Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

CHANGING THE FOCUS: WORKER-CENTRIC OPTIMIZATION IN HUMAN-IN-THE-LOOP COMPUTATIONS

by
Mohammadreza Esfandiari

A myriad of emerging applications from simple to complex ones involve human cognizance in the computation loop. Using the wisdom of human workers, researchers have solved a variety of problems, termed as "micro-tasks" such as, captcha recognition, sentiment analysis, image categorization, query processing, as well as "complex tasks" that are often collaborative, such as, classifying craters on planetary surfaces, discovering new galaxies (Galaxyzoo), performing text translation. The current view of "humans-in-the-loop" tends to see humans as machines, robots, or low-level agents used or exploited in the service of broader computation goals. This dissertation is developed to shift the focus back to humans, and study different data analytics problems, by recognizing characteristics of the human workers, and how to incorporate those in a principled fashion inside the computation loop.

The first contribution of this dissertation is to propose an optimization framework and a real world system to personalize worker's behavior by developing a worker model and using that to better understand and estimate task completion time. The framework judiciously frames questions and solicits worker feedback on those to update the worker model. Next, improving workers skills through peer interaction during collaborative task completion is studied. A suite of optimization problems are identified in that context considering collaborativeness between the members as it plays a major role in peer learning. Finally, "diversified" sequence of work sessions for human workers is designed to improve worker satisfaction and engagement while completing tasks.

CHANGING THE FOCUS: WORKER-CENTRIC OPTIMIZATION IN HUMAN-IN-THE-LOOP COMPUTATIONS

by
Mohammadreza Esfandiari

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
In Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Department of Computer Science

August 2020

APPROVAL PAGE

CHANGING THE FOCUS:
WORKER-CENTRIC OPTIMIZATION IN HUMAN-IN-THE-LOOP COMPUTATIONS

Mohammadreza Esfandiari

Dr. Senjuti Basu-Roy, Dissertation Advisor — Date
Assistant Professor of Computer Science,NJIT

Dr. Vincent Oria, Committee Member — Date
Professor of Computer Science, NJIT

Dr. Quentin Jones, Committee Member — Date
Associate Professor of Information Systems, NJIT

Dr. Yi Chen, Committee Member — Date
Professor of Business Data Science, School of Management, NJIT

Dr. Sihem Amer-Yahia, Committee Member — Date
Research Director at Laboratoire d'Informatique de Grenoble, University of Grenoble-Alpes, France

BIOGRAPHICAL SKETCH

Author: Mohammadreza Esfandiari

Degree: Doctor of Philosophy

Date: August 2020

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2020
- Bachelor of Science in Computer Science,
  Sharif University of Technology, Tehran, Iran, 2012

Major: Computer Science

Presentations and Publications:

Mohammadreza Esfandiari and Dong Wei and Sihem Amer-Yahia and Senjuti Basu Roy, "Optimizing Peer Learning in Online Groups with Affinities," *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp 1216-1226, 2019.

Mohammadreza Esfandiari and Senjuti Basu Roy and Sihem Amer-Yahia , "Explicit Preference Elicitation for Task Completion Time," *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp 1233-1242, 2018.

Mohammadreza Esfandiari and Kavan Bharat Patel and Sihem Amer-Yahia and Senjuti Basu Roy, "Crowdsourcing Analytics With CrowdCur," *Proceedings of the 2018 International Conference on Management of Data*, pp 1701-1704, 2018.

To Sizi. For her endless support and love.

ACKNOWLEDGMENT

I would like to express my deepest appreciation to Dr. Senjuti Basu Roy, who not only served as my research advisor, providing valuable technical and writing guidance but also constantly gave me support, encouragement, and reassurance through my years of PhD. Her extensive knowledge of life, great expertise in research, and humanity has affected me profoundly and benefited me significantly in the past and will be of important value in my future career.

Sincere appreciation goes to Prof. Sihem Amer-Yahia who has been the source of inspiration and constant support throughout my doctoratal study.

I also would to thank my committee members, Prof. Vincent Oria, Prof. Quentin Jones, Prof. Yi Chen for their support.

I would like to thank the Department of Computer Science and the National Science Foundation for providing support through my years of studies.

Last, but not least, I would like to thank Dong Wei, my friend, colleague, and collaborator, and students of Big Data Analytics Lab at NJIT from whom I have learned a lot about research and otherwise.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Overview

Bringing human in the computational loop has been the focus of research for the past couple of decades [75, 93, 92]. It usually involves very close relationship between the human workers and the application and one of the the main goals of such systems is to enable human workers and computers to cooperate in making decisions and solving problems. Recently, an emerging trend is to leverage online infrastructures to tap an under explored and richly heterogeneous pool of knowledge resident in the general population of consumers for innovative ideas, a practice termed as human-in-the-loop (HIL) computation [85, 82, 128, 47, 81, 77]. These problems usually range from simple like sorting [8], filtering, sentiment analysis [88], top-K ranking [108] to more knowledge-intensive problems like clustering [86, 129], anomaly detection [24], entity resolution [30, 20], or product design. The humans, so called workers, are involved in the process as individual contributors for so called "micro-tasks", or as a group to collaboratively work and solve a problem.

Human workers are better off in solving certain problems that are otherwise harder for machine algorithms. Studies suggest that human brains are uniquely capable for language processing, pattern recognition, and invention [84]. As an example, "The Last Tomb of Genghis Khan" project [76] sponsored by National Geographic have used the human pattern recognition power to tag a massive number of satellite images to resolve the age-old mystery about the location of the tomb of this ancient Mongolian emperor. Audio perception is another difficult problem where creating a learning algorithm to recognize the speech and transform it into a text requires a large amount of data but is rather trivial for human workers. Games like

Chess [56] and Go [22] that are computationally intensive for computers are quite well mastered by humans, or problems such as the Traveling Salesman [52, 58, 78]and Graph Coloring [76] that are computationally intractable by machines are easy to handle by humans even for large instances. The emerging trend of HIL computations draws inspirations from these myriad examples.

Our focus in this dissertation however is to understand and exploit optimization opportunities in such HIL computation process and design principled solutions at scale. Usually, these opportunities fall into one of the three categories: application-centric optimization, platform-centric optimization, and worker-centric optimization. In application-centric optimization, the factors of interest usually are task throughput which is the number of tasks that is done in unit of time, latency which measures how long does it take for a set of tasks to finish, cost which measures the monetary expenses or other expenses incur as a result of using human workers, and quality which measures how good the answer of the human worker is. Platform-centric optimization also considers factors similar to application-centric optimization. For example, revenue which is the amount of money the platform can make or providing simplification techniques for complex tasks. Existing works have studied these aspects to different extent: as an example, recent works have studied optimization opportunities in entity matching problems [70, 101] with the goal to optimize the quality. Magellan [70], Waldo [124] develop algorithms on human workers to minimize the cost incurred both in terms of monetary compensation while improving the quality of work. In [13, 12], workers are used to gather information and obtain recommendations for pattern mining. The idea is to enable users to pose general questions, mine the responses for potentially relevant data, and to receive concise yet relevant answers that represent frequent, significant data patterns. The optimization goal here is to minimize cost. For collaborative tasks, similar optimization problems are designed to minimize cost and latency or maximize quality [112, 105, 69, 114, 64].

For example, to find craters on different planets like Mars, non-expert human workers have been shown to have similar accuracy to experts by collaborating and learning about how to identify them more efficiently. In Crowd4U [64]collaborative tasks such as text translation or video subtitle generation are performed by a group of collaborating workers by optimizing quality and cost.

A major issue with the *existing HIL systems is that they tend to push humans further in the loop and largely ignore their role and contribution to the ecosystem. Human workers are mere agents and are used to pursue the broader computational goal. In fact, the existing literature of HIL computation has a clear bifurcation. The computational world of research has purely focused on treating humans as agents (as described in the paragraph above), whereas, the psychologists and social scientists [35, 109, 28, 62] have acknowledged and reasoned about the necessity to incorporate human characteristics in the computational loop. However, this latter genre of researchers have treated these problems purely empirically and are limited to field study and laboratory experiments.* Clearly, designing algorithms that recognize and capture human characteristics and study optimization in the HIL process has not been studied well yet. Combining these two different genre of research, in essence, is the primary contribution of this dissertation.

Similar to application and platform-centric factors, in this dissertation we look at several worker-centric factors. The first one is *preference* of the worker. An implicit assumption shared by most of the works in HIL systems is that the workers are willing to perform the tasks assigned to them. In practice, however, different task-performing intentions and preferences can lead to different types of behaviors. A worker is unlikely to honestly and promptly complete the assigned tasks when she is not interested in them, which cannot guarantee the quality of task results. Another interesting factor is the *skill* of the workers where it has a pronounced impact on the quality of the work returned by worker. This measure can be derived internally from the platform,

in terms of 'ranking' points, from common denominators including level of education or by using a questioner or exam [61]. *Boredom* and *fatigue* [117] are another factors of interest [113, 132, 68]. These factors contribute greatly to quality of the work or throughput of the system and can be measured by looking at response time and accuracy of each task.

Our goal in this dissertation is to *bring human back in the center stage of the computational loop and study optimization opportunities that arise to computationally solve these problems at scale. In particular, we investigate optimization opportunities that arise by modeling workers or studying factors that impact their performance inside such systems (such as skill, preference, boredom, etc) which in turn optimizes the underlying computational problems (in terms of quality, latency, cost).* We borrow inspirations from a handful of very recent works[111, 11, 96] that *intend to computationally model human characteristics in the loop*, such as *motivation*, *worker skill*, *motivation*, *relevance*, *pay off* but their treatment to human characteristics is still nascent. A recent study [83] argues that in order to maintain the attractiveness of HIL platforms, it is critical to enable *worker-centric optimization*. To that end, existing works have designed empirical studies to incentivize workers for long-lasting tasks [28, 62] or entertaining them during task completion [35]. More recently, researchers have tried to learn worker's behavior implicitly by creating a learning model that looks at worker's past history to adaptively perform task assignment [96]. Others [40] look at social media data of the worker in order to understand their preferences and assign an appropriate task to them.

In summary, we design models and algorithms to learn the worker's preferences by explicitly eliciting her feedback to estimate task completion time (Chapter 2). We also design real-world systems that demonstrate such models and algorithms in action on large scale real-world applications (Chapter 3). Next, we investigate how to improve worker's skills through peer interaction and learning, especially for group-

based tasks and collaborative environments (Chapter 4). Finally, we explore how to introduce diversity into a sequence of sets of tasks to improve worker satisfaction and reduce boredom and fatigue (Chapter 5).

1.2 Studied Problems

Although using the crowd as a computational building block is exciting and encouraging, it could also introduce some complications to the problem. As we discussed earlier, researchers have focused on solving this problem by optimizing how to assign tasks to workers, how to deal with the noise introduced in the process, or how to implicitly understand what a worker is interested in. *In this dissertation, we depart from the traditional view of optimizing the application and platform-centric parameters and bring workers to the center stage.* In particular, we investigate optimization opportunities that arise either by modeling worker behavior or enabling factors that impact their performance inside a human-in-the-loop system. In this section, we give an overview of the approach we have taken and the problems that we solve.

1.2.1 Eliciting Explicit Feedback

The main actors of a HIL platform are requesters and tasks, and workers who complete them. Understanding quality indicators in HIL computation has been a recent research focus [68, 35, 46, 48]. Some work focuses on estimating quality indicators such as engagement and motivation [80, 115, 96], and on revisiting this estimation periodically in an implicit manner. An important open question however is, **can we improve the estimation of quality indicators by seeking explicit preferences from workers?**

On Amazon Mechanical Turk or Prolific Academic, a task has factors such as *type* (e.g., image annotation, ranking, sentiment analysis), *payment* and *duration*, i.e.,

the time allotted to complete a task. Workers are characterized by their *preferences for task factors* [111, 11]. Our first contribution is to propose *an optimization framework* `ExPref`, within which an individual *worker model*, that captures *the preference of workers for task factors*, is learned and maintained to estimate *task completion time*. *Worker Model* is "supervised" in nature and it is initialized by deriving principles from *active learning* [34]. Indeed, worker preference on task factors, such as, payment, task types, are indicators of how much time she needs to complete the task. Task completion time is an important quality indicator in HIL platform, as deeper understanding and analysis of task completion time benefits customization of payment strategies [46, 48], task assignment, and appropriate recruitment of workforce for HIL platforms [60, 59, 112, 96].

Unless `ExPref` is updated periodically, it is likely to become outdated, as worker's preferences evolve over time (e.g., a worker's skills improve as she completes tasks). To update the model, we advocate the need to **explicitly elicit from a worker her preferences**. **That is a stark departure from the literature where workers are observed and their preferences computed implicitly**. An additional challenge arises to address the following question: *in what fashion these preferences should be extracted and used to produce a scalable and accurate model?*. To that end, we present **Question Selector** for optimizing preference elicitation that asks a worker to rank the k task factors that lead to minimizing error in the model. For example, a worker may be asked *"Rank task relevance and payment"*. A higher rank for payment will indicate the worker's preference for high paying tasks over those most relevant to her profile. We prove that optimally selecting k questions, i.e., k task factors, for explicit worker preference is NP-hard, even when the *Worker Model* is linear. Consequently, we develop an efficient alternative using an iterative greedy algorithm that has a provable approximation bound. Once the worker provides her preference, the *next challenge is how to consume that feedback to update Worker Model*

in a principled manner. We present **Preference Aggregator** to update the *Worker Model* with the elicited preferences. To ensure that `ExPref` does not necessarily incur additional burden, worker's response can be: *a total order over those k factors*; or *a partial* preference over a subset of those k factors, when the worker does not/ cannot provide total ordering. Of course, an extreme case is that the worker does not provide any preference and in that case, the *Worker Model* is updated implicitly. We formulate those choices as a constrained optimization problem and develop efficient solutions.

1.2.2 System CrowdCur

In order to showcase the power of `ExPref`, we design and deploy a real-world HIL platform that demonstrates the models and algorithms in action on a large scale real-world application. The platform consists of several components and at the heart of it lies the *Worker Curation* box which is in charge of understanding and modeling worker's preferences. The platform also provides an OLAP engine that gives insights to workers and requesters and platform owners. As a worker, she can understand her preference changes in the course of her task completion. She also can get curated suggestions based on the available task pool and her latest preferences. A requester can understand the structure of the workforce and get the latest information about her task and projected completion time of her tasks based on the current workforce. Platform owners also can benefit from this system by looking at different dimensions of the system namely, workers, tasks, and requesters. She also can understand the inter-connectivity of tasks and the workforce.

1.2.3 Peer Learning in Online Platforms

The emergence of platforms that support online networked technologies has changed the way we communicate, collaborate, and learn things together. Learning in groups has been studied extensively in the field of psychology and educational

research [63, 38, 23]. In computer science, existing works have focused on how to identify and rank groups and communities [25], how to efficiently form a set of groups to optimize different group recommendation semantics [110], or form groups for task assignment [14, 15, 72, 112, 104]. The effect of online collaboration however goes beyond, as it enables powerful and versatile strategies to improve knowledge of individuals and promote learning. For example, online critiquing communities,[1] social Q&A sites,[2] and crowdsourcing platforms[3] investigate how collaboration can promote knowledge and skill improvement of individuals [63]. *Learning potential*, is a key reason behind effective collaboration. It has been shown that the increase in learning one expects from collaboration yields fruitful coordination and higher quality contributions [3, 4]. For instance, in online fan-fiction communities, informal mentoring improves people's writing skills [43]. In this dissertation, we focus on improving skills and learning through peer interaction. Existing methods usually focus on ranking communities [25], effectively forming groups [110] or create groups of workers in crowdsourcing environments [14, 15, 72, 112, 104]. Again, by changing the focus on the workers and peers, we try to elevate their experience and more importantly their skillset. The two major components in this work are *Learning potential* and *Affinity*. Research [3, 4, 43, 15] indicates worker with a higher affinity toward each other and with varying degree of skills, will see a higher raise in their skills. Several problems need to be addressed. First and foremost, how to formalize learning and affinity in the context of online learning. These factors can take several structures which will have a different effect. Next, since this problem becomes bi-objective optimization, we need to present efficient algorithms to solve both factors with provable guarantees. This proves to be hard since all the instances of the problem are NP-Hard. We show this improves skills and knowledge of workers significantly.

[1] `https://movielens.org/`(accessed on Jan 17, 2019)
[2] `http://quora.com/`(accessed on Jan 17, 2019)
[3] `https://www.figure-eight.com/`(accessed on Jan 17, 2019)

1.2.4 Diversifying Recommendations

Finally, we explore satisfaction and boredom in session-based applications by looking at how diversity will affect different factors of a worker or a user of an online platform. 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. In HIL computation, 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, the problem of assigning tasks to workers *in a single session* has been studied extensively [44, 59, 60, 133, 105]. Most works focused on satisfying both workers' needs such as expected reward, and requesters' criteria such as budget [46, 48]. However, when a worker comes back to the platform multiple times (*each referred to as a session*), the diversity of tasks undertaken by the worker across sessions affects the worker's experience and performance. That was partially verified in recent work. In [97], it was shown empirically that task throughput improved when workers were given similar tasks. In [57], workers experienced boredom and fatigue after completing

similar tasks for a while. In [9], workers explicitly requested tasks posted by different requesters to build a reputation.

These aforementioned scenarios have three things in common: first, diversity needs to be accounted for in the design of a sequence of sessions. Second, both minimization and maximization of diversity are meaningful. Finally, the dimensions on which intra and inter-session diversities are expressed are *factors* that may not be related - hence they cannot be combined.

In this dissertation, we introduce the problem of optimizing *diversity* inside a set of items and between sets of items at the same time. More specifically, we are interested in optimizing diversity inside a set of items (*Intra Diversity*) and between consecutive sets of items (*Inter Diversity*). This gives rise to four different optimization problems which are all NP-Hard. We provide efficient and scalable solutions for all four variants and showcase what happens if one uses these in real settings like music recommendation or a HIL system.

CHAPTER 2

PREFERENCE ELICITATION FOR TASK COMPLETION TIME

2.1 Introduction

This section of dissertation focuses on the problem defined in Section 1.2.1. We start by formalizing the problem (Section 2.2.1) and present the inner working of the framework `ExPref`. Next, in Section 2.3 we focus on presenting efficient and scalable algorithms to solve those problems. Last, but not least, Section 2.4 presents an extensive empirical study with real workers on Amazon Mechanical Turk platform. To summarize the contribution of this chapter :

- `ExPref`, a framework that elicits explicit worker preference to better estimate task completion time. `ExPref` has a *Worker Model* that captures worker preferences for task factors.
- A formalization of two core problems: **Question Selector** that asks a worker to rank k task factors, and **Preference Aggregator** that updates the model with elicited preferences.
- An in-depth analysis and solutions with provable guarantees for the *Worker Model*, the **Question Selector**, and the **Preference Aggregator**.
- Extensive experiments that corroborate that explicit preference elicitation outperforms implicit preference computation [96] and that our framework scales well.

2.2 Framework and Formalism

We present our proposed framework and formalize the problems.

2.2.1 `ExPref` Framework

We propose an iterative framework `ExPref` (refer to Figure 2.1) that is designed to ask personalized questions to a worker to elicit her preferences. The rationale is that while task factors are stable, a worker's preference evolves as workers undertake tasks [57, 96]. We propose a *Worker Model* that consumes task factors and predicts

Table 2.1 Table of Important Notations

Notation	Definition
n, m	# tasks, # task factors
$\vec{t}$	task t represented by a vector of m factors
$\mathcal{T}$	Task factor matrix
$\vec{w}$	worker w's preference vector
$\mathcal{Q}^k$	a set of selected k questions
y_t	completion time of task t
$\vec{Y}$	vector representing y_t over a set of tasks
$\mathcal{F}$	Worker Model

for a task, how long will the worker spends on the task, *by inferring her preferences.* However, unless the *Worker Model* is refreshed or updated periodically, it is likely to become outdated, as worker preference evolves over time [57, 11, 96]. To update the model, one has to periodically invoke an explicit preference elicitation step, called **Question Selector** that selects a set of k task factors and asks worker w to rank them. Once the worker provides her preference, the *Worker Model* is updated by the **Preference Aggregator**.

This information could be used in many places to characterize the workforce of a crowdsourcing platform and enable several improvements such as the analysis of workers' fatigue [57] and motivation, and better task assignment to workers [60, 59, 112, 96].

Two computational problems form the heart of this framework. **1. Question Selector:** - when invoked, selects the best set of k questions to elicit a worker's preference for task factors. **2. Preference Aggregator:** - takes a worker's preference to the questions into account, and updates the *Worker Model*. The last two components work in sequence, given the necessity to refine the learned model. The technical challenge is to update the model while satisfying the preference the worker has provided.

2.2.2 Data Model and Problem Definitions

Task Factors. Task characteristics are commonly defined by the platform and their values by requesters. Each task t in a set of n given tasks is characterized by a set of m factors whose values are either explicitly present or could be extracted (such as keywords, duration, pay-off). For this work, we assume that for every task, its factors are given. This gives rise to a task factor matrix $\mathcal{T}$.

Example 1. *The matrix in Table 1 contains six tasks characterized by factors, such as* type, payoff, duration*. Example types are image annotation, ranking and sentiment analysis. Payoff determines the $ value the workers receives as payment, whereas, duration is an indication of the maximum time a worker needs to complete that task.*

$task-id$	*annotation*	*ranking*	*sentiment*	*payoff*	*duration*	*completion time*
$t1$	1	0	0	20	35	25
$t2$	1	0	0	5	5	35
$t3$	0	1	0	5	10	45
$t4$	0	1	0	5	40	5
$t5$	0	0	1	10	10	12
$t6$	0	0	1	20	30	23

Given a task t that a worker w undertakes (either via self-appointment or via an assignment algorithm), we are interested to understand and estimate task completion time, the time spent by the worker to perform the task. The last column of the task factor matrix indicates completion time of the individual tasks. When the worker is arriving in the platform for the first time, we use a budget b to initialize the *Worker Model* by asking workers b questions. Afterwards, we periodically update the model by seeking explicit feedback through **Question Selector** and update workers preference using **Preference Aggregator** in the *Worker Model*.

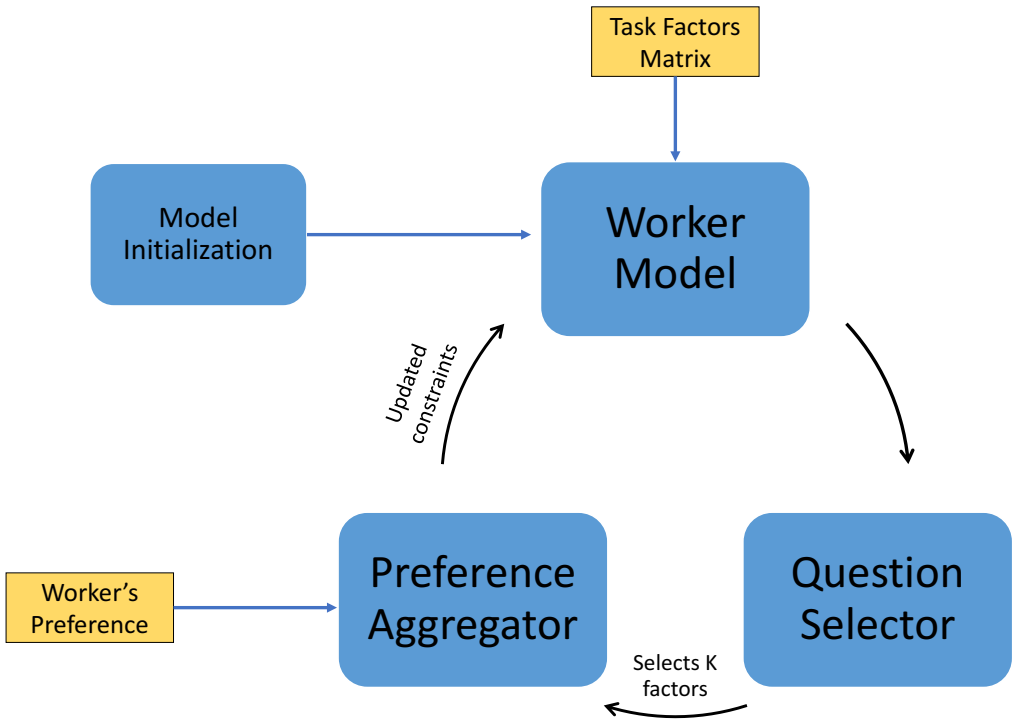


Figure 2.1 The `ExPref` framework.

Worker Preferences. The preferences of a worker w are represented by a vector $\vec{w}$ of length m that takes real values and determines the preferences over the task factors. Using Example 1, $\vec{w}$ could be represented as a set of weights for the task factors, such as, $\{duration, payment\}$.

Worker Model. Central to our framework is a model that consumes task factors and given a worker's history, infers her preference vector to estimate task completion time. It is easy to notice that task completion time is continuous in nature.

Explicit Questions. An explicit question q is asked to elicit w's preference on a particular task factor, assuming there is an one to one correspondence between the questions and task factors (thus, every task factor is a potential question and total possible questions m). A set of k questions is asked to obtain a preferred order among a set of k task factors (where k is part of the input). As an example, one

may ask to *" Rank task duration, annotation tasks, ranking tasks, sentiment analysis tasks, payment"*. A worker may provide a full order among these five factors as her preference, or may provide partial order of preference. As an example of the former, she may rank *payment*, then *duration*, then *ranking tasks*, followed by *sentiment analysis*, and finally *annotation tasks*. On the contrary, her preference is partial, when the worker prefers, *payment* over *duration*, but does not explicitly say anything about the rest.

Problem Definition: Worker Model Given the task factor matrix $\mathcal{T}$ of a set of n tasks, where each task t is described by m factors and associated with a continuous variable y_t denoting the time the worker spends on t, we are interested in estimating the worker preference vector $\vec{w}$. The *Worker Model* $\mathcal{F}$ is a linear aggregate function over $\mathcal{T}$ and $\vec{w}$, denoted as $\mathcal{F} = \vec{w}^T \cdot \mathcal{T}$.

Optimization goal. Our objective is to estimate $\vec{w}$ in $\mathcal{F}$, such that it minimizes the *reconstruction error* [134], i.e.,

$$\mathcal{E} = ||\vec{w}^T \cdot \mathcal{T} - \vec{Y}||_2^2 \tag{2.1}$$

Once the model is built, it can estimate the completion time of a future task by the worker. Using Example 1, $\mathcal{F}$ can estimate the completion time of any of the six tasks or other future tasks, by consuming $\mathcal{T}$.

Initialization. How to initialize the *Worker Model* is a challenge. Initially when a brand new worker w joins the platform, as no past history of w is available, she is treated akin to a "cold worker". Initially we have no information of task completion, hence $\vec{w}$ can't be estimated accurately. The objective of the model initialization is to select a subset of tasks $\mathcal{B}$ ($|\mathcal{B}| = b$, given as a budget) to be completed by w and build $\mathcal{F}$ using that.

One obvious choice is to randomly select b samples (tasks) to initialize the model. However, we argue there does exist further merit in careful selection of initial set of b tasks (training inputs), as a careful selection of the training examples (inputs), will generally need far fewer examples in comparison to selecting them at random from some underlying distribution.

Our proposed formalism is inspired from *active learning* in Machine Learning [34] that is iterative in nature and is optimized to select one more input (i.e., a task) at a time that maximizes the accuracy of the underlying model. For us, from the available candidate pool of tasks that are not yet undertaken by the worker, this translates to selecting one task t (input) at a time, the worker w undertakes it, and we record its completion time y_t.

Optimization goal. During the model initialization phase, in a single iteration, our objective is to select that task t that will give rise to a worker preference vector $\widehat{w}$, which is a good estimation of true worker preference $\vec{w}$ by minimizing the mean squared error of the maximum likelihood estimates between $\widehat{w}$ and $\vec{w}$.

$$E(||\widehat{w} - \vec{w}||)^2 \tag{2.2}$$

Given Example 1, if $b = 3$ and we have already selected two tasks $(t1, t2)$, the objective would be to select the next best task that minimizes the mean squared error between $\hat{w}$ and $\vec{w}$.

Problem Definition : Question Selector This module selects the best set of k questions for a worker w. The objective is to select those task factors that are responsible for the model's inaccuracy, i.e., removing them would improve the reconstruction error of $\mathcal{F}$ the most.

Optimization goal. Let $\mathcal{E}$ denote the current reconstruction error of $\mathcal{F}$ and $\hat{\mathcal{E}}$ denote it when k task factors are removed. Given $\mathcal{Q}$, the k questions are selected such that the model reconstruction error improves the most, i.e., $argmax_{\{\mathcal{Q}^k \in \mathcal{Q}: |\mathcal{Q}^k|=k\}}(\mathcal{E} - \hat{\mathcal{E}}_{\mathcal{Q}-\mathcal{Q}^k})$.

Using Example 1, if $k = 2$, this will select any two of the five task factors in the task factor matrix.

Problem Definition : Preference Aggregator The preferences provided by a worker for task factors, could be expressed as a set of constraints of the form, $i > j$, $j > l$. Worker preference could take one of the three following forms:

(1) **Full Order:** The worker can provide a full order over the k selected factors. The full ranking will be in the form of $i \succ j \succ k \succ l$, if i, j, k, l are the task factors (questions). This preference could be expressed as a set of $k - 1$ pairwise linear constraints of the form, $i > j$, $j > k$, and etc.

(2) **Partial Order:** The worker can provide a partial order instead, especially when she can not provide full order. Given the set of k factors, a partial order takes the form of $i \succ j$, but no preference is elicited between for k, l.

