Multi-Session Diversity to Improve User Satisfaction in Web Applications

Mohammadreza Esfandiari
New Jersey Institute of Technology
Newark, NJ, USA
me76@njit.edu

Ria Mae Borromeo
University of the Philippines
Laguna, Philippines
riamae@gmail.com

Sepideh Nikookar
New Jersey Institute of Technology
Newark, NJ, USA
sn627@njit.edu

Paras Sakharkar
New Jersey Institute of Technology
Newark, NJ, USA
ps863@njit.edu

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

Senjuti Basu Roy
New Jersey Institute of Technology
Newark, NJ, USA
senjuti.basuroy@njit.edu

Abstract

In various Web applications, users consume content in a series of sessions. That is prevalent in online music listening, where a session is a channel and channels are listened to in sequence, or in crowdsourcing, where a session is a set of tasks and task sets are completed in sequence. Content diversity can be defined in more than one way, e.g., based on artists or genres for music, or on requesters or rewards in crowdsourcing. A user may prefer to experience diversity within or across sessions. Naturally, *intra-session diversity* is set-based, whereas, *inter-session diversity* is sequence-based. This novel multi-session diversity gives rise to four bi-objective problems with the goal of minimizing or maximizing inter and intra diversities. Given the hardness of those problems, we propose to formulate *a constrained optimization problem that optimizes inter diversity, subject to the constraint of intra diversity*. We develop an efficient algorithm to solve our problem. Our experiments with human subjects on two real datasets, music and crowdsourcing, show our diversity formulations do serve different user needs, and yield high user satisfaction. Our large data experiments on real and synthetic data empirically demonstrate that our solution satisfy the theoretical bounds and is highly scalable, compared to baselines.

CCS Concepts

• **Information systems** → Crowdsourcing.

ACM Reference Format:
Mohammadreza Esfandiari, Ria Mae Borromeo, Sepideh Nikookar, Paras Sakharkar, Sihem Amer-Yahia, and Senjuti Basu Roy. 2021. Multi-Session Diversity to Improve User Satisfaction in Web Applications. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3442381.3450046

1 Introduction

Online content consumption is usually organized into sessions. That is prevalent in a variety of applications such as online music listening where 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. In 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 content inside (intra) and across (inter) sessions is natural for such applications to improve user satisfaction and engagement. In this paper, we define *multi-session diversity* and study its impact on user satisfaction in Web applications. *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* [3, 7, 12, 17, 22, 23, 26, 29–31, 34, 37–40].

Creating playlists to listen to 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 (by 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 assign tasks requiring similar skills within a session and different completion times across sessions. More generally, applications may require minimization or maximization of intra and inter diversities. Therefore, multi-session diversity aims to generate k sessions to a user, with a small number l of relevant items in each, yielding a total of $N = k \times l$ items. Intra and inter-session diversities can be either minimized or maximized which gives rise to 4 bi-objective problem variants.

Due to their hardness, we propose to formulate our 4 problems as a constrained optimization problem, with the goal of obtaining one point from the Pareto front **(Section 2.2)**. *The idea is to optimize inter diversity, subject to constraining intra diversity.* 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. We design an efficient algorithm to solve this problem. We design

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3450046

algorithms with provable guarantees for intra and inter problems individually and together **(Section 3)**.

We conduct four large-scale experiments: two with human subjects (music playlist and task recommendation), the other two with real and simulated data. 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 context.* In task recommendation, *we show that the benefit of diversity is more prominent for long sessions (with 5 sets and 10 tasks per set) compared to shorter sessions. In the latter, our algorithms achieve higher quality and worker satisfaction than a baseline with no diversity.* We investigate the scalability of our algorithms against several non-trivial baselines **(Section 4.2)**. We observe that in most cases, our algorithms produce approximation factors that are very close to 1. 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.

2 Data Model and Diversity Problem

We describe a simple running example in crowdsourcing.

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

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 generate 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:

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

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:

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

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 .

Other set-based [3] and sequence-based [40] definitions exist and could be considered in future work.

