Diversifying recommendations on sequences of sets

Sepideh Nikookar, Mohammadreza Esfandiari, Ria Mae Borromeo, Paras Sakharkar, Sihem Amer-Yahia, Senjuti Basu Roy

Noname manuscript No.
(will be inserted by the editor)

Diversifying Recommendations on Sequences of Sets (Author's copy)

Sepideh Nikookar · Mohammadreza Esfandiari · Ria Mae Borromeo · Paras Sakharkar · Sihem Amer-Yahia · Senjuti Basu Roy

the date of receipt and acceptance should be inserted later

Abstract Diversifying recommendations on a sequence of sets (or sessions) of items captures a variety of applications. Notable examples include recommending online music playlists, where a session is a channel and multiple channels are listened to in sequence, or recommending tasks in crowdsourcing, where a session is a set of tasks and multiple task sessions are completed in sequence. Item diversity can be defined in more than one way, e.g., as a genre diversity for music, or as a function of reward in crowdsourcing. A user who engages in multiple sessions may intend to experience diversity within and/or across sessions. *Intra session diversity* is set-based, whereas, *Inter session diversity* is naturally sequence-based. This novel formulation gives rise to four bi-objective problems with the goal of minimizing or maximizing *Inter* and *Intra* diversities. We prove hardness and develop efficient algorithms with theoretical guarantees. Our experiments with human subjects on two real datasets show that our diversity formulations do serve different user needs, and yield high user satisfaction. Our large scale experiments on real and synthetic data empirically demonstrate that our solutions satisfy our theoretical bounds and are highly scalable, compared to baselines.

Keywords Recommendation systems · Diversity algorithms

Sepideh Nikookar
New Jersey Institute of Technology
E-mail: sn627@njit.edu

Mohammadreza Esfandiari
New Jersey Institute of Technology
E-mail: me76@njit.edu

Ria Mae Borromeo
University of the Philippines
E-mail: rhborromeo@up.edu.ph

Paras Sakharkar
New Jersey Institute of Technology
E-mail: ps863@njit.edu

Sihem Amer-Yahia
CNRS, Univ. Grenoble Alpes
E-mail: sihem.amer-yahia@univ-grenoble-alpes.fr

Senjuti Basu Roy
New Jersey Institute of Technology
E-mail: senjutib@njit.edu

1 Introduction

Diversity aims to improve user experience by addressing the problem of over-specialization, where a user receives recommendations that are often too similar to each other. To create online music playlists, users organize songs into channels and listen to a few songs within the same channel before switching to the next channels to listen to other artists in the same genre, or to experience different music styles. On crowdsourcing platforms, workers complete a small set of tasks at a time (session) and *sequences of sessions* within a finite time (for example, half a day). Diversifying recommendations inside (*Intra*) and across (*Inter*) sessions is natural for such applications to improve user satisfaction and engagement.

Recommending playlists during a long-drive may need to minimize both *Intra* and *Inter* session diversities to generate songs by the same artist within a channel and similar beats across channels. Contrarily, designing playlists for a theme party is best done by composing songs from the same period within a channel (90's, 60's, etc) and different styles across channels (thereby minimizing *Intra* diversity on release date within a session and maximizing *Inter* diversity on style across sessions). Similarly, in crowdsourcing, it may be

ideal to recommend tasks requiring similar skills within a session and different completion times across sessions. Whereas, workers who have multiple expertise may be recommended tasks with different skills in a session and different rewards across sessions. More generally, applications may require minimization or maximization of *Intra* and *Inter* diversities.

These aforementioned scenarios have three things in common: first, diversity needs to be accounted for in the design of a sequence of sets of recommendations. Second, both minimization and maximization of diversity are meaningful. Finally, the dimensions on which *Intra* and *Inter* session diversities are expressed are item features that may not be related - hence they cannot be combined. We present a framework that satisfies all three requirements focusing purely on diversity and assuming that the items consumed by the framework are always suitable (relevant) to the user.

Our goal is to develop an algorithmic framework for *Inter* and *Intra* session diversities in tandem with the goal to recommend k sessions to a user, with a small number l of relevant items in each, yielding a total of $N = k \times l$ items [1]. *Intra* and *Inter* diversities can be either minimized or maximized which gives rise to *a bi-objective formalism to express four problem variants* **(Section 2.2)**. We also study the relaxed version of our proposed framework where the sessions are of varying lengths and the total number of items recommended to the user is a subset of N items. *To the best of our knowledge, our work is the first attempt to combine set and sequence diversities, two problems extensively studied individually in search and recommendation* [2, 6, 13, 22, 28, 29, 34, 37–39, 45, 48–51].

Our second contribution is theoretical. We first study each of the *Intra* and *Inter* diversity optimization problems individually and find that irrespective of minimization or maximization, the *Inter* problem is NP-hard **(Section 2.3)**. We also prove that the *Intra* minimization problem can be optimally solved in polynomial time. However, the complexity of each bi-objective problem remains NP-hard (because *Inter* optimization is NP-hard).

Our third contribution is algorithmic **(Section 3)**. We design principled solutions with provable guarantees for *Intra* and *Inter* problems individually. Algorithm `Ex-Min-Intra` runs in $O(NlogN)$ time and produces an exact solution of the *Min-Intra* problem. For *Min-Inter* and *Max-Inter*, algorithms `Ap-Min-Inter` and `Ap-Max-Inter` achieve $4 - 2/k$- and $\frac{1}{2}$-approximation factors, respectively. We also design an efficient $\frac{1}{2-1/k}$-approximation algorithm `Ap-Max-Intra` to solve the *Max-Intra* problem.

Additionally, we investigate an alternative formulation **(Section 2.4)** of all four problems to a corresponding constrained optimization problem, with the goal of obtaining one point from the Pareto front. *The idea is to optimize Inter diversity, subject to constraining Intra diversity.* The constraint on *Intra* is obtained by solving the *Intra* optimization first. There exists more than one benefit to this approach. First, in one of the two cases (i.e., Minimization) *Intra* is *tractable and easier to solve*, therefore, finding the optimal constraint value is computationally efficient. More importantly, the constrained optimization problem aims at finding one point in the Pareto front, which is perfectly acceptable, as the points in the Pareto front are qualitatively indistinguishable (unless further information is available). When *Inter* problems are optimized subject to constraining *Intra*, the combined solutions hold guarantees for 2 out of the 4 problems **(Section 3.4)**. Tables 2 and 3 summarize our theoretical and algorithmic results.

Our last contribution is experimental. We consider two real world applications and conduct multiple experiments involving 400 human subjects, as summarized in Table 4, for music and task recommendation. We additionally perform large scale experiments using real and simulation data to validate the properties of our designed algorithms. In music recommendation **(Section 4.1)**, *our results highlight, with statistical significance, that user satisfaction is higher when playlists are recommended considering diversity and the preferred diversity scenario depends on the underlying context.* In task recommendation, *our results show the benefit of diversification in task sessions across different session gaps or time intervals between sessions. Our algorithms achieve higher quality and worker satisfaction with statistical significance, than a baseline with No diversity in all the specified session gaps.*

(Section 4.2) investigates approximation factors and the scalability of our algorithms against several non-trivial baselines. We observe that in most cases, our algorithms produce approximation factors that are very close to 1. For the cases where the approximation factor is slightly worse, the solution is close enough. Finally, we also observe that our approach is faster and highly scalable when varying the number of items and the number of sessions considering different data distributions.

We present related works in Section 5, and conclude in Section 6.

[1] A preliminary version of this work has got accepted in The Web Conference, 2021 [20].

Task	Skill	Reward	Task	Skill	Reward	Task	Skill	Reward
$\mathbf{t_1}$	0.5	0.3	$\mathbf{t_2}$	0.51	0.4	$\mathbf{t_3}$	0.54	0.49
$\mathbf{t_4}$	0.59	0.50	$\mathbf{t_5}$	0.6	0.23	$\mathbf{t_6}$	0.63	0.4
$\mathbf{t_7}$	0.69	0.1	$\mathbf{t_8}$	0.7	0.60	$\mathbf{t_9}$	0.79	0.36
$\mathbf{t_{10}}$	0.8	0.12	$\mathbf{t_{11}}$	0.89	0.55	$\mathbf{t_{12}}$	0.93	0.34

Table 1 Task Skill and Reward

2 Formalism and Problem Analysis

For the purpose of illustration, we describe a simple running example on recommending task sessions in crowdsourcing. Same example could be used for the streaming music.

Example 1 Consider a set of $N = 12$ tasks, which are most relevant to a specific worker. Table 1 shows two dimensions of these tasks. The first dimension is the skill requirement of the task as provided by the requester. The second dimension is the task reward. We want to recommend 4 ($=k$) sessions, each containing 3 ($=l$) tasks.

2.1 Data Model

Item. An item has a set of dimensions. t_i^d represents the d-th dimension of the i-th item. Using Example 1, task t_1 is represented by two dimensions, $< 0.5, 0.30 >$. In the case of a song, examples of dimensions are artist, vibe, genre, etc.

Session. A session s consists of a set of l items that can be consumed in any order.

Sequence. A sequence of sessions is an ordering of k sessions $S =< s_1, s_2, \ldots, s_k >$ where sessions are presented to a user one after another.

***Intra* Diversity.** Given a dimension d, the diversity of a set of items in a single session s is referred to as *Intra* and defined by capturing how each item in that session deviates from the average, considering d, and taking an aggregate over l items as follows:

$$Intra^d(s) = \sum_{i=1}^{l} (t_i^d - \mu_s^d)^2 \quad (1)$$

where t_i^d is the value of dimension d of item t_i and μ_s^d is the average of d values of items in session s. *Intra* essentially captures variance of a set of items for a dimension d. Following Example 1, if the session s_1 consists of $\{t_1, t_3, t_5\}$, then $Intra^{skill}(s_1) = 0.005$.

***Inter* Diversity.** The diversity of items between two consecutive sessions is referred to as *Inter* and is defined for two consecutive sessions for a dimension d as follows:

$$Inter^d(s_i, s_{i+1}) = (\mu_{s_i}^d - \mu_{s_{i+1}}^d)^2 \quad (2)$$

which captures the difference between the average of two consecutive sessions. Given $S =< \{t_1, t_3, t_5\}, \{t_2, t_4, t_6\}, \{t_7, t_8, t_9\} >$, $Inter^{Reward}(S) = (0.34 - 0.433)^2 + (0.433 - 0.35)^2 = 0.015$ using Example 1 .

Changing the aggregation function from square to exact definition of variance (i.e., divide it by the number of items in the session), taking square root of the current definition, or changing the solutions to standard deviation will not require any changes in the solution and approximation factor, because these definitions are technically equivalent. In fact, the approximation factors remain unaltered for many popular distance functions that are part of the Minkowski family, such as, L_1, L_2, and L_∞. Other set-based [2] and sequence-based [51] definitions could be considered in future work.

For the simplicity of illustration, we use one dimension at a time to model diversity. For all practical purposes, both *Intra* and *Inter* dimensions could be designed to reflect multiple attributes by combining them and allowing overlap.

We explicitly chose to handle one attribute at a time because we believe that diversity becomes more difficult to perceive by users when combining several attributes. That is further exacerbated by the fact that users have to perceive *Intra* and *Inter* diversities at once. The use of a single attribute for *Intra* and for *Inter* allowed us to focus on the algorithmic and theoretical contributions. There is however a workaround to reduce any number of dimensions to one for each type of diversity by combining their values with a weighted linear function as in MMR [11].

2.2 Problem Definitions

We formalize our problems and propose to do that in two stages: we first focus on producing Fixed Length Sessions that consume all input items (Section 2.2.1), we then relax our problem to produce Variable Length Sessions that consume only a subset of input items (Section 2.2.2). This allows us to study Fixed and Variable Length Sessions in conjunction to consuming all versus some input items. The problems of Fixed Length Sessions with all input items and the problem of Variable Length Sessions with subset of input items are omitted as they are subsumed by the ones formalized in this paper.