(3) **No Preference:** The worker does not provide any preference.

Optimization goal. Worker's preferences are taken as hard constraints. Given her preference, the objective is to relearn $\mathcal{F}$ that satisfies the preferences such that its reconstruction error is minimized. The objective therefore is to minimize $\mathcal{E}$ such that the constraints are satisfied.

Using Example 1, if worker w explicitly states that she prefers `annotation` tasks to `ranking` tasks, this preference is translated into constraints expressed on the worker preference vector. Those are then used by the preference aggregator to update $\mathcal{F}$.

2.3 Algorithms

We present solutions that are efficient and come with guarantees.

2.3.1 Worker Model

As formalized in Section 2.2.2, the *Worker Model* $\mathcal{F}$ is a linear combination of task factors and worker preference over those factors. We design algorithms for the optimization problem that is intrinsic to the model, and then a solution for model initialization.

Efficient Algorithm. We first derive an alternative form of our optimization function and then show that we can use simple matrix algebraic techniques to solve the problem.

Equation (2.1) could be rewritten as

$$min_{\vec{w}}(\vec{w}^T \cdot \mathcal{T})^T(\vec{w}^T \cdot \mathcal{T}) = min_{\vec{w}}(\vec{w}^T\mathcal{T}^T\mathcal{T}\vec{w} - 2\vec{w}^T\mathcal{T}^T\vec{Y} + \vec{Y}^T\vec{Y}) \tag{2.3}$$

To minimize Equation (2.3), we compute the gradient and set it to zero. i.e.,

$$\nabla(\vec{w}) = 2\mathcal{T}^T\mathcal{T}\vec{w} - 2\mathcal{T}^T\vec{Y} = 0 \tag{2.4}$$

This allows us to solve $\vec{w}$ by computing the matrix inverse.

$$\mathcal{T}^T\mathcal{T}\vec{w} = \mathcal{T}^T\vec{Y} \tag{2.5}$$

Which finally gives,

$$\vec{w} = (\mathcal{T}^T\mathcal{T})^{-1}\mathcal{T}^T\vec{Y} \tag{2.6}$$

$(\mathcal{T}^T\mathcal{T})^{-1}\mathcal{T}^T$ in Equation (2.6) is known as the Moore-Penrose pseudo-inverse matrix of $\mathcal{T}$. This alternative representation is valid as long as the task factor matrix $\mathcal{T}$ is invertible (or could be inverted by adding an additional term, refer to [21]).

With this alternative representation in Equation (2.6), computing the best estimation of $\vec{w}$ is done by matrix inversion and multiplication techniques.

Running Time. The overall running time of the algorithm is dictated by matrix multiplication (multiplying $(\mathcal{T}^T\mathcal{T})$, Matrix inversion (inverting $\mathcal{T}^T\mathcal{T}^{-1}$), followed by matrix multiplication (to obtain $(\mathcal{T}^T\mathcal{T})^{-1}\mathcal{T}^T$), and a final matrix multiplication (to obtain $(\mathcal{T}^T\mathcal{T})^{-1}\mathcal{T}^T$). The asymptotic complexity is $\mathcal{O}(m^2n + m^3)$.

Initializing the Worker Model One major challenge to develop the "supervised" `Worker Model` is how to handle "cold workers" - brand new workers. If the platform does not have any information about such workers, we initialize the `Worker Model` $\mathcal{F}$ by judiciously selecting one task at a time and repeating this process b times.

Efficient Algorithm. As described in Section 2.2.2, we present an online problem formulation, which selects one task t at a time, the worker undertakes the task, we record the completion time and update $\mathcal{F}$. We repeat this process for b iterations. As expressed in Equation (2.2), our objective is to estimate $\widehat{w}$ which is a good estimator of $\vec{w}$ (the true preference vector of the worker). The model initialization algorithm has to select a task whose completion time is not yet known. Therefore, the challenge is, how to select a task without knowing its completion time, so that we meet the optimization goal.

We present an alternative representation of Equation (2.2) that lies at the heart of our algorithm. Interestingly, this alternative representation does not involve task completion time. Hence, just by looking at the task factor matrix, we can select the next best task that optimizes Equation (2.2). At a given run of this algorithm, let us assume then that we already have selected a few tasks that gives rise to $\mathcal{T}'$ task

factor matrix. Equation (2.2) can be rewritten as (we omit the details for brevity and refer to [134, 99] for details)

$$E(||\widehat{w} - \vec{w}||)^2|\mathcal{T}' = Trace[(\mathcal{T}'^T\mathcal{T}')]^{-1} \tag{2.7}$$

Therefore, minimizing the error is same as minimizing $Trace[(\mathcal{T}'^T\mathcal{T}')]^{-1}$, where $Trace[.]$ is the matrix trace, the sum of its diagonal components.

Now, we are ready to describe the algorithm. Consider Equation (2.7) and let us assume $A = (\mathcal{T}'^T\mathcal{T}')^{-1}$. (assuming it is already invertible). We are now trying to select another input task t that adds one additional row $[t]$ to $\mathcal{T}'$. Therefore, now $\mathcal{T}'$ becomes

$$\begin{bmatrix} \mathcal{T}' & t^T \end{bmatrix} \quad \begin{bmatrix} \mathcal{T}' \\ t^T \end{bmatrix} = \mathcal{T}'^T\mathcal{T}' + \begin{bmatrix} t \end{bmatrix} \begin{bmatrix} t \end{bmatrix} \tag{2.8}$$

This is equal to $AA^{-1} + tt^T$. Therefore, we would like to find a t that minimizes

$$Trace[(A^{-1} + tt^T)^{-1}] \tag{2.9}$$

This matrix inverse could actually be carried out in closed form and it becomes

$$Trace[(A^{-1} + tt^T)^{-1}] = Trace[A] - \frac{t^TAAt}{1 + t^TAt} \tag{2.10}$$

Now that $Trace[A] = Trace[(\mathcal{T}'^T\mathcal{T}')]^{-1}$, to minimize the mean squared error by adding a task, we choose that task that maximizes

$$\frac{t^T AAt}{1 + t^T At} \tag{2.11}$$

Therefore, at a given iteration, the algorithm selects a task that maximizes Equation (2.11). To select the set $\mathcal{B}$, this process is repeated b times.

Running Time. This algorithm takes $\mathcal{O}(m^2n + m^3)$ time to compute $\frac{t^T AAt}{1+t^T At}$ of a single candidate task. In each iteration, it has to consider the entire pool of tasks to decide the best candidate. If the number of tasks is upper-bounded by n, one iteration of this algorithm takes $\mathcal{O}(m^2n^2 + m^3n)$

Running Example: Suppose we have selected the first two tasks in Table 1. We describe how to select the third task to complete the set of $b = 3$ tasks for model initialization. Suppose the following two tasks have been selected in the previous iteration,

$$\mathcal{T} = \begin{array}{c} \begin{matrix} \textit{annotation} & \textit{ranking} & \textit{sentiment} & \textit{payoff} & \textit{duration} \end{matrix} \\ \left[\begin{matrix} 1 & 0 & 0 & 20 & 35 \\ 0 & 1 & 0 & 5 & 10 \end{matrix}\right] \end{array}$$

Assume that $A = (\mathcal{T}^T\mathcal{T})^{-1}$. After calculating the value of A, we start by examining the four remaining tasks one at a time and we will calculate the value of $\frac{t^T AAt}{1+t^T At}$ for each of them. Out of the four remaining tasks, one can see that $\frac{t^T AAt}{1+t^T At}$ is

maximized ($\frac{t^T AAt}{1+t^T At} = 0.512$) for the task $t4$,

$$t4 = \begin{bmatrix} \textit{annotation} & \textit{ranking} & \textit{sentiment} & \textit{payoff} & \textit{duration} \\ 0 & 1 & 0 & 5 & 40 \end{bmatrix}$$

Therefore, this task is selected and given to the worker. Once she returns it, task completion time is recorded.

2.3.2 Question Selector

The **Question Selector** intends to select the k-task factors (i.e., questions) whose removal maximizes the improvement of the *Worker Model* $\mathcal{F}$. The idea is to present those factors to the worker and seek her explicit preference. A careful review of the objective function (refer to Section 2.2.2) shows that since $\mathcal{E}$ is a constant at a given point - thus, maximizing $(\mathcal{E} - \hat{\mathcal{E}}_{\mathcal{Q}-\mathcal{Q}^k}) : \{\mathcal{Q}^k \in \mathcal{Q} : |\mathcal{Q}^k| = k\}$ is same as minimizing the reconstruction error of $\hat{\mathcal{E}}_{\mathcal{Q}-\mathcal{Q}^k}$, i.e., retaining the best $m-k$ factors (thus eliminating the worst k factors) that has the smallest reconstruction error of $\mathcal{F}$. The problem thus becomes selecting the best $m-k$ factors that have the smallest reconstruction error. The remaining k factors would therefore be chosen as the explicit questions for preference elicitation.

Theorem 1. *Optimally selecting k questions for explicit worker preference is NP-hard.*

Proof. (Sketch): When a linear model such as the one in Equation (2.1) is assumed, the problem of identifying and removing the k worst factors, i.e., retaining the best $m-k$ factors, is akin to selecting a subset of $m-k$ columns from the task factor matrix $\mathcal{T}$ such that the pseudo-inverse of this sub-matrix has the smallest norm.

Under the ℓ_2 norm, using the rigorous NP-hardness proof described in [21], our proof follows. Given an instance of that problem [21], we set k (the k worst factors to

remove) as the difference between the total number of columns and k' (k'= the best set of k' columns giving rise to the submatrix whose pseudo-inverse has the smallest norm). The rest of the proof is trivial and omitted for brevity. □

Efficient Algorithm Under the linear model such as the one described in Equation (2.1) and its equivalent representation using a pseudo-inverse matrix, the objective of identifying the set $\mathcal{Q}^k$ of k selected questions (thereby identifying $m-k$ best factors) out of a set $\mathcal{Q}$ of m questions (a task factor is a question) is equivalent to retaining the task factor submatrix with $m-k$ columns that is of the following form [99]:

$$\underset{\mathcal{Q}^k \subset \mathcal{Q}, |\mathcal{Q}^k|=k}{\operatorname{argmin}} \; Trace(\mathcal{T}_{Q\setminus Q^k}^T \mathcal{T}_{Q\setminus Q^k})^{-1} \tag{2.12}$$

We now describe a greedy algorithm `K-ExFactor` to identify k worst task factors (thus retaining $m-k$ best factors). Our algorithm makes use of Equation (2.12) and has a provable approximation guarantee. It works in a backward greedy manner and eliminates the factors iteratively. It works in k iterations, and in the i-th iteration, from the not yet selected set of factors, it selects a question q_j and eliminates it which marginally minimizes Trace$(\mathcal{T}_{Q\setminus q_j}^T \mathcal{T}_{Q\setminus q_j})^{-1}$. Once the k^{th} iteration completes the eliminated k questions are the selected k-factors for explicit elicitation. The pseudo code of the algorithm is presented in Algorithm 1.

Running Time. The algorithm runs in k iterations. Line 4 in Algorithm 1 requires a $\mathcal{O}(m^2 n)$ time for matrix multiplication and inversion for the question under consideration. Therefore, the overall complexity is $\mathcal{O}(km^2n^2)$. Notice that most of the complexity is actually in the process of recomputing the model error and the actual question selection is rather efficient.

Theorem 2. *Algorithm* ***k-ExFactor*** *has an approximation factor of* $\frac{m}{m-k}$.

Algorithm 1 Algorithm `k-ExFactor`: Greedy Question Selector

Require: Task factor matrix $\mathcal{T}$, set of questions $\mathcal{Q}$

1: $\mathcal{T}_Q \leftarrow \mathcal{T}$
2: $\mathcal{Q}^s \leftarrow \mathcal{Q}$
3: **for** $j \leftarrow 1$ to k **do**
4: $\quad q_j \leftarrow argmin_{q \in \mathcal{Q}}$ Trace$(\mathcal{T}_{Q \setminus q}^T \mathcal{T}_{Q \setminus q})^{-1}$
5: $\quad \mathcal{T}_Q \leftarrow \mathcal{T}_{Q \setminus j}$
6: $\quad \mathcal{Q}^s \leftarrow \mathcal{Q}^s \setminus q_j$
7: **end for**
8: Return $\mathcal{Q} - \mathcal{Q}^s$

Proof. The proof adapts from an existing result [18] that uses backward greedy algorithm for *subset selection* for matrices and retains a given smaller number of columns such that the pseudo-inverse of the smaller sub-matrix has the smallest norm possible. These results adapt, as this is akin to removing k worst task factors and retaining the best $m - k$ factors. It is also shown in recent work [21] that the objective function is not submodular, nor is it supermodular or monotone. □

Running Example: Using Example 1, if $k = 3$,
{sentiment, Payoff, Duration} are the three task factors for which worker feedback is solicited.

2.3.3 Preference Aggregator

We can now describe how to aggregate worker responses and incorporate her provided preferences into the model $\mathcal{F}$. Recall Section 2.2 and note that the worker provides either a full order among the selected questions (task factors), a partial order, or possibly no answer. For the last scenario, since the worker does not provide any feedback, we simply update $\mathcal{F}$ implicitly. This is done by updating the model without any constraints. However, for both full and partial orders, worker preference adds a set of linear constraints in the optimization function in $\mathcal{F}$.

Efficient Algorithm Our solution treats partial and full order in a similar fashion. In both cases, they add linear constraints to the objective function. With the linear constraints added to our objective function in Equation (2.1), updating the *Worker Model* under preference aggregation problem becomes a constrained least squares problem.

Specifically, our problem corresponds to a box-constrained least squares one as the solution vector must fall between known lower and upper bounds. The solution to this problem can be categorized into active-set or interior-point [87]. The active-set based methods construct a feasible region, compute the corresponding active-set, and use the variables in the active constraints to form an alternate formulation of a least squares optimization with equality constraints [120]. We use the interior-point method that is more scalable and encodes the convex set (of solutions) as a barrier function. It uses primal Newton Barrier method to ensure the KKT equality conditions to optimize the objective function [87].

Running Time. Our proposed primal Newton Barrier interior-point is iterative and the exact complexity depends on the barrier parameter and the number of iterations, but the algorithm is shown to be polynomial [120].

Running Example: Using Example 1 again, if the worker says that she prefers `Duration > Sentiment > Payoff`, then the new weights that the preference aggregator estimates for $\mathcal{F}$ are, tagging=0.1, ranking= 0.1, sentiment=0.12, payoff=0.11, duration=0.97. Notice that the order of the task factors provided by the worker is satisfied in the updated model.

2.4 Experimental Evaluations

We describe our experimental setup, steps, and findings in this section. All algorithms are implemented in Python 3.5.1 using Intel Core i7 4GHz CPU and 16GB of memory and Linux operating system. All the numbers are presented as an average of 10 runs.

2.4.1 Dataset Description

We use 165,168 micro-tasks from CrowdFlower. A task belongs to one of the 22 different categories, such as, tweet classification, searching information on the web, audio transcription, image tagging, sentiment analysis, entity resolution, etc. Each task type is assigned a set of keywords that best describe its content and a payment, ranging between \$0.01 and \$0.12. These are *micro-tasks* that take less than a minute to complete.

Initially, we group a subset of micro-tasks into 240 Human Intelligence Tasks (HITs) and publish them on Amazon Mechanical Turk. Each HIT contains 20 tasks and has a duration of 30 minutes. A worker who accepts a HIT is redirected to our platform to complete the tasks. A worker may complete several HITs in a work session and gets paid for every completed *micro-task*.

Task Factors. The task types along with other factors, such as, payment and duration, form the task factors. Our original data has 41 task factors that are continuous, categorical or binary. By involving domain experts, we binarized these them to obtain a total of 100 factors that uniquely characterize the tasks.

Worker and Keywords. Each hired worker has to previously complete at least 100 HITs that are approved, and to have an approval rate above 80%. Overall, 58 different workers complete tasks. When a worker is hired for the first time, she is asked to select a set of keywords from a given list of keywords that capture her preferences. We create a unique *Worker Model* for all the 58 different workers that participate in our experiments.

When a worker first joins, we ask her to choose the top-5 keywords of her preference. We use these chosen keywords for a case study, shown in Section 2.4.5.

Ground Truth. For each micro-task, we record the ground-truth, which is the amount of time the worker spent on it in seconds. This is encoded in the task completion time vector for the corresponding *Worker Model*.

2.4.2 Implemented Algorithms

Worker Model The linear model in Section 2.3.1 is implemented with a regularization parameter α. When implementing statistical models, this is a standard practice to avoid overfitting. The overall objective function thus becomes,

$$\min_{\vec{w} \in \mathbb{R}^m} \left\| y - \vec{w}^T \cdot \mathcal{T} \right\|_2 + \alpha \left\| \vec{w} \right\|_2^2 \tag{2.13}$$

The best value of α is chosen by generalized cross validation [87].

Model Initialization. We set a fixed budget b which we use to initialize the *Worker Model* iteratively (Section 2.2.2) and implement the following algorithms:

1. Random Initialization. `RandomInit` selects a randomly selected task iteratively, presents it to the worker and records the task completion time. The algorithm stops when the budget b is exhausted.

2. Active Initialization. `ActiveInit` implements our algorithm given in Section 2.3.1.

3. **Uniform Initialization.** `UniformInit` initializes the model by assigning uniform weights to the worker preference vector.

Explicit Feedback This has two important components - one is the **Question Selector** that selects the task factors for explicit preference elicitation, the other is **Preference Aggregator** that updates the *Worker Model* using elicited preferences.

Question Selector. We have implemented two algorithms to find the best set of questions to ask as described in Section 2.3.2.

1. Optimization-Aware Question Selector. `k-ExFactor` is our proposed algorithm described in Section 2.3.2.

2. k-random Question Selector. `k-Random` is a simple baseline that randomly selects k-task factors for preference elicitation.

Preference Aggregator: This is our implemented solution for preference aggregation, as described in Section 2.3.3.

Implicit Feedback We also implement implicit feedback computation to be compared against explicit feedback.

Algorithm `Implicit-1` is an adaption of recent work [96] that investigates how to implicitly capture worker motivation and use that for task assignment. While we do not necessarily focus on motivation as a factor in this work, we adapt the algorithm in [96] to estimate and update the worker preference vector over time. We do that by taking the average over the worker preference vector of the worker model obtained in different iterations. Since our focus is not on task assignment, once we estimate the worker preference vector using `Implicit-1`, we use that in conjunction with our *Worker Model* to predict a task completion time.

Algorithm `Implicit-2` is a further simplification. It relearns the *Worker Model* at the end of every iteration as the worker completes tasks and does not factor in the preference of the worker.

2.4.3 Invocation of `ExPref`

For quality experiments, `ExPref` is invoked iteratively and in an online fashion: in the beginning, we filter out the tasks and task completion history by worker id since the framework is personalized per worker. On average, a worker undertakes 200 tasks. We randomly divide the tasks into three subsets. We use 50% of each worker's data as a holdout over which error is computed. Half of the remaining tasks are used for training/developing the *Worker Model* and the rest as the pool of available tasks.

To conduct experiments only related to *Worker Model initialization* (specifically for "cold" workers), the training set is empty in the beginning and all tasks are in the available pool. We use the budget b to find a subset $\mathcal{B}$ of tasks based on our proposed solution in Equation (2.11).

After *Worker Model* is trained, in every iteration, we select a set x of 20 tasks (unless otherwise stated), randomly from the pool of available tasks and present them to the worker. After recording task completion time, we add those x tasks back to the training set. Next, we invoke the **Question Selector** that seeks explicit feedback from that worker. Upon receiving worker feedback, the *Worker Model* is updated using the **Preference Aggregator** and the new training set. We calculate the error over the holdout set after this. All these steps construe a single iteration of the `ExPref`.

For scalability experiments, we are only interested to measure the running time of the algorithms in `ExPref`. Thus, the experimental set up is rather simple there and we use the entire dataset.

Error. Unless otherwise specified, we calculate the quality of the *Worker Model* as the Mean Square Error (MSE) over validation set, defined as,

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\vec{w}^T\mathcal{T} - \vec{Y}\right)^2$$

Additionally, we present R^2 (co-efficient of determination) which indicates the proportion of the variance in the task completion time that is predictable from the task factors. R^2 takes values between $[-\infty, 1]$, where higher is better.

$$R^2 = 1 - \frac{\sum_t\left(y_t - \vec{w}^T\vec{t}\right)^2}{\sum_t\left(y_t - \bar{Y}\right)^2}$$

where, $\bar{Y}$ is the average task completion time.

Iteration. We define an iteration as the completion of a HIT (Human Intelligence Task) of 20 tasks, after which we compute the MSE and R^2 of the *Worker Model*.

Preference Elicitation. As described in Section 2.2.2, workers can provide their preference either as a full order, partial order, or they may not even provide any preference.

Preference History. For every worker, we also maintain her elicited preferences in all previous iterations (full history), preferences only in the current iteration (no history), or preferences in the last few iterations (partial history). *Worker Model* is updated accordingly.

2.4.4 Summary of Results

Our proposed explicit preference elicitation framework outperforms (with statistical significance) existing implicit ones after fewer iterations.

- We compare our approach `ExPref` with two other baseline algorithms `Implicit-1` [96] and `Implicit-2` (Section 2.4.5). We present MSE and R^2 with statistical significance results (standard error) and show that `ExPref` convincingly and significantly outperforms the other baselines under varying parameters : 1) Number of iterations (Figure 2.2), 2) Number of task factors (Figure 2.3), and 3) Number of tasks worker completes in each iteration (Figure 2.4).
- We compare the effect of different parameters of `ExPref` with appropriate baselines (Section 2.4.5). We show with a small number of questions k (Figure 2.5), `k-ExFactor` outperforms the baselines. Our results demonstrate that `ActiveInit` is an effective model initialization algorithm (Figure 2.8).
- Our results also indicate that task completion time is highly correlated to task outcome/quality of the completed task. This further justifies our investigation - indeed, deeper analysis of task completion time improves the quality of the crowdsourced tasks. Our case study results show that `ExPref` is capable to truly capture worker preference.

`ExPref` is scalable.

- We compare `ExPref` with other baselines under varying parameters: 1) Number of tasks, 2) Number of task factors, and 3) Number of questions (k). Unsurprisingly, `ExPref` is slower but it still scales very well.

- We compare our model initialization method `ActiveInit` by varying the budget b. `ActiveInit` is slower than the two other baselines. Despite that, it scales reasonably well. These results demonstrate the effectiveness of eliciting explicit preferences making `ExPref` usable in practice.

2.4.5 Quality Experiments

The objective of these experiments is to capture the effectiveness of our explicit feedback elicitation framework and compare it with appropriate baselines. Specifically, we are interested in answering the following questions :

1. How `ExPref` performs compared to implicit ones (Section 2.4.5).
2. Effect of different parameters in `ExPref` (Section 2.4.5).
3. Relationship between task completion time and task outcome (Section 2.4.5).
4. A case study on worker's explicit feedback (Section 2.4.5).
5. A case study on worker preferences (Section 2.4.5).

Parameter Setting. For a given worker, there are four parameters to vary: 1) Number of tasks in each iteration (x), 2) Number of task factors (m), 3) Number of questions asked (k), and 4) Number of iterations. For initializing the model, we additionally vary the budget b. Table 2.2 presents the default values alongside the experimental settings for each parameter. To select a different number of task factors, the best m features are retained by finding factors that are highly correlated to the target and discarding the rest. By default, we always maintain the full history of worker's preference while updating the *Worker Model* under varying iterations.

`ExPref` vs. Baselines We compare two explicit solutions with two implicit ones. We vary # iterations, # task factors, and x (# tasks assigned to a worker after which the framework is invoked).

Varying the number of iterations.

Figure 2.2 presents the error of the four *Worker Model*s in the course of ten iterations.

Table 2.2 Parameter Settings

Parameters	Range	Default
# tasks in each iteration (x)	$5, 10, 15, 20, 25$	20
# task factors (m)	$5, 10, 25, 50, 80$	80
# questions to ask (k)	$3, 5, 7, 9$	3
# iterations	$1, 2, 3, 4, 5, 6, 7, 8, 9, 10$	7
initialization budget (b)	$5, 15, 30, 50, 75$	50

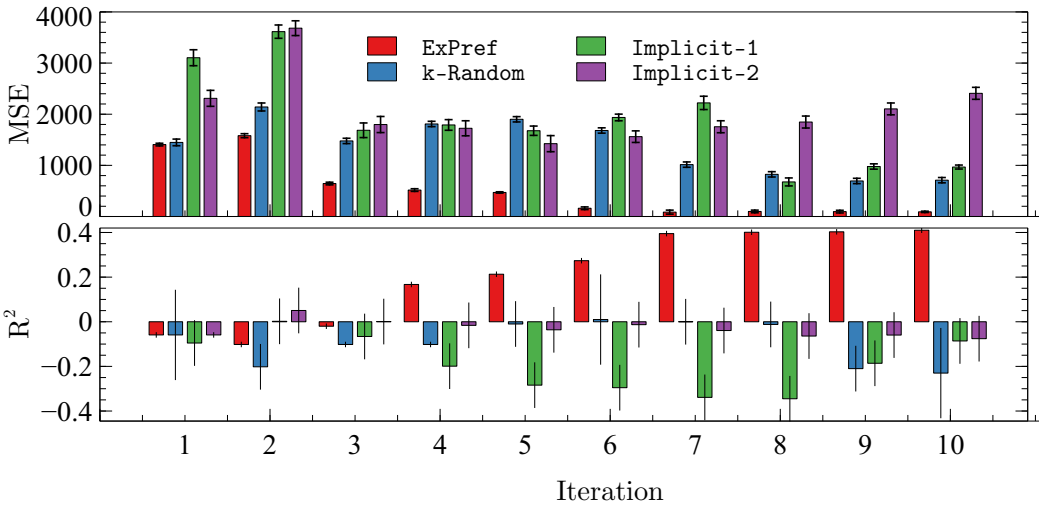


Figure 2.2 Comparison between the error of the four models after ten iterations.

We notice that after the seventh iteration, all the four models become stable and their corresponding errors vary only by a very small margin. This means that `ExPref` can achieve significantly better results than the other three baselines after few iterations. Additionally, we observed that `ExPref` achieve stability in far fewer iterations than the baselines. This confirms the fact that worker's preference will be helpful to the model.

Varying the number of task factors.

In Figure 2.3, we observed that by adding more task factors, all the models perform better but `ExPref` performs significantly better. We also see that the number of task factors extracted from the tasks on our *Worker Model* is minimal compared to the other three baselines.

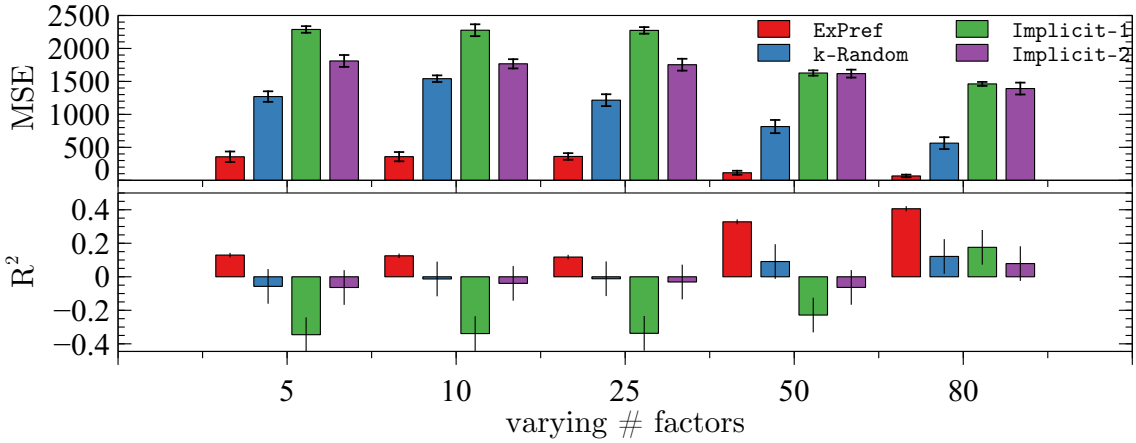


Figure 2.3 Error varying number of task factors.

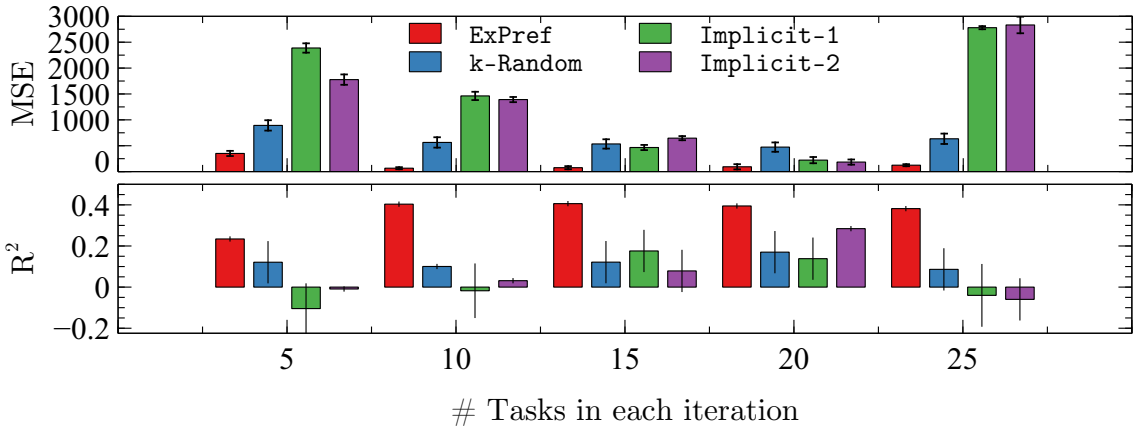


Figure 2.4 Error varying number of tasks in each iteration.