2.2 Diversity Problem

Given N items, we are interested in finding a sequence $S =< s_1, \ldots, s_k >$ of k sessions, each consisting of l items. Inter and intra-session diversities are maximized or minimized leading to four bi-objective optimization problems. Each one of those problems is NP-hard. In fact, two ((Min Inter, Max Intra) and (Max Inter, Max Intra)) out of the four problems are NP-hard on both objectives.[1] Therefore, we propose to formulate a constrained optimization problem with the goal of obtaining one solution among all non-dominated ones, i.e., one point from the Pareto front. *The idea is to optimize inter diversity, subject to the constraint of intra diversity.* We hence formulate the following problem:

$$\begin{aligned} &\min_S(\max) \sum_{i=1}^{k-1} (Inter^{d_2}(s_i, s_{i+1})) \\ &\text{s.t.} \\ &\sum_{i=1}^{k} (Intra(s_i))x <= OPT_{Intra^{d_1}} \\ &|S| = k,\ |s_i| = l, N = k \times l \end{aligned} \tag{3}$$

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 3). Table 2 summarizes our technical results.

3.1 Algorithm Min(Max)-Intra

We study Min Intra where the objective is to generate k sessions, each of length l, such that their aggregated *Intra* diversity is minimized. Specifically, if there are l values associated with a dimension in a session, the intra diversity is the variance of those points. With an abstract representation, once sorted, the dimension values of

[1] Full proofs will be made available in a deanonymized report.

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

N items, fall on a line. 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 we propose algorithm `Ex-Min-Intra` that 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. This algorithm produces an exact solution.

We now turn to Max Intra, an NP-hard problem.[2] 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 creates sessions that satisfy this property. First, it creates l bins, each has k different slots. Then, these bins are initialized in such a way that each bin contains a subset of k items from the set of items. The final two steps run in an iterative manner. In the third step, the algorithm scores the bins as follows:

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

, where μ_{global} is the global mean over all items. Finally, it greedily merges two bins. This process is repeated for $l - 1$ number of iterations.

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 $d(b_i)$. 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.

To merge $b - i$ with b_j, where $b - i$ has the largest score and b_j has the smallest score, we create a new bin b_{ij}^{merge} that contains the m-th smallest items of b_i and m-th largest items of b_j ($1 \leq m \leq k$). Considering Example 1, the new bin b_{23}^{merge} is created by combining b_2 and b_3 , such that

$$b_{23}^{merge} = \{[t_5, t_{12}], [t_6, t_{11}], [t_7, t_{10}], [t_8, t_9]\}$$

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

3.2 Algorithm Min(Max)-Inter

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 means 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.

[2]Full proofs will be made available in a deanonymized report.

Algorithm 1 Algorithm `Ap-Max-Intra`

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

1: $\mu_{global} \leftarrow$ Mean over 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_{ij}^{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

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 $w(s_i, s_j) = (\mu_{s_i} - \mu_{s_j})^2$ where μ_{s_i} (resp. μ_{s_j}) is the mean of session s_i (resp. s_j). 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 [25, 28]. 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 $O(k^2 + logk)$.

Using Example 1 to find *Max-Inter* of *Skill* dimension, we first apply the `Ex-Min-Intra` to find 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 $\mu_{s_4} = 0.873$. These sessions will become 4 nodes of a complete graph. The nodes of this graph are the sessions and the weight of each edge is the *Inter* value we get from Equation 2. We solve the longest path problem for this graph and we get the tour of $T = \{s_1 \rightarrow s_4 \rightarrow s_2 \rightarrow s_3 \rightarrow s_1\}$. We remove the edge $s_2 \rightarrow s_3$ since it has the smallest weight. The solution of *Max-Inter* is hence the sequence $S =< s_2, s_4, s_1, s_3 >$.

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.

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

3.3 Optimizing Inter with Intra as Constraint

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*.

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*.

Table 3 provides the summary of the theoretical guarantees of these algorithms.

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. Our code and data are available on GitHub.[3]

4.1 Experiments with Human Subjects

We validate how multi-session diversity improves user satisfaction in two Web applications: music recommendation, where we generate music channels, and task recommendation in crowdsourcing, where we generate task sessions.