2.2.1 Fixed Length Sessions

Given N items, we are interested in finding a sequence $S =< s_1, \ldots, s_k >$ of k sessions, each consisting of l

items. We consider 4 problem variants all of which are instances of a general problem formalized as follows:

Optimize-Intra, Optimize-Inter. Given a set of N items with two dimensions of interest d and d' on *Intra* and *Inter* respectively, we are interested in creating a sequence $S =< s_1, ..., s_k >$ of k sessions, each containing l items, s.t. $N = k \times l$ and

$$\begin{aligned} &\underset{S}{\text{optimize}} \sum_{i=1}^{k} (Intra^d(s_i)) \\ &\underset{S}{\text{optimize}} \sum_{i=1}^{k-1} (Inter^{d'}(s_i, s_{i+1})) \\ &\text{s.t.} \\ &\quad |S| = k,\ |s_i| = l, N = k \times l \end{aligned} \tag{3}$$

2.2.2 Variable Length Sessions

Given N items, we are interested in finding a sequence $S =< s_1, \ldots, s_k >$ of k sessions, with length $L =< l_1, \ldots, l_k >\ \ s.t.\ l_i \leq l;\ \forall i = 1, \ldots, k$. We consider 4 problem variants all of which are instances of a general problem formalized as follows:

Optimize-Intra, Optimize-Inter. Given a set of N items with two dimensions of interest d and d' on *Intra* and *Inter* respectively, we are interested in recommending a subset of items by creating a sequence $S =< s_1, ..., s_k >$ of k sessions with length $L =< l_1, \ldots, l_k >\ s.t.\ l_i \leq l;\ \forall i = 1, \ldots, k$, and $N = k \times l$

$$\begin{aligned} &\underset{S}{\text{optimize}} \sum_{i=1}^{k} (Intra^d(s_i)) \\ &\underset{S}{\text{optimize}} \sum_{i=1}^{k-1} (Inter^{d'}(s_i, s_{i+1})) \\ &\text{s.t.} \\ &\quad |S| = k,\ |s_i| = l_i,\ l_i \leq l,\ \sum_{\forall i} l_i < N,\ N = k \times l \end{aligned} \tag{4}$$

We refer to Section 4.2.5 for further details.

2.3 Analysis of the Problems Considering Fixed Length Sessions

We analyze the complexity of *Intra* and *Inter* diversities. This exercise allows us to analyze the nature of these problems and sheds light on designing principled solutions.

2.3.1 Intra Diversity Optimization

Theorem 1 *Min-Intra is Polynomial time solvable.*

Proof Minimizing *Intra* diversity is akin to grouping a set of points in a line to produce the smallest aggregated variance. This requires sorting the points based on the *Intra* dimension d and grouping every l points to create a session. Clearly, this is polynomial time solvable.

Theorem 2 *Optimizing Max-Intra is NP-Hard.*

Proof The proof of this theorem uses another claim that we prove later (Theorem 6). This latter theorem formally proves that *Max-Intra* happens ($\sum_{i=1}^{k}(Intra(s_i))$ is maximized) if the mean of each session is equal (or very close) to the global mean of all N items for the specific dimension d. We omit the superscript d from the proof and t_i is considered as the value of the item t_i for dimension d. Since groups have the same size l, the problem is akin to finding groups of items whose sum is equal:

$$\sum_{t_i \in s_1} t_i = \sum_{t_i \in s_2} t_i = \cdots = \sum_{t_i \in s_k} t_i \tag{5}$$

To prove NP-hardness we reduce an instance of the k-Equal Subset Sum of Equal Cardinality Problem (k-`ESSEC`) [14] to an instance of *Max-Intra*, as follows. Given an instance of k-`ESSEC` with $P = \{a_1, ..., a_N\}$ which are N positive integers and k, we set the items $t_i = a_i$ and k remains the same. A solution to the k-`ESSEC` with k disjoint subsets, each with equal value $sum(s_1) = sum(s_2) = \ldots = sum(s_k)$ occurs, iff a solution of the *Max-Intra* exists with $l = N/k$ and $\mu_{s_i} = \mu_{s_{global}} = \frac{\sum_{i=1}^{N} a_i}{N}$.

2.3.2 Inter Diversity Optimization

The *Inter* diversity problem aims to find a sequence of k sessions of length l that will optimize the aggregated *Inter* distance computed on a dimension d over all k sessions in that sequence.

Theorem 3 *Inter Problem (both Min and Max) is NP-Hard.*

Proof (Sketch) We show the NP-hardness for the *Min-Inter* case, and the maximization works analogously. To prove the NP-hardness of the *Min-Inter* problem, we reduce an instance of the known NP-hard problem Hamiltonian Path problem [23] to an instance of the *Min-Inter* problem. Consider an instance of the Hamiltonian Path problem with $G = (V, E)$, where V is the set of nodes and E is the set of edges. Each node $v_i \in V$

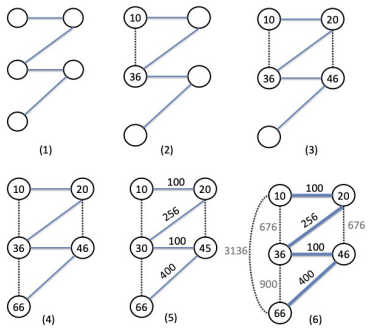


Fig. 1 Reduction: Hamiltonian Path to the *Inter* problem

represents l items with the same value on the dimension of interest. Essentially, these l items form a session. For assigning the *Inter* diversity of two sessions, we first deal with the non-edges in G. For each edge $(v_i, v_j) \notin E$, we set the μ_{s_i} and μ_{s_j} such that $||\mu_{s_i} - \mu_{s_j}|| > X$ (where X is an integer) and for each edge $(v_i, v_j) \in E$, we create $||\mu_{s_i} - \mu_{s_j}|| \leq X$. This creates an instance of *Min-Inter* problem with $|V|$ (i.e., k for *Min-Inter*) sessions, each with l items. Clearly, this reduction can be done in polynomial time. Figure 1 shows such a reduction from an example graph, where $X = 15$. Now a Hamiltonian Path exists in G, iff *Min-Inter* value is $< X^2 \times |V|$.

Theorem 4 *The bi-objective optimization problems combining Intra and Inter diversity are all NP-Hard.*

Proof (Sketch) We omit the formal proof for brevity - but it is easy to show that the NP-hardness remains for each of the 4 bi-objective problems, since the individual optimization problems are NP-hard.

2.4 Modified Problem Definitions of Fixed Length Sessions

As proved in Theorem 4, each of the 4 bi-objective optimization problems are NP-hard. In fact 2 ((*Min-Inter, Max-Intra*) and (*Max-Inter, Max-Intra*)) out of the 4 problems are NP-hard on both objectives. Upon careful investigation, we propose an alternative formulation of each of these bi-objective problems to a corresponding constrained optimization problem, with the goal of obtaining one point from the Pareto front. *The idea is to optimize Inter diversity, subject to the constraint of Intra diversity.*

The constraint on *Intra* is obtained by solving the *Intra* optimization first. There exists more than one benefit to this approach. First, in one of the two cases (i.e., Minimization) *Intra* is *tractable and easier to solve*, therefore, coming up with the optimal value of the constraint is computationally efficient. More importantly, the constrained optimization problem aims at finding one point in the Pareto front, which is perfectly acceptable, as the points in the Pareto front are qualitatively indistinguishable (unless further information is available).

Algorithm	*Running Time*	*Approx Factor*
`Ex-Min-Intra`	$O(NlogN)$	Exact
`Ap-Max-Intra`	$O(NlogN + Nl)$	$\frac{1}{2-1/k}$
`Ap-Min-Inter`	$O(NlogN + k^2 + logk)$	$4 - 2/k$
`Ap-Max-Inter`	$O(NlogN + k^2 + logk)$	$1/2$

Table 2 Optimization Algorithms and Results for Fixed Length Sessions

$$\begin{aligned} &\min_{S}(\max) \sum_{i=1}^{k-1} (Inter^{d_2}(s_i, s_{i+1})) \\ &\quad \text{s.t.} \\ &\quad \sum_{i=1}^{k} (Intra(s_i))x <= OPT_{Intra^{d_1}} \\ &\quad |S| = k,\ |s_i| = l, N = k \times l \end{aligned} \tag{6}$$

where OPT_{Intra} is the optimal solution of the *Intra* problem.

Using Example 1, the sequence

$$S =< \{t_5, t_6, t_7\}, \{t_1, t_2, t_3\}, \{t_9, t_{10}, t_{11}\} >$$

minimizes the $Intra^{Skill}$ score but at the same time maximizes the $Inter^{Reward}$ score whereas

$$S' =< \{t_1, t_2, t_3\}, \{t_9, t_{10}, t_{11}\}, \{t_5, t_6, t_7\} >$$

minimizes the $Intra^{Skill}$ and minimizes the $Inter^{Reward}$.

3 Optimization Algorithms

We design optimization algorithms for the *Intra* and *Inter* problems individually, following which, we study how to solve the constrained optimization problem (Equation 6). Table 2 summarizes our technical results.

3.1 Algorithm *Min-Intra*

3.1.1 Fixed Length Sessions

The objective here is to design k sessions, each of length l, such that the aggregated *Intra* diversity over the k

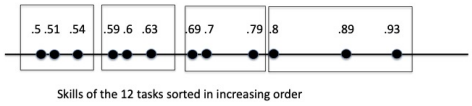


Fig. 2 Sorted *Intra* Diversity of Skills

sessions is minimized. Specifically, if there are l values associated with a dimension in a session, the *Intra* diversity is the variance of those points that is to be minimized here.

With an abstract representation, once sorted, the dimension values of N items, fall on a line, as shown in Figure 2. Therefore, if the aggregated variance is to be minimized, it is intuitive that the sessions need to be formed by grouping l values that are closest to each other.

Thus our proposed `Exact-Min-Intra` algorithm for minimizing *Intra* diversity first sorts the values of the dimension of interest. After that, it starts from the smallest value and finds each consecutive l points to form a session.

Theorem 5 *Algorithm* `Exact-Min-Intra` *is exact.*

Proof (Sketch) Let us assume that our algorithm does not produce an exact solution. That means there exists another algorithm which produces a solution with smaller *Intra* diversity than that of `Exact-Min-Intra`. Suppose this other algorithm uses another way to create the sessions. Of course, this is different from sorting the items in increasing value of the dimension of interests and grouping each l of them starting from the smallest one. However, that is a contradiction because then the latter algorithm will have larger *Min-Intra* value, as l non-consecutive points will have higher variance than consecutive ones. Hence the proof.

Lemma 1 *Algorithm* `Exact-Min-Intra` *runs in* $O(N\ logN)$.

Proof Since the only required operation is sorting, the running time of the algorithm will take $O(NlogN)$.

3.1.2 Variable Length Sessions

For the Variable Length Sessions problem, we group the items depending on the specified input length after sorting them. To clarify more, in Example 1, if we are given $[2, 3, 2, 3]$ as the sessions' length input, we choose the first two items after sorting as the first session, then the following three items as the second session, and so on.

3.2 Algorithm *Max-Intra*

As proved in Section 2.3, *Max-Intra* is NP-hard. What makes it computationally intractable is that when the objective is to maximize variance, the search space has to be combinatorially explored.

We show that *Max-Intra* is optimized when all sessions have the same mean, which is equal to the global mean μ_{global}. This proof is critical, as it helps us design our solution. Theorem 6 has the formal statement.

Theorem 6 $\sum_{i=1}^{k}(Intra(s_i))$ *is maximized when*

$$\mu^d_{s_1} = \mu^d_{s_2}, \ldots = \mu^d_{s_k} = \mu_{global} \tag{7}$$

Proof The theorem states that the objective is maximized when the means of all sessions are equal, which in turn are equal to the global mean. It is indeed true that when $\mu^d_{s_1} = \mu^d_{s_2}, \ldots = \mu^d_{s_k}$, the global mean $\mu_{global} = \frac{1}{N}\sum_{j=1}^{N}(t^d_j) = \frac{k \times \mu^d_{s_i}}{k} = \mu^d_{s_i}$

Our intention is to prove that $\sum_{i=1}^{k}(Intra(s_i))$ is maximized when this aforementioned scenario occurs. For ease of exposition, we omit the superscript d from the proof.

