1-p} \sum_{i=1}^{k-1} d(\pi(i), \pi(i+1)) \\
&\ge \ell_\pi - p^n \epsilon \sum_{i=1}^{k-1} d(\pi(i), \pi(i+1)) \\
&= \ell_\pi - \Theta(p^n).
\end{aligned} \tag{C.2}$$

Similarly, we simplify $\mathcal{H}(A^o)$:

$$\begin{aligned}
\mathcal{H}(A^o) &= \sum_{i=1}^{n-1} \frac{p^{i+1}}{1-p} d(\pi^o(i), \pi^o(i+1)) - \sum_{i=1}^{n-1} \frac{p^{n+1}}{1-p} d(\pi^o(i), \pi^o(i+1)) \\
&\le \sum_{i=1}^{n-1} \frac{p^{i+1}}{1-p} d(\pi^o(i), \pi^o(i+1)) \\
&= \sum_{t=1}^{T} \sum_{i=(t-1)(k-1)+1}^{t\cdot(k-1)} \frac{p^{i+1}}{1-p} d(\pi^o(i), \pi^o(i+1)) \\
&\le \ell_\pi (1 + p^{k-1} + \cdots + p^{(T-1)(k-1)}) \\
&\le \frac{\ell_\pi}{1-p^{k-1}}.
\end{aligned} \tag{C.3}$$

Combining Equations (C.2) and (C.3), we get

$$\begin{aligned}
\frac{\mathcal{H}(A^o)}{\mathcal{H}(A)} &\le \frac{\frac{\ell_\pi}{1-p^{k-1}}}{\ell_\pi - \Theta(p^n)} \\
&= \frac{1}{1-p^{k-1}} \cdot \frac{\ell_\pi}{\ell_\pi - \Theta(p^n)} \\
&= \frac{1}{1-p^{k-1}} \cdot \frac{\ell_\pi - \Theta(p^n) + \Theta(p^n)}{\ell_\pi - \Theta(p^n)} \\
&= \frac{1}{1-p^{k-1}} \left(1 + \frac{\Theta(p^n)}{\ell_\pi - \Theta(p^n)}\right) \\
&= \frac{1}{1-p^{k-1}} (1 + \Theta(p^n)).
\end{aligned}$$

We have demonstrated that the BkI algorithm provides an asymptotic $\frac{1}{1-p^{k-1}}$-approximation. □

Theorem 4. *Assume that $p_i = p$, for any $i \in U$. Let p be a function of n such that $\lim_{n\to\infty} p = 1$ and $\lim_{n\to\infty} p^n = 0$. Moreover, assume that $p = 1 - \frac{1}{t_n}$, where t_n is a increasing function of n and $t_n = o(n)$. The algorithm GM yields a $\left(\frac{16e^2}{3(e-1)} + O\left(\frac{1}{t_n}\right)\right)$-approximation for the MaxOHP problem.*

Proof. We let π be the ordering of U obtained from GM algorithm, and A be the corresponding sequence. Recall that

$$\mathcal{H}(A) = \sum_{i=1}^{n-1} W_{A_i} d(\pi(i), \pi(i+1)). \tag{C.4}$$

Since p^i decreases as i increases, we have $p^i \ge p^{t_n}$, for all $i \le t_n$. Hence, we obtain a simple lower bound on W_{A_i} for any $i \le t_n$.

$$W_{A_i} = \sum_{j=i+1}^{n} p^j \ge \sum_{j=i+1}^{t_n} p^j \ge (t_n - i) p^{t_n}. \tag{C.5}$$

Recall that, the π we choose satisfies the following two properties, which we are soon going to use to obtain a lower bound of $\mathcal{H}(A)$.

(P1) $d((\pi(2i-1), \pi(2i))) \ge d((\pi(2i+1), \pi(2i+2)))$ for $i < k$, and

(P2) $d((\pi(2i), \pi(2i+1))) \ge \frac{1}{2} d((\pi(2i-1), \pi(2i)))$ for $i \le k$.