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.

[3] https://github.com/Multi-Session-Diversity/WWW2021

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 4. The 5th playlist represents the baseline with no diversity. It consists of similar songs randomly selected from one of the dimensions. In this last experiment, 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 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 workers \$0.10 for profile collection and \$1.00 for their evaluations.

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

Table 4: Diversity dimensions per context

Summary of Results. We observe in Table 5 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) [32]. 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 Table 6. The sample size of 200 workers from the estimated 200,000 workers in AMT [14] gives our results a 99% confidence level and a 10% error margin (based on the Central Limit Theorem [33]). *In summary, our music experiment clearly shows that diversity is preferred over no diversity.* Additionally, diversity definitions depend on context, as observed in Table 6.

	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 5: Average evaluation scores across all contexts

	Scenario	Long Drive	Theme Party	Sunday Morning	Learn Music
1	*Min-Intra, Min-Inter*	**16.58**,3.64,3.62	14.86,3.52,3.88	14.76,3.64,3.34	14.42,3.46,3.32
2	*Min-Intra, Max-Inter*	15.82,3.70,3.76	**15.06**,3.50,3.72	14.12,3.82,3.66	**15.20**,3.61,3.50
3	*Max-Intra, Min-Inter*	16.52,3.70,**3.86**	13.64,3.54,**3.98**	14.30,3.58,3.56	14.38,3.54,3.44
4	*Max-Intra, Max-Inter*	16.24,**3.84**,3.76	13.96,**3.68**,3.80	**15.04**,**3.58**,**3.70**	13.40,**3.64**,**3.58**
5	*no diversity*	*14.10,3.34,3.60*	*11.92,3.30,3.42*	*13.62*,3.46,3.46	*11.68,3.30,3.28*

Table 6: Average number of selected songs, average diversity rating, and average user satisfaction per context

4.1.2 Task Recommendation. In these experiments, we recommend short and long task sessions to workers in crowdsourcing. The short sessions consist of 3 sets each with 3 tasks. The long sessions consist 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. For each session type (short and long), we collect 100 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. Additionally, we generate a *no-diversity* baseline session. In this session, we randomly pick a task type and tasks belonging to that type. Next, workers complete the recommended sessions. We measure *task throughput*, *quality* of the completed tasks with respect to a ground truth, and *worker satisfaction*. *Throughput* refers to the average number of tasks completed per minute. *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. *Satisfaction* refers to how satisfied workers are with the task sessions (a rating from 1 to 5 provided by each worker). We recruit 200 workers, pay each \$0.03 for profile collection and at least \$0.35 for task completion.

	Scenario	Through put	Quality (%)	Worker Satis-faction
1	*Min-Intra(creation date), Min-Inter(skill)*	6.13,6.95	**65.64**,68.33	4.47,4.48
2	*Min-Intra(skill), Max-Inter(reward)*	5.93,6.96	62.77,69.27	**4.43**,**4.5**
3	*Max-Intra(skill), Min-Inter (reward)*	5.91,6.96	*61.76*,**70.08**	4.44,4.41
4	*Max-Intra(duration), Max-Inter (skill)*	*5.35*,**6.98**	61.24,67.98	4.46,4.40
5	*no diversity*	**7.53**,*6.56*	64.38,*66.04*	*4.26,4.19*

Table 7: Task recommendation for short (first number) and long sessions (second number)

Summary of Results. We present the average throughput, quality, and worker satisfaction for short and long sessions in Table 7.

Similar to the music experiments, our sample size (n=200) allows our results to achieve 99% confidence level with 10% margin of error. We again used a one-way ANOVA to evaluate statistical significance. In short sessions, only throughput is statistically significant at $p = 0.05$. In long sessions, both quality and worker satisfaction are statistically significant at $p = 0.10$.

