CROWD DATA ANALYTICS AND OPTIMIZATION

by

HABIBUR RAHMAN

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2017

To My Parents

ACKNOWLEDGEMENTS

I would like to express my sincerest gratitude to my professor Dr. Gautam Das. I am very lucky to had him as my Ph.D supervisor. From the moment he became my adviser, he took a sincere interest in choosing my research area and have had long discussions with me. He understood my strength, ability and my interests. Early in my Ph.D life, he advised me to focus on "Crowdsourcing" even though I had no publication in this area at that time. Since then, I have published several research papers with him. He always made sure that I am on the right track by giving me counsel in a timely manner. More importantly, he always encourages me to do work which can create a considerable impact in our research community as well as in our society.

Second, I would like to thank my collaborator, my mentor Dr. Senjuti Basu Roy. I always told her that she is my second supervisor. I will always cherish the times that we spend on discussing, coming up with the plan for the experiments, solving critical problems.I cannot thank her enough for the impact she had in my Ph.D life.

I would like to thank my former lab members Saravanan and Mahashweta, who introduced me to this world of academia and mentored me early in my Ph.D career. I

would also like to thank all DbxLab members (Azade, Abol, Farhad, Jees and others) for giving me encouragement and support.

I would like to thank all my family members (Ammu, Abbu, Tushar and Sony) for giving me the support to go to USA for higher studies. Finally, I would like to thank my wonderful wife Ummy Sattar for her constant support in completion of my Ph.D.

April 24, 2017

ABSTRACT

CROWD DATA ANALYTICS AND OPTIMIZATION

Habibur Rahman, Ph.D.

The University of Texas at Arlington, 2017

Supervising Professor: Gautam Das

Crowdsourcing can be defined as *outsourcing with crowd*, where crowd refers to the online workers who are willing to complete simple tasks for small monetary compensation. The overwhelming reach of internet has enabled us to exploit crowd in an unprecedented way. Crowdsourcing, nowadays, is considered as a tool to solve both simple tasks (such as labeling ground truth, image recognition etc.) and complex tasks (such as collaborative writing, citizen journalism etc.). Furthermore, it is also used to solve computational problems such as Entity Resolution, Top-k, Group-by etc. While crowdsourcing provides us with plenty of opportunities, it also presents us with a plenty of challenges due to the complex interplay between the tasks and the workers.

In this dissertation, we first study the aspect of collaboration on solving complex task with crowdsourcing. Collaborative crowdsourcing acknowledges as one of the most promising areas of next-generation crowdsourcing. It refers to a specific form of human-based computation involving skilled workers forming groups and solving problems that require advanced skills in different domains in a collaborative manner. A number of emerging applications, such as collaborative document editing, sentence translation, and citizen journalism require human workers with complementary skills and expertise to form groups in order to achieve a complex goal. There are several key challenges to overcome in collaborative crowdsourcing- i) Task Assignment in such applications are primarily self-coordinated to date, or sometimes such assignments are performed manually by the domain experts in a task-specific manner without considering the affinity among the workers and ii) lack of principled solutions for estimating human factors such as skill/cost etc. We first initiate the investigation of the task assignment optimization problem for collaborative crowdsourcing and show how to incorporate team-based factors such as *affinity* and *critical mass*. Then, we demonstrate the deployment of our task assignment algorithm in a real-time collaborative system named *Crowd4u*. Finally, we present comprehensive optimization based formulations for estimation of the skill of workers in collaborative systems.

Then we propose a novel technique for task recommendation in crowdsourcing. There are some notable differences between task recommendation and the area of item recommendation. The main challenge here is that the worker does not explicitly says which task she likes (or dislikes). So, we propose several methods for task recommendation, which consider both the implicit signals and task similarity.