Let us simplify $\mathcal{H}(A)$ by applying the above observation. In the following analysis, we define $\mathcal{T}_n \coloneqq \lfloor (t_n - 1)/2 \rfloor$.

$$\begin{aligned}
\mathcal{H}(A) &\overset{(a)}{\geq} \sum_{i=1}^{\mathcal{T}_n} \left(W_{A_{2i-1}} d(\pi(2i-1), \pi(2i)) + W_{A_{2i}} d(\pi(2i), \pi(2i+1)) \right) \\
&\overset{(b)}{\geq} \sum_{i=1}^{\mathcal{T}_n} \left(W_{A_{2i-1}} + \frac{1}{2} W_{A_{2i}} \right) d(\pi(2i-1), \pi(2i)) \\
&\overset{(c)}{\geq} \frac{1}{\mathcal{T}_n} \sum_{i=1}^{\mathcal{T}_n} \left(W_{A_{2i-1}} + \frac{1}{2} W_{A_{2i}} \right) \sum_{i=1}^{\mathcal{T}_n} d(\pi(2i-1), \pi(2i)) \\
&\overset{(d)}{\geq} \frac{p^{t_n}}{\mathcal{T}_n} \sum_{i=1}^{\mathcal{T}_n} \left((t_n - 2i + 1) + \frac{1}{2}(t_n - 2i) \right) \sum_{i=1}^{\mathcal{T}_n} d(\pi(2i-1), \pi(2i)) \\
&= \frac{p^{t_n}}{\mathcal{T}_n} \sum_{i=1}^{\mathcal{T}_n} \left(-3i + 1 + \frac{3}{2} t_n \right) \sum_{i=1}^{\mathcal{T}_n} d(\pi(2i-1), \pi(2i)) \\
&= \left(p^{t_n} \right) \left(\frac{-3 - 3\mathcal{T}_n}{2} + 1 + \frac{3}{2} t_n \right) \sum_{i=1}^{\mathcal{T}_n} d(\pi(2i-1), \pi(2i)).
\end{aligned} \tag{C.6}$$

Inequality (a) holds as we decompose $\mathcal{H}(A)$ into two parts, the series of $W_{A_{2i-1}} d(\pi(2i-1), \pi(2i))$ are related with the top-$\mathcal{T}_n$ matching, and the series of $W_{A_{2i}} d(\pi(2i), \pi(2i+1))$ are related with the edges that connect this matching.

Inequality (b) holds according to the property **(P2)**.

Inequality (c) holds as we apply Lemma 9 by property **(P1)**.

Inequality (d) holds as we reformulate $W_{A_{2i-1}}$ and $W_{A_{2i}}$ according to Equation (C.5).

Let $\ell_{t_n\text{-max}}$ denote the largest weights of a path that consists of t_n nodes chosen from U, i.e.,

$$\ell_{t_n\text{-max}} = \max_{\pi} \sum_{i=1}^{t_n - 1} d(\pi(i), \pi(i+1)).$$

Notice that as $\{\pi(2i-1), \pi(2i)\}_{i=1}^{\mathcal{T}_n}$ is a greedy matching of the first $\mathcal{T}_n$ edges, hence, it consists of at least $\frac{1}{4}$ of the largest weights of the path that consists of $2\mathcal{T}_n$ edges. We can get

$$\sum_{i=1}^{\mathcal{T}_n} d(\pi(2i-1), \pi(2i)) \geq \frac{1}{4} \frac{t_n - 2}{t_n - 1} \ell_{t_n\text{-max}},$$

where the factor $\frac{t_n - 2}{t_n - 1}$ is added as $2\mathcal{T}_n$ can either be $t_n - 1$ or $t_n - 2$.

We substitute the above result into Equation (C.6), and we get

$$\begin{aligned}
\mathcal{H}(A) &= \left(p^{tn} \right) \left(\frac{-3 - 3\mathcal{T}_n}{2} + 1 + \frac{3}{2} t_n \right) \frac{1}{4} \frac{t_n - 2}{t_n - 1} \ell_{t_n\text{-max}} \\
&\overset{(a)}{>} \left(p^{tn} \right) \left(\frac{3}{4} t_n - \frac{1}{2} \right) \frac{1}{4} \frac{t_n - 2}{t_n - 1} \ell_{t_n\text{-max}} \\
&= \left(p^{tn} \right) \left(\frac{3}{16} t_n - \frac{1}{8} \right) \frac{t_n - 2}{t_n - 1} \ell_{t_n\text{-max}},
\end{aligned} \tag{C.7}$$

where inequality (a) holds by substituting an upper bound $\frac{t_n}{2}$ of $\mathcal{T}_n$ into $\mathcal{T}_n$,