Varying the number of tasks in each iteration.

Figure 2.4 presents the results for varying the number of tasks a worker receives in each iteration (denoted by x). `ExPref` outperforms the other three baseline by a large margin in terms of achieving smaller error. Notice that for $x = 20$ all the algorithms perform well but `ExPref` is better that the other three. For this reason, we set the default value of x to be 20.

Varying the number of questions.

Since the results of `Implicit-1` and `Implicit-2` do not change with k, we present the results in the next section (Section 2.4.5).

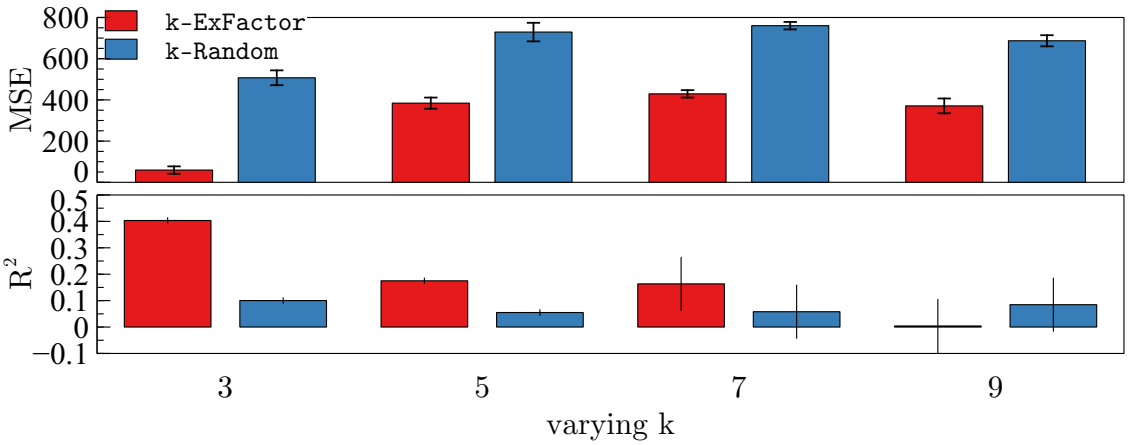


Figure 2.5 Error varying number of questions.

Effect of Different Parameters We do a comparative study on the effect of different parameters, namely, how we track worker's history (recent history vs full history vs partial history), the number of questions we ask a worker (k), and the budget we use for model initialization (b).

Number of explicit questions to ask.

As the number of questions increases, the quality of the *Worker Model* decreases (Figure 2.5). This happens for two reasons. First, when the number of questions is higher than five, worker responses are inconsistent. Second, since we keep the full history of responses for the worker, the number of constraints imposed in our optimization problem grows significantly which in turn affects performance.

Similarly, as we ask more questions from workers, most of the answers provided are in the form of partial ranking rather than full ranking. It's likely that the workers simply picked the most important factors and ignored the rest.

Full, partial, no history.

We present a comparative study between fully or partially capturing the history of the worker's preferences versus no history, i.e., using the most recent preferences only. To better understand the difference between the three, as an example, consider we ask four explicit questions to a worker in each iteration, after the third iteration, her full history size is 12, whereas, her most recent history size is four; i.e., the

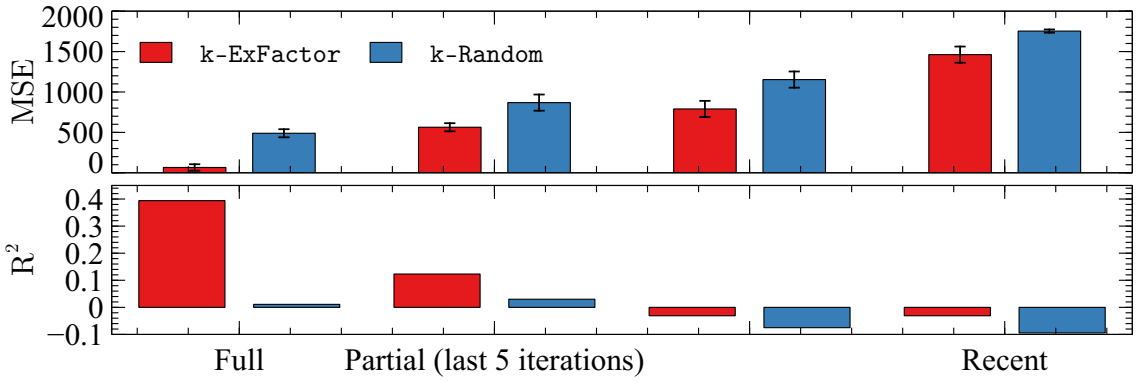


Figure 2.6 Recent history vs partial history vs full history.

recent history represents the number of feedback in the current iteration. Assuming an expiration time of two, the partial history is defined as the preference of the worker based on the last two iterations which is of size 8. Figure 2.6 presents the results of `k-ExFactor` against `k-Random` in the last iteration for four scenarios. We omit the results for the other two implicit models since their results is not affected by the worker's history. Clearly, maintaining a full history for the worker performs significantly better. Intuitively, this shows that the model can understand the worker better when all the information about her is maintained. This means that the *Worker Model* can handle variations in worker's behavior significantly better compared to using recent information only.

Full, partial, no order of preference.

Figure 2.7 presents the results for different types of feedback a worker can provide. We set the number of questions that we ask in each iteration to $k = 4$. Notice that when the worker does not provide any answer, the results are very similar to `Implicit-2`. This means that if the worker does not provide any answer, we fall back to the implicit model. Similarly, note that the performance of the *Worker Model* is not affected by the type of answer a worker provides. This means that as long as the worker provides some feedback, albeit partially, our **Preference Aggregator** can help the *Worker Model* better estimate the task completion time.

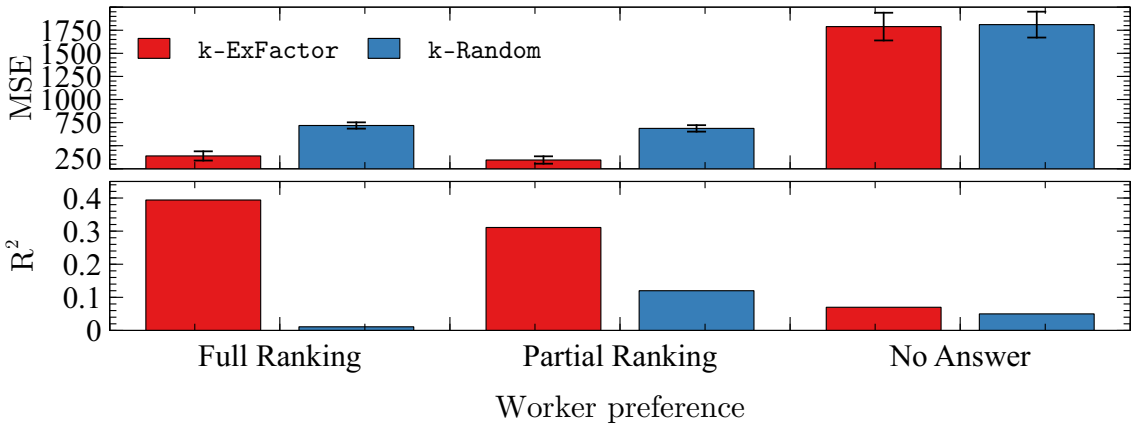


Figure 2.7 Error varying worker preference type.

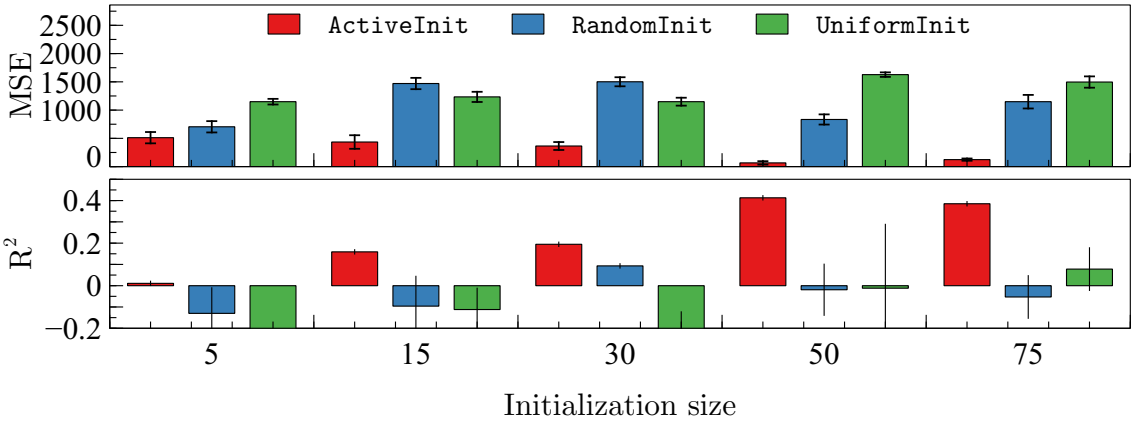


Figure 2.8 Evaluation of model initialization algorithms.

Model Initialization Figure 2.8 presents the difference in the reconstruction error between the three initialization methods. `ActiveInit` performs better overall and for $b = 50$ it has the lowest reconstruction error between the three methods. Notice that as the size of initialization set grows, the reconstruction error drops. This is attributed to the fact that the *Worker Model* needs a reasonable amount of training data to be able to predict the task completion duration. Another important phenomena is that increasing b beyond 50 results in an increase in the reconstruction error. This is because the *Worker Model* will overfit the training data and lose its predictive power.

Task Completion Time vs. Task Outcome We notice that completed tasks that have correct answers take more time on average to complete than the tasks

that are not. Using *Chi-squared test*, we observe a high positive correlation between task completion time and its outcome/quality (with $\chi^2 = 3796.99$ and $p - value = 0.00001$). This in fact is one of the motivations behind our study, as the correct estimation of task completion time will help us better understand task outcome.

Worker's Feedback We profile all 58 workers over the course of their participation and notice that they always provide some feedback (partial/full). We notice that if the number of questions is less than four, the workers are more likely to provide full feedback. As we increase the number of questions, worker's tend to give partial feedback.

A Case Study We profile three workers randomly from our database and analyze their models in conjunction with the keywords they have initially chosen. Table 2.3 presents the five keywords chosen by the workers and the top-2 worker preferences. It is easy to notice that they are highly correlated, which shows that our proposed model successfully captures worker preference.

Table 2.3 Worker Keywords and Preference Learned By *Worker Model*

Worker no	Worker Keywords	Top-2 preference
1	dress,google street view, airlines, classification, scene	dress, scene
2	business, body parts, google street view, health, classification	classification, google street view
3	image, south Asia, disease, animals, text	image, text

2.4.6 Scalability Experiments

We are interested in answering the following questions :

1. How `ExPref` scales compared to implicit preference computation (Section 2.4.6).

2. Effect of different parameters in `ExPref` (Section 2.4.6).
3. Time Profile of each component of `ExPref` (Section 2.4.6).

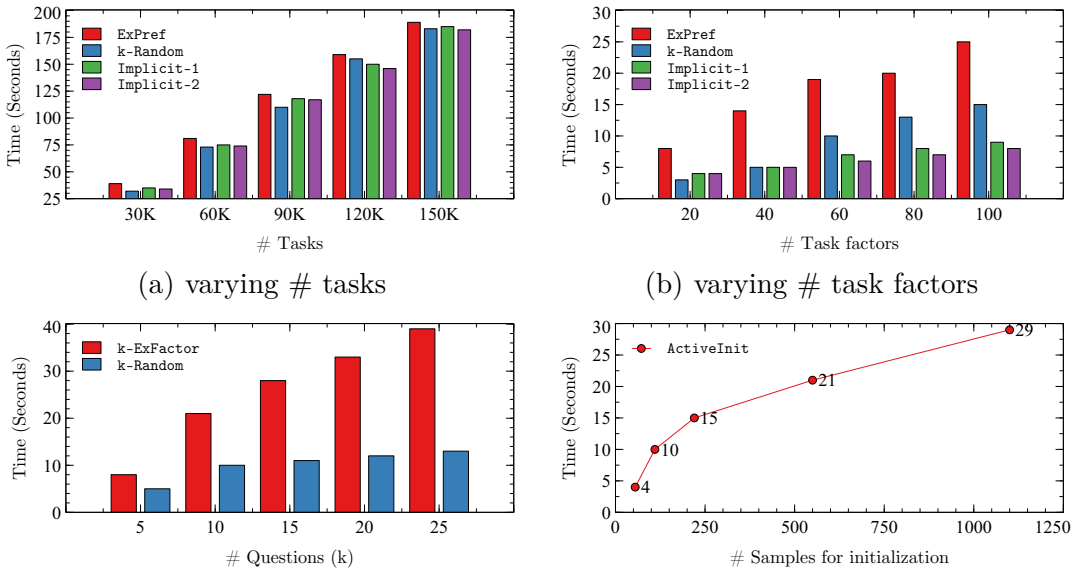
Unless otherwise stated, we report running times in seconds.

Parameter Setting. Our dataset contains 165, 168 tasks and 80 task factors obtained from 58 workers. In these experiments, we vary the following parameters: # tasks, # task factors, k, and the initialization budget b. Unless otherwise stated, all the numbers present the average running time of a single iteration over all the 58 workers. The default values are set as # tasks = 30, 000, # task factors = 50, $k = 3$, and $b = 20$. By default, we consider full order of worker preference since that adds more constraints to the problem. For the *Worker Model* initialization comparison, only the appropriate three methods are compared.

Efficiency of `ExPref` vs. Baselines Figure 2.9(a) presents the running times of the four algorithms with varying number of tasks. Of course, our proposed solution `k-ExFactor` makes a lot more computation to ensure optimization and hence has the highest running time. However, it is easy to notice that with an increasing number of tasks, it scales well and the running time is comparable to the other competing algorithms. A similar observation holds when we vary the number of task factors, as shown in Figure 2.9(b). `k-ExFactor` scales well and never takes more than 80 seconds.

Effect of Parameters on Efficiency Figure 2.9(c) represents the running times by varying k, the number of task factors chosen for preference elicitation. Here only `k-ExFactor` is compared with `k-Random`, as the other two algorithms do not rely on explicit preference elicitation. Unsurprisingly, `k-Random` is faster, but our proposed solution `k-ExFactor` scales well and has a comparable running time. Finally, in Figure 2.9(d), we vary the initialization sample size and present the running time of `ActiveInit`. Our initialization model scales well and does not take much time

as the size of initialization set grows. The other two baselines do not perform any computation and take negligible time to terminate.



(a) varying # tasks

(b) varying # task factors

(c) varying k

(d) varying b

Figure 2.9 Scalability study

Profiling `ExPref` for Efficiency We further profile the individual running time of `ExPref` with the default settings; i.e., # tasks = 30,000, # task factors= 50, $k = 3$. It takes eight seconds to train the *Worker Model*, 6.15 seconds to solve **Question Selector** that finds the best k factors, and 9.1 seconds to run **Preference Aggregation** that updates the *Worker Model* with the added constraints. These results demonstrate that the individual components of the framework take comparable time.

2.5 Conclusion

We present a framework `ExPref` for eliciting explicit worker's preference for task completion time in crowdsourcing platforms by developing and maintaining a personalized *Worker Model*. Around this model, we define two core optimization problems; **Question Selector** that selects the best set of questions to obtain

a worker's preference in the form of a full/partial ranking of task factors, and **Preference Aggregator** that updates the worker model with provided preferences. We present theoretical results showing the hardness of our problems and algorithms with theoretical guarantees. We conduct large-scale experiments with 165, 168 tasks from CrowdFlower involving 58 workers hired from Amazon Mechanical Turk. Our quality experiments corroborate the necessity of explicit preference elicitation by comparing that with state of the art implicit preference computation. Our scalability results demonstrate that our framework is practical and could be used in real crowdsourcing platforms. As an ongoing work, we are investigating how to adapt this problem, when the explicit feedback is erroneous or has bias due to malicious user behavior.

The related work can be classified into three categories: preference elicitation from the crowd, leveraging worker preferences in crowdsourcing processes, and worker models.

Preference Elicitation. In [51, 98, 37], the crowd was solicited to perform max/top-k and clustering operations with the assumption that workers may make errors. These papers study the relationship between the number of comparisons needed and error. Efficient algorithms are proposed with a guarantee to achieve correct results with high probability. A similar problem was addressed in [50] in the case of a skyline evaluation. In that setting, it is assumed that items can only be compared through noisy comparisons provided by the crowd and the goal is to minimize the number of comparisons. A recent work studies the problem of computing the all pair distance graph [103] by relying on noisy human workers. The authors addressed the challenge of how to aggregate those feedback and what additional feedback to solicit from the crowd to improve other estimated distances. *While we also rely on inputs from the crowd, the elicited input represents each worker's preference for different factors (as opposed to completing actual tasks), and is hence not assumed*

to be noisy or erroneous. However, as worker preferences evolve over time, we propose an iterative approach with the goal of improving task completion time overall.

Leveraging Preferences. Worker preferences for task factors are heavily leveraged in all crowdsourcing processes. Very few of these efforts focused on leveraging them in *task completion* [28, 35, 116]. Authors of [68] investigated 13 worker *motivation* factors and found that workers were interested in *skill variety* or *task autonomy* as much as *task reward*. Chandler and Kapelner [28] empirically showed that workers *perceived meaningfulness* of a task improved throughput without degrading quality. Shaw et al. [116] assessed 14 incentives schemes and found that incentives based on *worker-to-worker comparisons* yield better crowd work quality. Hata et al. [57] studied worker *fatigue* and it affects how work quality over extended periods of time. Other efforts focused on gradually increasing pay during task completion to improve worker retention [48]. Lately, adaptive task assignment were studied with a particular focus on maximizing the quality of crowdwork [44, 59, 60, 96] but primary for improved task assignment. *Existing work showed the importance of leveraging implicit worker preferences for task assignment. In contrast, we show explicit elicitation of worker preferences results in a more accurate model that leads to better estimation of task completion time.*

CHAPTER 3

CROWDSOURCING ANALYTICS WITH CROWDCUR

3.1 Introduction

After the initial introduction of `ExPref` in Chapter 2, we demonstrate CrowdCur, a system that provides that capability and allows requesters and workers to examine various analytics of interest.

CrowdCur is a platform that provides better transparency and generates less frustration among workers. This should incur better worker retention. The idea behind CrowdCur is not new, and several proposals have addressed transparency in crowdsourcing from requester and platform perspectives. Requester transparency reveals details ,such as, recruitment criteria, the conditions under which work may be rejected, and the time before workers' contributions are approved [54]. Platform transparency, e.g., showing feedback to workers on their performance, has also been addressed [68]. Several tools and forums have been developed to disclose information to workers. For example, Turkopticon [65] provides a plug-in that helps workers determine which HITs do not pay fairly and which requesters have been reviewed by other workers. CrowdFlower displays a panel with a worker's estimated accuracy so far.

CrowdCur is designed following the `ExPref` framework. It relies on three core components. The worker and task curation components continuously monitor the platform and produce basic statistics, and the analytics component that aggregates those statistics. We describe the architecture of CrowdCur and demonstration scenarios. CrowdCur is implemented in such a way that it can be used as a plugin with existing platforms such as AMT, CrowdFlower and Prolific Academic and it's designed only for micro-tasks.

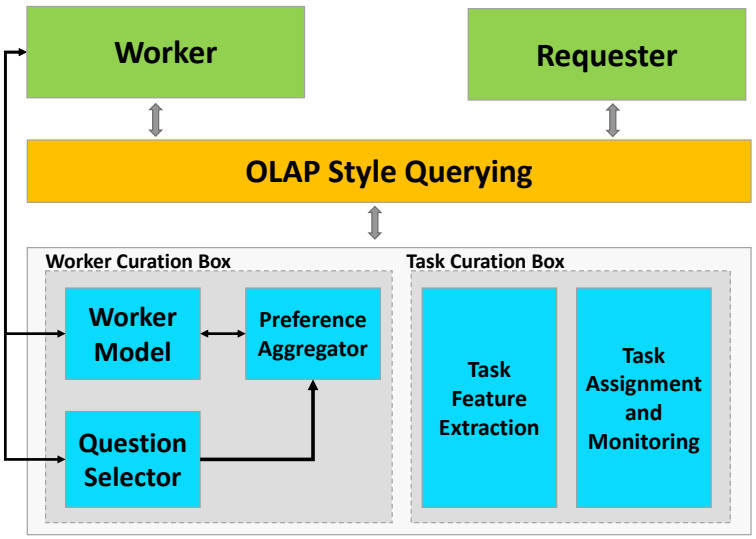


Figure 3.1 CrowdCur architecture overview

3.2 CrowdCur platform

Figure 3.1 outlines the architecture of our platform. CrowdCur is a plugin that the worker or requester installs on their system. The web-based interface enables workers to select and work on tasks, requesters to monitor the completion of their submitted jobs and to get meaningful statistics and recommendations. The back-end consists of three components, *Worker Curation*, *Task Curation*, and *OLAP Style Querying*.

3.2.1 Worker Curation

The ability to characterize workers with their desired preferences is of great importance in crowdsourcing [111, 11, 68]. Most platforms such as, Amazon Mechanical Turk or CrowdFlower rely solely on implicit monitoring of worker preferences. In CrowdCur, our overarching goal is to interleave implicit observations with explicitly seeking feedback from workers to maintain an accurate worker model for effective task completion duration. To achieve this goal, we use *Worker Curation* to keep track of their preferences and progress as they complete tasks. We now describe briefly the three primary components of *Worker Curation*. The inner

working of the algorithms and solutions used in *Worker Curation* have been studied in Chapter 2.

Worker Model: To capture the preference of workers, CrowdCur learns and maintains a learning model. Once this model is developed, one can use it to appropriately estimate future outcomes - in our case to estimate the completion time of a future task. An understandable and practical simplification is that the preference of a worker for a task factor is represented by a weight in the model. We define *Worker Model* as a linear combination of task factors, where the learning parameters are the worker preference. Similarly, we define the task completion duration as the amount of time it took for the worker to complete that task. This frames the learning model into a regression problem. Specifically, we describe a linear regression problem where the objective is to estimate a worker's preferences which minimize the mean square error between the predicted duration and the real duration, i.e.,

$$\underset{\vec{w}\in\mathbb{R}^m}{\operatorname{argmin}} ||\vec{w}^T \cdot \mathcal{T} - \vec{Y}||_2^2 \tag{3.1}$$

where $\vec{Y}$ is the outcome of a set of tasks, $\mathcal{T}$ is a matrix of task factors and $\vec{w}$ is the worker preference. This learning model keeps track of which task factors are important to a worker. Clearly, as the worker completes different tasks, her preferences evolve. This will result in an error in the prediction of the model.

Question Selector: In order to quickly resolve the error in the *Worker Model*, CrowdCur finds a subset of the task factors that cause error in the worker model. This problem is akin to selecting a subset of columns from the task factor matrix such that the pseudo-inverse of the sub-matrix has the smallest norm.

$$\underset{\mathcal{Q}_k\subset\mathcal{Q},|\mathcal{Q}_k|=k}{\operatorname{argmin}} Trace(\mathcal{T}_{Q\setminus Q_s}^T \mathcal{T}_{Q\setminus Q_s})^{-1} \tag{3.2}$$

where $\mathcal{Q}$ is the set of task factors, $\mathcal{Q}_s$ is a subset of task factors. The result of `Question Selector` is a set of task factors which the model is not sure about and there is a chance that the preference of the worker is different than what the model has learned so far.

Preference Aggregator: To understand the preference of the worker, we present the result of *Question Selector* to the worker and ask for their input. Specifically, we are interested to see the worker's full order or partial order among the selected task factors. This will result in a set of linear constraints of the form $i \geq j$. We introduce a set of linear constraints to the *Worker Model* which transform the learning model to a constrained optimization problem. In order to keep track of the latest preference of the worker, we keep track of the history of the preferences that we acquired from the worker. If there is a conflicting constraint, we ignore the older information since there is a chance that the worker's preference has changed.

3.2.2 Task Curation

Tasks come from different sources and have different content. *Task Curation* is in charge of monitoring the flow of tasks and also creating descriptive factors for each task to be used in *Worker Curation*. *Task Curation* has two components: (a) *Task Feature Extraction*, and (b) *Task Assignment and Monitoring*. These components are fully customizable.

Task Feature Extraction: Popular platforms, such as Mechanical Turk or Prolific Academic, characterize tasks using factors, such as type, payment and duration. In CrowdCur, we create a set of factors using unsupervised feature extraction methods. One of the well studied methods for this is Autoencoders [32, 125]. The autoencoder learns a function $f_w(x) \approx x$, i.e., an approximation to the identity function, so as to output $f_w(x)$ that is similar to x. We use the output of the learned function from the Autoencoder (i.e $f_w(x)$) to create the artificial features

by giving the content of the tasks to the function. For example, given a set of images for an image tagging task, we create the artificial features by passing each image to our learned Autoencoder. These extracted features alongside the provided factors for the tasks constitute the foundation of CrowdCur.

Task Assignment and Monitoring: Rather than show the same set of tasks to all workers, the added value of CrowdCur is its ability to integrate a task recommendation component that displays tasks to which workers can self-assign themselves. CrowdCur does not assume a specific task assignment algorithm. It advocates self-assignment where workers decide which task to partake in, since that reveals more information about workers and their preferences.

3.2.3 OLAP Style Querying

Different initiatives implement transparency in crowdsourcing platforms through plug-ins. Turkopticon [65] is a plug-in for AMT that lets workers review tasks and requesters. Crowd-Workers and Turkbench [54] provide expected hourly wages when workers browse tasks. The MobileWorks platform [71] facilitates worker-to-worker communication and assigns manager roles to some workers, allowing workers to monitor each other and benefit from each other's experience, which results in higher quality contributions. CrowdCur complements these plug-ins by providing the ability to reason over computed statistics by querying and comparing the statistics of different tasks and workers.

The third component of our platform is an OLAP style querying component which will expose the information gathered by the previous two components to the workers and requesters. One can consider the information in an online crowdsourcing environment as a cube with multiple dimensions (Figure 3.2). That enables querying and aggregating workers and tasks and comparing them on various dimensions. For example, a worker can compare oneself to alike workers, and a requester could

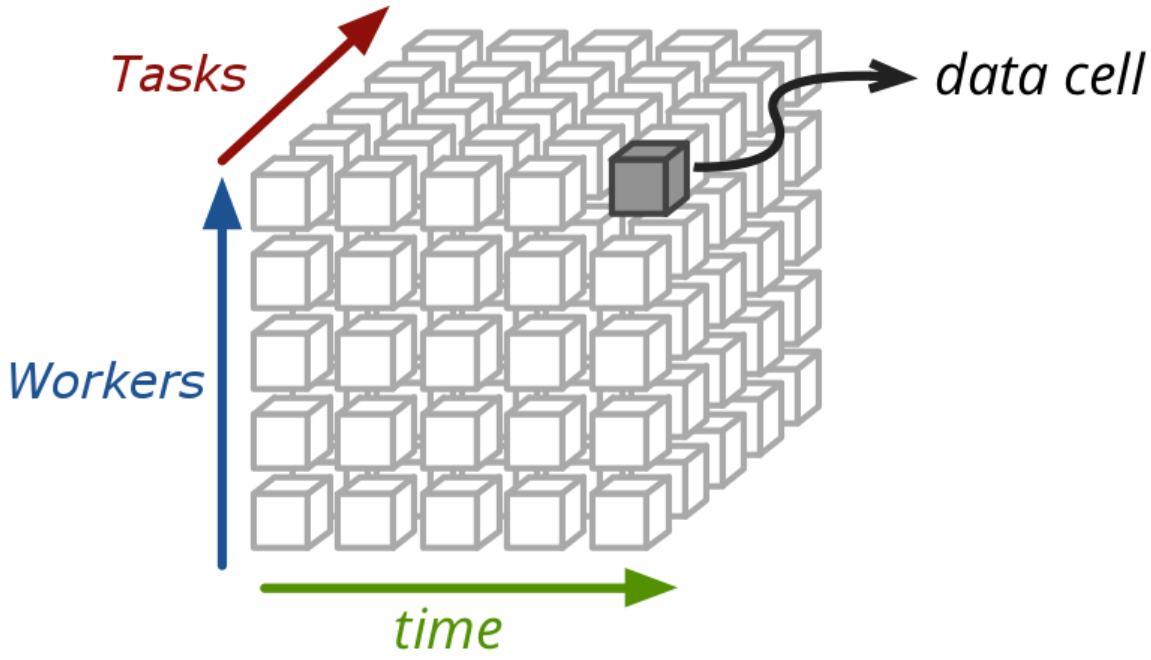


Figure 3.2 Cube of tasks, workers, and time.