Our results indicate that short sessions generated by our algorithms do not significantly differ from the *no-diversity* baseline in terms of quality and worker satisfaction. On the other hand, the throughput of *no-diversity* is significantly higher than sessions generated by our algorithms. 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 [15]. As the number of tasks per session increases (long sessions) however, this observation changes. Throughput decreases for *no-diversity* and sessions generated by our algorithms obtain higher quality and worker satisfaction with statistical significance. *In summary, our experiments show that the benefit of diversity in task recommendation is more prominent for sessions comprising many tasks. Diversity tends to bring positive effect to avoid boredom which is prominent for sessions with many 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 [2, 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[4].

`Baseline-MMR`. This baseline is inspired by the MMR algorithm [10] 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.

Clustering Algorithms are not applicable to be used as baselines since they do not control session size, and they are not adapted to sequences.

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, our algorithms is highly scalable and is much more efficient than 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.

Our Scenarios	N=8192 , k=16		N=1024 , k=128	
	Intra	Inter	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 8: Approximation factors on 1-Million Song Dataset

Comparison against Optimal. Table 8 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 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.

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

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. Figure 1 shows how *Inter* scores change as we vary N from 2^{10} to 2^{16} for `Baseline-MMR`, `Random` and our algorithms. We have omitted the plots for Synthetic data experiments since those results closely follow the result for 1-Million Songs dataset. Figures 1(a)(c) confirms 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 1(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.

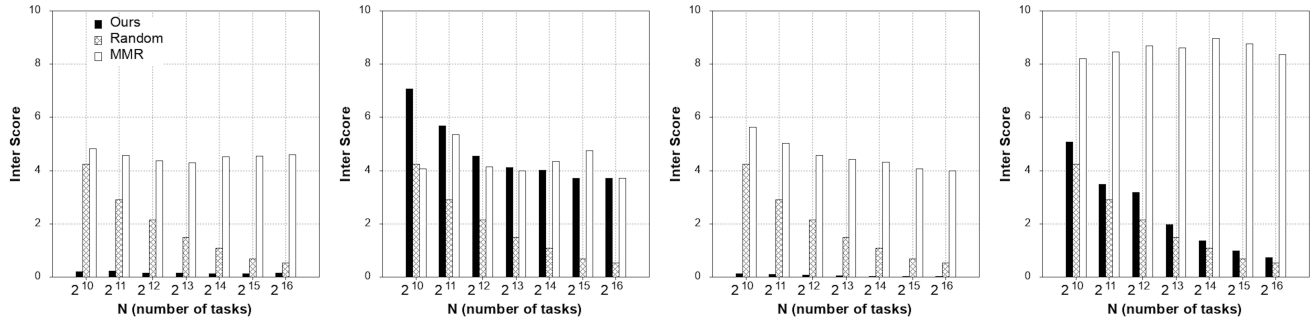
We also compare *Intra* scores whilst varying N. Table 9 showcases the approximation factors of our algorithm's *Intra* considering `Optimal` for $N \leq 2^{13}$ and $N > 2^{13}$. A ratio of 1 means that the algorithm produces the best or optimal solution. These results showcase that our solutions achieve even better bound empirically compared to the theoretical bounds. Table 9 also shows that although `Baseline-MMR` performs better in *Max-Inter* problem, but it performs poorly for both *Min-Intra* and *Max-Intra* problems.

Interestingly, `Random` produces an approximation factor close to 1 for $N > 2^{13}$ when maximizing *Intra*. This is because *Intra* is maximized when the variance of the sessions are maximized which is one of the side effects of `Random`. 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 2 presents 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}. The synthetic dataset also mimics this trend. We observe figures 2(a)(c) that our algorithm performs significantly better than other baselines in minimizing *Inter* diversity. For Figures 2(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 more sessions are present, it allows for more diversity between each session.