We do the proof by the method of contradiction. Consider two different sets of k sessions, S and S'. For $S = s_1, s_2, \ldots, s_k$, we have $\mu_{s_1} = \frac{1}{l}\sum_{t \in s_1} t$ and similarly for other $s_i \in S$. For $S' = s'_1, s'_2, \ldots, s'_k$ where

$$\mu_{s'_1} = \mu_{s'_2} = \ldots = \mu_{s'_k} = \mu_{global} = \frac{1}{k * l}\sum t \tag{8}$$

We also assume, $Intra(S) > Intra(S')$.

$$\begin{aligned} Intra(S) &= \sum_{s_i} Intra(s_i) \\ &= \sum_{t \in s_1}(t - \mu_{s_1})^2 + \ldots + \sum_{t \in s_k}(t - \mu_{s_k})^2 \\ &= \sum_{i=1}^{N} t_i^2 - l(\mu^2_{s_1} + \mu^2_{s_2} + \ldots + \mu^2_{s_k}) \end{aligned} \tag{9}$$

$$Intra(S') = \sum_{i=1}^{N} t_i^2 - kl\mu^2_{global} \tag{10}$$

According to our assumption, $Intra(S) > Intra(S')$ this means that,

$$\sum_{i=1}^{N} t_i^2 - l(\mu^2_{s_1} + \mu^2_{s_2} + \ldots + \mu^2_{s_k}) > \sum_{i=1}^{N} t_i^2 - kl\mu^2_{global} \tag{11}$$

which after considering $\mu_{opt} = \frac{\mu_{s_1} + \mu_{s_2} + \ldots + \mu_{s_k}}{k}$ we get,

$$\mu^2_{s_1} + \mu^2_{s_2} + \ldots + \mu^2_{s_k} < 0 \tag{12}$$

which is a clear contradiction, hence the proof.

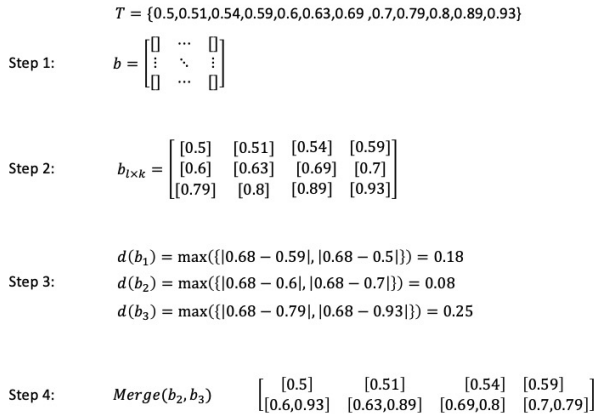


Fig. 3 `Ap-Max-Intra` steps on Example 1

3.2.1 Fixed Length Sessions

Theorem 6 provides a useful insight, that is, to maximize the *Intra*, we need to form the k sessions in such a way that the means of all the sessions are equal or very close to each other. Algorithm `Ap-Max-Intra` is iterative and greedy and it relies on this principle to create sessions that satisfy this property. First, it creates l bins, each has k different slots. The bins are then initialized so that each contains a subset of k items from the set of items that are sorted in ascending order. Then, in the third step, each of the l bins are scored using a scoring function described in Definition 1 , which captures the maximum difference between the average of all items and the ones in each bin. Finally, it merges the two bins with the highest and lowest scores greedily. The final two steps are repeated iteratively. This process is repeated for $l-1$ times.

Definition 1 (**Score of the i-th bin:**)

$$d(b_i) = \max\{|\mu_{global} - \underset{\forall j}{\operatorname{argmax}}\, b_{ij}|, |\mu_{global} - \underset{\forall j}{\operatorname{argmin}}\, b_{ij}|\}$$

This scoring function captures the largest difference between items in a bin and the global average, allowing the highest and lowest scoring bins to be merged. If we do this, as we proved in Theorem 6, the sessions created at the last step have an average near to μ_{global}, which maximizes the *Intra* value.

To illustrate the solution further, b_{ij} represents the j-th slot in bin i, which is kept as a placeholder for j-th session. To initialize the bins, we first sort the items in an increasing order on the dimension of interests. Next, in the i-th bin $1 \leq i \leq l$, we put the sorted items $t_{(i)*k+j}$ in b_{ij}. Using Example 1, this amounts to creating 3 bins of tasks where $b_1 = \{[t_1], [t_2], [t_3], [t_4]\}$, $b_2 = \{[t_5], [t_6], [t_7], [t_8]\}$, and $b_3 = \{[t_9], [t_{10}], [t_{11}], [t_{12}]\}$. In step 3, each bin is scored, based on $d(b_i)$, as presented in Definition 1. Then two bins i and j are merged that have the largest and smallest score respectively. Going back to the Example 1, the scores are calculated as follows $d(b_1) = 0.18$, $d(b_2) = 0.08$, and $d(b_3) = 0.25$ and b_2 and b_3 are merged. Figure 3 details these steps.

To merge b with b', where b has the largest score and b' has the smallest score, we create a new bin b^{merge} such that, b^{merge}_{ij} contains the m-th smallest items of b and m-th largest items of b' $(1 \leq m \leq k)$. Considering Example 1, the new bin b^{merge} is created by combining b_2 and b_3 , such that

$$b^{merge} = \{[t_5, t_{12}], [t_6, t_{11}], [t_7, t_{10}], [t_8, t_9]\}$$

This process is then repeated until only a single bin is left.

Algorithm 1 Algorithm `Ap-Max-Intra`

Require: N, Number of sessions k, Length of session l

1: $\mu_{global} \leftarrow$ Average of all items
2: Initialize l bins each with k slots $\leftarrow$
3: $b_i \leftarrow \{b_{i1} = [t_{il+1}], b_{i2} = [t_{il+2}, ..., b_{ik} = [t_{il+k}]]\}$
4: **while** number of bins > 1 **do**
5: pick b_i and b_j with the largest and smallest scores
6: b^{merge}=merge b_i and b_j
7: Delete b_i and b_j
8: number of bins $\leftarrow l-1$
9: Return the final merged bin

Theorem 7 `Ap-Max-Intra` *runs in* $O(NlogN + Nl)$.

Proof Getting the average of the items takes $O(N)$. The partitioning of items into k bins takes $O(NlogN)$ which is done by sorting items first and then putting each item in their corresponding bin by iterating over them once more. Now there are $l-1$ iterations of the algorithm to merge the bins. Each bin merge takes at most $O(kl)$ since there are k sessions with at most l members which means for $l-1$ iterations we will have $O(kl^2)$. Overall, the running time of the algorithm will be $O(NlogN + Nl)$

Theorem 8 *Algorithm* `Ap-Max-Intra` *has* $\frac{1}{2-1/k}$ *approximation factor.*

Proof (Sketch) The detail proof of this problem makes use of an approximation-preserving reduction. Basically the idea of an approximation-preserving reduction is as follows: we need to show that an instance of `Ap-Max-Intra` is reducible to an instance of another known

NP-hard problem, Balanced Number Partitioning problem [33] and by applying Algorithm `BLDM`, which is an approximation algorithm for the latter problem produces a solution for the problem `Ap-Max-Intra`. The proof sketch makes use of two arguments: the first is that an instance of *Max-Intra* could be reduced to an instance of the Balanced Number Partitioning problem [33] in polynomial time. Then, it can be shown that the `BLDM` algorithm has one-on-one correspondence with `Ap-Max-Intra`. `Ap-Max-Intra` will accept $\frac{1}{2-1/k}$ approximation factor, since `BLDM` holds $2-1/k$ approximation factor.

3.2.2 Variable Length Sessions

The solution of *Max-Intra* for Variable Length Sessions is identical to aforementioned one, except the last step. If the length of each of the k bins is smaller than the length of the input variable, we merge them; otherwise, we skip that one and move on to the next one to merge. To clarify, if we want to merge one more time after step 4 in Figure 3, we skip the first column because the first session must have length 2, but we merge [0.51] and [0.63, 0.89] in the second column to get a session with length 3 as specified in the input length $[2, 3, 2, 3]$.

3.3 Algorithm *Min(Max)-Inter*

3.3.1 Fixed Length Sessions

Optimization of *Inter* diversity, both minimization and maximization variants, is NP-hard, and they bear remarkable similarity to each other. Given a set of N items, the *Min(Max)-Inter* problems will try to find an ordering of k sessions, each with l items, such that the aggregated differences between the average of two consecutive sessions is minimized (maximized). To better understand these problems, we break them into two steps. We only present these steps for the *Max-Inter* problem and note that the *Min-Inter* version works analogously, only by inverting the optimization goals inside the algorithm. For example, for optimizing *Max-Inter*, our goal is to find a sequence of sessions that maximizes Equation 2. One intuition is that *Inter* diversity increases if the means of individual sessions (on the dimension of interest) are highly different from each other. Indeed, if the k sessions have the same exact mean, no matter how one orders them, *Inter* diversity will be zero. As we prove in Lemma 2, this relates to *forming a set of k sessions with the goal to minimize Intra diversity.* So, the first step of our algorithm is to produce a set of sessions with the smallest *Intra* diversity. The next step is to order these sessions, such that the resulting sequence has the *Inter* value maximized. This is our guiding principle in creating the algorithms to solve this problem.

Our proposed solution `Ap-Max-Inter` for *Max-Inter* works as follows: we first find k sessions obtained by running Algorithm `Ap-Min-Intra`. This is needed, since it will generate sessions with means as different from each other as possible. After that, we create a graph of k nodes, each represents one of the k sessions. The weight of each edge (s_i, s_j) is defined as $w(s_i, s_j) = (\mu_{s_i} - \mu_{s_j})^2$. After that, the goal is to run an algorithm for the Longest path problem for *Max-Inter*. Since the graph is complete with positive weights on the edges, the Longest Path Problem could be solved by replacing the positive weights with negative values and running a traveling salesman problem (TSP) over it. In our implementation, we use the simple yet effective 2-approximation algorithm for TSP in metric space, described in [32, 36]. The algorithm starts by finding the Minimum Spanning Tree of the input graph using Prim's algorithm. Next, it lists the nodes in Minimum Spanning Tree in a pre-order walk and adds the edge to the starting vertex to the end. This path will create an ordering of sessions by following from the starting vertex s_i to the ending vertex s_j. This algorithm runs in $O(k^2 logk)$ which is dominated by the running time of the Prim's algorithm. We further improve this running time by using Fibonacci heaps and obtain $O(k^2 + logk)$.

Inversely, Algorithm `Ap-Min-Inter`, designed for *Min-Inter* first solves the *Min-Inter* problem to create sessions with the largest *Intra* diversity. Then, we create the graph same as we have done in `Ap-Max-Inter` but the edge weights do not need to be negated. Finally, we run TSP [36] to generate a sequence of sessions for minimizing *Inter* diversity of those sessions. For both problems, the obtained solution is a cycle and has one extra edge. We simply remove the edge with the smallest (largest) value in the solution. This produces an ordering of the sessions. Algorithm 2 presents the pseudo code of *Max-Inter* algorithm.

Algorithm 2 Algorithm `Ap-Max-Inter`

Require: N items, Number of sessions k, Length of session l

1: $S_{init} \leftarrow Min - Intra(N, k, l)$
2: $G = (S, E) \leftarrow$ complete graph with k nodes
3: $w(s_i, s_j) = (\mu_{s_i} - \mu_{s_j})^2$
4: Run Longest path algorithm on G
5: Longest path contains the ordering of the sessions.

Theorem 9 *Both* `Ap-Max-Inter` *and* `Ap-Min-Inter` *run in* $O(NlogN + k^2 + logk)$.

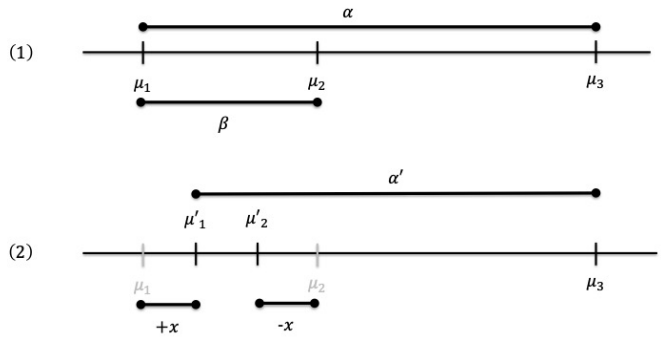


Fig. 4 Relationship between *Min-Intra* and *Max-Inter* when defined on the same dimension (1) If $S =< s_2, s_1, s_3 >$ is the *Min-Intra* solution and $\mu_{s_1} \leq \mu_{s_2} \leq \mu_{s_3}$, the *Inter* value reaches its maximum value, which is $\alpha + \beta$; (2) If a task is swapped between sessions s_1 and s_2, the *Inter* value for the new sessions will be $\alpha + \beta - 3x$ which is smaller and cannot be the solution of *Max-Inter*.

Proof The running time of the algorithm `Ap-Max-Inter` is dominated by the first step which is getting the solution of *Min-Intra* (for `Ap-Min-Inter` it is *Max-Intra*). The algorithm for TSP takes $O(k^2 + logk)$. This means that the overall running time will be $O(NlogN + k^2 + logk)$.