We let π^o be the optimal ordering for the MaxOHP, and A^o be the corresponding optimal sequence. Hence,

$$\mathcal{H}(A^o) = \sum_{i=1}^{n-1} W_{A_i^o} d(\pi^o(i), \pi^o(i+1)).$$

We can obtain a simple upper bound on $W_{A_i^o}$ by

$$\begin{aligned}
W_{A_i^o} &= \sum_{j=i+1}^{n} p^j \\
&= \frac{p^{i+1}}{1-p} - \frac{p^{n+1}}{1-p} \\
&\leq \frac{p^{i+1}}{1-p} \\
&\overset{(a)}{=} p^{i+1} t_n \\
&\leq p^{\lfloor \frac{i+1}{t_n} \rfloor \cdot t_n} t_n,
\end{aligned} \tag{C.8}$$

where equality (a) holds as we can use $p = 1 - \frac{1}{t_n}$.

By letting $T = \left\lceil \frac{n}{t_n - 1} \right\rceil$ we can derive an upper bound of $\mathcal{H}(A^o)$ as follows

$$\begin{aligned}
\mathcal{H}(A^o) &= \sum_{t=0}^{T} \sum_{i=(t-1)(t_n-1)+1}^{t \cdot (t_n - 1)} p^{t \cdot t_n} t_n d(\pi^o(i), \pi^o(i+1)) \\
&\leq t_n \cdot \ell_{t_n\text{-max}} \sum_{t=0}^{T} p^{t \cdot t_n} \\
&\leq \frac{t_n \cdot \ell_{t_n\text{-max}}}{1 - p^{t_n}}.
\end{aligned} \tag{C.9}$$

Combining Equations (C.7) and (C.9) leads to

$$\begin{aligned}
\frac{\mathcal{H}(A^o)}{\mathcal{H}(A)} &\leq \frac{\frac{t_n \cdot \ell_{t_n\text{-max}}}{1 - p^{t_n}}}{(p^{tn}) \cdot (\frac{3}{16} t_n - \frac{1}{8}) \frac{t_n - 2}{t_n - 1} \ell_{t_n\text{-max}}} \\
&\overset{(a)}{=} \frac{1}{(1 - \frac{1}{e}) \frac{1}{e} \frac{3}{16} - \Theta(\frac{1}{t_n})} \\
&= \frac{16e^2}{3(e-1)} + O(\frac{1}{t_n}),
\end{aligned}$$

where (a) holds as we apply Lemma 10, and use

$$p^{t_n} = (1 - t_n)^{t_n} = \frac{1}{e} - \Theta\left(\frac{1}{t_n} \right).$$

Thus we have proven that GM is a $\left(\frac{16e^2}{3(e-1)} + O\left(\frac{1}{t_n} \right) \right)$-approximation algorithm.

□

Theorem 5. *Assume that $p_i = p$, for all $i \in U$. Moreover, assume that $\lim_{n\to\infty} p = 1$ and $p^n = c - \frac{1}{\Theta(n)}$, where c is a constant. Then any ordering of the items of U yields a $\left(\frac{1}{c} + \frac{1}{\Theta(n)}\right)$-approximation for the MaxSSD problem.*

Proof. We let π be any ordering of U, and A be the corresponding sequence. We let π^* be the optimal order for the problem MaxSSD, and let A^* be the corresponding sequence. We have

$$\begin{aligned}\mathcal{S}_+(A) &= \sum_{i=1}^{n-1} p_{A_{i+1}} d(\pi(i+1), A_i)\\ &\geq \sum_{i=1}^{n-1} \left(c - \Theta\left(\frac{1}{n}\right)\right) d(\pi(i+1), A_i)\\ &= \left(c - \Theta\left(\frac{1}{n}\right)\right) \sum_{i=1}^{n-1} d(\pi(i+1), A_i)\\ &\overset{(a)}{\geq} \left(c - \Theta\left(\frac{1}{n}\right)\right) \mathcal{S}_+(A^*),\end{aligned}$$

where inequality (a) holds because $\mathcal{S}_+(A^*)$ is not larger than the sum of all pairs of diversities.