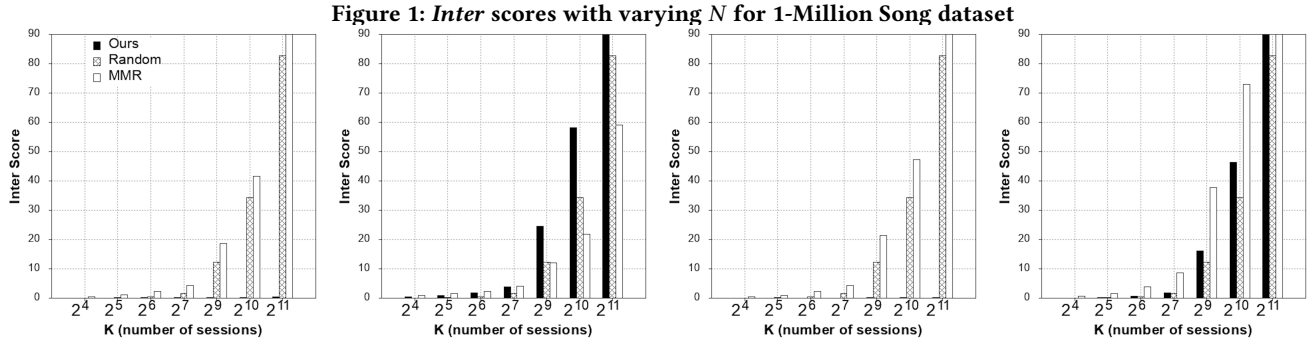
The approximation factors of *Intra*, hold when varying k (results omitted for space reasons).

Varying distribution. Figures 3 and 4 present how our algorithm and other baselines perform as we vary data distributions. We set N to 2^{13} and k to 2^7.



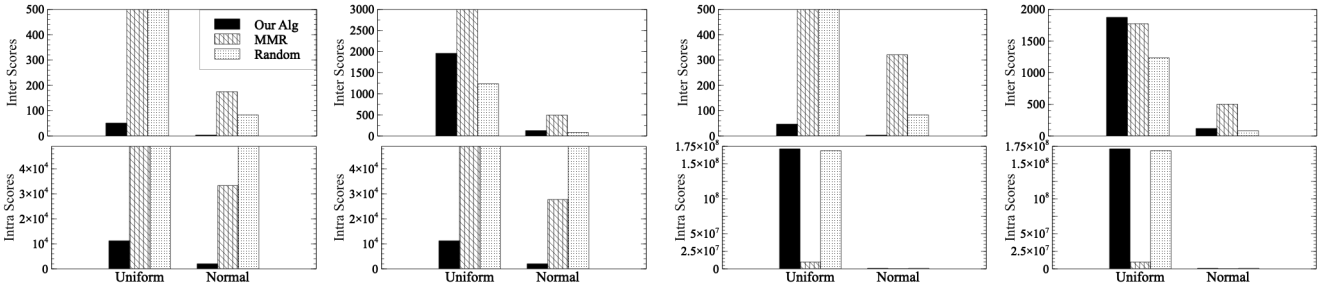
(a) Ap-Min-Inter*(Min-Intra)* (b) Ap-Max-Inter *(Min-Intra)* (c) Ap-Min-Inter *(Max-Intra)* (d) Ap-Max-Inter *(Max-Intra)*

Figure 1: *Inter* scores with varying N for 1-Million Song dataset



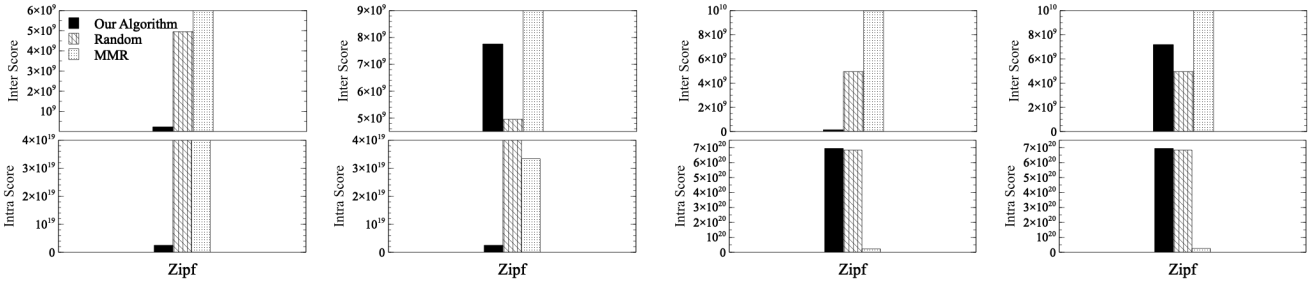
(a) Ap-Min-Inter *(Min-Intra)* (b) Ap-Max-Inter *(Min-Intra)* (c) Ap-Min-Inter *(Max-Intra)* (d) Ap-Max-Inter *(Max-Intra)*