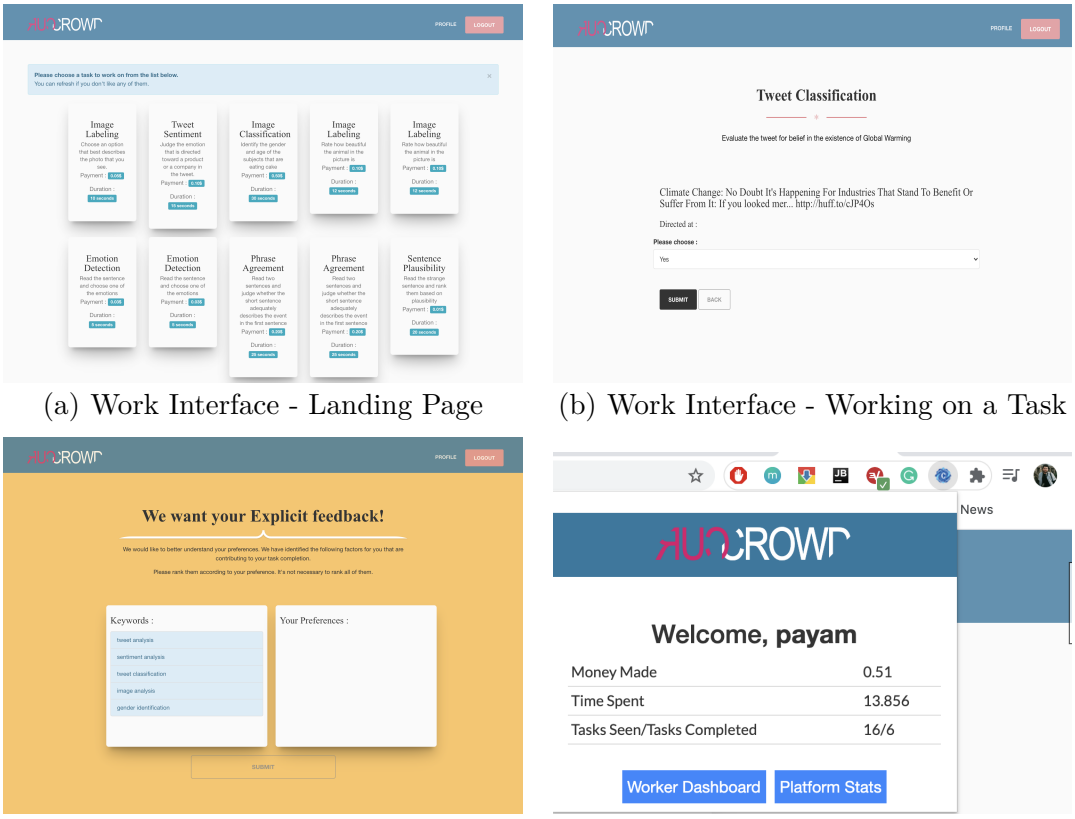
get statistics on a certain kind of tasks. We advocate for an expressive analytics exploration approach. Existing approaches can be classified into by-example [94], by-query [67], by facet (question-answering) [122], and by-analytics [10].

The front end of CrowdCur is implemented using Chrome Plugin Development Toolkit which is a popular front-end javascript library provided by Google. We also use the D3.js library to visualize the information. The back-end has been implemented completely in Python 3.6. For *Task Feature Extraction*, we use Theano [6] to implement various unsupervised feature extraction methods. *Worker Curation Box* is implemented using SciPy optimization framework. At the heart of CrowdCur, we use Django REST web framework and PostgreSQL server as the web server and back-end database.

3.3 System Demonstration

In order to demonstrate the effectiveness of the system, we present different perspectives one might assume in a crowdsourcing environment.

We use 5,000 different tasks in five different categories, such as tweet classification, image tagging, sentiment analysis, and etc. We obtain these tasks from CrowdFlower data for everyone [1] alongside their corresponding responses. In this dissertation, we present two end-to-end scenarios of interacting with CrowdCur 1) How a new worker will experience the CrowdCur enhanced crowdsourcing platform and how the provided analytics can improve her overall satisfaction, and 2) How a requester can get deep insights about tasks and workers.



(a) Work Interface - Landing Page (b) Work Interface - Working on a Task

(c) Eliciting Explicit Feedback (d) CrowdCur Plugin

Figure 3.3 CrowdCur important components.

The three scenarios that we look at in this dissertation are :

As a worker: Since we are interested in simulating different types of workers, we ask the users to complete a set of tasks and provide an explicit feedback when needed. These information will simulate a worker with different rates of task

[1] https://www.crowdflower.com/data-for-everyone/

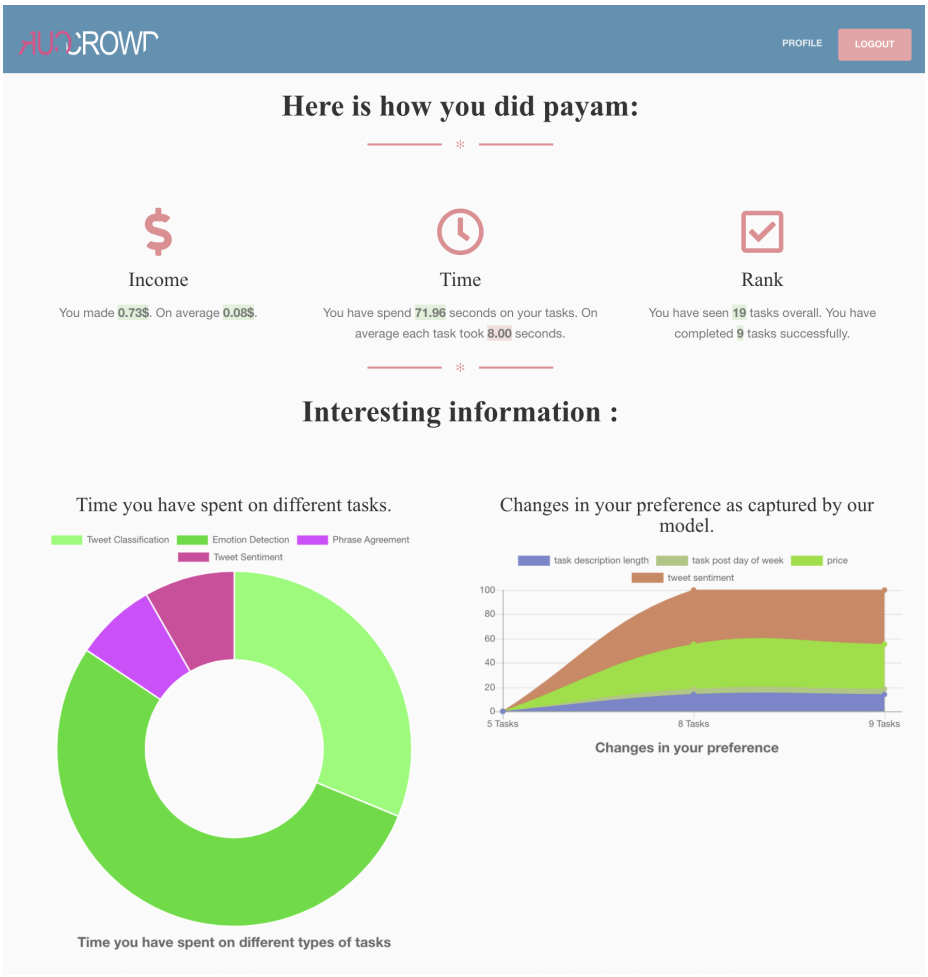


Figure 3.4 CrowdCur worker landing page at the end of a work session.

completion on different task types and different preferences: their preference changes as a function of time. In each round of task completion, the users see the probability of successful completion and the predicted task completion duration of the task currently chosen. After completion of every five tasks, the `Question Selector` is invoked and the users are presented with a set of factors that they can rank according to their preferences. The users will choose which factor ranks higher. These ranked factors are then given to `Preference Aggregator` which in turn will update the model.

At any time during the work session, CrowdCur will generate a list of recommended tasks based on `Worker Model` of tasks they have previously completed. The user also will get information about how similar workers performed and how

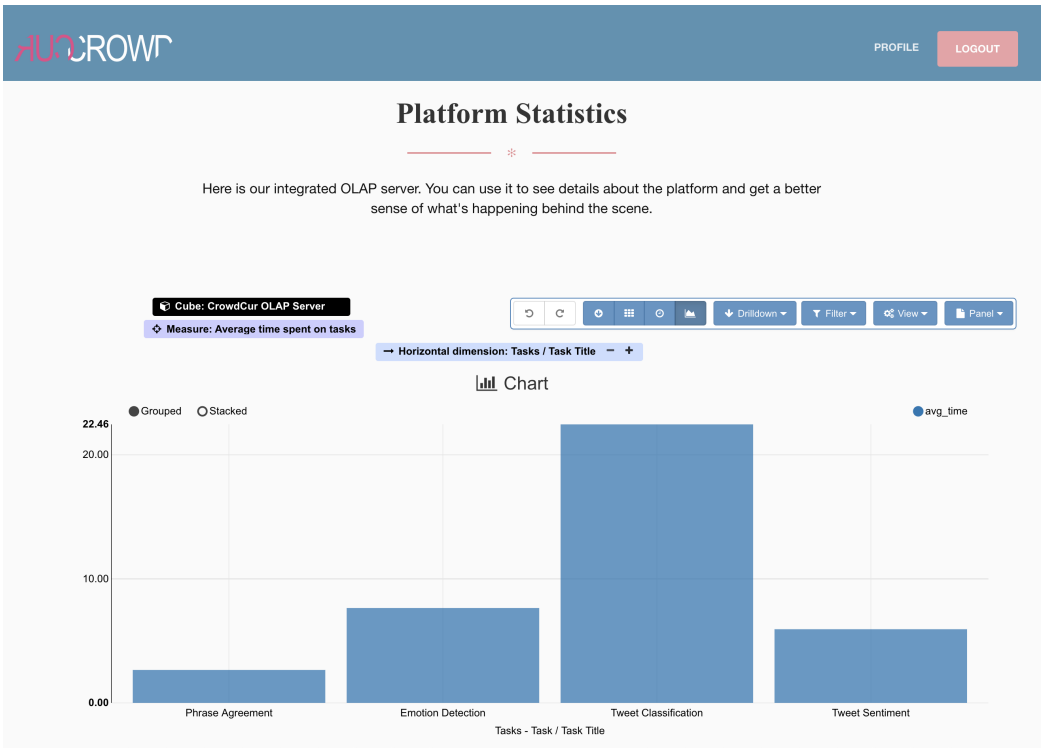


Figure 3.5 CrowdCur requester dashboard.

the worker's preference have changed as a function of time. CrowdCur will also provide insightful information about different aspects of worker's performance such as how much money they have made, how fast they have worked and etc. It also will produce a set of advices for the worker on how to improve their performance based on comparing `Worker Models`. For example, for a worker interested in making more money, the system will check how similar workers acted and which tasks they have chosen and their performances, then it will generate simple guides that will help the worker boost her earnings. Figures 3.3 and 3.4 shows the interface of CrowdCur if the user has logged in as a worker. Users get access to different interactive plots which they can explore. They can also see a list of advices that CrowdCur generated for the workers.

As a requester: Requesters can get different analytics about the workers that are working on their tasks. They can get the structure of the whole workforce that is currently working on the platform and look at their preferences as a function of

time. Users also have access to different analytics about their tasks and also how the tasks are getting completed on the platform. For example, one might be interested in finding out how long does an audio transcription task takes on a platform and what type of worker usually work on these tasks. This will be helpful to a prospective requester since they can decide based on this information which platform to deploy their tasks on or how much they should reward workers. Figures 3.5 shows the user interaction with CrowdCur, if the attendee has logged in as a requester. The user can use the interactive plots to examine how workers' preferences have changed as a function of time. They also can see the top preferences of people who work on a type of task. As a requester, they also can interact with information about the tasks they have submitted such as the rate at which they are getting done.

OLAP style querying: Users will have access to CrowdCur powerful OLAP tool at the end of each experience. For example, as a worker, they can create a query that will compare the current worker profile to other profiles on user defined metrics, or ask about how changes in worker's preference impacts the rate of task completion. Users will also get to use the CrowdCur OLAP system as a requester. For example, as a requester, they can make queries about what type of workers are more likely to work on their tasks, or how long does a specific type of task take to finish.

3.4 Conclusion

We have developed an end-to-end crowdsourcing platform which keeps the focus on the worker instead of requester. CrowdCur follows closely the framework `ExPref` presented in Chapter 2. CrowdCur enables all the three players in the crowdsourcing environment to have a unified access to their data. The *Worker Curation* integrates `ExPref` in the platform and is in charge of periodically updating *Worker Model* by explicitly eliciting worker's preference. *Task Curation* box helps platform owner to identify task factors and fine tune what workers will see in terms

of factors. It also uses state-of-art algorithms to automatically extract important factors from each micro-task in order for the platform to have micro understanding of each task. Lastly, OLAP engine helps bring all the components together. It enables unified access and transparency. The worker can use OLAP system to understand how she did and the *Worker Curation* box will help her to follow her preferences. As requester, CrowdCur provide an overview of what the crowd looks like and how her tasks are performing and even she can get an estimate of when her tasks are done.

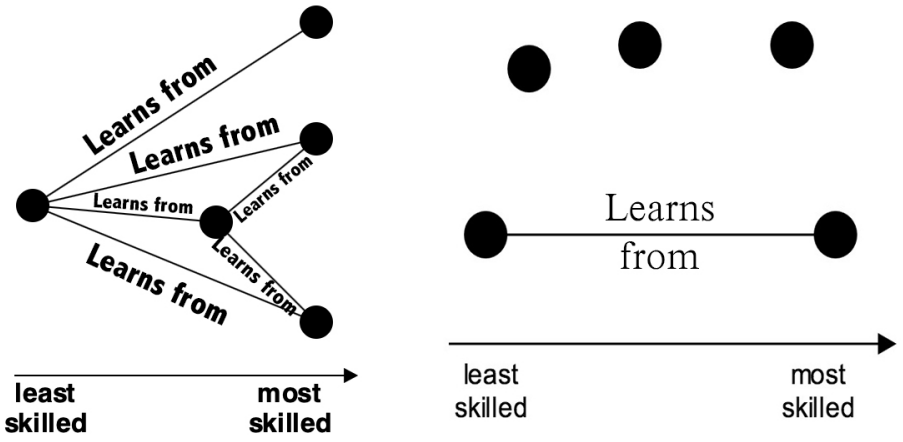
CHAPTER 4

OPTIMIZING PEER LEARNING WITH AFFINITIES

4.1 Introduction

In this chapter, we explore how affinity between group members improves peer learning and address modeling, theoretical, and algorithmic challenges. To the best of our knowledge, this work is the first to examine algorithmic group formation with affinities for peer learning. Group formation in online communities has been studied primarily in the context of task assignment [14, 15, 72, 112, 104]. The problem is often stated as: given a set of individuals and tasks, form a set of groups for the tasks that optimize some aggregated utility subject to constraints such as group size, maximum workload etc. Utility can be aggregated in different ways: the sum of individual skills, their product, etc [15]. Group formation is combinatorial in nature and proposed algorithms solve the problem under different constraints and utility definitions (e.g., [72]). Unlike these problems, we study how to form groups with the goal of maximizing peer learning under different affinities.

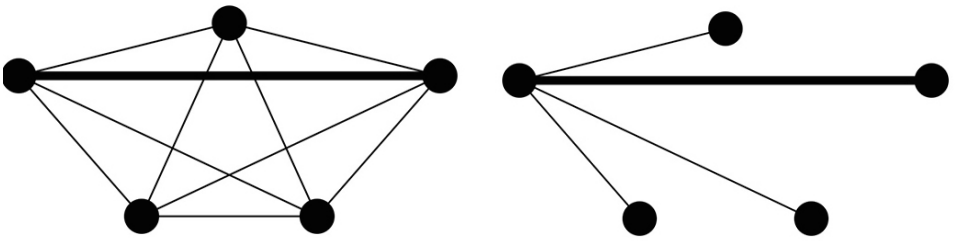
Our first contribution is to present principled models to formalize peer learning and affinity structures. We assume that a peer can only learn from another peer if the skill of the latter is strictly higher than the skill of the former [4]. The learning potential of a peer from a more skilled peer can then naturally be defined as the skill difference between the latter and the former [3, 4, 2]. The learning potential of the latter from the former is null. We use that to formulate two common learning models (Figure 4.1): LpA where each member learns from all higher skilled ones, and LpD where the least skilled member (resp., the most skilled) learns from (resp. teaches to) all others.



(a) (b)

Figure 4.1 Illustration of LpA and LpD- (a) LpA: members learn from higher-skilled ones. (b) LpD: the least skilled member learns from the most skilled one.

Affinity, on the other hand, depends on the application and can be expressed using common socio-demographic attributes or more generally, using models that capture psychological traits. We study our two learning models in conjunction with two common affinity structures (Figure 4.2): AffD where group affinity is the smallest between all members, and AffC where group affinity is the smallest between a designated member (e.g., the least skilled or the most skilled) and all others. We investigate these two affinity scenarios through fact-checking and fact-learning applications.



(a) (b)

Figure 4.2 Illustration of affinity structures - (a) AffD: smallest affinity between all pairs. (b) AffC: smallest affinity between one member and others.

Our second contribution is to study the formalized models systematically and present our theoretical findings. In its general form, our problem formulation is a

bi-objective optimization, with the goal to build k equi-sized groups over a set of n members that maximize both learning potential and affinity. Interestingly, we prove that no variant of optimizing learning potential alone is hard to solve (LPD and LPA), however, the problems become NP-hard when affinity and group size constraints are considered. Therefore, our solution first finds k groups that yield the highest possible learning potential value and then transforms our two-objective problem into *a constrained optimization* that looks for k groups that optimize affinity, with that learning potential value as a constraint.

Our third contribution is algorithmic. We present a suite of scalable algorithms that form groups to maximize learning potential and optimize affinity within *constant approximation factors*. To attain their approximation guarantees, these algorithms assume that affinity satisfies triangle inequality [72]. Many similarity/distance measures such as Jaccard distance and edit distance are known to satisfy metric properties and these properties are usually assumed to design algorithms with guarantees [72]. Our technical contributions are summarized in Table 4.2.

4.2 Modeling and Problem Definition

We present our models following which we define the problems we tackle in this work.

Example 2. *We have a set of fact-checking tasks to be completed by* 12 *individuals with varying skills. We design questions to compute one skill per individual (e.g., on the British royal wedding) and obtain the skill values:* $\{2, 3, 1, 5, 6, 4, 9, 8, 10, 12, 14, 17\}$. *Each pair of individuals has an affinity that reflects how effectively they can collaborate based on their socio-demographics. Therefore, there are* $\binom{12}{2}$ *pairs of affinities forming a complete graph. We show a subset of that graph in Table 4.1 where each worker is identified by her skill. Our goal is to divide the workers into three equi-sized groups of four members each.*

Table 4.1 Partial Affinity Table for Example 2

member	2	3	1	5	6	4	9	8	10	12	14	17
2	-	20	-	-	-	-	3	-	-	-	11	17
3	20	-	13	-	-	-	17	-	-	-	11	4
1	-	13	-	-	-	-	10	-	-	-	14	21
...												
17	17	4	21	-	-	-	-	-	-	-	6	-

4.2.1 Modeling

Group. A group is a set of individuals who will complete a task together. The group size is constant throughout task completion.

Skill. Each individual has an approximated skill reflecting an ability to perform a task. We obtain skills from standard tests and questionnaires to assess expertise level. Other approaches such as inferring skills from completed tasks [106], are also possible.

Affinity. Between every pair of individuals working on a task, affinity captures how well they get along. We express affinity as a similarity measure (higher values are better). We assume affinity satisfies triangle inequality [72]. Group affinity is the aggregation of affinities between its members.

Learning Potential. We define the learning potential between two individuals as the difference between their skill values. The learning potential is not a metric since for the person with the higher skill value, we set it to 0 [3, 4]. The learning potential for a group is the sum of learning potentials of its members.

We have a set of n individuals who are working on a collaborative task. Each w_i has a skill value $w_i^s \in R \geq 0$ representing an ability to complete a task. Our goal is to group them into k equi-sized groups such that the aggregated learning potential and affinity of the groups are maximized. Before formalizing the problem, we investigate variants of learning potential and affinity.

Learning Potential Models Intuitively, the higher the learning potential of a group, the more likely its members will learn from each other. We examine two definitions.

Learning Potential - Diameter. We define LPD as the difference in skills between the most skilled and the least skilled members. This reflects that for a group, we are interested in maximizing the highest learning potential of the least skilled individual in that group.

$$\text{LPD}(g) = \max_{w_i \in g}(w_i^s) - \min_{w_j \in g}(w_j^s) \tag{4.1}$$

In Example 2, if we create the following three groups (we use a member's skill to represent her). $\mathcal{G} = \{g_1 = (2, 3, 1, 5), g_2 = (6, 4, 9, 8), g_3 = (10, 12, 14, 17)\}$ then, $\text{LPD}(g_1) = 5 - 1 = 4$, $\text{LPD}(g_2) = 9 - 4 = 5$, and $\text{LPD}(g_3) = 17 - 10 = 7$. The aggregated learning potential of the grouping $\mathcal{G}$ is 16.

Learning Potential - All. We define LPA as the sum of differences between each member's skill and that of all other members with higher skills.

$$\text{LPA}(g) = \sum_{w_i \in g} \sum_{(w_j \in g, s.t. w_i^s < w_j^s)} (w_j^s - w_i^s) \tag{4.2}$$

Given the previous grouping, we can compute $\mathcal{G} = \{g_1 = (2, 3, 1, 5), g_2 = (6, 4, 9, 8), g_3 = (10, 12, 14, 17)\}$, $\text{LPA}(g_1) = |1 - 2| + |1 - 3| + |1 - 5| + |2 - 3| + |2 - 5| + |3 - 5| = 13$, $\text{LPA}(g_2) = 17$, and $\text{LPA}(g_3) = 23$. The aggregated learning potential of the grouping under LPA is 53.

Affinity Models It is important to look at the effect of affinity on learning since members with higher affinities are likely to learn better from each other and collaborate more effectively [43, 90, 104]. We examine two affinity variants.

Affinity - Diameter. We can formalize affinity as a complete graph $G = (V, E)$ where V is the set of n individuals and E contains weighted edges that correspond to the affinity between every pair of them. In this case, affinity satisfies triangle inequality. We refer to this case as AFFD and define the affinity of a group as the minimum pairwise affinity of all its members as follows:

$$\text{AFFD}(g) = \min_{w_i, w_j \in g} \mathit{aff}(w_i, w_j) \tag{4.3}$$

According to Example 2, if $g = \{2, 12, 14\}$ then
$\text{AFFD}(g) = \min\{\mathit{aff}(2, 12), \mathit{aff}(2, 14), \mathit{aff}(12, 14)\}$, i.e., 4.

Affinity - Center. Affinity can also be defined based on the relationship between one member and all others. We refer to that as AFFC and capture it as a graph $G = (V, E)$ where edges are defined between one designated member w_D and all others.

$$\text{AFFC}(g) = \min_{w_D, w_j \in g} \mathit{aff}(w_D, w_j) \tag{4.4}$$

AFFC captures the cases where the designated member is the least skilled or the most skilled. Similarly to AFFD, in Example 2, if $g = \{2, 12, 14\}$ and the group center is 14 then $\text{AFFC}(g) = \min\{\mathit{aff}(2, 14), \mathit{aff}(12, 14)\}$ which corresponds to $\mathit{aff}(12, 14) = 6$.

For instance, when the task is collaborative fact-checking (e.g., check facts related to the British royal wedding), LpA reflects that each group member will learn from other members with higher skills and AffD captures agreement between the most two disagreeing members in the group. When LpA is combined with AffC, we can capture a task such as text editing where group members collaborate to correct grammar and spelling mistakes in text. In that case, one can intuitively assume that each group member will learn from other members with higher skills and that everyone must have affinity with the highest skilled member. Another example, AffD LpD captures a task where group members are asked to produce facts they believe to be true. In that case, the least skilled member learns from all others and all get along when stating facts.

4.2.2 Problem Definition

Given a set $W = \{w_1, ..., w_n\}$ of individuals with their corresponding skill values w_i^s, our goal is to form a grouping $\mathcal{G}$ that contains k equi-sized groups $g_1, g_2, ..., g_k$ that maximizes two objective functions, aggregated learning potential and aggregated affinity. More formally:

$$\begin{aligned} \underset{\mathcal{G}}{\text{maximize}} \quad & \sum_{i=1}^{k} LP(g_i), \sum_{i=1}^{k} \mathit{Aff}(g_i) \\ \text{s.t.} \quad & |\mathcal{G}| = k,\ |g_i| = \frac{n}{k} \end{aligned} \tag{4.5}$$

where $LP(g_i)$ (resp. $\mathit{Aff}(g_i)$) refers to any of the learning potential (resp. affinity) definitions above.

Since the two objectives are incompatible with one another, our problem qualifies as *multi-objective*. Upon examining the learning potential expressions, we notice that these are polynomial time solvable problems, simply because the primary

Table 4.2 AFF-* LP-* NP-Hardness And Technical Results

Problem	Algo.	Approx.	Time
(AFFC LPD)	GRAFFC-LPD	exact LPD, 3 AFFC	$O(klogn + nlogk)$
(AFFC LPA)	GRAFFC-LPA	exact LPD, 3 AFFD	$O(nlogn)$
(AFFD LPD)	GRAFFD-LPD	exact LPA, 6 AFFC	$O(klogn + nlogk)$
(AFFD LPA)	GRAFFD-LPA	exact LPA, 6 AFFD	$O(nlogn)$

operation that these problems require is sorting. We present exact algorithms for the two learning potential problems in Section 4.3.1. The complexity of our problem lies within the affinity structure and the group size constraint. One way to solve our bi-objective optimization problem is therefore to transform it into a single-objective problem with constraints. We can rewrite Equation (4.5) as follows:

$$\begin{aligned} \underset{\mathcal{G}}{\text{maximize}} \quad & \sum_{i=1}^{k} Aff(g_i) \\ \text{s.t.} \quad & \sum_{i=1}^{k} LP(g_i) \geq OptLP \\ & |\mathcal{G}| = k,\ |g_i| = \frac{n}{k} \end{aligned} \tag{4.6}$$

where *OptLP* is the optimal solution for learning potential maximization. Essentially, we are interested in finding a solution for the affinity objective on the Pareto front, that has the highest learning potential. In Section 4.4, we present approximation algorithms that find a feasible grouping (that maximizes learning potential) and offer provable constant approximation for affinities.

4.3 Optimization

In this section, we first study how to optimize each of our two objectives individually, learning potential and affinity, and in the last subsection we begin studying our bi-objective optimizations by translating them into constrained optimization problems.

This exercise has many benefits - (a) it offers a deeper understanding of the individual problems and (b) it provides perspective on how to combine them and design scalable solutions with provable guarantees (refer to Section 4.4).

4.3.1 Optimizing Learning Potential

Our algorithmic endeavor begins by first describing solutions to group formation that maximize learning potential (LP) alone. Once we obtain the optimal LP value, we use that as a constraint when optimizing affinity (Equation 4.6 in Section 4.2.2). Fortunately, both LP problems are computationally tractable, and we present efficient algorithms that form a grouping with exact solutions. While different, our algorithms are designed in the same spirit as those designed to solve the value-based group formation [3] and the p-percentile partitioning problem [4]. A central idea to those algorithms is to create a grouping based on sorting group members on skill values.

Learning Potential LpD We want to form k groups that maximize the aggregated learning potential which in LpD is the maximum pairwise skill difference (Equation 4.4). LpD of a group is always determined by a single pair of its members, the least skilled and the most skilled ones. Therefore, if we have to form a single group, we just need to select the most and least skilled members and make them part of that group. The other members in the group could be any as their participation does not increase or decrease the LpD value. This seemingly simple logic sufficiently extends to forming k groups. To form k groups, we need to find two buckets with a total of $2k$ people, the most skilled bucket containing the k highest skilled workers (the i-worker in that bucket is referred to as $w_i^{s.high}$), and the least skilled bucket containing the k least skilled workers (the i-worker in that bucket is referred to as $w_i^{s.low}$). We can then form k pairs by grouping one member in the least skilled bucket with one in the most skilled bucket and placing them in the same group. The remaining $n - 2k$ workers

can be distributed arbitrarily across the k groups, while keeping the group size the same (pseudo-code in Algorithm 2).

Applied to Example 2, this is akin to forming the least skilled bucket with participants of skill values $\{1, 2, 3\}$, the most skilled one with values $\{12, 14, 17\}$, and forming three pairs, each one representing a group of size two, by pairing members across the buckets. We can state the following theorem:

Theorem 3. *Any pairing across the least skilled and most skilled buckets produces the optimal aggregated value for LPD.*

Proof. Consider the set of k least skilled members and k most skilled members. It is easy to see that changing the assignment of the least skilled members would not change the overall sum of the skill difference. LPD of the grouping is:

$$OPT\text{LPD} = (w_1^{s.high} - w_1^{s.low}) + (w_2^{s.high} - w_2^{s.low}) + \\ \ldots + (w_k^{s.high} - w_k^{s.low})$$

Indeed, any possible grouping across the buckets over these $2k$ members will not affect the sum, and thus the LPD value. □

Based on Theorem 3, we can state that multiple groupings maximize the LPD value. This corollary is important, because it provides intuition on the challenges that arise when combining affinity with learning potential.

Corollary 3.1. *There are* $\frac{k! \times (n-2k)!}{(n/k-2)!^k}$ *possible groupings to maximize LPD.*

Proof. The members in the highest and the lowest skilled buckets could be paired in $k!$ groupings. The remaining $(n-2k)$ members are to be placed over $(n/k-2)$ positions in each group, and a total over k groups. This gives rise to $\frac{k! \times (n-2k)!}{(n/k-2)!^k}$ groupings. □

Lemma 1. *Computing one optimal grouping for LPD takes* $O(n + klogn)$.