Hence,

$$\frac{\mathcal{S}_+(A^*)}{\mathcal{S}_+(A)} \leq \frac{1}{c - \Theta\left(\frac{1}{n}\right)} = \frac{1}{c} + \Theta\left(\frac{1}{n}\right).$$

□

D OMITTED PROOFS FROM SECTION 7

Theorem 6. *Assume $p_i \in [a, b]$, for all $i \in U$, for $0 \leq a \leq b < 1$. Then, Algorithm BkI with BI_k defined as in Equation (7.1) yields a $\frac{a^2+kb^k}{a^2(1-b-2b^{k-1})}$-approximation guarantee for the MaxOHP problem for any constant integer $2 \leq k \leq n$.*

Proof. Let π be the ordering of U obtained from our algorithm. Let $A_\pi = (\pi(i))_{i=1}^n$ denote the ordered sequence according to π and write $A = A_\pi$. Recall that

$$\mathcal{H}(A) = \sum_{i=1}^{n-1} W_{A_i} d(\pi(i), \pi(i+1)).$$

Let π^o be the optimal ordering for the MaxOHP. Let $A_{\pi^o} = (\pi^o(i))_{i=1}^n$ and write $A^o = A_{\pi^o}$. For the optimal ordering A^o, let $\mathcal{H}(A^o) = \sum_{i=1}^{n-1} W_{A_i^o} d(\pi^o(i), \pi^o(i+1))$.

Recall that

$$W_{A_i} := \sum_{j=i+1}^{n} p_{A_j}.$$

Hence,

$$\begin{aligned}\mathcal{H}(A) &= \sum_{i=1}^{n-1} W_{A_i} d(\pi(i), \pi(i+1))\\ &\geq \sum_{i=1}^{k-1} W_{A_i} d(\pi(i), \pi(i+1))\\ &\geq \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} p_{A_j} d(\pi(i), \pi(i+1)).\end{aligned}$$

We define

$$\ell_k := \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} p_{A_j} d(\pi(i), \pi(i+1)).$$

It is not hard to see that ℓ_k is the function value maximized in Equation (7.1). Let $T := \left\lceil \frac{n-1}{k-1} \right\rceil$.

$$\begin{aligned}\mathcal{H}(A^o) &= \sum_{i=1}^{n-1} W_{A_i^o} d(\pi^o(i), \pi^o(i+1))\\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} p_{A_j^o} d(\pi^o(i), \pi^o(i+1))\\ &= \sum_{t=1}^{T} \sum_{i=(t-1)(k-1)+1}^{t(k-1)} \sum_{j=i+1}^{n} p_{A_j^o} d(\pi^o(i), \pi^o(i+1))\\ &= \sum_{t=1}^{T} \sum_{i=(t-1)(k-1)+1}^{t(k-1)} \left(\sum_{j=i+1}^{t\cdot k} p_{A_j^o} + \sum_{j=t\cdot k+1}^{n} p_{A_j^o} \right) d(\pi^o(i), \pi^o(i+1))\end{aligned}$$

Notice that for each fixed t, $\mathcal{H}(A^o)$ is the sum of two parts:

$$\sum_{i=(t-1)(k-1)+1}^{t(k-1)} \sum_{j=i+1}^{t\cdot k} p_{A_j^o} d(\pi^o(i), \pi^o(i+1)) \tag{D.1}$$

$$\sum_{i=(t-1)(k-1)+1}^{t(k-1)} \sum_{j=t\cdot k+1}^{n} p_{A_j^o} d(\pi^o(i), \pi^o(i+1)) \tag{D.2}$$

Let us first discuss the part (D.1)

$$\begin{aligned}&\sum_{i=(t-1)(k-1)+1}^{t(k-1)} \sum_{j=i+1}^{t\cdot k} p_{A_j^o} d(\pi^o(i), \pi^o(i+1))\\ &\overset{(a)}{\leq} \ell_k p_{A^o_{(t-1)(k-1)}}\\ &\leq \ell_k b^{(t-1)(k-1)}.\end{aligned} \tag{D.3}$$