Figure 2: *Inter* scores with varying k for 1-Million Song dataset



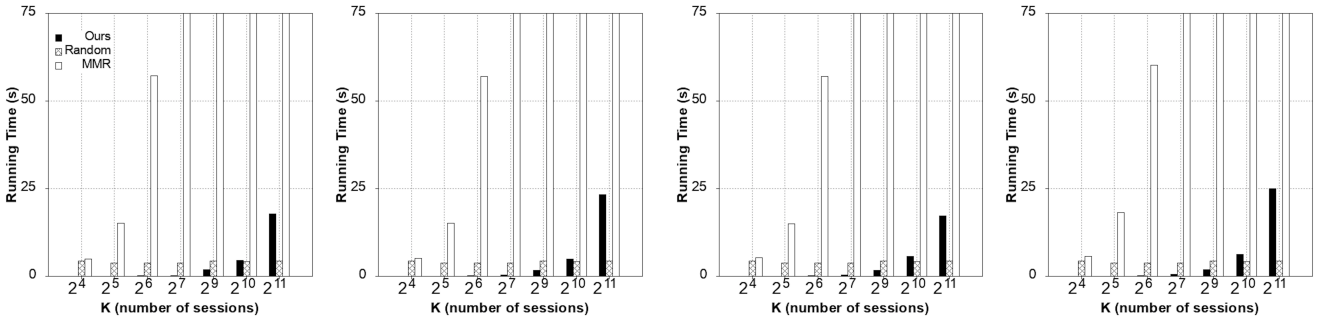
(a) *Min-Intra,Min-Inter* (b) *Min-Intra,Max-Inter* (c) *Max-Intra,Min-Inter* (d) *Max-Intra,Max-Inter*

Figure 3: Synthetic Data: *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*

Figure 4: Synthetic Data: Zipf Distribution



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

Figure 5: Running times varying k for 1-Million Song dataset

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

Table 9: *Intra* approximation factors varying N on 1-Million Song

Considering *Intra* scores, our algorithm performs the best using Uniform distribution for all 4 scenarios and using a Zipf distribution produces a similar trend. 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 3(b)(d), `Baseline-MMR` outperforms our algorithm due to the same reasons mentioned in the varying k and N section of this paper.

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.

4.2.5 Scalability Evaluation. We compare the running time of the three algorithms for 1-Million dataset.

In Figures 5, we vary k and set N to 2^{13}. Naturally, as N increases, the running time of our algorithms increases. We observe that our algorithms scale very well but are 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, with increasing values of k, our algorithms are consistently faster. We also observe that as we vary N with $k = 2^7$, our algorithms are the fastest in all diversity scenarios (plots are omitted for space reasons).

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 [3, 7, 12, 17, 22, 23, 26, 29–31, 34, 36–40], 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 [20, 35] have also acknowledged the need for diversity and sequence based modeling in different recommendation applications. Recent works in crowdsourcing [18, 27] 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 [11]. Amer-Yahia et al. [6] 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 [18] involving crowd. Even though completing similar tasks lead to faster completion time [15], but such composition lead to fatigue and boredom, and task abandonment [13, 19, 21]. Aipe and Gadiraju[4] empirically observe that workers who perform similar tasks achieve higher accuracy and faster task completion time compared to workers who complete 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 [5]. 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 [3, 17, 26, 29, 30, 34] or sequence-based only [7, 12, 22, 40]. As an example, in [40], the authors study sequence-based diversity that is defined as the diversity of any permutation of the items. Another example is [7], in which taxonomies are used to sample search results to reduce homogeneity. In [3], the authors propose an algorithm with a provable approximation factor to find relevant and diverse news articles. In the database context, Chen and Li [12] propose to post-process structured query results, organizing them in a decision tree for easier navigation. In [8, 24], 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 [16], recommendation diversity is formulated as a set-cover problem.