Algorithm 2 Algorithm to maximize LpD

input: set of workers W, k

output: a grouping $\mathcal{G}$, OPTLpD

procedure `LpD`(W, k)

OPTLpD $\leftarrow 0$

create highest and lowest skill buckets with k workers each

$\mathcal{G} \leftarrow$ a set of k empty groups

for i in $(1, ..., k)$ **do**

pick $w_m \in most\_skilled$ and $w_l \in least\_skilled$

$g_i \leftarrow \{w_m, w_l\}$

$W \leftarrow W \setminus \{w_l, w_m\}$

OPTLpD $\leftarrow OPT$LpD $+ (w_m^s - w_l^s)$

end for

while W is not empty **do**

Assign $w_i \in W$ in g_i, s.t $g_i \leq n/k$.

end while

end procedure

Learning Potential LpA The LPA of a group is the sum of skill differences between every member with every other more skilled member (Equation 4.3). The LPA of a set of k groups is the sum over the LPA of each group. What becomes intuitively apparent is that if one has to form one group to maximize LPA, one should always group the most skilled member with the remaining less skilled ones. This logic extends to creating k groups by sorting members on skills (in increasing or decreasing order), and creating n/k buckets, each with k members. To form a group of size n/k, we choose a member from each bucket and repeat this process k times.

Using Example 2, this is akin to sorting the skills of the participants and forming a total of four buckets:

$$\{1, 2, 3\}, \{4, 5, 6\}, \{8, 9, 10\}, \{12, 14, 17\}$$

We form the first group by arbitrarily selecting one member from each of these four buckets, for example, those with skills $\{1, 4, 8, 12\}$. Then we repeat the process twice to get the two other groups, e.g., $\{2, 5, 9, 14\}$ and $\{3, 6, 10, 17\}$.

This algorithm turns out to be optimal - moreover, just like for LPD, all possible groupings across n/k buckets are permissible and will produce the same optimal LPA value.

Theorem 4. *Any grouping across the n/k buckets produces the optimal aggregated value for LPA.*

Proof. The proof is very similar to the proof of LPD. Consider a grouping that we get by running the above algorithm. We sort the members based on their skill values and we create x buckets. The first k workers will go into the first bucket and so on. We create a group by choosing one member of each bucket. After k iterations, we obtain our grouping. Without loss of generality, assume that the highest skilled

member of group i has the skill value of S_i and the other members have s_i^j where j denotes the bucket that this member has been chosen from. Consider the grouping $G = \{g_1 = (S_1, s_1^1, s_1^2), g_2 = (S_2, s_2^1, s_2^2), \ldots, g_k = (S_k, s_k^1, s_k^2)\}$. We can show that swapping two members from the same bucket will not change the LPA optimal value. Without loss of generality, consider a new grouping G' where the position of s_i^j and $s_{i'}^j$ is swapped. Now consider the LPA value for G and G'. We can show that the difference between these two scores is 0. This holds when we pick the lowest skilled member.

$$LP(\mathcal{G}) = \sum_{i=1}^{k} \sum_{j=1}^{x-1} (S_i - s_i^j) \tag{4.7}$$

In the Equation (4.7), the only difference between G and G' is in group i and i'. More accurately, we need to show that $(S_i - s_i^j) + (S_{i'} - s_{i'}^j)$ in the grouping G is equal to $(S_i - s_{i'}^j) + (S_{i'} - s_i^j)$. It's easy to see that these two are identical; hence the proof.

□

Corollary 4.1. *There are $k!^{n/k}$ possible groupings for LPA.*

Lemma 2. *Computing one optimal grouping for LPA takes $O(nlogn)$.*

4.3.2 Optimizing Affinity

Since we express affinity as similarity, optimizing it amounts to minimizing distance. AFFC takes the affinity graph over n members and a subset of k members as centers (teachers) as input, and intends to partition the remaining $n-k$ members into k equi-sized groups such that the sum of the radii (maximum distance between the center and a member in each group) is minimized. AFFD, on the other hand, only takes the affinity graph over n members and k to partition the members into k equi-size groups, such that, the sum of the diameters (diameter of a group is the maximum

pairwise distance in the group) of the grouping is minimized. We first present some theoretical results on the hardness of these two problems.

Even though it intuitively appears that AffC is an easier problem than AffD, both problems are NP-hard. The hardness of AffC is due to the group size constraint.

Theorem 5. *The decision version of the AffC problem in NP-Complete.*

Proof. (sketch) It is easy to see that the problem is in NP. To prove the NP-hardness, the reduction is straightforward (there is a one-to-one correspondence) if we consider the uniform p-centered min-max partition problem as the source problem, which is proved to be NP-hard [73] for general graphs. □

Theorem 6. *The decision version of the AffD problem is NP-Complete.*

Proof. For simplicity, we consider a simpler scenario, where affinity (distance) is binary - 0/1.

For this binary scenario, the decision version of the Affinity-All problem is as follows: given a set of n members, is there a grouping of k equal sized groups, such that the sum of diameters of the grouping is k?

It is easy to see that the problem is in NP. To prove NP-hardness, we use the well-known exact cover by 3-Sets (X3C) for reduction. The decision version of X3C is as follows: given a finite set X with $|X| = 3q$ elements and a collection C of 3-element subsets of X, does C contain an exact cover for X, that is, a sub-collection $C' \subseteq C$, where C' contains exactly q subsets, such that every element of X occurs in exactly one member of C'?

Given an instance of X3C, we reduce it to an instance of AffD in the following way: Each element in X is a member. Therefore, the total number of members $n = 3q$. The affinity graph is a weighted complete graph among the n members and it involves adding edge weights between every pair of members. Each subset of three

elements in C represents three nodes in this graph, and the edge weights between them gets the value 1. This is a polynomial time operation and the number of operations involved in this is the size of C. After that, we need to resolve all the edge weights that are across the subsets. For that, we start considering all the triangles with some unresolved edge weights.

There are three possible scenarios to handle in this process: (1) all three edges unresolved, (2) one edges unresolved, (3) one edge unresolved. For the first case, we can safely add the weight of 0 to each of such edge (this happens when the subsets are fully disjoint). For the second scenario (this happens when two nodes in the triangle are part of the same subset but the third node is part of a different subset), one of the unresolved edges gets 1 and the other gets 0. Finally, for the third scenario (this happens when one member in the triangle is part of both subsets), the unresolved edge gets a 0. We note that this step is again fully polynomial and takes at most $O\binom{n}{3}$ time. After completing this step, we will have assigned all the edge weights in the affinity graph. It could be shown that the affinity graph constructed this way satisfies triangle inequality.

After that, we set $k = q$. Now, the reduction is complete. Notice that X3C $\leq_P$ AffD. There exists a solution to the X3C problem if and only if a solution to our instance of AffD problem exists with the total diameter value q (or k). This completes the proof.

□

4.3.3 Optimizing Affinity with Learning Potential as a Constraint

Finally, we turn our attention to studying the four constrained optimization problems, with the objective to optimize affinity, while satisfying the learning potential value obtained from the algorithms in Section 4.3.1. Since affinity is modeled as a distance, our goal is to minimize that distance, considering the underlying affinity structure.

Recall that LpD and LpA are polynomial-time problems and that we presented exact solutions for both in the previous section. To ease exposition, we will henceforth call the optimal values obtained for the LP problems as Lp-* (it is either LpD or LpA). Our focus now is to study how to optimize Aff-* (AffC or AffD), with Lp-* as constraints.

Theorem 7. *The decision versions of Aff-* Lp-* problems are NP-Complete.*

Proof. The proof is straightforward. □

Our technical deep dive into these four problems is described in Section 4.4. We develop greedy algorithms that are extremely lean in computational time with constant approximation factors. Table 4.2 summarizes the four problem variants and our technical results.

4.4 Constrained Optimization

We now present a suite of algorithms with theoretical guarantees to solve the four different variants of optimizing affinity with learning potential as a constraint. As our problems are NP-hard, we develop approximation algorithms that are scalable and bear theoretical guarantees. Our results are summarized in Table 4.2.

Our algorithms are greedy and use the following intuition: Lp-* problems are first solved and these solutions produce an intermediate grouping that has the optimal LP values. Our algorithms start from these solutions and greedily choose the rest of the members to output the final grouping that is guaranteed to have optimal LP values and provable constant approximation factors for affinity.

4.4.1 Algorithm for AffC LpD

Our discussion of LpD in Section 4.3.1 stated that only $2k$ members (k most skilled and k least skilled) are needed to produce the optimal grouping. Our proposed Algorithm GrAffC-LpD starts from there (recall Algorithm 2) - that is, it first

identifies the $2k$ members which will guarantee the optimal LpD value (thereby satisfying the constraint of the optimization problem). These outputs are referred to as boundary members. That means, two members in each group are decided by now and a total of $2k$ members are decided for the grouping. In each group, the highest skilled member is the teacher and acts as the center for that group since AffC is formalized as the maximum distance between that member and anyone else in the group. The rest of the grouping is performed in a greedy manner. For the remaining $n-2k$ members, all we have to do is assign them to their respective closest center. Since each group has a size constraint, this greedy assignment may lead to sub-optimality - since for a member w_i, the group with the closest distance between its center and w_i may have reached its size and w_i may need to be assigned to a group such that the distance between w_i and its center c_i is larger (potentially worsening the AffC value). But as we shall prove later, this greedy assignment cannot be arbitrarily worse, since affinity between members satisfies triangle inequality (pseudo-code in Algorithm 3).

Going back to Example 2, based on GrAffC-LpD, initially we will have the following partial grouping : $g_1 = \{1, 12\}$, $g_2 = \{2, 17\}$, $g_3 = \{3, 14\}$. After that, Algorithm GrAffC-LpD greedily adds two more members in each group that are not yet part of any group. For example, for g_1, it will add the member who has the highest affinity with 12 and the process will repeat.

Algorithm 3 Algorithm GrAffC-LpD

input: a set W of n participants, k groups

output: a grouping $\mathcal{G}$

B = Call LpD(W, k)

C = the k highest skilled members in B that are k centers

Assign $w_i \in \{W - B\}$ to the closest center c_j s.t., $|g_j| \leq n/k$

Theorem 8. *Algorithm GrAffC-LpD accepts a* 3 *approximation factor to optimize AffC.*

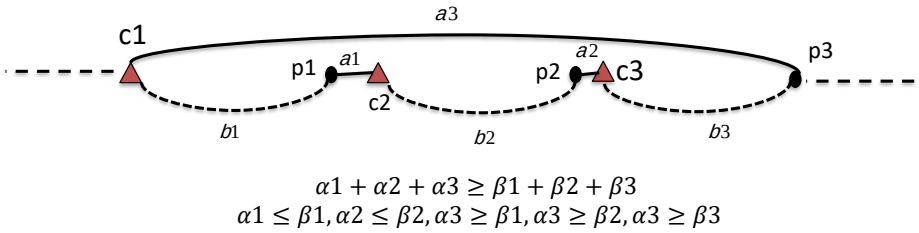


Figure 4.3 Upper bound of approximation factor for GrAffC-LpD.

Proof. Without loss of generality, let us assume a worst case scenario of three groups as shown in Figure 4.3, where members p_1, p_2, p_3 dictate the AffC score of these three groups that are centered around c_2,c_3,c_1, respectively. Because of the greedy assignment, p_1 is assigned to the center c_2, p_2 is assigned to c_3, but at the end because of the size constraints p_3 gets a really bad assignment of c_1. Distance between p_1 and c_2, i.e., $d(p_1, c_2) = \alpha_1$, similarly $d(p_2, c_3) = \alpha_2$, and $d(p_3, c_1) = \alpha_3$. The optimal assignment would have given rise to a different assignment though (as shown in the dotted line), where $p_1 \in c_1, p_2 \in c_2, p_3 \in c_3$. $d(p_1, c_1) = \beta_1, d(p_2, c_2) = \beta_2$, and $d(p_3, c_3) = \beta_3$. Let OPT denote the optimum AffC value, such that $OPT = \beta_1 + \beta_2 + \beta_3$. Of course, $\alpha_1 + \alpha_2 + \alpha_3 \geq \beta_1 + \beta_2 + \beta_3$. But it is easy to notice that

$$\alpha_3 \leq (\alpha_1 + \beta_1 + \alpha_2 + \beta_2 + \beta_3) \tag{4.8}$$

$$\alpha_3 \leq (2\beta_1 + 2\beta_2 + \beta_3) \tag{4.9}$$

Because of the triangle inequality, this is indeed true, $\alpha_1 + \beta_1 \leq 2\beta_1$ (because $\alpha_1 \leq \beta_1$ what the greedy algorithm GrAffC-LpD will ensure. Therefore, we can write that,

$$\alpha_1 + \alpha_2 + \alpha_3 \leq (3\beta_1 + 3\beta_2 + \beta_3) \tag{4.10}$$

$$\leq 3(\beta_1 + \beta_2 + \beta_3) \tag{4.11}$$

$$\leq 3 \times OPT \tag{4.12}$$

It is easy to notice that this argument easily extends to an arbitrary number of groups; hence the proof.

□

Corollary 8.1. *Running time of GrAffC-LpD is* $O(klogn + nlogk)$

4.4.2 Algorithm for AffC LpA

The idea of this greedy algorithm GrAffC-LpA is similar to the previous one, that is start with the partial grouping that LpA returns. However, unlike LpD, LpA creates a set of n/k buckets (or partitions) (see Section 4.3.1) that dictate that forming an intra-partition group is forbidden, and any possible inter-partition groups will result in the same optimal LpA value. In fact, as Corollary 4.1 suggests, there are $(k!)^{n/k}$ possible groupings that yield the optimal LpA value. The challenge is to find one grouping that optimizes affinity.

GrAffC-LpA begins by invoking the LpA procedure to compute n/k buckets that are sorted in increasing order of skills. It selects the teachers as the k members in the last buckets (they are the centers and they have the k highest skills). After that, GrAffC-LpA operates in a greedy fashion. For the remaining $n - k$ members, it

follow a similar approach as Algorithm GrAffC-LpD. At each iteration, it chooses a member from the bucket and assigns it to the closest center. In Example 2, we create a total of four buckets:

$$\{1, 2, 3\}, \{4, 5, 6\}, \{8, 9, 10\}, \{12, 14, 17\}$$

Next, we assign each high skilled member in the last bucket to a group and consider them as centers. As an example, $g_1 = \{12\}$, $g_2 = \{14\}$, and $g_3 = \{17\}$. Next, for the members of the first bucket, based on their affinities, 1 is assigned to g_1, 2 to g_3, and 3 to g_2. The process continues until all the buckets are empty.

Since each group has a size constraint, this greedy assignment may lead to sub-optimality - as it happened in GrAffC-LpD. However, this greedy assignment cannot be arbitrarily worse, because affinity between members satisfies triangle inequality.

Theorem 9. *Algorithm GrAffC-LpA accepts a* 3 *approximation factor for AffC.*

Proof. Similar to the proof for Theorem 8, since the two algorithms follow the same exact process. □

Corollary 9.1. *Running time of Algorithm GrAffC-LpA is* $O(nlogn)$.

4.4.3 Algorithms for AffD Lp-*

There is an interesting relationship between AffC and AffD that merits further delineation. In the AffC problem, we want to minimize the distance from a center to the farthest member in the group (i.e., minimizing the radii). In AffD, we do not have any member as the center, rather we are interested to form groups to minimize the maximum distance (i.e., the diameter). The next theorem states that any solution for the former problem is a solution for the latter that is at most two times worse.

Based on that, the greedy algorithms in Sections 4.4.1 and 4.4.2 could be used to solve AffD,Lp-* problems. We refer to these algorithms as GrAffD-LpD and GrAffD-LpA, respectively for the AffD LpD and AffD LpA problems. GrAffD-LpD is identical to GrAffC-LpD, and GrAffD-LpA is identical to GrAffC-LpA operationally. Their respective running times are the same as their counterparts.

Theorem 10. *Any solution for AffC gives a* 2 *approximate solution for AffD.*

Proof. Consider a solution to AffC. Consider that for a group g_i, the distance from the center c_i to the farthest member is α_i. Assume that w_i is the member with this distance equal to α_i. Based on the triangle inequality, we can easily show that in the worst case, there is another member $w_j \in g_i$ where $d(w_i, w_j) < d(w_i, c_i) + d(w_j, c_i) < 2 \times \alpha_i$. Hence, any algorithm that solves AffC also solves AffD with a 2 approximation factor. □

Corollary 10.1. *Algorithms GrAffD-LpD and GrAffD-LpA have a* 6 *approximation factor for AffD.*

Proof. Proofs are direct derivatives of Theorem 10. □

4.5 Experimental Evaluations

Our experimental effort goes in two directions. In Section 4.5.1, we involve actual human workers and collaborative tasks. In the remaining three subsections, we describe synthetic data experiments.

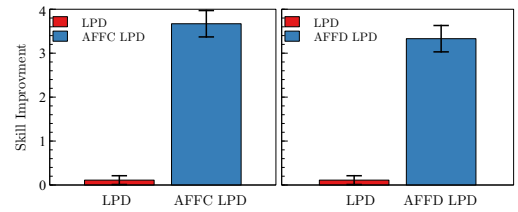
4.5.1 Real Data Experiments

These experiments examine *if affinity brings added utility in peer learning.* They are designed for *collaborative fact-checking and fact-learning* and involve Amazon Mechanical Turk (AMT) workers in three stages: 1) pre-task skill assessment, 2) task completion in a group, and 3) post-task completion skill assessment.

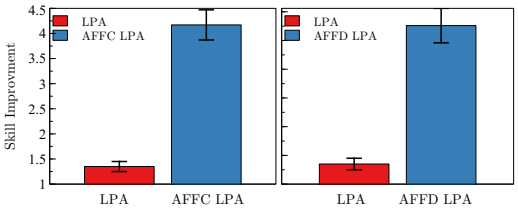
Experimental setup and design. We design four HITs for the four variants of our problem that consider both AFF and LP, as well as four additional HITS that only consider LP (without AFF), a model similar to [3]. We recruit 100 workers from AMT who are redirected to an internal website. Each HIT contains the three aforementioned stages. One of our fact-checking tasks is about the British royal family. These experiments are run in three different stages. We first run a pre-task skill test to assess the skills of each worker using eight true/false questions for which we know the true answer. We set questionnaires for that purpose. Next stage, we set up a collaborative document that contains five facts about the royal family and where workers in the same group can collaborate, comment, and edit. Workers are asked to discuss if these facts are true, and provide further evidence that support their answer. Finally, each worker takes a post-task completion skill test that is again eight true/false questions on the royal family. We also explicitly ask each worker what they have learned by completing that task. We design similar studies for fact-learning, where workers have to actually propose facts with supporting evidence. To keep this experiment tractable, we form groups of size three and run three different samples of the same experiment. This also allows us to analyze results with statistical significance. We pay each worker $2, if all three stages are completed. Each experiment must run over a window of 24 hours to account for differences in time.

Affinity calculation. For simplicity, we capture affinity as the Euclidean distance between their socio-demographic data (specifically, age, country, education) obtained from AMT. There exists other sophisticated measures such as MBTI tests for project-based learning [90]. We nevertheless note that the simple measures that we have used have been shown to be useful affinity indicators [104].

Evaluation criteria. In order to evaluate the effectiveness of affinity in peer learning, we measure the difference between each member's skills before and after



(a)



(b)

Figure 4.4 Skill improvement with and without affinity in LPD (a) and LPA (b).

task completion, and refer to that as *skill improvement*. We also measure the average number of comments in each group and the quality of contributions. These two criteria help us interpret worker engagement and skill improvement.

Summary of results. We compare with and without affinity counterparts for each problem variant (e.g., AFFC LPD with LPD). Our results confirm that affinity improves learning potential substantially with statistical significance. Figure 4.4(a) contains the average skill improvement comparison of LPD with and without affinity and shows the important role of affinity. This is consistent with LPA (Figure 4.4(b)). We also observe that LPA has higher improvements, possibly because facts have many facets that one learns from more skillfull peers. Additionally, Figure 4.5 presents two sample interactions between two workers during task completion. In the first question, *Worker 2* provides a new piece of information about the Queen, which is set as one of the questions in the post-task skill assessment. This additional information helps *Worker 1* improve her skill during post-task assessment.

Finally, we anecdotally observe that higher learning potential yields higher quality task outcomes. On average, quality (computed as the average number of

The Queen does not need a passport to travel True or False ?	• Worker 1: True. All British Passports are issued in the Name of Her Majesty, The Queen. • Worker 2 : I found an article which agrees with your findings. **Fun fact: she also doesn't need a driver's license or a license plate on her car.**
Members of the royal family have to accept absolutely all gifts.	• Worker 1 : (Mostly false; Large true in practice.) While I couldn't find any law requiring the Royals to accept all gifts. • Worker 2 : I found an article which says they make a list of all gifts they receive throughout the year and release it publicly. In addition, they donate many of their gifts.

Figure 4.5 Sample of worker interactions with each other.

facts correctly identified by the groups), is higher for groups built with affinity (4.3 facts out of 5 are correct), compared to their counterparts built without affinity (3.9 out of 5).

4.5.2 Synthetic Experiments Setup

These experiments evaluate the qualitative guarantees and the scalability of our algorithms. All algorithms are implemented in Python 3.6 using Intel Core i7 4GHz CPU and 16GB of memory and Windows operating system. All numbers are averages of 10 runs.

Implemented Algorithms. The closest works to ours [3, 4] do not consider affinity and cannot be used for synthetic data experiments. Hence, we implement three additional baselines:

Optimal. We formalize the ILP solution using the below formulation. It's easy to implement this formulation in any Linear Programming software. For this experiments we use PuLP, a popular python based ILP package.

$$\underset{\mathcal{G}}{\text{optimize}} \sum_{i=1}^{k} \text{AFF-*}(g_i)$$

$$\text{s.t.} \sum_{i=1}^{k} LP(g_i) \geq \text{LP-*}$$

$$\text{AFF-*}(g_i) = \sum_{j=1}^{n} \sum_{m=1}^{n} x_{i,j} * x_{i,m} * \mathit{Aff}(w_j, w_m), \forall i = 1 \ldots k$$

$$\sum_{j=1}^{n} x_{i,j} = n/k, \forall i = 1 \ldots k$$

$$x_{i,j} = 0/1 \;\; (i = 1...k \;\&\; j = 1 \ldots n)$$

The rationale behind implementing ILP is to demonstrate that the theoretical approximation factors of our algorithms hold in practice. Since ILP is NP-hard, the algorithm does not terminate beyond $k = 3$ and $n = 50$.

Baseline-1 (clustering-based). This baseline is motivated by the popular k-means algorithm. It starts with a random grouping and greedily swaps members across groups as long as that improves affinity, while satisfying group size. Once the grouping converges based on affinity, we check if it satisfies the optimum learning potential value (which could be derived efficiently in polynomial time). If not, we perform another set of swaps to move members across the groups until we find a grouping that reaches the optimal learning value.

Baseline-2. This is a simpler and efficient baseline. It first solves the learning potential problem and finds the seed members in each group that dictate the optimal learning value. The rest of the members are assigned randomly to groups by considering group size.

These solutions are compared with four of our algorithms GRAFFC-LPD, GRAFFC-LPA, GRAFFD-LPD, GRAFFD-LPA (refer to Section 4.4, whenever applicable).

Experimental Setup. We simulate a group of workers with two functions that capture the relationship between skill and affinity. Specifically, there are two random number generators, one produces the skill of each member and the other generates pairwise affinities that satisfy triangle inequality.

We consider two skill and affinity distributions: (a) *Normal*, where the mean and standard deviations are set to $\mu = 100$, $\sigma = 20$, respectively; (b) *Zipf*, where the value of the exponent α is set to 1.5.

Parameters: We vary n (the total number of individuals), k (the number of groups), and the skill and affinity distributions.

Summary of Results. 1. Our algorithms exhibit tighter approximation factors than the bounds we proved. Our algorithms also outperform the two baselines.
2. The approximation factors of the algorithms with a Normal skill distribution are better than Zipf.
3. All algorithms are highly scalable considering up to 10^6 members and 160 groups and only take seconds to run.

Table 4.3 Approximation Factors

Algo.	Parameters	App. Factor
(GRAFFC-LPD)	$[n = 15, k = 3]$	1.13(0.12)
	$[n = 50, k = 3]$	1.23(0.02)
(GRAFFC-LPA)	$[n = 15, k = 3]$	1.04(0.07)
	$[n = 50, k = 3]$	1.02(0.03)
(GRAFFD-LPD)	$[n = 15, k = 3]$	1.21(0.14)
	$[n = 50, k = 3]$	1.31(0.04)
(GRAFFD-LPA)	$[n = 15, k = 3]$	1.18(0.11)
	$[n = 50, k = 3]$	1.19(0.12)

4.5.3 Quality Experiments (Synthetic)

We assess quality by measuring the approximation factor and the objective function value. Both of these are described considering affinity, per our problem definition. LP values are always optimal (as the algorithms for LP are exact). **Default parameter setting.** Unless otherwise stated, k is set to 25 and n to 1000.

Comparison against ILP: Table 4.3 presents the approximation factor of the four algorithms on a small dataset generated from Normal Distribution. For all the four algorithms, the approximation factor in practice is very tight and the deviation is always between 1 and 2.

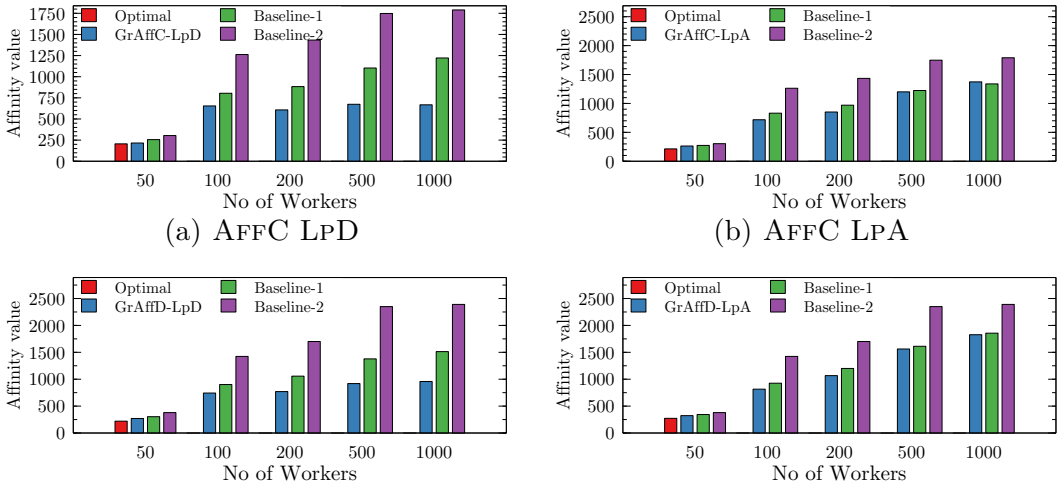


Figure 4.6 Aff-* Lp-* values varying n for Normal distribution.

Varying n : Figure 4.6 reports the results of varying n for Normal distribution. Of course, ILP does not scale beyond $n = 50$, but it is easy to notice that for all the cases we could compare, our greedy solutions attain a very tight approximation factor (close to 1.5). Figure 4.7 shows the affinity values for Zipf distribution. Similar observations hold. Our proposed algorithms perform significantly better.

There are two interesting observations in both Figures 4.6 and 4.7. Firstly, the grouping generated by our algorithm attains smaller objective value as the number of workers grow. Secondly, for Normally distributed data, we observe a consistent

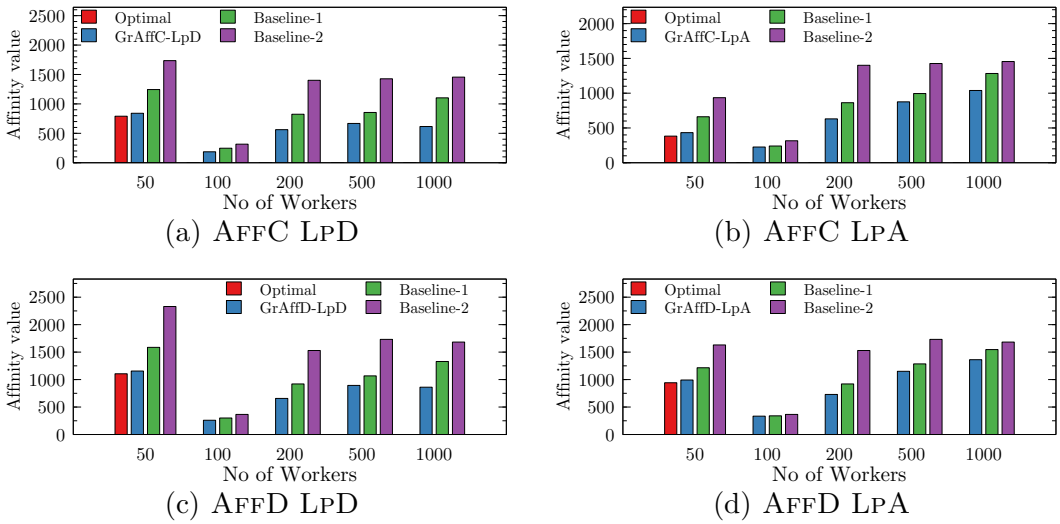


Figure 4.7 Aff-* Lp-* values varying n for Zipf distribution.

growth of objective value as the number of workers increases. This is not the case for the Zipfian distributed data. We conjecture that this is caused by the skew in the values generated from the Zipfian distribution. Some values are very large and others are vary small. Another important factor is that the data sampled from a Zipfian distribution consists of mostly duplicate values.