Similarly, for part (D.2)

$$\begin{aligned}&\sum_{i=(t-1)(k-1)+1}^{t(k-1)} \sum_{j=t\cdot k+1}^{n} p_{A_j^o} d(\pi^o(i), \pi^o(i+1))\\ &\overset{(b)}{\leq} \sum_{i=(t-1)(k-1)+1}^{t(k-1)} \sum_{j=t\cdot k+1}^{n} b^j d(\pi^o(i), \pi^o(i+1))\\ &\leq \sum_{i=(t-1)(k-1)+1}^{t(k-1)} \frac{b^{t\cdot k+1}}{1-b} d(\pi^o(i), \pi^o(i+1))\\ &\leq (k-1) d_{\max} \frac{b^{t\cdot k+1}}{1-b},\end{aligned} \tag{D.4}$$

where $d_{\max} := \max_{u,v\in U} d(u, v)$. Inequality (a) holds by the definition of ℓ_k and our algorithm, and inequality (b) holds because $p_{A_j^o} \leq b^j$.

We combine Equations (D.3) and (D.4) and we continue to bound $\mathcal{H}(A^o)$:

$$\begin{aligned}\mathcal{H}(A^o) &\leq \sum_{t=1}^{T}\left(\ell_k b^{(t-1)(k-1)} + (k-1)d_{\max}\frac{b^{t\cdot k+1}}{1-b}\right) \\ &\leq \frac{\ell_k}{1-b^{k-1}} + (k-1)d_{\max}\frac{b^{k+1}}{(1-b)(1-b^k)},\end{aligned} \tag{D.5}$$

Hence,

$$\frac{\mathcal{H}(A^o)}{\mathcal{H}(A)} \leq \frac{1}{1-b^{k-1}} + \frac{(k-1)d_{\max}}{\ell_k}\cdot\frac{b^{k+1}}{(1-b)(1-b^k)}. \tag{D.6}$$

Next, we derive an upper bound of the second term of Equation (D.6). To do so, we first derive an upper bound of ℓ_k. Let $\hat{\pi}$ be any order, which places the edge with edge weight $d_{\max}$ to the first two positions. Let $\hat{A}$ be the corresponding sequence. We have:

$$\ell_k \overset{(a)}{\geq} \ell_2 \overset{(b)}{\geq} p_{\hat{A}_2} d(\hat{\pi}(1), \hat{\pi}(2)) \geq a^2 d_{\max}. \tag{D.7}$$

Notice that inequality (a) holds as ℓ_k should consist of at least one edge, and inequality (b) holds because of the optimality of ℓ_k. Thus,

$$\begin{aligned}\frac{\mathcal{H}(A^o)}{\mathcal{H}(A)} &\leq \frac{1}{1-b^{k-1}} + \frac{(k-1)}{a^2}\frac{b^{k+1}}{(1-b)(1-b^k)} \\ &\leq \frac{a^2 + (k-1)b^{k+1}}{a^2(1-b-b^{k-1}-b^k)} \\ &\leq \frac{a^2 + kb^k}{a^2(1-b-2b^{k-1})}.\end{aligned}$$

This completes the proof. □

E MORE EXPERIMENTAL RESULTS

Run times of our methods and the baselines on different datasets are presented in Table 5.

Table 5: Run times of proposed algorithms and baselines (sec)

Dataset	Random	MSD	MMR	DPP	DUM	BkI (k=2),	BkI-H (k=2)	BkI (k=3)	BkI (k=4)
Coat	0.93	1.32	1.68	1.56	1.23	2.22	1.19	1.96	40.88
Netflix-Prize	10.91	16.19	15.18	28.61	27.02	26.08	13.18	14.47	69.81
Movielens	46.61	119.05	89.79	135.14	142.92	260.03	81.54	87.24	189.52
Yahoo-R2	61.38	213.13	119.75	237.30	237.28	488.53	125.82	135.44	746.87
KuaiRec-2.0	28.85	45.67	77.36	447.59	490.80	87.38	43.09	45.21	57.26