Lemma 2 *Given a set of N items forming k sessions (each with l items), when defined on the same dimension of interest, Inter diversity of the k sessions is maximized (minimized), when Intra diversity of those k sessions is minimized(maximized).*

Proof (Sketch)

***Inter* Minimization Case:** For the case of *Max-Intra*, the solution will require the averages of all groups to be the same (Recall Theorem 6). This results in having *Min-Inter* with value 0, leading to the optimal solution. Hence the proof.

***Inter* Maximization Case:** We prove by contradiction for *Min-Intra* and *Max-Inter*, for $k = 3$. For the purpose of illustration, consider the sequence $S =< s_2, s_1, s_3 >$ where $\mu_{s_1} \leq \mu_{s_2} \leq \mu_{s_3}$. Consider s_1, s_2, s_3 are the solution of *Min-Intra* and $\mu_{s_3} - \mu_{s_1} = \alpha$ and $\mu_{s_2} - \mu_{s_1} = \beta$. Figure 4 presents one such solution. Now consider that we swap a task between s_1 and s_2. After this swap, the value of μ_{s_1} will increase by x amount and the value of μ_{s_2} will decrease by the same x. Now it is easy to see that if the value of *Inter* is $\alpha + \beta$ for the solution of *Min-Intra*, then the value of the new solution will be $\alpha + \beta - 3x$ which is smaller. This argument extends to $k > 3$.

Theorem 10 `Ap-Max-Inter` *produces an answer that is at least* $1/2$ *of the the optimal solution.*

Proof The approximation of `Ap-Max-Inter` occurs in step 2, while solving the longest path problem (Since `Min-Intra` has an exact solution)). Since the longest path algorithm has the $1/2$ approximation factor, the overall algorithm `Ap-Max-Inter` has $1/2$ approximation factor.

Theorem 11 *Algorithm* `Ap-Min-Inter` *has* $4-2/k$ *approximation factor.*

Proof Similar to the proof of `Ap-Max-Inter`, using Lemma 2, the first step of `Ap-Max-Inter` is finding a set of sessions which are closest to each other. Using algorithm `Ap-Max-Intra` provides these sessions with $2 - 1/k$ approximation. The next step multiplies this error by a factor of 2 since the composition of the groups is not changed and we only find an ordering over the fixed groups. This yields an approximation factor of $4 - 2/k$.

3.3.2 Variable Length Sessions

The *Inter* solution of the Variable Length Sessions is the same as Fixed Length Sessions for both minimization and maximization problem.

3.4 Optimizing *Inter* with *Intra* as Constraint for Fixed Length Sessions

We now develop algorithms for the constrained optimization problems defined in Section 2.4. When *the values of the item dimension used for Intra diversity are all unique, two of these four algorithms have provable approximation factors.* Table 3 provides the summary of our technical results.

Algorithm	*Running Time*	*Approximation Factor*
Alg-Min-Intra,Min-Inter	$O(NlogN + k^2)$	$(OPT, 4 - 2/k)$
Alg-Min-Intra, Max-Inter	$O(NlogN + k^2)$	$(OPT, 1/2)$
Alg-Max-Intra, Min-Inter	$O(NlogN + Nl + k^2)$	*heuristic*
Alg-Max-Intra, Max-Inter	$O(NLogN + Nl + k^2)$	*heuristic*

Table 3 Optimization Algorithms and Results for Fixed Length Sessions

To optimize *Inter* with *Min-Intra* as a constraint, we design two algorithms *Alg-Min-Intra, Min-Inter* and *Alg-Min-Intra, Max-Inter*. For both, we start from the solution of the *Min-Intra* problem using algorithm `Ex-Min -Intra`. This solution is an exact algorithm for solving *Min-Intra* and gives a set of k sessions as the the output. After that, we run `Ap-Max-Inter` in *Alg-Min-Intra, Min-Inter* and `Ap-Min-Inter` in *Alg-Min-Intra, Max-Inter*.

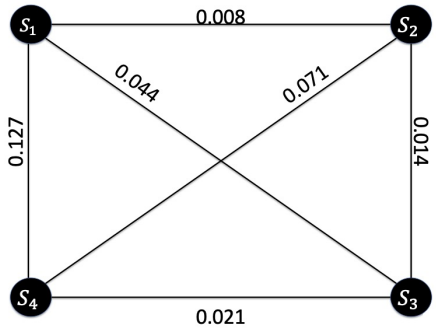


Fig. 5 `Ap-Max-Inter` graph of Example 1

On the other hand, to optimize *Inter* with *Max-Intra* as a constraint, we start from the solution of the *Max-Intra* using algorithm `Ap-Max-Intra`. This solution is an approximation algorithm for solving *Max-Intra* and returns a set of k sessions. After that, we run `Ap-Max-Inter` for *Max-Intra, Max-Inter* and `Ap-Min-Inter` for the *Max-Intra, Min-Inter*. Using Example 1, to solve *Alg-Min-Intra, Max-Inter* where the *Intra* dimension is on *Skill* and *Inter* dimension is on *Skill* as well, we first call the `Exact-Min-Intra` subroutine which sorts N items and group these items that are close to each other and obtain the following sessions, $s_1 = \{t_1, t_2, t_3\}$, $s_2 = \{t_4, t_5, t_6\}$, $s_3 = \{t_7, t_8, t_9\}$, and $s_4 = \{t_{10}, t_{11}, t_{12}\}$ where $\mu_{s_1} = 0.516$, $\mu_{s_2} = 0.6066$, $\mu_{s_3} = 0.726$, and $\mu_{s_4} = 0.873$ (see Figure 2). Given the solution of these 4 sessions, we then create a complete graph (Figure 5) by considering each session as a node. The weight of each edge in this graph is the *Inter* value of adjacent sessions on the *Skill* dimension that are calculated using Equation 2. The `Ap-Max-Inter` is then akin to the longest path problem. We convert these positive weights to negative weights by introducing a minus sign and then apply our proposed 2-approximation algorithm for the traveling salesman problem (TSP) on metric space that gives us the following tour $T = \{s_1 \rightarrow s_4 \rightarrow s_2 \rightarrow s_3 \rightarrow s_1\}$. We remove the edge $s_2 \rightarrow s_3$ since it has the smallest weight. The solution of *Max-Inter* is hence the sequence $S =< s_2, s_4, s_1, s_3 >$.

Alg-Min-Intra, Min-Inter problem is solved in a similar manner by following the steps outlined above. The only distinction is that we don't have to convert weight to negative weight.

Algorithm 3 presents the generic pseudo code. These two algorithms are based on heuristics and may not have any provable bounds.

Theorem 12 *Algorithm Alg-Min-Intra, Max-Inter has* $(1, 1/2)$ *approximation factor Min-Intra,Max-Inter problem and Alg-Min-Intra, Min-Inter has* $(1, 4 - 2/k)$ *approximation factor Min-Intra,Min-Inter problem, as long as items in Intra dimension have unique values.*

Algorithm 3 Algorithm for maximizing *Inter* with *Intra* as a constraint

Require: N items, Number of sessions k, Length of session l, dimensions d_1 and d_2

1: $S_{init} \leftarrow k$ sessions of l items each, obtained by running *Intra* optimization algorithm on d_1
2: $G = (V, E) \leftarrow$ complete graph with nodes of S_{init} and edge weights are calculated based on d_2 values between a pair of sessions
3: Call Subroutine `Ap-Max-Inter` or `Ap-Min-Inter` on G

Proof (Sketch) We provide the proof for *Alg-Min-Intra, Max-Inter* and the proof of Alg-Min-Intra, Min-Inter works analogously. `Ex-Min-Intra` is optimal. Since items have unique values on *Intra* diversity dimension, there exists *one and only one set of k sessions* that minimizes *Intra* diversity values. The second step of the algorithm *Alg-Min-Intra, Max-Inter* creates an ordering over these sessions. In that subset of the search space, i.e containing only solutions that start with the sessions of `Ex-Min-Intra` where the *Min-Intra* is optimal, our *Max-Inter* algorithm produces a solution which is 1/2 the optimal solution. Hence the $(1, 1/2)$ approximation factor holds for *Min-Intra,Max-Inter* problem. Similarly, the $(1, 4 - 2/k)$ approximation factor holds for *Min-Intra,Min-Inter* problem.

4 Experimental Evaluations

We first conduct experiments involving human subjects on music playlist recommendation and task recommendation in crowdsourcing to observe the effect of diversity on user satisfaction (in both applications) and worker performance (in crowdsourcing). Then, using large scale real data and synthetic data, we examine the quality of our algorithms in comparison to baselines, and evaluate the scalability of our approach.

Except for Section 4.2.5, which is related to the Variable Length Sessions, the rest of the section focuses on the Fixed Length Sessions.

4.1 Experiments with Human Subjects

We validate our framework in two applications: music recommendation, where we generate music playlists, and task recommendation in crowdsourcing, where we generate task sessions. These experiments are summarized in Table 4.

Experiment	# of workers	Setup	Observed data	Findings
1. Music Recommendation	**200**	Users rate playlists. Each playlist has 5 channels. Each channel has 10 songs.	user satisfaction no. of selected songs diversity rating	User satisfaction, no. of selected songs, and diversity rating are higher in diversified playlists.
2. Task Recommendation	**200 (total)**	Workers complete and rate task sessions. Each task session has 5 task sets. Each task set has 10 tasks.	worker satisfaction quality throughput	Worker satisfaction and quality are higher in diversified task sessions.
2.1. Controlled Session Gap	102	Workers complete task sessions in session gaps of 1 minute, 5 minutes, and 10 minutes.	workers' preference (diverse vs. similar task sessions) factors affecting workers' satisfaction (diversity, relevance, others)	Worker satisfaction is higher in diversified task sessions across all session gaps and peaks at the 5-minute session gap. Quality is higher in diversified task sessions across all session gaps. Workers prefer diverse sessions diversity is the main factor in 55% of the workers' ratings.
2.2. Random Session Gap	98	Workers complete tasks anytime.		

Table 4 Summary of experiments with human subjects

4.1.1 Music Recommendation.

We generate music playlists for users and consider different contexts namely music for long drive, theme party, Sunday morning, and learning a new music style, to observe how diversity affects user satisfaction in different contexts. Each playlist contains 5 distinct channels, each with 10 songs.

Dataset. The dataset consists of 727 songs from 54 albums, 47 artists, and 10 genres. The songs are from albums that won the Grammy Best Album of the Year Award between 1961 and 2020. The list of albums and their corresponding genres are from Wikipedia while the other information such as artist, period, popularity, tempo, and duration are from Spotify.

Experiments Flow. We first collect preferred genres and artists from users to form their profiles. We then generate 5 music playlists for each user. Each playlist has 5 channels, and each channel has 10 songs. The first 4 playlists are generated using the algorithms in Table 3, with dimensions specified for each context in Table 5. The 5th playlist represents the baseline with No diversity. It consists of similar songs randomly selected from the same dimension. Specifically, in this case, all songs from the period 2000's. Lastly, users evaluate the playlists by selecting songs they would actually listen to, rating how much they like diversity in the sessions, and providing an overall rating of the entire playlist. The ratings are based on a 5-pt Likert scale where 1 is the lowest and 5 is the highest. We measure user satisfaction using the overall rating provided by users. We recruit 200 workers (50 per context) from Amazon Mechanical Turk (AMT). We pay the workers \$0.10 for profile collection and \$1.00 for their evaluations.

Summary of Results. We observe in Table 6 that user satisfaction in diversified playlists (Scenarios $1-4$) is higher compared to the *No diversity* baseline. This observation is statistically significant at $p = 0.10$ using a one-way Analysis of Variance (ANOVA) [42]. The results are consistent with other measures: workers select the smallest number of songs from the *No diversity* playlist and the *No diversity* playlist receives the lowest average diversity ratings. Moreover, these observations extend to different contexts, as shown in Tables 7, 8, and 9. These results are summarized in Table 4 - Experiment 1. The sample size of 200 workers from the estimated $200,000$ workers in AMT [17] gives our results a 99% confidence level and a 10% error margin (based on the Central Limit Theorem [44]). *In summary, our music experiment clearly shows that diversity is preferred over No diversity.* Additionally, diversity definitions depend on context, as observed in Tables 7, 8, and 9.