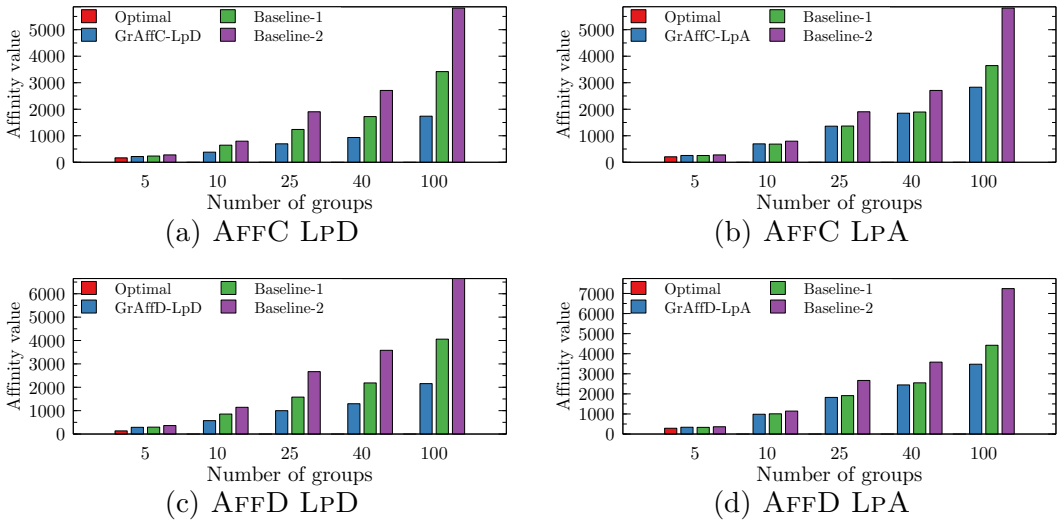


Figure 4.8 Aff-* Lp-* values varying k for Normal distribution.

Varying k : Figures 4.8 and 4.9 present the results of varying k for Normal and Zipf distributions. The ILP for $k = 5$ is ran on $n = 50$. Our presented algorithms

consistently outperform the other baselines. We observe that the change in k affects the objective value significantly more than change in the number of workers (n). We believe this is because larger k signifies more centers to assign workers to. Remember in Algorithm 3, we need to assign workers to the closest center. This means for larger values of k, we would diverge from the optimal solution easier. In fact, k impacts our algorithm more than n.

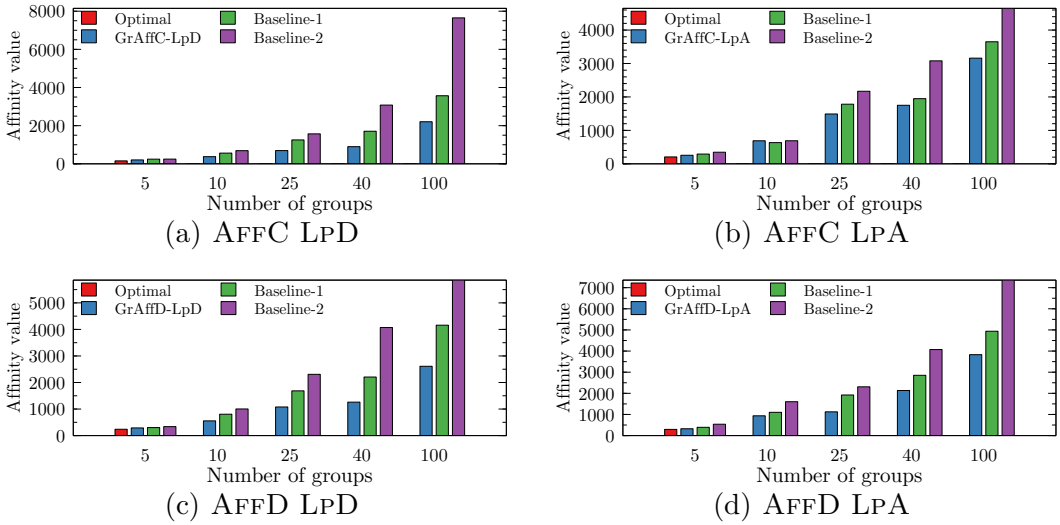


Figure 4.9 Aff-* Lp-* values varying k for Zipf distribution.

4.5.4 Scalability Experiments (Synthetic)

We measure running times and compare with **Baseline-1**. We exclude ILP since it does not scale, and **Baseline-2** since it produces inferior objective values. Running time is reported in seconds.

Default parameter settings. We found that Normal and Zipf skill distributions have identical running times for each variant of Aff-* Lp-* problems. We also note that, as proved in Section 4.4.3, the running time of AffD is identical to that of AffC, considering their respective Lp-* counterparts. Therefore, we only present results for Aff-* LpD and Aff-* LpA. We vary n and k with defaults set to $n = 100000$ and $k = 5$.

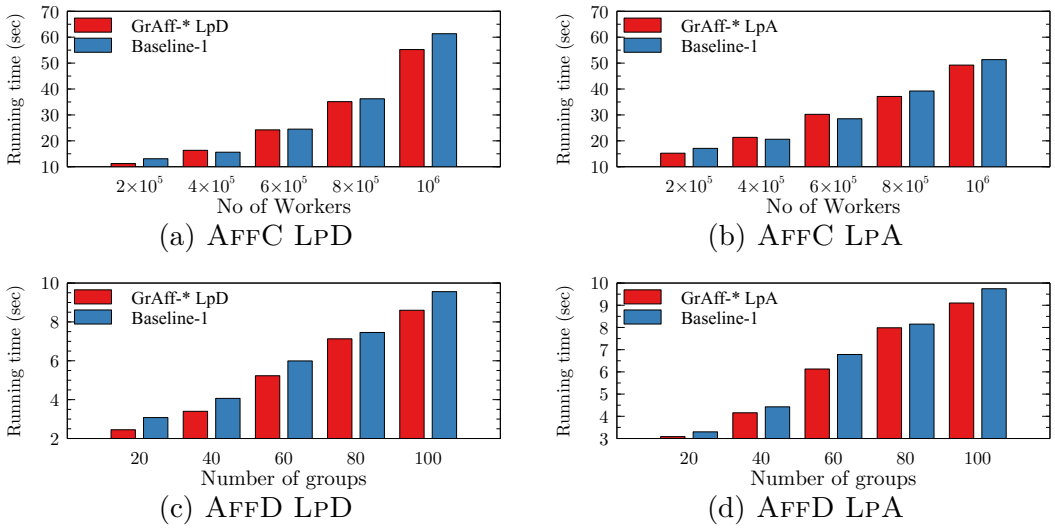


Figure 4.10 Scalability results for Aff-* Lp-*

Results. Figure 4.10 presents results. Our algorithms are highly scalable and take seconds only. The algorithms run linearly with varying n and k which confirms our theoretical analysis.

4.6 Conclusion

We examine online group formation where members seek to increase their *learning potential* via task completion with two learning models and affinity structures. We formalize the problem of forming a set of k groups with the purpose of optimizing peer learning under different affinity structures and propose constrained optimization formulations. We show the hardness of our problems and develop four scalable algorithms with constant approximation factors. Our experiments with real workers demonstrate that considering affinity structures drastically improves learning potential, and our synthetic data experiments corroborate the qualitative and scalability aspects of our algorithms. Our work studies computational aspects and relates to team formation and computer-supported learning.

Team formation was first studied to form a single group with one objective and later a 2-approximation algorithm was proposed for bi-criteria team formation in

social networks [72]. In [14, 15], Anagnostopoulos et al. propose online algorithms for the balanced social task assignment problem. Capacitated assignment was studied in a follow up work [79]. Generalized density sub-graph algorithms were later proposed [107]. [112, 104] study the problem of forming teams for task assignment considering affinity. In [110], the hardness of forming groups to optimize group satisfaction is studied under different group satisfaction semantics.

Computer-Supported Collaborative Learning (CSCL). Social science has a long history of studying non-computational aspects of computer-supported collaborative learning [33, 36]. With the development of online educational platforms (such as, Massive Open Online Courses or MOOCs), several parameters were identified for building effective teams: (1) individual and group learning and social goals, (2) interaction processes and feedbacks [119], (3) roles that determine the nature and group idiosyncrasy [36].

CHAPTER 5

DIVERSIFYING RECOMMENDATIONS ON SEQUENCES OF SETS

5.1 Introduction

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. Intra and inter diversities can be either minimized or maximized which gives rise to *a bi-objective formalism to express four problem variants* **(Section 5.2.2)**. *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* [16, 29, 1, 135, 131, 130, 123, 102, 91, 95, 44, 59, 60, 133, 105].

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 5.2.3)**. 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 5.3)**. 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 5.2.4)** 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 two out of the four problems **(Section 5.3.4)**. Tables 5.2 and 5.3 summarize our theoretical and algorithmic results.

Our last contribution is experimental. We conduct 4 large-scale experiments: two with human subjects (music playlist and task recommendation), the other two with large real and simulated data. In music recommendation **(Section 5.4.1)**, *our results highlight, with statistical significance, 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 that the benefit of diversification is more prominent for long sessions (contains 5 sets and 10 tasks per set compared to shorter sessions that contain 3 sets and 3 tasks per set), as for those, our algorithms achieve higher quality and worker satisfaction with statistical significance, than a baseline with no diversity.* **(Section 5.4.2)** 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 to the optimal. 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.

5.2 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.

Table 5.1 Example of Task Recommendation 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

Example 3. *Consider a set of* $N = 12$ *tasks, which are most relevant to a specific worker. Table 5.1 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.*

5.2.1 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 3, 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:

$$Intra^d(s) = \sum_{i=1}^{l}(t_i^d - \mu_s^d)^2 \tag{5.1}$$

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 3, 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:

$$Inter^d(s_i, s_{i+1}) = (\mu_{s_i}^d - \mu_{s_{i+1}}^d)^2 \tag{5.2}$$

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 3 .

Other set-based [1] and sequence-based [135] definitions exist and could be considered in future work.

5.2.2 Problem Definitions

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 four 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

$$\begin{aligned} &\underset{S}{\text{optimize}} \sum_{i=1}^{k} (Intra^{d}(s_i)) \\ &\underset{S}{\text{optimize}} \sum_{i=1}^{k-1} (Inter^{d'}(s_i, s_{i+1})) \\ &\text{s.t.} \\ &|S| = k,\ |s_i| = l, N = k \times l \end{aligned} \tag{5.3}$$

5.2.3 Problem Analysis

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.

Intra Diversity Optimization

Theorem 11. *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 12. *Optimizing Max-Intra is NP-Hard.*

Proof. The proof of this theorem uses another claim that we prove later (Theorem 16). This latter theorem formally proves that *Max-Intra* happens (i.e., $\sum_{i=1}^{k}(Intra(s_i))$ 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. Since groups have the same size l, the problem is akin to finding groups of items whose sum is equal:

$$\sum_{t_i \in s_1} t_i = \sum_{t_i \in s_2} t_i = \cdots = \sum_{t_i \in s_k} t_i \tag{5.4}$$

To prove NP-hardness we reduce an instance of the k-Equal Subset Sum of Equal Cardinality Problem (k-`ESSEC`) [31] to an instance of *Max-Intra*, as follows. Given an instance of k-`ESSEC` with $S = \{a_1, ..., a_N\}$ 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 $\mu_{s_i} = \mu_{s_{global}} = \frac{\Sigma_{i=1}^{N} a_i}{N}$.

□

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

Proof. 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 [49] 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 $v_i \in V$ represents l items with 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

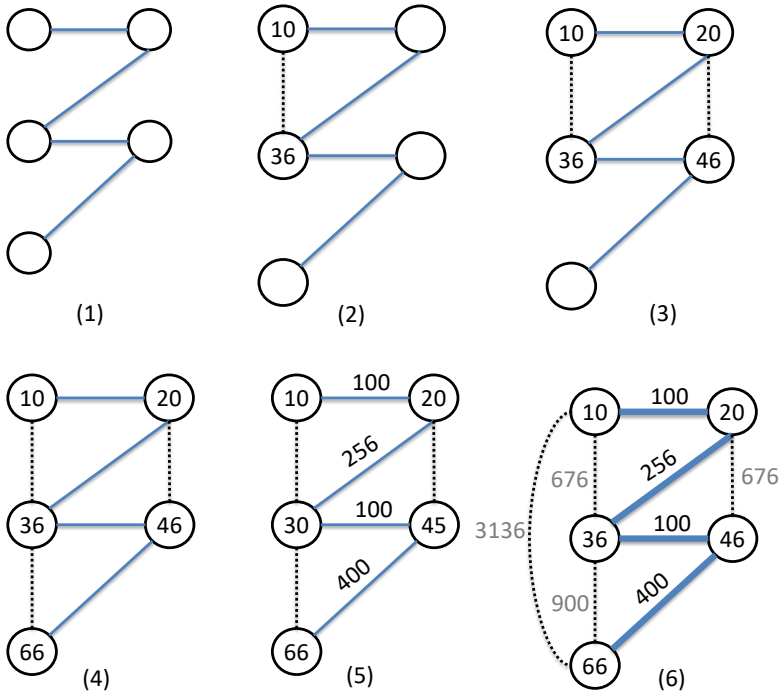


Figure 5.1 Reduction: Hamiltonian Path to the *Inter* problem.

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 5.1 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 14. *The bi-objective optimization problems combining intra and inter diversity are all NP-Hard.*

Proof. Consider one of the problem variations, *Max-Intra-Min-Inter*. Since both multi-objective functions correspond to a NP-complete, we can easily argue that if we have an algorithm to solve the bi-objective problem to optimality, then we can replace one of the objective functions with a constant factor and solve the other NP-hard problem optimally which is impossible. □

5.2.4 Modified Problem Definitions

As proved in Theorem 14, each of the four bi-objective optimization problems are 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. 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).

$$\begin{aligned} &\min_{S}(\max) \sum_{i=1}^{k-1}(Inter^{d_2}(s_i, s_{i+1})) \\ &\quad \text{s.t.} \\ &\qquad \sum_{i=1}^{k}(Intra(s_i))x <= OPT_{Intra^{d_1}} \\ &\qquad |S| = k,\ |s_i| = l, N = k \times l \end{aligned} \tag{5.5}$$

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

Using Example 3, the sequence

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

Table 5.2 Algorithms and Approximation Factors

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`	$O(NlogN + k^2 + logk)$	$1/2$

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 $Inter^{Reward}$.

5.3 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 5.5). Table 5.2 summarizes our technical results.

5.3.1 Algorithm Min-Intra

The objective here is to design k sessions, each of length l, such that the aggregated *Intra* diversity over the k 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 5.2. 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.

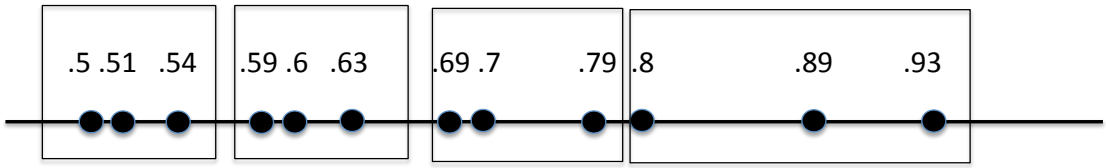


Figure 5.2 Sorted *Intra-Diversity* of skills.

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 15. *Algorithm* `Exact-Min-Intra` *is exact.*

Proof. 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 3. *Algorithm* `Exact-Min-Intra` *runs in* $O(NlogN)$.

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

5.3.2 Algorithm Max-Intra

As proved in Section 5.2.3, 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_T. This proof is critical, as it helps us design our solution. Theorem 16 has the formal statement.

Theorem 16. $\sum_{i=1}^{k}(Intra(s_i))$ *is maximized when*

$$\mu^d_{s_1} = \mu^d_{s_2}, \ldots = \mu^d_{s_k} = \mu_{global} \tag{5.6}$$

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

$$\mu_{s'_1} = \mu_{s'_2} = \ldots = \mu_{s'_k} = \mu_{global} = \frac{1}{k * l}\sum t \tag{5.7}$$

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

$$\begin{aligned} Intra(S) &= \sum_{s_i} Intra(s_i) \\ &= \sum_{t \in s_1} (t - \mu_{s_1})^2 + \ldots + \sum_{t \in s_k} (t - \mu_{s_k})^2 \\ &= \sum_{i=1}^{N} t_i^2 - l(\mu_{s_1}^2 + \mu_{s_2}^2 + \ldots + \mu_{s_k}^2) \end{aligned} \tag{5.8}$$

$$Intra(S') = \sum_{i=1}^{N} t_i^2 - kl\mu_{global}^2 \tag{5.9}$$

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

$$\sum_{i=1}^{N} t_i^2 - l(\mu_{s_1}^2 + \mu_{s_2}^2 + \ldots + \mu_{s_k}^2) > \sum_{i=1}^{N} t_i^2 - kl\mu_{global}^2 \tag{5.10}$$

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

$$\mu_{s_1}^2 + \mu_{s_2}^2 + \ldots + \mu_{s_k}^2 < 0 \tag{5.11}$$

which is a clear contradiction, hence the proof. □

Algorithm: Theorem 16 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. 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

$$T = \{0.5,0.51,0.54,0.59,0.6,0.63,0.69\ ,0.7,0.79,0.8,0.89,0.93\}$$

Step 1: $$b = \begin{bmatrix} [] & \cdots & [] \\ \vdots & \ddots & \vdots \\ [] & \cdots & [] \end{bmatrix}$$

Step 2: $$b_{l\times k} = \begin{bmatrix} [0.5] & [0.51] & [0.54] & [0.59] \\ [0.6] & [0.63] & [0.69] & [0.7] \\ [0.79] & [0.8] & [0.89] & [0.93] \end{bmatrix}$$

Step 3:
$$d(b_1) = \max(\{|0.68 - 0.59|, |0.68 - 0.5|\}) = 0.18$$
$$d(b_2) = \max(\{|0.68 - 0.6|, |0.68 - 0.7|\}) = 0.08$$
$$d(b_3) = \max(\{|0.68 - 0.79|, |0.68 - 0.93|\}) = 0.25$$

Step 4: $$Merge(b_2, b_3) \qquad \begin{bmatrix} [0.5] & [0.51] & [0.54] & [0.59] \\ [0.6,0.93] & [0.63,0.89] & [0.69,0.8] & [0.7,0.79] \end{bmatrix}$$

Figure 5.3 `Ap-Max-Intra` steps on Example 3.

run in an iterative manner. In the third step, the algorithm scores the bins according to a scoring function defined in Definition 1. Finally, it greedily merges two bins. This process is repeated for $l - 1$ number of iterations.

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

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

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 3, 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 3, 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 5.3 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_{ij}^{merge} contains the m-th smallest items of b and m-th largest items of b' $(1 \leq m \leq k)$. Considering Example 3, 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 4 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: **end while**
10: Return the final merged bin

Theorem 17. `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 18. *Algorithm* `Ap-Max-Intra` *has* $\frac{1}{2-1/k}$ *approximation factor.*

Proof. The 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 [89] 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 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 [89] 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. □

5.3.3 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 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 (5.2). 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 4, 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 [74, 100]. 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 [100] 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 5 presents the pseudo code of *Max-Inter* algorithm.

Using Example 3 to find *Max-Inter* of *Skill* dimension, we first apply the `Exact-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 four 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 (5.2). 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 5 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.

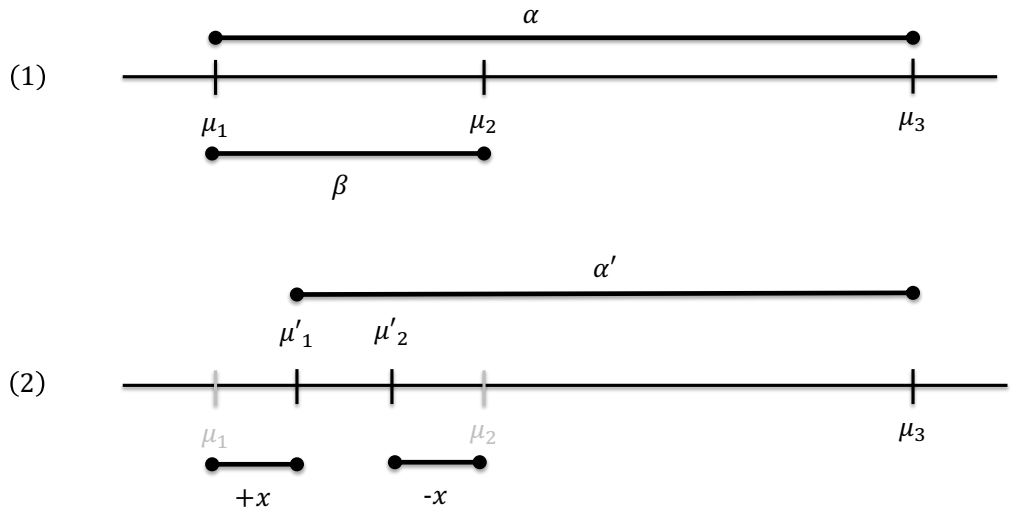


Figure 5.4 Relationship between *Min-Intra* and *Max-Inter*.

Theorem 19. *Both* `Ap-Max-Inter` *and* `Ap-Min-Inter` *run in* $O(NlogN + k^2 + logk)$.

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 4. *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. For the case of *Max-Intra*, the solution will require the averages of all groups to be the same (Recall Theorem 16). This results in having *Min-Inter* with value 0, leading to the optimal solution. Hence the proof.

To show the relationship between *Min-Intra* and *Max-Inter*, we show that when $k = 3$, *Min-Intra* corresponds to maximizing inter on the same dimension. 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 5.4 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 20. `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 21. *Algorithm* `Ap-Min-Inter` *has* $4 - 2/k$ *approximation factor.*

Proof. Similar to the proof of `Ap-Max-Inter`, using Lemma 4, 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 $4 - 2/k$. □

5.3.4 Optimizing Inter with Intra as Constraint

We now develop algorithms for the constrained optimization problems defined in Section 5.2.4. When *the values of the item dimension used for intra diversity are all unique, two of these four algorithms have provable approximation factors.* Table 5.3 provides the summary of our technical results.

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

Table 5.3 Algorithms and Approximation Factors for Problem Combinations

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*

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*. Algorithm 6 presents the generic pseudo code. These two algorithms are based on heuristics and may not have any provable bounds.

Algorithm 6 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

Theorem 22. *Algorithm Alg-Min-Intra, Max-Inter has* $(1, 1/2)$ *approximation factor Min-Intra,Max-Inter problem and Alg-Min-Intra, Min-Inter has* $(1, 4 - 2/k)$ *approxi-*

mation factor Min-Intra,Min-Inter problem, as long as items in intra dimension have unique values.

Proof. 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. □

5.4 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.

5.4.1 Experiments with Human Subjects

We validate our framework in two applications: music recommendation, where we generate music channels, and task recommendation in crowdsourcing, where we generate task sessions.

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.

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 five music playlists for each user. Each playlist has five channels, and each channel has 10 songs. The first four playlists are generated using the algorithms in Table 5.3, with dimensions specified for each context in Table 5.4. 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.

Table 5.4 Diversity Dimensions Per Context

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

Table 5.5 Average Evaluation Scores Across All Contexts

	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*

Summary of Results. We observe in Table 5.5 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) [121]. 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 5.6, 5.7, and 5.8. The sample size of 200 workers from the estimated $200,000$ workers in AMT [41] gives our results a 99% confidence level and a 10% error margin (based on the Central Limit Theorem [118]). *In summary, our music experiment clearly shows that diversity is preferred over no diversity.* Additionally, diversity definitions depend on context, as observed in Tables 5.6, 5.7, and 5.8.

Table 5.6 Average Number of Selected Songs Per Context

	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 5.7 Average Diversity Rating 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 5.8 Average User Satisfaction 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*

Task Recommendation. In these experiments, we recommend short and long task sessions to workers in crowdsourcing. The short sessions consist of three sets each with three tasks. The long sessions consist of five sets and each set consists of 10 tasks.

Dataset. The dataset consists of $20,000$ tasks from Figure Eight's open data library. 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 5.3 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.

Table 5.9 Task Recommendation: Short Sessions

	Scenario	Through put	Quality (%)	Worker Satis-faction
1	*Min-Intra(creation date), Min-Inter(skill)*	6.13	**65.64**	4.47
2	*Min-Intra(skill), Max-Inter(reward)*	5.93	62.77	**4.43**
3	*Max-Intra(skill), Min-Inter (reward)*	5.91	*61.76*	4.44
4	*Max-Intra(duration), Max-Inter (skill)*	*5.35*	61.24	4.46
5	*no diversity*	**7.53**	64.38	*4.26*

Table 5.10 Task Recommendation: Long Sessions

	Scenario	Through put	Quality (%)	Worker Satis-faction
1	*Min-Intra(creation date), Min-Inter(skill)*	6.95	68.33	4.48
2	*Min-Intra(skill), Max-Inter(reward)*	6.96	69.27	**4.5**
3	*Max-Intra(skill), Min-Inter (reward)*	6.96	**70.08**	4.41
4	*Max-Intra(duration), Max-Inter (skill)*	**6.98**	67.98	4.40
5	*no diversity*	*6.56*	*66.04*	*4.19*

Summary of Results. We present the average throughput, quality, and worker satisfaction for short sessions in Table 5.9 and long sessions in Table 5.10. 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 [39]. 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.*

5.4.2 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.

Data Sets. *a. 1-Million Song:* We use the Million Songs Dataset [19] that has 1 million songs (please note the Spotify dataset used in Section 5.4.1 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.

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[1]. The IP equivalent of the *Min-Intra,Max-Inter* problem is described in below. Other formulations (*Max-Intra,Min-Inter*, *Max-Intra,Max-Inter*, *Min-Intra,Min-Inter*) can be expressed similarly.

[1] *https://www.gurobi.com/resource/switching-from-open-source/ (accessed on Feb 1, 2020)*

$$\underset{\mathcal{G}}{\text{maximize}} \sum_{i=1}^{k} Inter(g_i)$$

$$\text{s.t.} \sum_{i=1}^{k} \sum_{j=1}^{l} (d_j^1 x_{ij} - \mu_i)^2 \leq opt$$

$$\sum_{j} x_{ij} = l, \forall i = 1 \ldots k$$

$$\sum_{i} x_{ij} = 1, \forall j = 1 \ldots l$$

$$x_{ij} \in \{0, 1\}$$

where d_j^1 is the value of the first dimension of the task j and *opt* is the value of optimal solution from `Ex-Min-Intra`.

`Baseline-MMR`. This baseline is inspired by the MMR algorithm [26] 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. We note that even though at a high level our studied problems may bear similarity with existing clustering problems, there is more than one significant difference that preclude the applicability of clustering algorithms as solutions to our problems. First and foremost, clustering algorithms do not allow to control the size of the sessions,

which is one of our key requirements. Second, clustering algorithms are not suitable to optimize sequence, which is one of our primary goals of *Inter* diversity. Finally, clustering algorithms cannot be adapted easily to solve multi-objective optimization problems, such as ours.

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.

Comparison against Optimal. Table 5.11 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.

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.

Table 5.11 Approximation Factors on 1-Million Songs Dataset

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

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 four problems we attempt to optimize.

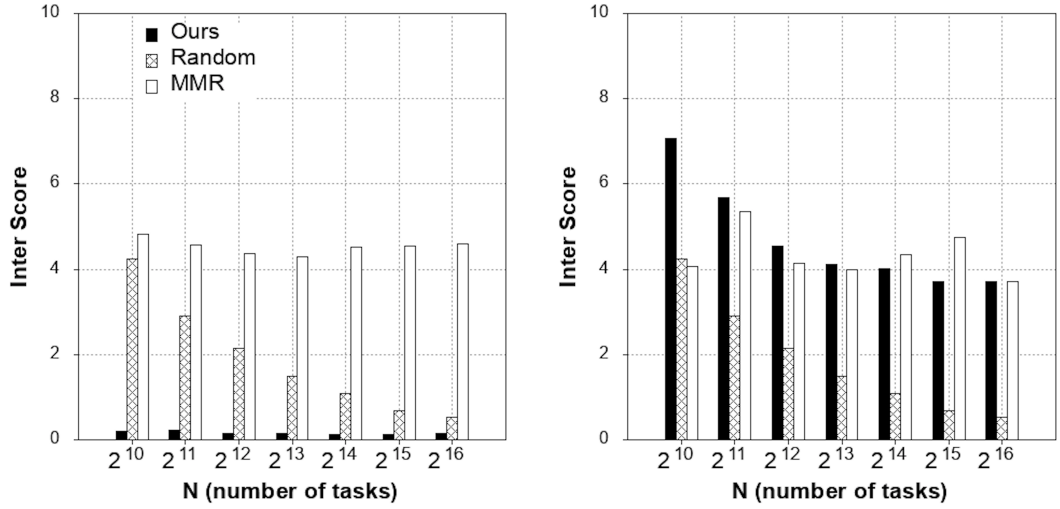
Varying N**.** Figure 5.5 shows how *Inter* scores change as we vary N from 2^{10} to 2^{16} for `Baseline-MMR`, `Random` and our algorithms. We have omitted the plots for Synthetic data experiments since those results closely follow the result for 1-Million Songs dataset. Figures 5.5(a)(c) confirm that our algorithm performs best when *Inter* diversity is minimized. The objective function improves with increasing N. On the other hand, as seen in Figures 5.5(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 5.12 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 5.12 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` 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 5.6 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 5.6(a)(c) that our algorithm performs significantly better than other baselines in minimizing *Inter* diversity. For Figures 5.6(b)(d), our observation is similar to the case of varying N, `Baseline-MMR` performs slightly better. Overall, *Inter* diversity increases for all four scenarios as k increases. This is because of the fact that when more sessions are present, it allows for more diversity between each session.