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 scalable algorithmic framework and experimental studies to address multi-session diversity to improve user satisfaction in Web 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 develop algorithms to solve our problems. 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.

This work opens up more than one research directions: an immediate extension of our work is to observe users as they consume items and learn how diversity could be personalized.

Acknowledgments

The work of Mohammadreza Esfandiari, Sepideh Nikookar, Paras Sakharkar, and Senjuti Basu Roy are supported by the National Science Foundation, CAREER Award #1942913,IIS #2007935, IIS #1814595, and by the Office of Naval Research Grant No:N000141812838.

References

[1] [n.d.]. Figure Eight - Data For Everyone. https://www.figure-eight.com/data-for-everyone/. Accessed 25 January 2019.
[2] [n.d.]. *Million Song Database.* http://millionsongdataset.com/
[3] Sofiane Abbar, Sihem Amer-Yahia, Piotr Indyk, and Sepideh Mahabadi. 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.* 1–12.
[4] Alan Aipe and Ujwal Gadiraju. 2018. SimilarHITs: Revealing the Role of Task Similarity in Microtask Crowdsourcing.. In *HT*. 115–122.
[5] Maha Alsayasneh, Sihem Amer-Yahia, Eric Gaussier, Vincent Leroy, Julien Pilourdault, Ria Mae Borromeo, Motomichi Toyama, and Jean-Michel Renders. 2017. Personalized and diverse task composition in crowdsourcing. *IEEE Transactions on Knowledge and Data Engineering* 30, 1 (2017), 128–141.
[6] Sihem Amer-Yahia, Eric Gaussier, Vincent Leroy, Julien Pilourdault, Ria Mae Borromeo, and Motomichi Toyama. 2016. Task composition in crowdsourcing. In *Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on*. IEEE, 194–203.
[7] Aris Anagnostopoulos, Andrei Z. Broder, and David Carmel. 2006. Sampling Search-Engine Results. *World Wide Web* 9, 4 (2006), 397–429.
[8] Albert Angel and Nick Koudas. 2011. Efficient diversity-aware search. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. 781–792.
[9] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. 2011. The million song dataset. (2011).
[10] Jaime G Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries.. In *SIGIR*, Vol. 98. 335–336.
[11] Dana Chandler and Adam Kapelner. 2012. Breaking Monotony with Meaning: Motivation in Crowdsourcing Markets. *CoRR* abs/1210.0962 (2012).
[12] Zhiyuan Chen and Tao Li. 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*. 641–652.
[13] Peng Dai, Jeffrey M. Rzeszotarski, Praveen Paritosh, and Ed H. Chi. 2015. And Now for Something Completely Different: Improving Crowdsourcing Workflows with Micro-Diversions. In *ACM CSCW*. 628–638.
[14] Djellel Difallah, Elena Filatova, and Panos Ipeirotis. 2018. Demographics and dynamics of mechanical Turk workers. In *Proceedings of the eleventh acm international conference on web search and data mining*. ACM, 135–143.
[15] Djellel Eddine Difallah, Michele Catasta, Gianluca Demartini, and Philippe Cudré-Mauroux. 2014. Scaling-up the crowd: Micro-task pricing schemes for worker retention and latency improvement. In *Second AAAI Conference on Human Computation and Crowdsourcing*.
[16] Khalid El-Arini, Gaurav Veda, Dafna Shahaf, and Carlos Guestrin. 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*. 289–298.
[17] Ju Fan, Guoliang Li, Beng Chin Ooi, Kian-lee Tan, and Jianhua Feng. 2015. iCrowd: An Adaptive Crowdsourcing Framework. In *SIGMOD*. 1015–1030.
[18] Ju Fan, Meiyu Lu, Beng Chin Ooi, Wang-Chiew Tan, and Meihui Zhang. 2014. A hybrid machine-crowdsourcing system for matching web tables. In *2014 IEEE 30th International Conference on Data Engineering*. IEEE, 976–987.