	Long Drive	**Theme Party**	**Sunday Morning**	**Learn Music**
Intra	tempo	period	popularity	genre
Inter	popularity	genre	genre	tempo

Table 5 Diversity dimensions per context

	Scenario	**No. of Selected Songs**	**Diversity Rating**	**User Satisfaction**
1	*Min-Intra, Min-Inter*	**15.16**	3.57	3.54
2	*Min-Intra, Max-Inter*	15.05	3.66	3.66
3	*Max-Intra, Min-Inter*	14.71	3.59	**3.71**
4	*Max-Intra, Max-Inter*	14.66	**3.69**	**3.71**
5	*No diversity*	*12.83*	*3.35*	*3.44*

Table 6 Average evaluation scores across all contexts

	Scenario	Long Drive	Theme Party	Sunday Morning	Learn Music
1	*Min-Intra, Min-Inter*	**16.58**	14.86	14.76	14.42
2	*Min-Intra, Max-Inter*	15.82	**15.06**	14.12	**15.20**
3	*Max-Intra, Min-Inter*	16.52	13.64	14.30	14.38
4	*Max-Intra, Max-Inter*	16.24	13.96	**15.04**	13.40
5	*No diversity*	*14.10*	*11.92*	*13.62*	*11.68*

Table 7 Average number of selected songs per context

	Scenario	Long Drive	Theme Party	Sunday Morning	Learn Music
1	*Min-Intra, Min-Inter*	3.64	3.52	3.64	3.46
2	*Min-Intra, Max-Inter*	3.70	3.50	3.82	3.61
3	*Max-Intra, Min-Inter*	3.70	3.54	3.58	3.54
4	*Max-Intra, Max-Inter*	**3.84**	**3.68**	**3.58**	**3.64**
5	*No diversity*	*3.34*	*3.30*	3.46	*3.30*

Table 8 Average diversity rating per context

	Scenario	Long Drive	Theme Party	Sunday Morning	Learn Music
1	*Min-Intra, Min-Inter*	3.62	3.88	3.34	3.32
2	*Min-Intra, Max-Inter*	3.76	3.72	3.66	3.50
3	*Max-Intra, Min-Inter*	**3.86**	**3.98**	3.56	3.44
4	*Max-Intra, Max-Inter*	3.76	3.80	**3.70**	**3.58**
5	*No diversity*	*3.60*	*3.42*	3.46	*3.28*

Table 9 Average user satisfaction per context

	Session Type	Worker Satisfaction	Quality (%)	Through put
1	*Min-Intra(creation date), Min-Inter(skill)*	4.26	0.67	7.72
2	*Min-Intra(skill), Max-Inter(reward)*	**4.30**	**0.68**	7.85
3	*Max-Intra(skill), Min-Inter (reward)*	4.29	0.66	7.60
4	*Max-Intra(duration), Max-Inter (skill)*	4.28	**0.68**	7.71
5	*No diversity*	4.01	0.62	**7.92**

Table 10 Task recommendation sessions

		Session Gap (minutes)		
	Session Type	**1**	**5**	**10**
Worker Satisfaction	Diversified	3.76	4.30	4.21
	Non-diversified	3.48	4.09	3.91
Quality	Diversified	0.68	0.64	0.65
	Non-diversified	0.62	0.55	0.59
Throughput	Diversified	8.50	8.82	8.01
	Non-diversified	8.57	10.67	8.18

Table 11 Throughput, quality, and worker satisfaction of diversified and non-diversified sessions with different session gaps

4.1.2 Task Recommendation.

In these experiments, we recommend task sessions to workers in crowdsourcing. Each task session consists of 5 sets and each set consists of 10 tasks.

Dataset. The dataset consists of 20,000 tasks from Figure Eight's open data library [1]. Each task belongs to one of 10 types such as tweet classification, image transcription, and sentiment analysis. Each task type is represented as a set of keywords that best describe required skills. Additionally, each task has a creation date, an expected completion time (less than a minute), and a reward that varies between \$0.01 - \$0.05.

Experiments flow. We collect a total of 200 user profiles where workers indicate (from 1 to 5) their interest in completing tasks, which are described by a given set of keywords. For each user profile, we generate task sessions using the algorithms in Table 3 and a combination of the following dimensions: skill, reward, duration, and creation date. We also generate a *No diversity* baseline session. In this session, we randomly pick a task type and randomly select similar tasks belonging to that type. Next, workers complete and rate the recommended sessions. We measure *worker satisfaction*, *quality* of the completed tasks with respect to a ground truth, and *task throughput*.

Satisfaction refers to how satisfied workers are with the task sessions (a rating from 1 to 5 provided by each worker). *Quality* refers to the percentage of correct answers from the tasks completed by a worker. To measure quality, we use the answers obtained from the dataset as the ground truth. We use a naïve script that relies on basic equality to evaluate answer correctness. *Throughput* refers to the average number of tasks completed per minute.

We also investigate the impact of *session gap*, the time interval between completing sessions, in our proposed algorithms. We conduct Experiment 2.1 in Table 4 where we assign 102 of the 200 workers to complete the sessions in 3 fixed time intervals of 1, 5, and 10 minutes between sessions (34 workers for each fixed time interval). We also ask these workers their preference between diversified and non-diversified sessions, and to indicate which factor mainly affects their satisfaction.

In total, we recruit 200 workers, pay each \$0.03 for profile collection and at least \$0.35 for task completion.

Summary of Results. Table 10 presents the average worker satisfaction, quality, and throughput in the task recommendation experiments. We observe that worker satisfaction and quality in diversified sessions are higher compared to the *No diversity* baseline. This observation is statistically significant at $p = 0.01$ using a one-way Analysis of Variance (ANOVA) [42]. On the other hand, throughput is marginally higher for the *No diversity* case. This observation confirms previous studies where workers get more proficient in completing similar (and hence not diverse) tasks, allowing them to become faster at task completion [18].

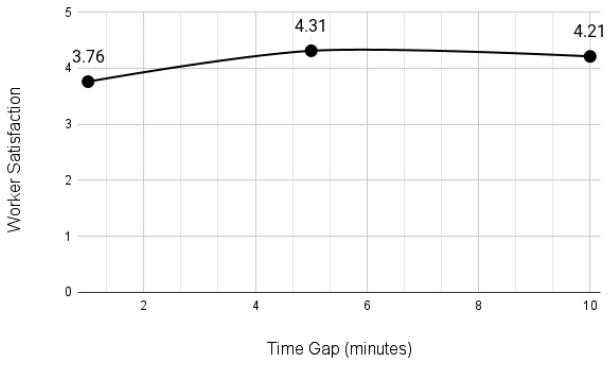


Fig. 6 Worker satisfaction in varying session gaps

In Table 11, we present the average worker satisfaction, quality, and throughput grouped by session gap. The values are from the 102 workers in the Experiment 2.1 in Table 4 where we control the session gap. For each session gap (1, 5, 10 minutes), 34 workers complete the 5 session types listed in 10. The *Diversified* rows in Table 11 show the aggregated values obtained in the 4 task sessions generated by our algorithms while the *Non-diversified* rows show the values obtained in the baseline or *No diversity* session.

Our findings show that quality and worker satisfaction are better in the diversified tasks sessions generated by our proposed algorithms across all session gaps. These observations are significant at $p = 0.10$ using a *t-test* [15]. Moreover, it is interesting to note that worker satisfaction peaks at the 5-minute session gap as seen in Figure 6. For the task sessions, 5 minutes may be the ideal break time workers need to alleviate fatigue or boredom [41]. We also note that our findings are consistent with the 102 workers' responses where 72.5% of the workers prefer diversified sessions over non-diversified sessions; 55% of the workers consider diversity in their ratings, 23.3% consider relevance, and 21.7% consider other factors.

In summary, our task recommendation experiments clearly show the benefit of diversity in the workers' satisfaction and the quality of crowdsourced tasks.

4.2 Large Data Experiments

The goal here is to evaluate our algorithms with appropriate baselines (including exact solutions) and compare them qualitatively (approximation factors, objective function value) and scalability-wise (running time). All algorithms are implemented in Python 3.6 on a 64-bit Windows server machine, with Intel Xeon Processor, and 16 GB of RAM. All numbers are presented as the average of five runs. For brevity we present a subset of results that are representative.

4.2.1 Data Sets.

a. 1-Million Song: We use the Million Songs Dataset [9] that has 1 million songs (please note the Spotify dataset used in Section 4.1 is small in scale). We have normalized the data to be between $[0, 1]$. This dataset also includes artist popularity and hotness, genre, release date and etc. The presented results are representative and consider tempo and loudness as dimensions.
b. Synthetic dataset: The presented results on this are the ones that vary distributions of the underlying dimensions. We sample from three distributions: Normal, Uniform, and Zipfian. For Normal distribution, data is sampled with mean and standard deviation, $\mu = 250$, $\sigma = 10$. For Uniform, dataset is sampled from Uniform distribution between [0,500], and for Zipfian distribution default exponent is set to $\alpha = 1.01$. We produce a pool of 2^{30} items for each of our three distributions.

4.2.2 Implemented Baselines.

In addition to `Random` where we generate random sequences, we implement different baselines and compared the performance of our algorithms.

`Optimal.` The optimal baseline is based on an Integer Programming (IP) algorithm that solves the problem optimally on small instances. The rationale behind implementing IP is to verify the theoretical approximation factors of our algorithms against the optimal solution. We used Gurobi as the solver[2].

`Baseline-MMR.` This baseline is inspired by the MMR algorithm [11] used in diversifying web search results. MMR takes a search query and returns relevant and diverse results. Hence, our mapping to MMR optimizes *Intra* session diversity only. At each iteration, `Baseline-MMR` considers an item to be included or not in the result and calculates two scores: the *Intra* score of adding a new item to a session and the *max* (resp., *min Inter*) score of a new session to all other sessions in the case of *Max-Inter* (resp., *Min-Inter*). It then picks the highest or the lowest weighted sum of these two scores based on the *Intra* part of the problem. The item with that score is chosen to be added to the session. This process is repeated until there is no item left.

`Baseline-Constrained-KMean.` This is a clustering technique similar to the one proposed in [10] , which uses the K-Mean Clustering approach to produce a set of k clusters, each containing exactly l items. Following

[2] *https://www.gurobi.com/resource/switching-from-open-source/*

Our Scenarios	N=8192 , k=16		N=1024 , k=128	
	Intra	*Intra*	*Intra*	*Inter*
Min-Intra , Min-Inter	1	1.05	1	1
Min-Intra , Max-Inter	1	0.35	1	0.49
Max-Intra , Min-Inter	0.99	1.06	0.98	1.04
Max-Intra , Max-Inter	0.99	0.58	0.95	0.69

Table 12 Approximation factors on 1-Million Song dataset

that, these clusters are sorted by increasing mean to generate a sequence of sets as the final result.

4.2.3 Summary of Results.

Overall, for our problems, where both Intra and Inter diversity are to be optimized, our algorithms are the unanimous choice considering both quality and scalability. When the *Intra* and *Inter* diversity is studied individually, our algorithms outperform all the baselines and empirically produce approximation factors close to 1, across varying k, N, and different distributions. The only exception to this latter observation is `Baseline-MMR`, which performs better in maximizing *Inter* diversity (while performing very poorly for *Intra* optimization), which is due to its focus on optimizing *Inter* diversity only. Moreover `Baseline-Constrained-KMean` performs poorly for the maximization problems, our algorithm convincingly outperforms it in both *Intra* and *Inter* minimization and maximization. This baseline also exhibits poor scalability. Contrarily, our algorithms are highly scalable and are much more efficient compared to the baselines.

4.2.4 Quality Evaluation.

We vary k (the number of sessions), N (the number of items), and the data distribution. The default values are N=2^{13} and k=2^7 with a uniform distribution.

Comparison against Optimal. Table 12 shows the approximation factors for our algorithms for two default settings: ($N = 2^{13}$, $k = 2^4$) and ($N = 2^{10}$, $k = 2^7$) using 1-Million Song dataset. We can see that our algorithms produce an approximation factor equal to 1 when *Intra* diversity is minimized and a factor very close to 1 when *Intra* diversity is maximized.

When *Inter* diversity is minimized, the resulting approximation factors are close to 1. However, when *Inter* diversity is maximized, the approximation factors are slightly low as our algorithm solves the *Intra* part of the problem before ordering the sessions to maximize *Inter* diversity. It is hence bound by the constraints of the solution to *Intra*. Nevertheless, the solution formulated by our algorithm for *Min-Intra,Max-Inter* and *Min-Intra,Min-Inter* is able to produce a point on the Pareto Front in the optimal solution region which meets both the *Intra* and *Inter* objectives. The synthetic dataset mimics this trend as well.