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

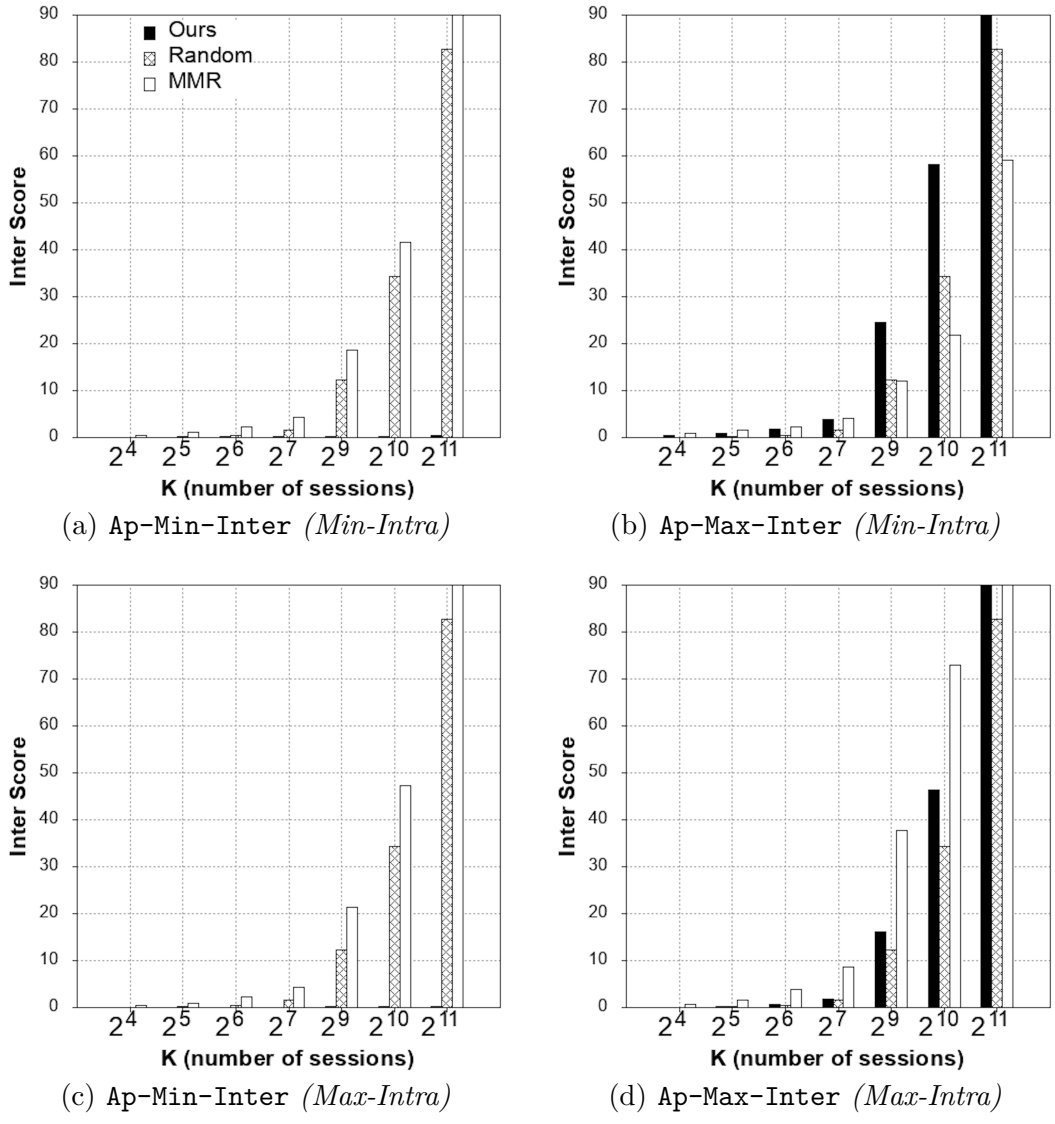
(c) `Ap-Min-Inter` *(Max-Intra)* (d) `Ap-Max-Inter` *(Max-Intra)*

Figure 5.5 *Inter* scores with varying N for 1-Million Song dataset.

We present approximation factors of *Intra* in Table 5.13 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 5.7 and 5.8 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 four scenarios and using a Zipf distribution produces a similar



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

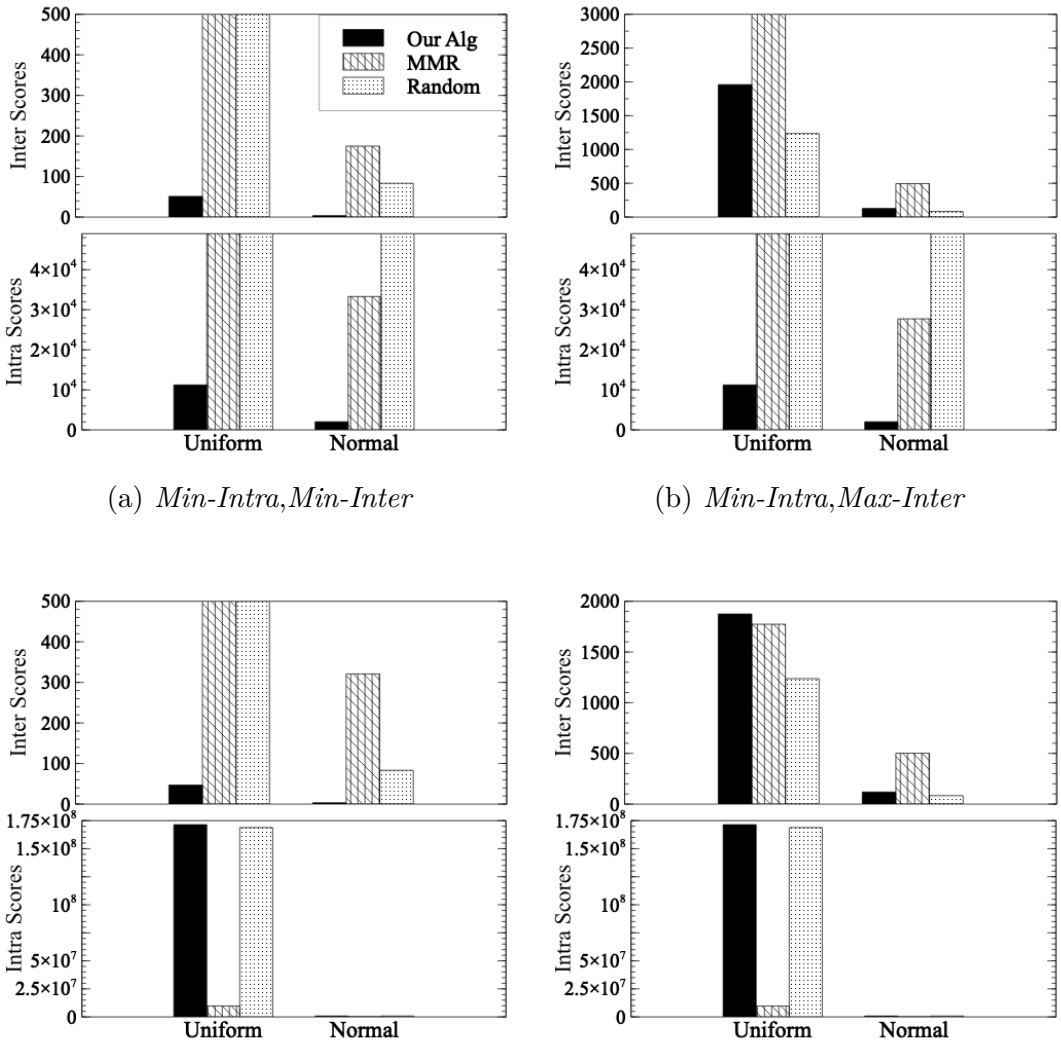
(c) `Ap-Min-Inter` *(Max-Intra)* (d) `Ap-Max-Inter` *(Max-Intra)*

Figure 5.6 *Inter* scores with varying k for 1-Million Songs dataset.

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 5.7(b)(d), `Baseline-MMR` outperforms our algorithm due to the same reasons mentioned in Section 5.4.2.

We also observe that across all four scenarios, Zipf produces scores much larger in magnitude than Normal or Uniform distribution. This is due to the range of values

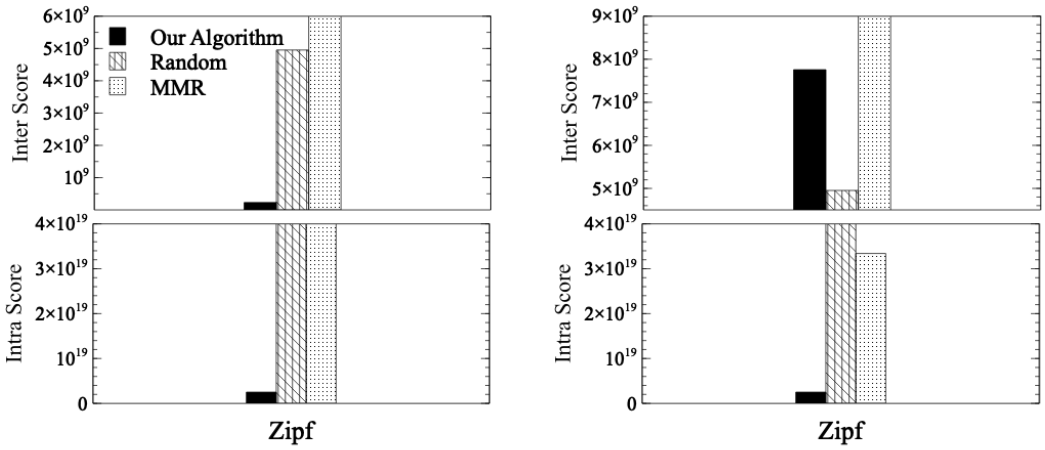


(a) *Min-Intra,Min-Inter* (b) *Min-Intra,Max-Inter*

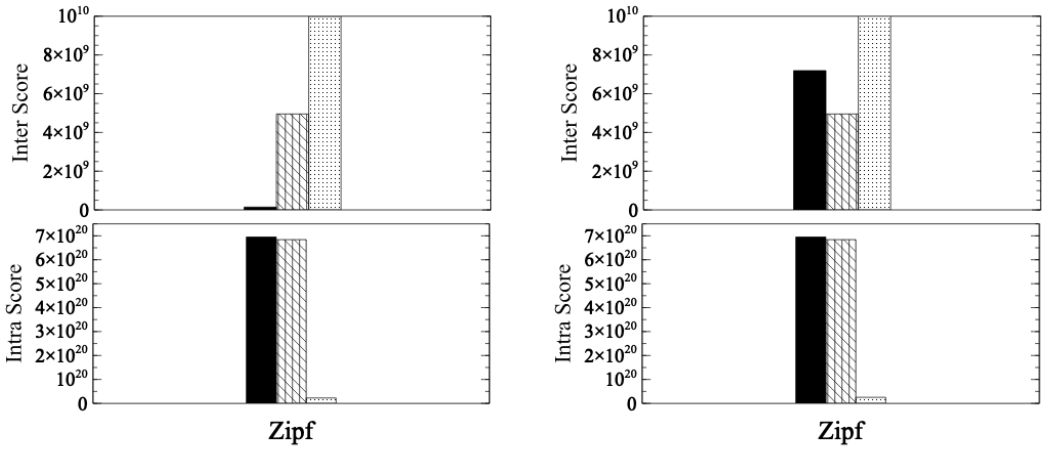
(c) *Max-Intra,Min-Inter* (d) *Max-Intra,Max-Inter*

Figure 5.7 Synthetc Data: *Inter* and *Intra* scores varying distributions.

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.



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



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

Figure 5.8 Synthetc Data: Zipf distribution.

Table 5.12 *Intra* Approximation Factors of The Three Algorithms Varying N on 1-Million Songs

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

Table 5.13 *Intra* Approximation Factors of The Three Algorithms Varying k on 1-Million Songs

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

Scalability Evaluation. Figures 5.9 and 5.10 compare the running time of the three algorithms for 1-Million dataset. Naturally, as N increases, the running time of our algorithm increases. We also observe that as we vary N with $k = 2^7$, our algorithm is the fastest in all diversity scenarios.

In Figures 5.10, 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.

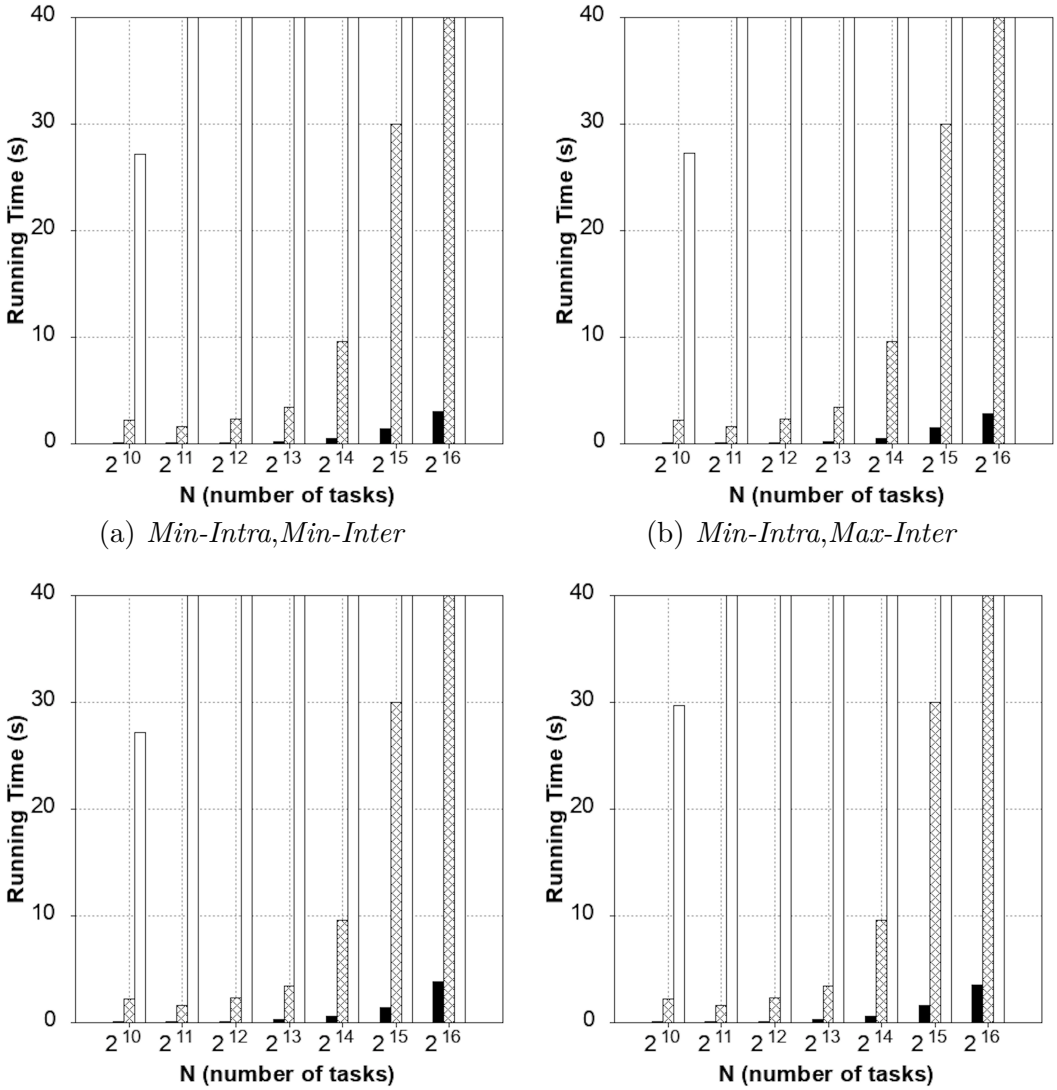
Overall, we find that our algorithms are highly scalable and produce results within a few seconds for very large values of N and k, while some of the baselines take hours to complete.

5.5 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.

Applications Diversity has been extensively studied in recommendation and search applications [16, 29, 1, 135, 131, 130, 123, 102, 91, 95, 44, 59, 60, 133, 105, 127], 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

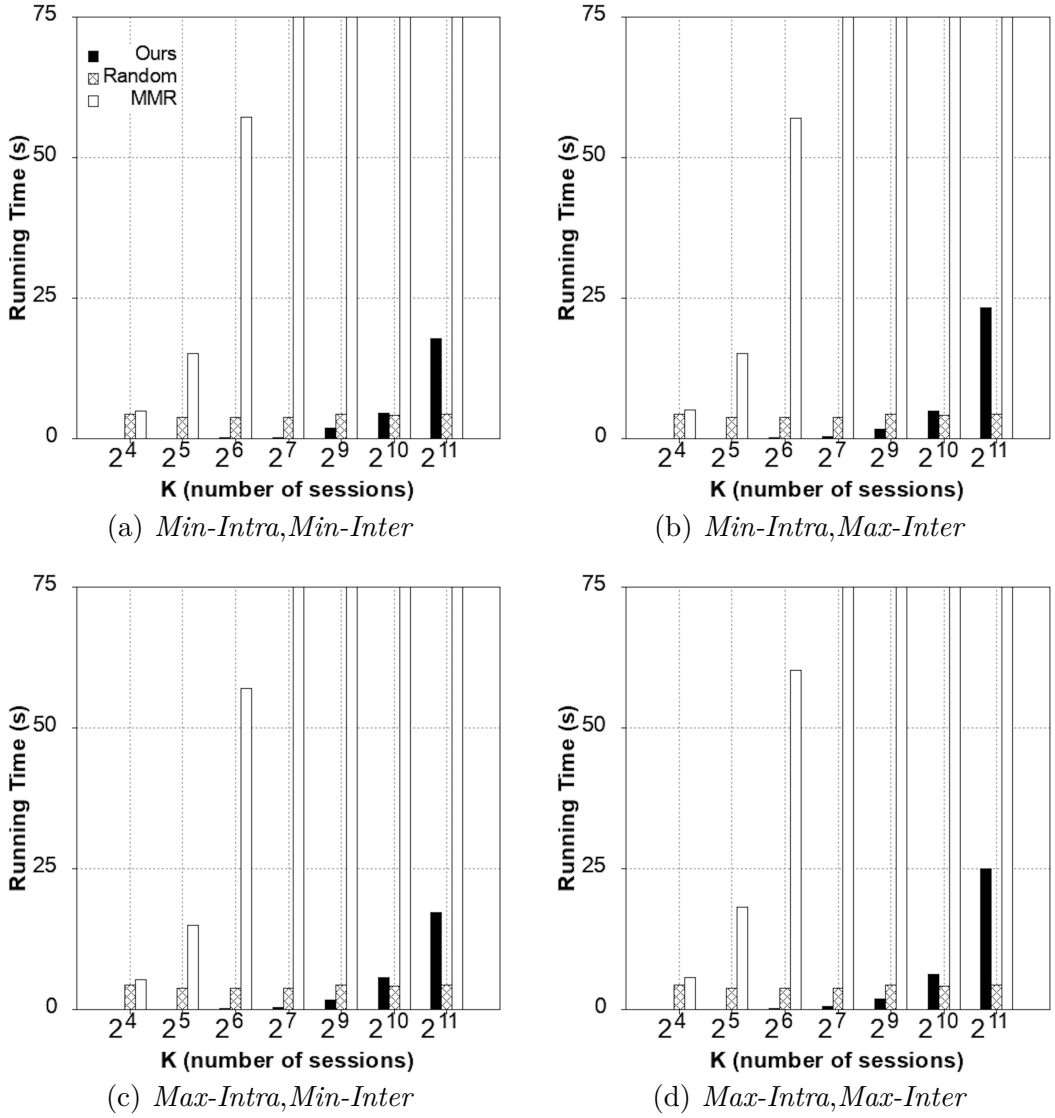


(a) *Min-Intra,Min-Inter* (b) *Min-Intra,Max-Inter*

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

Figure 5.9 Running times varying N for 1-Million Songs dataset.

heterogeneity. Existing works [55, 126] have also acknowledged the need for diversity and sequence based modeling in different recommendation applications. Recent works in crowdsourcing [45, 96] 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 [28]. Amer-Yahia et al. [9] 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



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

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

Figure 5.10 Running times varying k for 1-Million Songs dataset.

and inter-table influence in web table matching [45] involving crowd. Even though completing similar tasks lead to faster completion time [39], but such composition lead to fatigue and boredom, and task abandonment [57, 35, 53]. Aipe and Gadiraju[5] 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 [7]. 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 [1, 44, 91, 95, 123, 102] or sequence-based only [59, 16, 29, 135]. As an example, in [135], the authors study sequence-based diversity that is defined as the diversity of any permutation of the items. Another example is [16], in which taxonomies are used to sample search results to reduce homogeneity. In [1], the authors proposed an algorithm with a provable approximation factor to find relevant and diverse news articles. In the database context, Chen and Li [29] propose to post-process structured query results, organizing them in a decision tree for easier navigation. In [17, 66] 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 [42], recommendation diversity was formulated as a set-coverage 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.

This work opens up more than one research direction: 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.

CHAPTER 6

SUMMARY AND FUTURE WORK

Our goal in this dissertation is to *bring human back in the center stage of the computational loop and study optimization opportunities that arise to computationally solve these problems at scale.* In this chapter we summarize our contribution(Section 6.1), and explore some of the future areas of research (Section 6.2).

6.1 Summary

We start the discussion and why we need to change our focus to users in Chapter 1. We argue that looking at workers as mere agents without understating their needs, goals, and motivations will eventually result in subpar performance and lower engagement. We also outlined some of the recent research in this area and showcased how this new approach in the HIL system has increased different metrics in crowdsourcing environments, online collaborations, and massive online learning platforms.

`ExPref` framework in Chapter 2 is designed to study how one might include human factors in an optimization problem. We focus on the problem of predicting *Task Completion Time* which is an interesting and important metric both for worker and requester of a task. In Section 2.2, we dive deeper into different components of `ExPref` and introduce several NP-hard problems and the overall approach in which *Worker Model* works alongside *Question Selector* and *Preference Aggregator* to understand the preference of a worker. We showcase that by using *Question Selector* as our way of getting feedback from workers and *Preference Aggregator*, we successfully integrated explicit worker feedback into the *Worker Model*. In Section 2.3, we present `ActiveInit` as a method of initializing the *Worker Model*, `k-ExFactor` for choosing the best set of questions to ask from the worker. In Section 2.4, our extensive set of experiments showcases the power of `ExPref` in real platforms for real

workers. This gives us a great platform in which implicit and explicit feedbacks are combined.

Chapter 4 presents another equally important problem that arises in the context of peer learning. This problem is important for collaborative HIL problems, such as text editing or translation. Similar to Chapter 2, by looking at the affinity between peers and learning potential of a group, we intend to improve the skillset of the human workers while completing on tasks. In Section 4.2, we define four variants of the problem LP-* AFF-*. We show LP-* problems have polynomial-time algorithms and AFF-* problems are NP-hard. In Section 4.3.1, we solve four constraint optimization problems by setting the optimal value of LP-* problem as a constraint on AFF-*. We also run real data experiments on Amazon Mechanical Turk in Section 4.5.1 and explore the scalability and efficiency of the algorithms in Section 4.5.2. Overall, we show a statistically significant increase in the skill of a worker participating in a group that was formed by one of the four variations against different baselines.

Finally, after understanding user's preferences and knowing how to increase their skill set, we change our focus to recommending task sessions for the human workers such that their preferences are satisfied over important dimensions. Chapter refchapter:kdd explores such a problem in the context of the diversity of items in a session and the diversity of sequence of sessions. In Section 5.2 we introduced the problem of *Optimize-Intra, Optimize-Inter* which tries to optimize both *Intra* and *Inter* diversities simultaneously. It also presents efficient solutions for each problem if considered separately. Section 5.3.4 provides solutions for the combination of these problems. Our approximation algorithms are fast with theoretical bounds. Section 5.4 presents two sets of experiments. The first one explores two real-life applications of these problems and measures user satisfaction and performance. We show that that user satisfaction in diversified playlists is higher compared to the *no diversity* baseline. We also measure average throughput, quality and worker satisfaction in a

crowdsourcing environment and our experiments show that the benefit of diversity in task recommendation is more prominent for sessions comprising many tasks. Diversity tends to bring a positive effect to avoid boredom which is prominent for sessions with many tasks. Section 5.4.2 presents quality experiments that aim to measure the approximation factor and scalability of the proposed algorithms. 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.

6.2 Future Work

This dissertation opens up many interesting directions. The explicit feedback approach can learn user preferences more accurately and immediately by asking users directly but it is not always obvious what questions to ask in order to best capture user preferences. Some factors are explicit such as the task category, but other factors may be latent or hard to express, such as user fatigue or mood. Furthermore, asking for explicit user feedback may overwhelm users, who may provide sparse input or even get driven away from the system. *On the other hand*, the implicit feedback approach can more easily capture latent user preferences, but it also has its own shortcomings. As user preferences change, an implicit worker model will adapt more slowly.

The above brings us to an interesting research question: *How can we combine explicit and implicit feedback in a worker model leveraging the best of both worlds*? This problem is challenging:

- The relationship between worker's preference and tasks is not straight-forward. Workers may prefer attributes that are not defined in tasks, such as fatigue or enjoyment.
- Latent variables such as fatigue, skill improvements are usually very hard to deal with.

- Explicit feedback may be in the form of ratings on specific factors, while implicit feedback may be expressed differently, e.g. using latent factors. Then, it is not straightforward how we can effectively combine explicit and implicit feedback. For example, in Chapter 2 we use a linear regression model to combine explicit and implicit feedback by introducing the feedback as a set of constraints on the linear model and finding a solution that satisfies all the constraints. Although this is an interesting solution there are a few issues with this. Firstly, these linear constraints will make the optimization problem more complex. Secondly, it's not clear what to do with the previous constraints. Lastly, the set of questions that they ask is on the factors that make the model worst which makes sense but it's not clear why adding constraints between those factors will make the model perform better in general.

How to use this preference model for task recommendation: As the number of tasks uploaded by requesters daily can be large, the crowdsourcing platform should aim to recommend tasks to each worker that he will likely accept and complete as close to the expected time as possible in order to optimize the platform productivity. On the other hand, workers desire the top-k best-fitting tasks being promptly and effectively recommended to them. Others have looked into this problem from the worker's point of view [27]. As an example,[27], use a linear combination of task factors a worker factors to create a model for assigning tasks to workers whose completed tasks have the best quality.

Ideally, given that we can predict the task completion time for each worker and task, we can recommend tasks to a user such that their deadlines (i.e., expected completion time as given by the requester) are as close as possible to the predicted task completion time for the worker and task.

However, if for each worker we find the best tasks based on what we described above, our recommendation model needs to be fair in two aspects: (a) making sure that all tasks are recommended to at least a users and (b) making sure that all users are recommended at least b tasks.

When to interview a user: A problem that naturally arises is: when do we interview the user to refresh our knowledge on explicit user preferences? User preferences change in a crowdsourcing system, and it is important that we can

quickly adapt to such changes. Any model that only relies on implicit feedback has to collect enough "evidence" to "sense" a shift in user preferences and change its output. Refreshing our explicit preferences knowledge can help us adapt faster and more smoothly. The challenges that we have here are: (a) We do not want to invoke too often because users will be annoyed, (b) We do not want to invoke too rarely because preferences change and we will have a stale explicit model.

In diversifying recommendations on sequences of sets we explored diversifying a set of items based on two dimensions. An immediate extension of this work is to generalize these problems on a variable number of dimensions. One might ask the following research questions: *How can we optimize diversity on an arbitrary number of dimensions*?

- The immediate consequence of this question is that *Min-Intra* problem becomes NP-hard and akin to a clustering problem.
- Since the problem is multi-dimensional, the choice of distance metric becomes tangled to the use case. For example, diversifying a set of songs might react differently than diversifying a set of documents.

REFERENCES

[1] Sofiane Abbar, Sihem Amer-Yahia, Piotr Indyk, and Sepideh Mahabadi. Real-time recommendation of diverse related articles. In *Proceedings of the 22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil*, pages 1–12, 2013.

[2] Rakesh Agrawal, Behzad Golshan, and Evangelos E. Papalexakis. Toward data-driven design of educational courses: A feasibility study. In *Proceedings of the 9th International Conference on Educational Data Mining, EDM 2016, Raleigh, North Carolina, USA*, page 6, 2016.

[3] Rakesh Agrawal, Behzad Golshan, and Evimaria Terzi. Grouping students in educational settings. In *Proceedings of the 20th SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA*, pages 1017–1026, 2014.

[4] Rakesh Agrawal, Sharad Nandanwar, and Narasimha Murty Musti. Grouping students for maximizing learning from peers. In *Proceedings of the 10th International Conference on Educational Data Mining, EDM 2017, Wuhan, Hubei, China*, 2017.

[5] Alan Aipe and Ujwal Gadiraju. Similarhits: Revealing the role of task similarity in microtask crowdsourcing. In *Proceedings of the 29th on Hypertext and Social Media, HT 2018, Baltimore, MD, USA*, pages 115–122, 2018.