[19] Lei Han, Kevin Roitero, Ujwal Gadiraju, Cristina Sarasua, Alessandro Checco, Eddy Maddalena, and Gianluca Demartini. 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*. 321–329.
[20] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2012. Context-aware music recommendation based on latenttopic sequential patterns. In *Proceedings of the sixth ACM conference on Recommender systems*. 131–138.
[21] Kenji Hata, Ranjay Krishna, Fei-Fei Li, and Michael S. Bernstein. 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*. 889–901.
[22] Chien-Ju Ho, Shahin Jabbari, and Jennifer Wortman Vaughan. 2013. Adaptive Task Assignment for Crowdsourced Classification. In *ICML*. 534–542.
[23] Chien-Ju Ho and Jennifer Wortman Vaughan. 2012. Online Task Assignment in Crowdsourcing Markets. In *AAAI*.
[24] Anoop Jain, Parag Sarda, and Jayant R Haritsa. 2004. Providing diversity in k-nearest neighbor query results. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 404–413.
[25] Charles Eric Leiserson, Ronald L Rivest, Thomas H Cormen, and Clifford Stein. 2001. *Introduction to algorithms*. Vol. 6. MIT press Cambridge, MA.
[26] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functionsâĂŤI. *Mathematical programming* 14, 1 (1978), 265–294.
[27] Julien Pilourdault, Sihem Amer-Yahia, Dongwon Lee, and Senjuti Roy. 2017. Motivation-aware task assignment in crowdsourcing. In *EDBT*.
[28] Abraham Punnen, FrancoiS Margot, and Santosh Kabadi. 2003. TSP heuristics: domination analysis and complexity. *Algorithmica* 35, 2 (2003), 111–127.
[29] 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.
[30] Lijing Qin and Xiaoyan Zhu. 2013. Promoting diversity in recommendation by entropy regularizer. In *Twenty-Third International Joint Conference on Artificial Intelligence*.
[31] Habibur Rahman, Senjuti Basu Roy, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. 2019. Optimized group formation for solving collaborative tasks. *VLDB J.* 28, 1 (2019), 1–23.
[32] Michael R Stoline. 1981. The status of multiple comparisons: simultaneous estimation of all pairwise comparisons in one-way ANOVA designs. *The American Statistician* 35, 3 (1981), 134–141.
[33] SurveyMonkey. [n.d.]. Calculating the Number of Respondents You Need. https://help.surveymonkey.com/articles/en_US/kb/How-many-respondents-do-I-need.
[34] Saúl Vargas, Linas Baltrunas, Alexandros Karatzoglou, and Pablo Castells. 2014. Coverage, redundancy and size-awareness in genre diversity for recommender systems. In *Proceedings of the 8th ACM Conference on Recommender systems*. 209–216.
[35] Maksims Volkovs, Himanshu Rai, Zhaoyue Cheng, Ga Wu, Yichao Lu, and Scott Sanner. 2018. Two-stage model for automatic playlist continuation at scale. In *Proceedings of the ACM Recommender Systems Challenge 2018*. 1–6.
[36] Dongjing Wang, Shuiguang Deng, and Guandong Xu. 2018. Sequence-based context-aware music recommendation. *Information Retrieval Journal* 21, 2-3 (2018), 230–252.
[37] Cong Yu, Laks Lakshmanan, and Sihem Amer-Yahia. 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*. 368–378.
[38] Mi Zhang and Neil Hurley. 2008. Avoiding monotony: improving the diversity of recommendation lists. In *Proceedings of the 2008 ACM conference on Recommender systems*. 123–130.
[39] Yudian Zheng, Jiannan Wang, Guoliang Li, Reynold Cheng, and Jianhua Feng. 2015. QASCA: A Quality-Aware Task Assignment System for Crowdsourcing Applications. In *SIGMOD*. 1031–1046.
[40] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. 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*. 22–32.