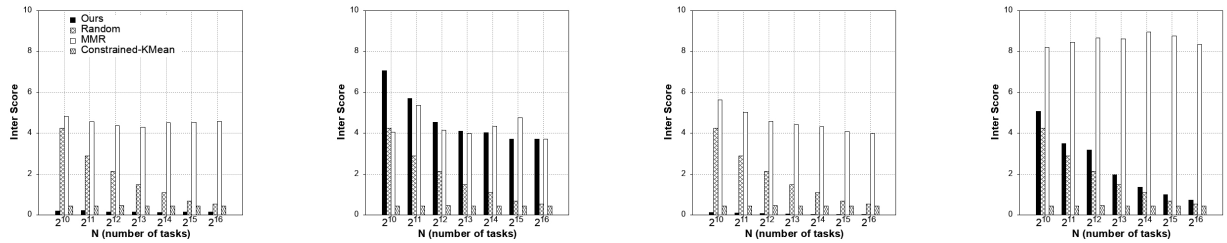
Based on the approximation factor results and the above analysis, we conclude that our algorithms produce good and in some cases the best possible solution for the 4 *problems we attempt to optimize.*

Varying N. Figures 7 and 8 show how *Inter* scores change as we vary N from 2^{10} to 2^{16} for `Baseline-MMR`, `Baseline-Constrained-KMean`, `Random` and our algorithms for 1-Million Song and synthetic dataset respectively. Figures 7(a)(c) and 8(a)(c) confirm that our algorithm performs best when *Inter* diversity is minimized. The objective function improves with increasing N. On the other hand, as seen in Figures 7(b)(d) and 8(b)(d),when *Inter* diversity is maximized, `Baseline -MMR` outperforms our algorithm with increasing N. This is because our algorithm is subject to the constraints imposed by optimizing *Intra* diversity first then maximizing the *Inter* diversity, while `Baseline-MMR` focuses on the *Inter* dimension only.

We also compare *Intra* scores whilst varying N. Tables 13 and 14 showcase the approximation factors of our algorithm's *Intra* considering `Optimal` for $N \leq 2^{13}$ and $N > 2^{13}$. A ratio of 1 means that the algorithm produces the best or optimal solution. These results showcase that our solutions achieve even better bound empirically compared to the theoretical bounds. Tables 13 and 14 also show that although `Baseline-MMR` performs better in *Max-Inter* problem, but it performs poorly for both *Min-Intra* and *Max-Intra* problems.

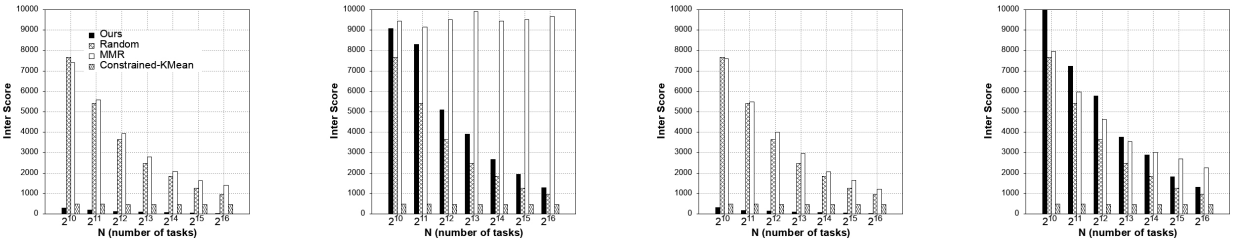
Interestingly, `Random` often times produces approximation factor close to 1 for $N > 2^{13}$ when maximizing *Intra*. This is due to the fact that *Intra* is maximized when the variance of the sessions are maximized which is one of the side effects of `Random` algorithm. However, `Baseline-MMR` and `Random` produce very poor approximation factors when minimizing *Intra*. Contrarily, our solutions stay close to 1 approximation factor for both minimization and maximization of *Intra* diversity. As N increases, the *Intra* scores do not see any drastic change in approximation factors, and always stays close to 1.

Varying k. Figure 9 and 10 present how *Inter* scores evolve as we vary k between 2^4 and 2^{11} for different baselines compared to our algorithm. We keep N constant at 2^{13}. We observe figure 9(a)(c) and 10(a)(c) that our algorithm performs significantly better than other baselines in minimizing *Inter* diversity. For Figures 9(b)(d) and 10(b)(d), our observation is similar to the case of varying N, `Baseline-MMR` performs slightly better. Overall, *Inter* diversity increases for all 4 scenarios as k increases. This is because of the fact that when



(a) `Ap-Min-Inter` *(Min-Intra)* (b) `Ap-Max-Inter` *(Min-Intra)* (c) `Ap-Min-Inter` *(Max-Intra)* (d) `Ap-Max-Inter` *(Max-Intra)*

Fig. 7 *Inter* scores with varying N for 1-Million Song dataset



(a) `Ap-Min-Inter` *(Min-Intra)* (b) `Ap-Max-Inter` *(Min-Intra)* (c) `Ap-Min-Inter` *(Max-Intra)* (d) `Ap-Max-Inter` *(Max-Intra)*

Fig. 8 *Inter* scores with varying N for Synthetic dataset

more sessions are present, it allows for more diversity between each session.

We present approximation factors of *Intra* in Tables 15 and 16 and observe similar trend as to when we vary N. Also, similar to varying N for *Intra* scores, the approximation factors here also stay close to 1 for our algorithm.

Varying distribution. Figures 11 and 12 present how our algorithm and other baselines perform as we vary data distributions. We set N to 2^{13} and k to 2^7.

Considering *Intra* scores, our algorithm performs the best using Uniform distribution for all 4 scenarios. However, Normal performs slightly worse at times with our algorithm when we attempt to maximize *Intra*.

When we compare *Inter* scores, our algorithm performs best with Uniform distribution. In Figures 11(b)(d), `Baseline-MMR` outperforms our algorithm due to the same reasons mentioned in the varying k and N section of this paper.

`Baseline-Constrained-KMean` outperforms our algorithms for minimizing *Intra* and *Inter* when using the Zipf distribution.

We also observe that across all 4 scenarios, Zipf produces scores much larger in magnitude than Normal or Uniform distribution. This is due to the range of values in Zipf, which results in larger *Intra* and *Inter* scores. Overall, our algorithms stand out to be the best choice, with its best performance being on Uniform distribution.

Min-Intra (Minimizing & Maximizing *Inter*)	N	Algorithms			
		MMR	Random	Constrained-KMean	Ours
	<= 8192	0.008	6.41E-05	0.165	1
	>8192	0.002	5.42E-05	0.167	1
Max-Intra (Minimizing & Maximizing *Inter*)	N	Algorithms			
		MMR	Random	Constrained-KMean	Ours
	<= 8192	0.22	0.98	0.0173	0.99
	>8192	0.021	0.92	0.0187	0.99

Table 13 *Intra* approximation factors varying N on 1-Million Song dataset

Min-Intra (Minimizing & Maximizing *Inter*)	N	Algorithms			
		MMR	Random	Constrained-KMean	Ours
	<= 8192	0.006	7.53E-05	0.018	1
	>8192	0.005	6.56E-05	0.0086	1
Max-Intra (Minimizing & Maximizing *Inter*)	N	Algorithms			
		MMR	Random	Constrained-KMean	Ours
	<= 8192	0.19	0.94	0.007	0.99
	>8192	0.019	0.99	0.0076	0.99

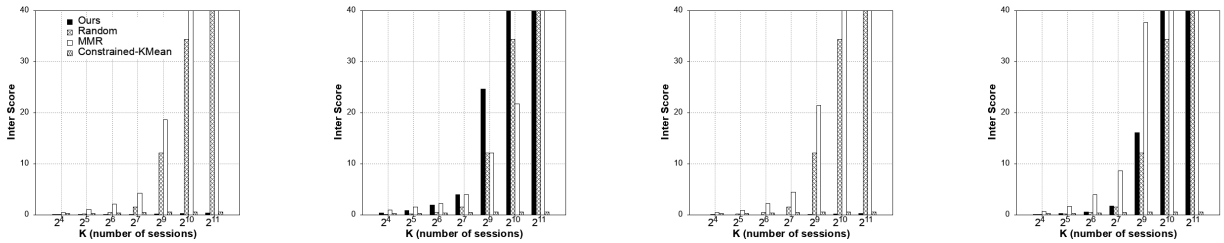
Table 14 *Intra* approximation factors varying N on Synthetic dataset

Min-Intra (Minimizing & Maximizing *Inter*)	k	Algorithms			
		MMR	Random	Constrained-KMean	Ours
	<= 128	0.011	0.0021	0.263	1
	>128	0.0012	4.95E-06	0.069	1
Max-Intra (Minimizing & Maximizing *Inter*)	k	Algorithms			
		MMR	Random	Constrained-KMean	Ours
	<= 128	0.035	0.92	0.05	0.99
	>128	0.29	0.85	0.0027	0.99

Table 15 *Intra* approximation factors varying k on 1-Million Song dataset

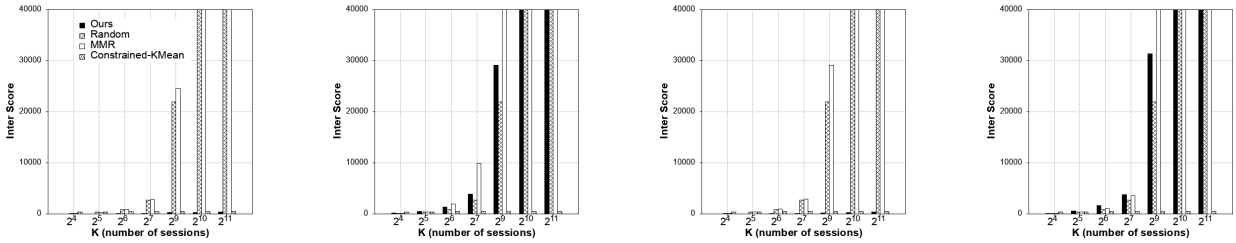
4.2.5 Variable Length Sessions

We relaxed two limitations of the Fixed Length Sessions problem in this section: a. session lengths varies; b. just a subset of items is recommended. The complexity of the *Min-Intra* problem remains unchanged, while the NP-hardness of *Max-Intra* still holds. Finally, *Inter* problems' NP-hardness remains intact. We vary k between between 2^4 and 2^{11} and keep N as its de-



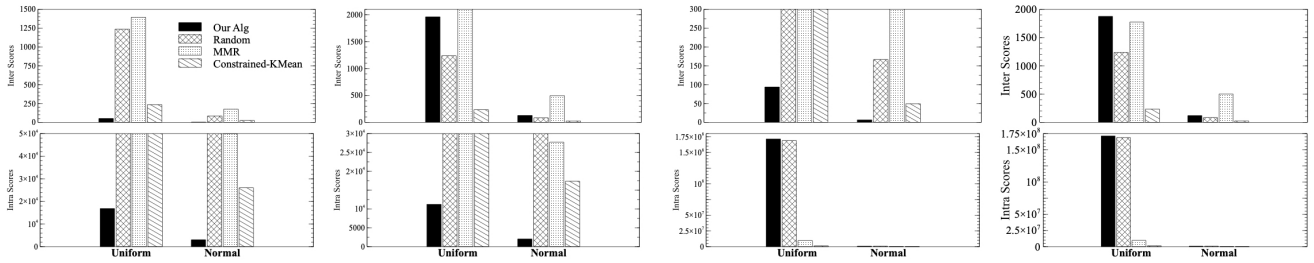
(a) `Ap-Min-Inter` *(Min-Intra)* (b) `Ap-Max-Inter` *(Min-Intra)* (c) `Ap-Min-Inter` *(Max-Intra)* (d) `Ap-Max-Inter` *(Max-Intra)*

Fig. 9 *Inter* scores with varying k for 1-Million Song dataset



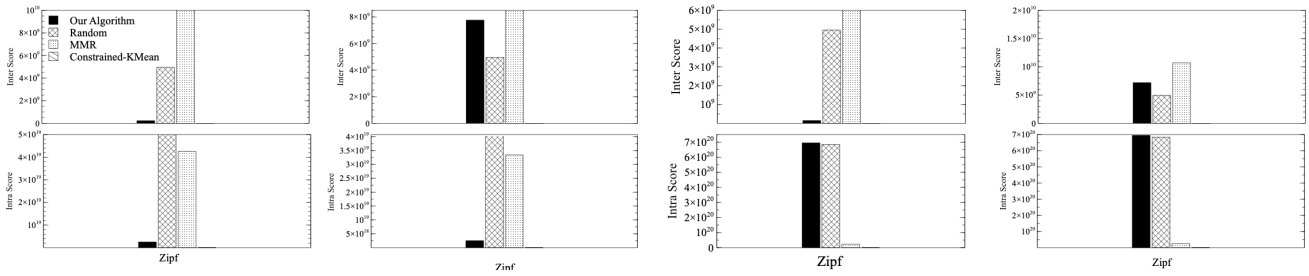
(a) `Ap-Min-Inter` *(Min-Intra)* (b) `Ap-Max-Inter` *(Min-Intra)* (c) `Ap-Min-Inter` *(Max-Intra)* (d) `Ap-Max-Inter` *(Max-Intra)*