[6] Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermüller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, Alexander Belopolsky, Yoshua Bengio, Arnaud Bergeron, James Bergstra, Valentin Bisson, Josh Bleecher Snyder, Nicolas Bouchard, Nicolas Boulanger-Lewandowski, Xavier Bouthillier, Alexandre de Brébisson, Olivier Breuleux, Pierre Luc Carrier, Kyunghyun Cho, Jan Chorowski, Paul F. Christiano, Tim Cooijmans, Marc-Alexandre Côté, Myriam Côté, Aaron C. Courville, Yann N. Dauphin, Olivier Delalleau, Julien Demouth, Guillaume Desjardins, Sander Dieleman, Laurent Dinh, Melanie Ducoffe, Vincent Dumoulin, Samira Ebrahimi Kahou, Dumitru Erhan, Ziye Fan, Orhan Firat, Mathieu Germain, Xavier Glorot, Ian J. Goodfellow, Matthew Graham, Çaglar Gülçehre, Philippe Hamel, Iban Harlouchet, Jean-Philippe Heng, Balázs Hidasi, Sina Honari, Arjun Jain, Sébastien Jean, Kai Jia, Mikhail Korobov, Vivek Kulkarni, Alex Lamb, Pascal Lamblin, Eric Larsen, César Laurent, Sean Lee, Simon Lefrançois, Simon Lemieux, Nicholas Léonard, Zhouhan Lin, Jesse A. Livezey, Cory Lorenz, Jeremiah Lowin, Qianli Ma, Pierre-Antoine Manzagol, Olivier Mastropietro, Robert McGibbon, Roland Memisevic, Bart van Merriënboer, Vincent Michalski, Mehdi Mirza, Alberto Orlandi, Christopher Joseph Pal, Razvan Pascanu, Mohammad Pezeshki, Colin Raffel, Daniel Renshaw, Matthew Rocklin, Adriana Romero, Markus Roth, Peter Sadowski, John Salvatier, François Savard, Jan Schlüter, John Schulman, Gabriel Schwartz, Iulian Vlad Serban, Dmitriy Serdyuk, Samira Shabanian, Étienne Simon, Sigurd Spieckermann, S. Ramana Subramanyam, Jakub Sygnowski, Jérémie Tanguay, Gijs van Tulder, Joseph P. Turian, Sebastian Urban, Pascal Vincent, Francesco Visin, Harm de Vries, David Warde-Farley, Dustin J. Webb, Matthew Willson, Kelvin Xu, Lijun Xue, Li Yao, Saizheng Zhang, and Ying Zhang. Theano: A python framework for fast computation of mathematical expressions. *Computing Research Repository*, abs/1605.02688, 2016.

[7] Maha Alsayasneh, Sihem Amer-Yahia, Éric Gaussier, Vincent Leroy, Julien Pilourdault, Ria Mae Borromeo, Motomichi Toyama, and Jean-Michel Renders. Personalized and diverse task composition in crowdsourcing. *IEEE Transactions on Knowledge and Data Engineering*, 30(1):128–141, 2018.

[8] Robert A. Amar, James Eagan, and John T. Stasko. Low-level components of analytic activity in information visualization. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis 2005), Minneapolis, MN, USA*, pages 111–117, 2005.

[9] Sihem Amer-Yahia, Éric Gaussier, Vincent Leroy, Julien Pilourdault, Ria Mae Borromeo, and Motomichi Toyama. Task composition in crowdsourcing. In *Proceedings of the 2016 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2016, Montreal, QC, Canada*, pages 194–203, 2016.

[10] Sihem Amer-Yahia, Sofia Kleisarchaki, Naresh Kumar Kolloju, Laks V. S. Lakshmanan, and Ruben H. Zamar. Exploring rated datasets with rating maps. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia*, pages 1411–1419, 2017.

[11] Sihem Amer-Yahia and Senjuti Basu Roy. Human factors in crowdsourcing. *Proceedings of the VLDB Endowment*, 9(13):1615–1618, 2016.

[12] Yael Amsterdamer, Susan B. Davidson, Tova Milo, Slava Novgorodov, and Amit Somech. OASSIS: query driven crowd mining. In *Proceedings of the International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA*, pages 589–600, 2014.

[13] Yael Amsterdamer, Yael Grossman, Tova Milo, and Pierre Senellart. Crowdminer: Mining association rules from the crowd. *Proceedings of the VLDB Endowment*, 6(12):1250–1253, 2013.

[14] Aris Anagnostopoulos, Luca Becchetti, Carlos Castillo, Aristides Gionis, and Stefano Leonardi. Power in unity: forming teams in large-scale community systems. In *Proceedings of the 19th Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada*, pages 599–608, 2010.

[15] Aris Anagnostopoulos, Luca Becchetti, Carlos Castillo, Aristides Gionis, and Stefano Leonardi. Online team formation in social networks. In *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France*, pages 839–848, 2012.

[16] Aris Anagnostopoulos, Andrei Z. Broder, and David Carmel. Sampling search-engine results. In *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan*, pages 245–256, 2005.

[17] Albert Angel and Nick Koudas. Efficient diversity-aware search. In *Proceedings of the SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece*, pages 781–792, 2011.

[18] Haim Avron and Christos Boutsidis. Faster subset selection for matrices and applications. *SIAM Journal on Matrix Analysis and Applications*, 34(4):1464–1499, 2013.

[19] Thierry Bertin-Mahieux, Daniel P. W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA*, pages 591–596, 2011.

[20] Indrajit Bhattacharya and Lise Getoor. Collective entity resolution in relational data. *IEEE Transactions on Knowledge and Data Engineering*, 1(1):5, 2007.

[21] Sampoorna Biswas, Laks V. S. Lakshmanan, and Senjuti Basu Roy. Combating the cold start user problem in model based collaborative filtering. *Computing Research Repository*, abs/1703.00397, 2017.

[22] Bruno Bouzy and Tristan Cazenave. Computer go: An AI oriented survey. *Artificial Intelligence*, 132(1):39–103, 2001.

[23] Barry Bozeman and Mary K Feeney. Toward a useful theory of mentoring: A conceptual analysis and critique. *Administration & society*, 39(6):719–739, 2007.

[24] Iker Burguera, Urko Zurutuza, and Simin Nadjm-Tehrani. Crowdroid: behavior-based malware detection system for android. In *Proceedings of the 1st Workshop Security and Privacy in Smartphones and Mobile Devices*, pages 15–26, 2011.

[25] HongYun Cai, Vincent Wenchen Zheng, Fanwei Zhu, Kevin Chen-Chuan Chang, and Zi Huang. From community detection to community profiling. *Proceedings of the VLDB Endowment*, 10(7):817–828, 2017.

[26] Jaime Carbinell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 51(2):335–336, 1998.

[27] Silvana Castano, Alfio Ferrara, and Stefano Montanelli. Crowdsourcing task assignment with online profile learning. In *On the Move to Meaningful Internet Systems. OTM 2018 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2018, Valletta, Malta*, volume 11229 of *Lecture Notes in Computer Science*, pages 226–242, 2018.

[28] Dana Chandler and Adam Kapelner. Breaking monotony with meaning: Motivation in crowdsourcing markets. *Computing Research Repository*, abs/1210.0962, 2012.

[29] Zhiyuan Chen and Tao Li. Addressing diverse user preferences in sql-query-result navigation. In *Proceedings of the International Conference on Management of Data, SIGMOD 2007, Beijing, China*, pages 641–652, 2007.

[30] Peter Christen. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Data-Centric Systems and Applications. 2012.

[31] Mark Cieliebak, Stephan J. Eidenbenz, Aris Pagourtzis, and Konrad Schlude. On the complexity of variations of equal sum subsets. *Nordic Journal of Computing*, 14(3):151–172, 2008.

[32] Adam Coates, Andrew Y. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15, pages 215–223, 2011.

[33] Elizabeth G Cohen. Restructuring the classroom: Conditions for productive small groups. *Review of educational research*, 1994.

[34] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. In *Proceedings of the Advances in Neural Information Processing Systems 7, NIPS Conference, Denver, Colorado, USA*, pages 705–712, 1994.

[35] Peng Dai, Jeffrey M. Rzeszotarski, Praveen Paritosh, and Ed H. Chi. And now for something completely different: Improving crowdsourcing workflows with micro-diversions. In *Proceedings of the 18th Conference on Computer Supported Cooperative Work & Social Computing, CSCW 2015, Vancouver, BC, Canada, March 14 - 18, 2015*, pages 628–638, 2015.

[36] Thanasis Daradoumis, Montse Guitert, Ferran Giménez, Joan Manuel Marquès, and T. Lloret. Supporting the composition of effective virtual groups for collaborative learning. In *Proceedings of the International Conference on Computers in Education, ICCE 2002, Auckland, New Zealand*, pages 332–336, 2002.

[37] Susan B. Davidson, Sanjeev Khanna, Tova Milo, and Sudeepa Roy. Top-k and clustering with noisy comparisons. *ACM Transactions on Database Systems (TODS).*, 39(4):35:1–35:39, 2014.

[38] Phillip Dawson. Beyond a definition: Toward a framework for designing and specifying mentoring models. *Educational Researcher*, 43(3):137–145, 2014.

[39] Djellel Eddine Difallah, Michele Catasta, Gianluca Demartini, and Philippe Cudré-Mauroux. Scaling-up the crowd: Micro-task pricing schemes for worker retention and latency improvement. In *Proceedings of the Second AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2014*, 2014.

[40] Djellel Eddine Difallah, Gianluca Demartini, and Philippe Cudré-Mauroux. Pick-a-crowd: tell me what you like, and i'll tell you what to do. In *Proceedings of the 22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil*, pages 367–374, 2013.

[41] Djellel Eddine Difallah, Elena Filatova, and Panos Ipeirotis. Demographics and dynamics of mechanical turk workers. In *Proceedings of the Eleventh International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*, pages 135–143, 2018.

[42] Khalid El-Arini, Gaurav Veda, Dafna Shahaf, and Carlos Guestrin. Turning down the noise in the blogosphere. In *Proceedings of the 15th SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France*, pages 289–298, 2009.

[43] Sarah Evans, Katie Davis, Abigail Evans, Julie Ann Campbell, David P. Randall, Kodlee Yin, and Cecilia R. Aragon. More than peer production: Fanfiction communities as sites of distributed mentoring. In *Proceedings of the 2017 Conference on Computer Supported Cooperative Work and Social Computing, CSCW 2017, Portland, OR, USA, February 25 - March 1, 2017*, pages 259–272, 2017.

[44] Ju Fan, Guoliang Li, Beng Chin Ooi, Kian-Lee Tan, and Jianhua Feng. icrowd: An adaptive crowdsourcing framework. In *Proceedings of the 2015 SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia*, pages 1015–1030, 2015.

[45] Ju Fan, Meiyu Lu, Beng Chin Ooi, Wang-Chiew Tan, and Meihui Zhang. A hybrid machine-crowdsourcing system for matching web tables. In *Proceedings of the IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, pages 976–987, 2014.

[46] Siamak Faradani, Bjoern Hartmann, and Panagiotis G. Ipeirotis. What's the right price? pricing tasks for finishing on time. In *Human Computation, Papers from the 2011 AAAI Workshop, San Francisco, California, USA*, volume WS-11-11 of *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[47] Michael J. Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. Crowddb: answering queries with crowdsourcing. In *Proceedings of the International Conference on Management of Data, SIGMOD 2011, Athens, Greece*, pages 61–72, 2011.

[48] Yihan Gao and Aditya G. Parameswaran. Finish them!: Pricing algorithms for human computation. *Proceedings of the VLDB Endowment*, 7(14):1965–1976, 2014.

[49] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. 1979.

[50] Benoît Groz and Tova Milo. Skyline queries with noisy comparisons. In *Proceedings of the 34th Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia*, pages 185–198, 2015.

[51] Stephen Guo, Aditya G. Parameswaran, and Hector Garcia-Molina. So who won?: dynamic max discovery with the crowd. In *Proceedings of the International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA*, pages 385–396, 2012.

[52] Gregory Gutin and Abraham P Punnen. *The traveling salesman problem and its variations*, volume 12. 2006.

[53] Lei Han, Kevin Roitero, Ujwal Gadiraju, Cristina Sarasua, Alessandro Checco, Eddy Maddalena, and Gianluca Demartini. All those wasted hours: On task abandonment in crowdsourcing. In *Proceedings of the Twelfth International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, pages 321–329, 2019.

[54] Benjamin V. Hanrahan, Jutta K. Willamowski, Saiganesh Swaminathan, and David B. Martin. Turkbench: Rendering the market for turkers. In *Proceedings of the 33rd Annual Conference on Human Factors in Computing Systems, CHI 2015, Seoul, Republic of Korea, April 18-23, 2015*, pages 1613–1616, 2015.

[55] Negar Hariri, Bamshad Mobasher, and Robin D. Burke. Context-aware music recommendation based on latenttopic sequential patterns. In *Proceedings of the Sixth Conference on Recommender Systems, RecSys '12, Dublin, Ireland*, pages 131–138, 2012.

[56] Dap Hartmann. How computers play chess. *International Computer Games Association*, 14(2):79–80, 1991.

[57] Kenji Hata, Ranjay Krishna, Fei-Fei Li, and Michael S. Bernstein. A glimpse far into the future: Understanding long-term crowd worker quality. In *Proceedings of the 2017 Conference on Computer Supported Cooperative Work and Social Computing, CSCW 2017, Portland, OR, USA, February 25 - March 1, 2017*, pages 889–901, 2017.

[58] Arthur V Hill. An experimental comparison of human schedulers and heuristic algorithms for the traveling salesman problem. *Journal of Operations Management*, 2(4):215–223, 1982.

[59] Chien-Ju Ho, Shahin Jabbari, and Jennifer Wortman Vaughan. Adaptive task assignment for crowdsourced classification. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 534–542, 2013.

[60] Chien-Ju Ho and Jennifer Wortman Vaughan. Online task assignment in crowdsourcing markets. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, Toronto, Ontario, Canada*, 2012.

[61] Chiung Ching Ho and Choo-Yee Ting. Measuring crowd sourced analytics: A review. *Inter-national Information Institute (Tokyo). Information*, 19(10B):4891, 2016.

[62] John Joseph Horton and Lydia B. Chilton. The labor economics of paid crowdsourcing. In *Proceedings 11th Conference on Electronic Commerce (EC-2010), Cambridge, Massachusetts, USA*, pages 209–218, 2010.

[63] Russell L Huizing. Mentoring together: A literature review of group mentoring. *Mentoring & Tutoring: Partnership in Learning*, 20(1):27–55, 2012.

[64] Kosetsu Ikeda, Atsuyuki Morishima, Habibur Rahman, Senjuti Basu Roy, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. Collaborative crowdsourcing with crowd4u. *Proceedings of the VLDB Endowment*, 9(13):1497–1500, 2016.

[65] Lilly Irani and M. Six Silberman. Turkopticon: interrupting worker invisibility in amazon mechanical turk. In *Proceedings of the 2013 SIGCHI Conference on Human Factors in Computing Systems, CHI '13, Paris, France*, pages 611–620, 2013.

[66] Anoop Jain, Parag Sarda, and Jayant R. Haritsa. Providing diversity in k-nearest neighbor query results. In *Proceedings of the Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference, PAKDD 2004, Sydney, Australia*, volume 3056 of *Lecture Notes in Computer Science*, pages 404–413, 2004.

[67] Prasanth Jayachandran, Karthik Tunga, Niranjan Kamat, and Arnab Nandi. Combining user interaction, speculative query execution and sampling in the DICE system. *Proceedings of the VLDB Endowment*, 7(13):1697–1700, 2014.

[68] Nicolas Kaufmann, Thimo Schulze, and Daniel Veit. More than fun and money. worker motivation in crowdsourcing - A study on mechanical turk. In *Proceedings of the 17th Americas Conference on Information Systems, AMCIS 2011, Detroit, Michigan, USA*, 2011.

[69] Aniket Kittur, Boris Smus, and Robert E. Kraut. Crowdforge: crowdsourcing complex work. In *Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Extended Abstracts Volume, Vancouver, BC, Canada*, pages 1801–1806, 2011.

[70] Pradap Konda, Sanjib Das, Paul Suganthan G. C., AnHai Doan, Adel Ardalan, Jeffrey R. Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeffrey F. Naughton, Shishir Prasad, Ganesh Krishnan, Rohit Deep, and Vijay Raghavendra. Magellan: Toward building entity matching management systems. *Proceedings of the VLDB Endowment*, 9(12):1197–1208, 2016.

[71] Anand Kulkarni, Philipp Gutheim, Prayag Narula, David Rolnitzky, Tapan S. Parikh, and Björn Hartmann. Mobileworks: Designing for quality in a managed crowdsourcing architecture. *IEEE Internet Computing*, 16(5):28–35, 2012.

[72] Theodoros Lappas, Kun Liu, and Evimaria Terzi. Finding a team of experts in social networks. In *Proceedings of the 15th SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France*, pages 467–476, 2009.

[73] Isabella Lari, Federica Ricca, Justo Puerto, and Andrea Scozzari. Partitioning a graph into connected components with fixed centers and optimizing cost-based objective functions or equipartition criteria. *Networks*, 67(1):69–81, 2016.

[74] Charles Eric Leiserson, Ronald L Rivest, Thomas H Cormen, and Clifford Stein. *Introduction to algorithms*, volume 6. 2001.

[75] Joseph CR Licklider. Man-computer symbiosis. *IRE transactions on human factors in electronics*, (1):4–11, 1960.

[76] Albert Yu-Min Lin, Andrew Huynh, Gert Lanckriet, and Luke Barrington. Crowdsourcing the unknown: The satellite search for genghis khan. *PLoS ONE*, 9(12):e114046, 2014.

[77] Xuan Liu, Meiyu Lu, Beng Chin Ooi, Yanyan Shen, Sai Wu, and Meihui Zhang. CDAS: A crowdsourcing data analytics system. *Proceedings of the VLDB Endowment*, 5(10):1040–1051, 2012.

[78] James N. MacGregor and Yun Chu. Human performance on the traveling salesman and related problems: A review. *The Journal of Problem Solving*, 3(2), 2011.

[79] Anirban Majumder, Samik Datta, and K. V. M. Naidu. Capacitated team formation problem on social networks. In *Proceedings of the The 18th International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China*, pages 1005–1013, 2012.

[80] Andrew Mao, Ece Kamar, and Eric Horvitz. Why stop now? predicting worker engagement in online crowdsourcing. In *Proceedings of the First Conference on Human Computation and Crowdsourcing, HCOMP 2013, Palm Springs, CA, USA*, 2013.

[81] Adam Marcus, Eugene Wu, David R. Karger, Samuel Madden, and Robert C. Miller. Human-powered sorts and joins. *Proceedings of the VLDB Endowment*, 5(1):13–24, 2011.

[82] Adam Marcus, Eugene Wu, Samuel Madden, and Robert C. Miller. Crowdsourced databases: Query processing with people. In *CIDR 2011, Fifth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 9-12, 2011, Online Proceedings*, pages 211–214, 2011.

[83] David B. Martin, Benjamin V. Hanrahan, Jacki O'Neill, and Neha Gupta. Being a turker. In *Computer Supported Cooperative Work, CSCW '14, Baltimore, MD, USA, February 15-19, 2014*, pages 224–235, 2014.

[84] Mark P Mattson. Superior pattern processing is the essence of the evolved human brain. *Frontiers in neuroscience*, 8, 2014.

[85] Arya Mazumdar and Barna Saha. Clustering via crowdsourcing. *Computing Research Repository*, abs/1604.01839, 2016.

[86] Arya Mazumdar and Barna Saha. Clustering with noisy queries. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5788–5799, 2017.

[87] Jodi L Mead and Rosemary A Renaut. Least squares problems with inequality constraints as quadratic constraints. *Linear Algebra and its Applications*, 432(8):1936–1949, 2010.

[88] Walaa Medhat, Ahmed Hassan, and Hoda Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, 5(4):1093–1113, 2014.

[89] Wil Michiels, Jan H. M. Korst, Emile H. L. Aarts, and Jan van Leeuwen. Performance ratios for the differencing method applied to the balanced number partitioning problem. In *STACS 2003, 20th Annual Symposium on Theoretical Aspects of Computer Science, Berlin, Germany, February 27 - March 1, 2003, Proceedings*, volume 2607 of *Lecture Notes in Computer Science*, pages 583–595, 2003.

[90] Vicente Rodríguez Montequín, Joaquín Villanueva Balsera, José Manuel Mesa Fernández, and Javier De Cos Juez. Using myers-briggs type indicator (MBTI) for assessment success of student groups in project based learning. In *Proceedings of the Second International Conference on Computer Supported Education, Valencia, Spain, April 7-10, 2010 - Volume 2*, pages 156–160, 2010.

[91] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Math. Program.*, 14(1):265–294, 1978.

[92] Allen Newell, John Calman Shaw, and Herbert A Simon. Chess-playing programs and the problem of complexity. *IBM Journal of Research and Development*, 2(4):320–335, 1958.

[93] JD North. The rational behavior of mechanically extended man. *Boulton Paul Aircraft Ltd., Wolverhampton, Eng*, 1954.

[94] Behrooz Omidvar-Tehrani, Sihem Amer-Yahia, and Alexandre Termier. Interactive user group analysis. In *Proceedings of the 24th International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia*, pages 403–412, 2015.

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

[96] Julien Pilourdault, Sihem Amer-Yahia, Dongwon Lee, and Senjuti Basu Roy. Motivation-aware task assignment in crowdsourcing. In *Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017*, pages 246–257, 2017.

[97] Julien Pilourdault, Sihem Amer-Yahia, Senjuti Basu Roy, and Dongwon Lee. Task relevance and diversity as worker motivation in crowdsourcing. In *Proceedings of the 34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*, pages 365–376, 2018.

[98] Vassilis Polychronopoulos, Luca de Alfaro, James Davis, Hector Garcia-Molina, and Neoklis Polyzotis. Human-powered top-k lists. In *Proceedings of the 16th International Workshop on the Web and Databases 2013, WebDB 2013, New York, NY, USA*, pages 25–30, 2013.

[99] Friedrich Pukelsheim. *Optimal design of experiments*. 2006.

[100] Abraham P. Punnen, François Margot, and Santosh N. Kabadi. TSP heuristics: Domination analysis and complexity. *Algorithmica*, 35(2):111–127, 2003.

[101] Kun Qian, Lucian Popa, and Prithviraj Sen. Systemer: A human-in-the-loop system for explainable entity resolution. *Proceedings of the VLDB Endowment*, 12(12):1794–1797, 2019.

[102] Lijing Qin and Xiaoyan Zhu. Promoting diversity in recommendation by entropy regularizer. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China*, pages 2698–2704, 2013.

[103] Habibur Rahman, Senjuti Basu Roy, and Gautam Das. A probabilistic framework for estimating pairwise distances through crowdsourcing. In *Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017*, pages 258–269, 2017.

[104] Habibur Rahman, Senjuti Basu Roy, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. Task assignment optimization in collaborative crowdsourcing. In *Proceedings of the 2015 IEEE International Conference on Data Mining, ICDM 2015, Atlantic City, NJ, USA*, pages 949–954, 2015.

[105] Habibur Rahman, Senjuti Basu Roy, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. Optimized group formation for solving collaborative tasks. *The International Journal on Very Large Data Bases*, 28(1):1–23, 2019.

[106] Habibur Rahman, Saravanan Thirumuruganathan, Senjuti Basu Roy, Sihem Amer-Yahia, and Gautam Das. Worker skill estimation in team-based tasks. *Proceedings of the VLDB Endowment*, 8(11):1142–1153, 2015.

[107] Syama Sundar Rangapuram, Thomas Bühler, and Matthias Hein. Towards realistic team formation in social networks based on densest subgraphs. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil*, pages 1077–1088, 2013.

[108] Christopher Ré, Nilesh N. Dalvi, and Dan Suciu. Efficient top-k query evaluation on probabilistic data. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, pages 886–895, 2007.

[109] Jakob Rogstadius, Vassilis Kostakos, Aniket Kittur, Boris Smus, Jim Laredo, and Maja Vukovic. An assessment of intrinsic and extrinsic motivation on task performance in crowdsourcing markets. In *Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain*, 2011.

[110] Senjuti Basu Roy, Laks V. S. Lakshmanan, and Rui Liu. From group recommendations to group formation. In *Proceedings of the 2015 SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia*, pages 1603–1616, 2015.

[111] Senjuti Basu Roy, Ioanna Lykourentzou, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. Crowds, not drones: Modeling human factors in interactive crowdsourcing. In *Proceedings of the First VLDB Workshop on Databases and Crowdsourcing, DBCrowd 2013, Riva del Garda, Trento, Italy*, volume 1025, pages 39–42, 2013.

[112] Senjuti Basu Roy, Ioanna Lykourentzou, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. Task assignment optimization in knowledge-intensive crowdsourcing. *The International Journal on Very Large Data Bases*, 24(4):467–491, 2015.

[113] Jeffrey M Rzeszotarski, Ed Chi, Praveen Paritosh, and Peng Dai. Inserting micro-breaks into crowdsourcing workflows. In *First AAAI Conference on Human Computation and Crowdsourcing*, 2013.

[114] Tim Schlippe, Chenfei Zhu, Daniel Lemcke, and Tanja Schultz. Statistical machine translation based text normalization with crowdsourcing. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada*, pages 8406–8410, 2013.

[115] Avi Segal, Ya'akov (Kobi) Gal, Ece Kamar, Eric Horvitz, Alex Bowyer, and Grant Miller. Intervention strategies for increasing engagement in crowdsourcing: Platform, predictions, and experiments. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 3861–3867, 2016.

[116] Aaron D. Shaw, John J. Horton, and Daniel L. Chen. Designing incentives for inexpert human raters. In *Proceedings of the 2011 Conference on Computer Supported Cooperative Work, CSCW 2011, Hangzhou, China, March 19-23, 2011*, pages 275–284, 2011.

[117] Jianhua Shen, Joseph Barbera, and Colin M Shapiro. Distinguishing sleepiness and fatigue: focus on definition and measurement. *Sleep medicine reviews*, 10(1):63–76, 2006.

[118] Julio M. Singer. Central limit theorems. In *International Encyclopedia of Statistical Science*, pages 228–234. 2011.

[119] Ivan Srba and Mária Bieliková. Dynamic group formation as an approach to collaborative learning support. *IEEE Transactions on Learning Technologies*, 8(2):173–186, 2015.

[120] Philip B Stark and Robert L Parker. Bounded-variable least-squares: an algorithm and applications. *Computational Statistics*, 10:129–129, 1995.

[121] Michael R Stoline. The status of multiple comparisons: simultaneous estimation of all pairwise comparisons in one-way anova designs. *The American Statistician*, 35(3):134–141, 1981.

[122] Huan Sun, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang. Open domain question answering via semantic enrichment. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy*, pages 1045–1055, 2015.

[123] Saúl Vargas, Linas Baltrunas, Alexandros Karatzoglou, and Pablo Castells. Coverage, redundancy and size-awareness in genre diversity for recommender systems. In *Proceedings of the Eighth Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA*, pages 209–216, 2014.

[124] Vasilis Verroios, Hector Garcia-Molina, and Yannis Papakonstantinou. Waldo: An adaptive human interface for crowd entity resolution. In *Proceedings of the 2017 International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA*, pages 1133–1148, 2017.

[125] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the Twenty-Fifth International Conference on Machine Learning (ICML 2008), Helsinki, Finland*, volume 307 of *ACM International Conference Proceeding Series*, pages 1096–1103, 2008.

[126] Maksims Volkovs, Himanshu Rai, Zhaoyue Cheng, Ga Wu, Yichao Lu, and Scott Sanner. Two-stage model for automatic playlist continuation at scale. In *Proceedings of the Recommender Systems Challenge, RecSys Challenge 2018, Vancouver, BC, Canada*, pages 9:1–9:6, 2018.

[127] Dongjing Wang, Shuiguang Deng, and Guandong Xu. Sequence-based context-aware music recommendation. *Journal of Information Retrieval*, 21(2-3):230–252, 2018.

[128] Jiannan Wang, Tim Kraska, Michael J. Franklin, and Jianhua Feng. Crowder: Crowdsourcing entity resolution. *Proceedings of the VLDB Endowment*, 5(11):1483–1494, 2012.

[129] Jinfeng Yi, Rong Jin, Anil K. Jain, Shaili Jain, and Tianbao Yang. Semi-crowdsourced clustering: Generalizing crowd labeling by robust distance metric learning. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems 2012. December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1781–1789, 2012.

[130] Cong Yu, Laks V. S. Lakshmanan, and Sihem Amer-Yahia. It takes variety to make a world: diversification in recommender systems. In *Proceedings of the 12th International Conference on Extending Database Technology, Saint Petersburg, Russia, March 24-26, 2009, Proceedings*, volume 360 of *ACM International Conference Proceeding Series*, pages 368–378, 2009.

[131] Mi Zhang and Neil Hurley. Avoiding monotony: improving the diversity of recommendation lists. In *Proceedings of the 2008 Conference on Recommender Systems, RecSys 2008, Lausanne, Switzerland*, pages 123–130, 2008.

[132] Ying Zhang, Xianghua Ding, and Ning Gu. Understanding fatigue and its impact in crowdsourcing. In *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*, pages 57–62. IEEE, 2018.

[133] Yudian Zheng, Jiannan Wang, Guoliang Li, Reynold Cheng, and Jianhua Feng. QASCA: A quality-aware task assignment system for crowdsourcing applications. In *Proceedings of the 2015 International Conference on Management of Data, Melbourne, Victoria, Australia*, pages 1031–1046, 2015.

[134] Eric R. Ziegel. Multivariate statistical modelling based on generalized linear models. *Technometrics*, 44(1):94, 2002.

[135] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan*, pages 22–32, 2005.