Fig. 10 *Inter* scores with varying k for Synthetic dataset



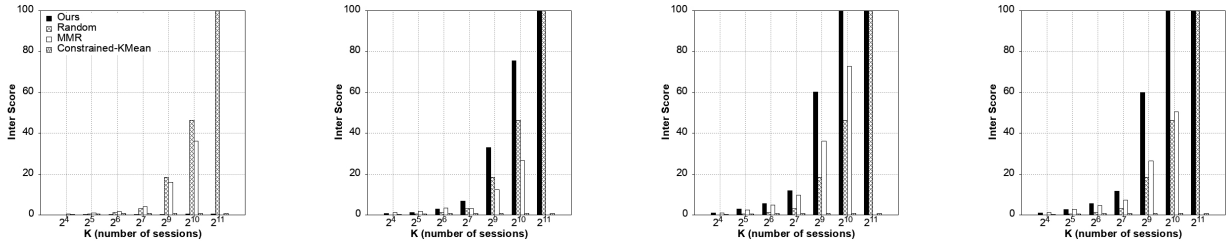
(a) *Min-Intra,Min-Inter* (b) *Min-Intra,Max-Inter* (c) *Max-Intra,Min-Inter* (d) *Max-Intra,Max-Inter*

Fig. 11 Synthetic dataset: *Inter* and *Intra* scores varying distributions



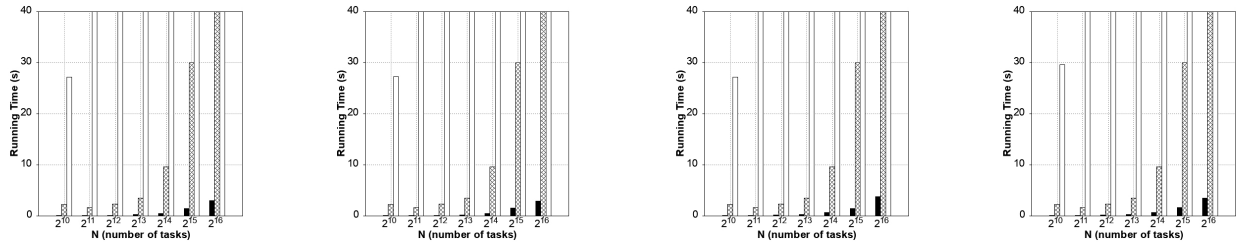
(a) *Min-Intra,Min-Inter* (b) *Min-Intra,Max-Inter* (c) *Max-Intra,Min-Inter* (d) *Max-Intra,Max-Inter*

Fig. 12 Synthetic dataset: Zipf Distribution



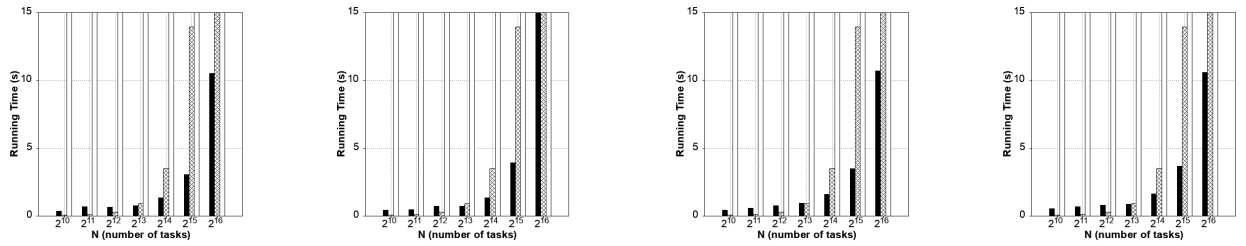
(a) `Ap-Min-Inter` *(Min-Intra)* (b) `Ap-Max-Inter` *(Min-Intra)* (c) `Ap-Min-Inter` *(Max-Intra)* (d) `Ap-Max-Inter` *(Max-Intra)*

Fig. 13 *Inter* scores with varying k for 1-Million Song dataset for Variable Length Sessions



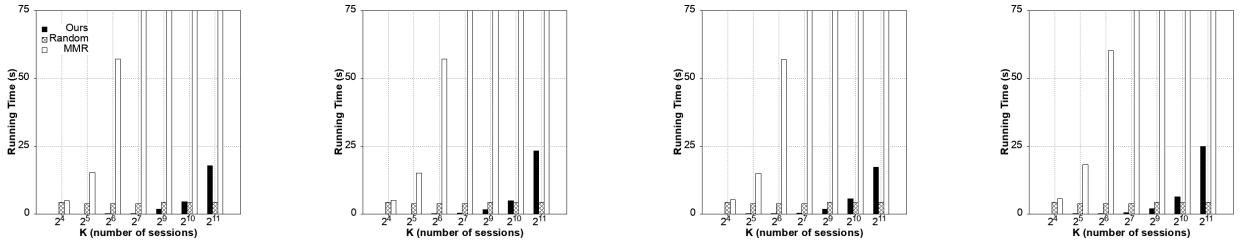
(a) *Min-Intra,Min-Inter* (b) *Min-Intra,Max-Inter* (c) *Max-Intra,Min-Inter* (d) *Max-Intra,Max-Inter*

Fig. 14 Running times varying N for 1-Million Song dataset



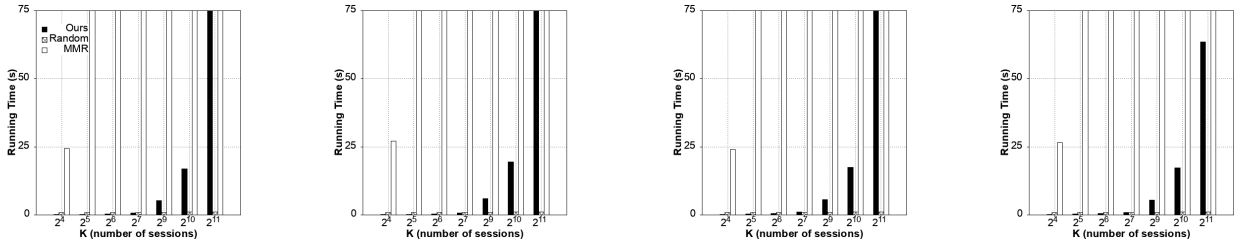
(a) *Min-Intra,Min-Inter* (b) *Min-Intra,Max-Inter* (c) *Max-Intra,Min-Inter* (d) *Max-Intra,Max-Inter*

Fig. 15 Running times varying N for Synthetic dataset



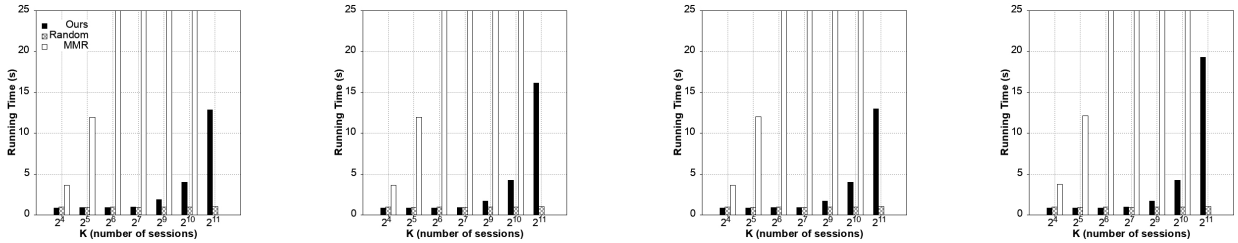
(a) *Min-Intra,Min-Inter* (b) *Min-Intra,Max-Inter* (c) *Max-Intra,Min-Inter* (d) *Max-Intra,Max-Inter*

Fig. 16 Running times varying k for 1-Million Song dataset



(a) *Min-Intra,Min-Inter* (b) *Min-Intra,Max-Inter* (c) *Max-Intra,Min-Inter* (d) *Max-Intra,Max-Inter*

Fig. 17 Running times varying k for Synthetic dataset



(a) *Min-Intra,Min-Inter* (b) *Min-Intra,Max-Inter* (c) *Max-Intra,Min-Inter* (d) *Max-Intra,Max-Inter*

Fig. 18 Running times varying k for 1-Million Song dataset for Variable Length Sessions

Min-Intra **(Minimizing & Maximizing *Inter*)**	**k**	**Algorithms**			
		MMR	**Random**	**Constrained-KMean**	**Ours**
	<= 128	0.035	0.0013	0.03	1
	>128	0.0008	5.05E-06	0.0045	1
Max-Intra **(Minimizing & Maximizing *Inter*)**	**k**	**Algorithms**			
		MMR	**Random**	**Constrained-KMean**	**Ours**
	<= 128	0.055	0.99	0.03	0.99
	>128	0.42	0.85	0.001	0.99

Table 16 *Intra* approximation factors varying k on Synthetic dataset

Min-Intra **(Minimizing & Maximizing *Inter*)**	**k**	**Algorithms**			
		MMR	**Random**	**Constrained-KMean**	**Ours**
	<= 128	0.215	0.0127	0.39	0.725
	>128	0.0485	0.0001	0.058	0.76
Max-Intra **(Minimizing & Maximizing *Inter*)**	**k**	**Algorithms**			
		MMR	**Random**	**Constrained-KMean**	**Ours**
	<= 128	0.435	0.98	0.033	0.714
	>128	0.39	0.78	0.0013	0.79

Table 17 *Intra* approximation factors varying k on 1-Million Song dataset for Variable Length Sessions

fault value for different baselines compared to our algorithms. This experiments incorporate an extra input that is generated at random and indicates the length of each session as a list of k values between 2 and l. In Example 1, $N = 12$ and $k = 4$, therefore $l = 3$. As a consequence, $[2, 3, 2, 3]$ is the length list containing random integers between 2 and 3, which is our l. As we have 8192 items in our experiments for each k value, averaging the items that are recommended in each scenario yields 4568 items out of 8192.

The results are presented in Figure 13 and Table 17. When maximizing *Intra*, Random produces an approximation factor close to 1 for $k \leq 128$. This is due to the same reason that is explained in the Varying k section that when *Intra* is maximized, the variance of the sessions is also maximized. Except for the *Alg-Max-Intra, Min-Inter* problem, all of the scenarios follow the same trend of Fixed Length Sessions. In comparison to the other baselines, Variable Length Sessions achieves a higher *Intra* approximation factor, as shown in Table 17.

Since `Baseline-MMR` could not finish in reasonable time for scenario when k is 2^{11}, we leave it blank in Figure 13.

4.2.6 Scalability Evaluation.

Figures 14 , 15 and 16, 17 compare the running time of the three algorithms for 1-Million Song and synthetic dataset. The running time of `Baseline-Constrained-KMean` was not included in these figures since some scenarios took many days to complete. Naturally, as N increases, the running time of our algorithms increase. We also observe that as we vary N with $k = 2^7$, our algorithms are the fastest in all diversity scenarios.

In Figures 16 and 17 , we vary k and set N to 2^{13}. We observe that our algorithms scale very well but is sometimes slightly slower than `Random`. This is unsurprising, as `Random` does not even have to do much work to generate sessions (recall that however it performs poorly qualitatively). However, we observe that our algorithm is consistently faster with increasing values of k. The scalability evaluation plots for relaxed problem experiments closely resemble those of the original problem for the 1-Million Song dataset, as seen in Figure 18. *Overall, we find that our algorithms are highly scalable and produce results within a few seconds for very large values of N and k, while some of the baselines take hours to complete.*

5 Related Work

Applications Diversity has been extensively studied in recommendation and search applications [2, 6, 13, 22, 28, 29, 34, 37–39, 45, 47–51], to return items that are relevant as well as cover full range of users interests. The goal is to achieve a compromise between relevance and result heterogeneity. Existing works [26, 46] have also acknowledged the need for diversity and sequence based modeling in different recommendation applications. Recent works in crowdsourcing [21, 35] have demonstrated the importance of diversity in task recommendation. Task diversity is grounded in organization theories and has shown to impact the motivation of the workers [12]. Amer-Yahia et al. [5] propose the notion of composite tasks (CT), a set of similar tasks that match workers' profiles, comply with their desired reward and task arrival rate. Their experiments show that diverse CTs contribute to improving outcome quality. A recent work has studied *Intra* and *Inter*-table influence in web table matching [21] involving crowd. Even though complet-

ing similar tasks lead to faster completion time [18], but such composition lead to fatigue and boredom, and task abandonment [16, 25, 27]. Aipe and Gadiraju[3] empirically observe that workers who perform similar tasks achieve higher accuracy and faster task completion time compared to workers who worked on diverse tasks. However, they find that these workers experience fatigue the most. Alsayasneh et al. integrate the concept of diversity in composite tasks and empirically find a positive effect of diversity in outcome quality [4]. In [43], The authors investigated a sequential group recommender that is aware of the group's previous interactions with the system by adding the concept of satisfaction, which characterizes how relevant the recommended items are to each group member.
For all of these applications, diversity is studied set-based or sequence based only.

These applications call for a deeper examination of diversity and a powerful framework to capture its variants, which is our focus here.

Set and Sequence Diversities Existing works on diversification could be classified as set-based only [2, 22, 34, 37, 38, 45] or sequence-based only [6, 13, 29, 31, 51]. As an example, in [51], the authors study sequence-based diversity that is defined as the diversity of any permutation of the items. Another example is [6], in which taxonomies are used to sample search results to reduce homogeneity. In [2], the authors proposed an algorithm with a provable approximation factor to find relevant and diverse news articles. In the database context, Chen and Li [13] propose to post-process structured query results, organizing them in a decision tree for easier navigation. In [8, 30] the notion of diversity is used in the results of queries to produce closest results such that each answers is different from the rest. In recommender systems, results are typically post-processed using pair-wise item similarity to generate a list that achieves a balance between relevance and diversity. For example, in [19], recommendation diversity was formulated as a set-coverage problem. By distinguishing between item and user diversity and focusing on various definitions of each, [31] investigated a diversity-aware recommender system for a single user or a group of users.

To the best of our knowledge, existing works have focused on achieving diversity in a single set. We solve set-based and sequence-based diversities in tandem and develop algorithms with guarantees.

6 Conclusion

We initiate the study of a formal and algorithmic framework to address diversity for s sequence of sets that has natural recommendation applications (from song playlists to task recommendations in crowdsourcing). The combination of *Intra* and *Inter* session diversities gives rise to four bi-objective optimization problems. We propose algorithms with guarantees. Our extensive empirical evaluation, conducted using human subjects, as well as large scale real and simulated data, shows the need for diversity to improve user satisfaction and the superiority of our algorithms against multiple baselines.

In addition to theoretical questions, this work opens up interesting directions that are of empirical interests: an immediate extension of our work is to observe users as they consume items and learn how diversity dimensions and their respective definitions could be personalized for different users. Similarly, we are empirically exploring how to choose the preferred diversity dimensions depending on the underlying context for different applications. Finally, an interesting open problem is to understand how time affects underlying contexts and fine tune diversified recommendations based on that.

In terms of other widely used diversification functions, there exist diversity functions that consider radius (maximum/minimum distance) [24], or sum (sum of distance). One can maximize or minimize these based on the underlying optimization goal. Many of these problems relate to the Facility Allocation Problem [7] and its variants, as well as Graph Partitioning problems [40]. These problems are known to be NP-hard. Our produced greedy solutions could be adapted to solve these variants. However, whether these solutions would be just heuristic or they would accept provable approximation factors would require revisiting and analyzing each of them and that can be studied in the future work.

We are also going to study the approximation factors of the proposed algorithms for the Variable Length Sessions in the future.

Acknowledgements The work of Sepideh Nikookar, Paras Sakharkar, and Senjuti Basu Roy are supported by the NSF CAREER Award #1942913, IIS #2007935, IIS #1814595, PPoSS: Planning #2118458, and by the Office of Naval Research Grants No, N000141812838, N000142112966.

References

1. (2019) Figure eight - data for everyone. `https://www.figure-eight.com/data-for-everyone/`
2. Abbar S, Amer-Yahia S, Indyk P, Mahabadi S (2013) Real-time recommendation of diverse related articles. In: 22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013, pp 1–12

3. Aipe A, Gadiraju U (2018) Similarhits: Revealing the role of task similarity in microtask crowdsourcing. In: HT, pp 115–122
4. Alsayasneh M, Amer-Yahia S, Gaussier E, Leroy V, Pilourdault J, Borromeo RM, Toyama M, Renders JM (2017) Personalized and diverse task composition in crowdsourcing. IEEE Transactions on Knowledge and Data Engineering 30(1):128–141
5. Amer-Yahia S, Gaussier E, Leroy V, Pilourdault J, Borromeo RM, Toyama M (2016) Task composition in crowdsourcing. In: Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on, IEEE, pp 194–203
6. Anagnostopoulos A, Broder AZ, Carmel D (2006) Sampling search-engine results. World Wide Web 9(4):397–429
7. Andreev K, Racke H (2006) Balanced graph partitioning. Theory of Computing Systems 39(6):929–939
8. Angel A, Koudas N (2011) Efficient diversity-aware search. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, pp 781–792
9. Bertin-Mahieux T, Ellis DP, Whitman B, Lamere P (2011) The million song dataset. In: Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)
10. Bradley PS, Bennett KP, Demiriz A (2000) Constrained k-means clustering. Microsoft Research, Redmond 20(0):0
11. Carbonell JG, Goldstein J (1998) The use of mmr, diversity-based reranking for reordering documents and producing summaries. In: SIGIR, vol 98, pp 335–336
12. Chandler D, Kapelner A (2012) Breaking monotony with meaning: Motivation in crowdsourcing markets. CoRR abs/1210.0962
13. Chen Z, Li T (2007) Addressing diverse user preferences in sql-query-result navigation. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007, pp 641–652
14. Cieliebak M, Eidenbenz S, Pagourtzis A, Schlude K (2008) On the complexity of variations of equal sum subsets. Nord J Comput 14(3):151–172
15. Cressie N, Whitford H (1986) How to use the two sample t-test. Biometrical Journal 28(2):131–148
16. Dai P, Rzeszotarski JM, Paritosh P, Chi EH (2015) And now for something completely different: Improving crowdsourcing workflows with micro-diversions. In: ACM CSCW, pp 628–638
17. Difallah D, Filatova E, Ipeirotis P (2018) Demographics and dynamics of mechanical turk workers. In: Proceedings of the eleventh acm international conference on web search and data mining, ACM, pp 135–143
18. Difallah DE, Catasta M, Demartini G, Cudré-Mauroux P (2014) Scaling-up the crowd: Micro-task pricing schemes for worker retention and latency improvement. In: Second AAAI Conference on Human Computation and Crowdsourcing
19. El-Arini K, Veda G, Shahaf D, Guestrin C (2009) Turning down the noise in the blogosphere. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009, pp 289–298
20. Esfandiari M, Borromeo RM, Nikookar S, Sakharkar P, Amer-Yahia S, Basu Roy S (2021) Multi-session diversity to improve user satisfaction in web applications. In: Proceedings of the Web Conference 2021, pp 1928–1936
21. Fan J, Lu M, Ooi BC, Tan WC, Zhang M (2014) A hybrid machine-crowdsourcing system for matching web tables. In: 2014 IEEE 30th International Conference on Data Engineering, IEEE, pp 976–987
22. Fan J, Li G, Ooi BC, Tan Kl, Feng J (2015) icrowd: An adaptive crowdsourcing framework. In: SIGMOD, pp 1015–1030
23. Garey MR, Johnson DS (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman
24. Gonzalez TF (1985) Clustering to minimize the maximum intercluster distance. Theoretical computer science 38:293–306
25. Han L, Roitero K, Gadiraju U, Sarasua C, Checco A, Maddalena E, Demartini G (2019) All those wasted hours: On task abandonment in crowdsourcing. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019, pp 321–329
26. Hariri N, Mobasher B, Burke R (2012) Context-aware music recommendation based on latenttopic sequential patterns. In: Proceedings of the sixth ACM conference on Recommender systems, pp 131–138
27. Hata K, Krishna R, Li F, Bernstein MS (2017) A glimpse far into the future: Understanding long-term crowd worker quality. In: Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing, CSCW 2017, Portland, OR, USA, February 25 - March 1, 2017, pp 889–901
28. Ho C, Vaughan JW (2012) Online task assignment in crowdsourcing markets. In: AAAI

29. Ho C, Jabbari S, Vaughan JW (2013) Adaptive task assignment for crowdsourced classification. In: ICML, pp 534–542
30. Jain A, Sarda P, Haritsa JR (2004) Providing diversity in k-nearest neighbor query results. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, pp 404–413
31. Kyriakidi M, Stefanidis K, Ioannidis Y (2017) On achieving diversity in recommender systems. In: Proceedings of the ExploreDB'17, pp 1–6
32. Leiserson CE, Rivest RL, Cormen TH, Stein C (2001) Introduction to algorithms, vol 6. MIT press Cambridge, MA
33. Michiels W, Korst J, Aarts E, Van Leeuwen J (2003) Performance ratios for the differencing method applied to the balanced number partitioning problem. In: Annual Symposium on Theoretical Aspects of Computer Science, Springer, pp 583–595
34. Nemhauser GL, Wolsey LA, Fisher ML (1978) An analysis of approximations for maximizing submodular set functions—i. Mathematical programming 14(1):265–294
35. Pilourdault J, Amer-Yahia S, Lee D, Roy S (2017) Motivation-aware task assignment in crowdsourcing. In: EDBT
36. Punnen A, Margot F, Kabadi S (2003) Tsp heuristics: domination analysis and complexity. Algorithmica 35(2):111–127
37. Puthiya Parambath SA, Usunier N, Grandvalet Y (2016) A coverage-based approach to recommendation diversity on similarity graph. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp 15–22
38. Qin L, Zhu X (2013) Promoting diversity in recommendation by entropy regularizer. In: Twenty-Third International Joint Conference on Artificial Intelligence
39. Rahman H, Roy SB, Thirumuruganathan S, Amer-Yahia S, Das G (2019) Optimized group formation for solving collaborative tasks. VLDB J 28(1):1–23
40. Rosenkrantz DJ, Tayi GK, Ravi S (2000) Facility dispersion problems under capacity and cost constraints. Journal of combinatorial optimization 4(1):7–33
41. Rzeszotarski JM, Chi E, Paritosh P, Dai P (2013) Inserting micro-breaks into crowdsourcing workflows. In: First AAAI Conference on Human Computation and Crowdsourcing
42. Stoline MR (1981) The status of multiple comparisons: simultaneous estimation of all pairwise comparisons in one-way anova designs. The American Statistician 35(3):134–141
43. Stratigi M, Nummenmaa J, Pitoura E, Stefanidis K (2020) Fair sequential group recommendations. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing, pp 1443–1452
44. SurveyMonkey (1999) Calculating the number of respondents you need. `https://help.surveymonkey.com/articles/en_US/kb/How-many-respondents-do-I-need`
45. Vargas S, Baltrunas L, Karatzoglou A, Castells P (2014) Coverage, redundancy and size-awareness in genre diversity for recommender systems. In: Proceedings of the 8th ACM Conference on Recommender systems, pp 209–216
46. Volkovs M, Rai H, Cheng Z, Wu G, Lu Y, Sanner S (2018) Two-stage model for automatic playlist continuation at scale. In: Proceedings of the ACM Recommender Systems Challenge 2018, pp 1–6
47. Wang D, Deng S, Xu G (2018) Sequence-based context-aware music recommendation. Information Retrieval Journal 21(2-3):230–252
48. Yu C, Lakshmanan L, Amer-Yahia S (2009) It takes variety to make a world: diversification in recommender systems. In: Proceedings of the 12th international conference on extending database technology: Advances in database technology, pp 368–378
49. Zhang M, Hurley N (2008) Avoiding monotony: improving the diversity of recommendation lists. In: Proceedings of the 2008 ACM conference on Recommender systems, pp 123–130
50. Zheng Y, Wang J, Li G, Cheng R, Feng J (2015) QASCA: A quality-aware task assignment system for crowdsourcing applications. In: SIGMOD, pp 1031–1046
51. Ziegler C, McNee SM, Konstan JA, Lausen G (2005) Improving recommendation lists through topic diversification. In: Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005, pp 22–32