Finally, we propose a probabilistic framework for estimating all pair distance of a set of objects with crowdsourcing. This problem is appropriate for several distance-based machine learning applications such as clustering,k-nearest neighbor etc. We

divide the overall problem into three smaller problems- i)Aggregate crowd inputs ii) Estimate distance of the unknown object pairs from the known set of distances iii) Find the next best question to be asked to the crowd.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

CHAPTER 1

Introduction

The emergence of Web 2.0 has given rise to a virtual workforce with unlimited opportunities. *Crowdsourcing* is a term, widely used to describe a framework, which enables these virtual workers to carry out a set of tasks through an online platform. Examples of such platforms are Amazon Mechanical Turk(AMT), Crowdflower etc. While crowdsourcing is generally used for solving simple tasks (also known as micro-tasks), it is also envisioned as a framework for solving complex, knowledge-intensive tasks such as creating a document or creating subtitle in different languages. These days, crowdsourcing gained an immense popularity as people can get their work done by numerous virtual workers for very low compensation. This new form of out-sourcing has created a plenty of opportunity for business owners, academicians and various other entities. Businesses use crowdsourcing primarily to i) get feedback for their products, ii) promote their products on social media and other platforms. Researchers, on the other hand, primarily use crowdsourcing to get ground truth(labeled) data, or to solve complex computational problems where machine performs badly. Citizen Science is a popular crowdsourcing platform, where scientists use volunteers to gather data for their scientific expedition.

The two main components of crowdsourcing are "worker" and "task". The role of workers varies significantly with different types of tasks. Here, we try to give a brief overview of these two components.

Worker: We refer to "worker" as an online user, who registers into the crowdsourcing platform, willing to complete tasks for monetary compensation. The quality of a crowdsourcing task depends on the skill of the workers. Generally, crowdsourcing platforms use simple evaluation metrics for workers, such as acceptance ratio or average task rating etc. Both *estimation of skill of workers* and *worker compensation for tasks* are studied thoroughly in the context of crowdosurcing [1, 2, 3]. In this dissertation, we will explore skill estimation of the workers for collaborative tasks.

Tasks: A "task" is considered as a single unit of work, which is to be completed by one or multiple workers. Sometimes, a task is referred as a microtask or a Human Intelligence Task(HIT). Initially, the crowdsourcing platforms only facilitate simple tasks. Nowadays, with more computational power and more people being available online, crowdsourcing proves to be an effective tool for solving complex tasks.

Although crowdsourcing promises a new way of solving tasks, there are several challenges that need to be addressed. Broadly, existing research on crowdsourcing generally falls into two categories-

i) **Optimization in crowdsourcing framework**: These set of problems stems from the necessity to improve the existing framework while keeping an application in mind. Examples include the optimization of the worker to task assignment, estimation of the skill of workers, recommending a set of tasks to the workers based on the past history of the task completion [4, 5]. Solving these problems would improve the latency of task completion and would improve the quality of the completed tasks.

ii) **Solving novel problems with crowdsourcing** Another challenge is to demonstrate how the power of crowd can be used to solve complicated problems.

Crowdsourcing has been used to solve a number of problems such as top-k, group-by, ranking, database join, sorting, entity resolution etc [6, 7, 8].

In view of this challenges, next, we are going to present the overview and the impact of this dissertation.

1.1 Dissertation Overview and Impact

In this dissertation, first, we study the aspect of collaboration on solving complex task with crowdsourcing. Collaborative crowdsourcing is widely considered as the most important area of next generation crowdsourcing which is designed for solving complex tasks [9]. However, the task assignments in such applications are primarily self-coordinated to date, or sometimes such assignments are performed manually by domain experts in a task-specific manner. In the latter scenario, domain experts also manually evaluate the expertise of the workers. This leads us to explore the following problems. First, we formalize the notion of collaboration among crowd workers and propose a comprehensive optimization model for task assignment in a collaborative crowdsourcing environment. Then, we integrate our task assignment module in an academic crowdsourcing platform named *Crowd4u* [10]. Furthermore, we present comprehensive optimization based formulations for estimation of the skill of workers in team-based tasks.

Next, we propose techniques for task recommendation based on the previous history of completed tasks by the worker. A common problem for the crowdsourcing platform is that the workers suffer a huge latency to find a suitable task. To alleviate that, we propose two alternative formulations for task recommendation that exploit the implicit feedback and similarity of the tasks.

Finally, we propose a framework for estimating pairwise distances with crowdsourcing. This problem has applications in data mining and machine learning. Here,

we describe how to aggregate the crowd inputs when they are uncertain (or, have a probability of being incorrect). We also describe how to select the best question to be asked to the crowd.

We present a high-level overview of different chapters below.

Task Assignment Optimization in Collaborative Crowdsourcing: The current systems in "Collaborative Crowdsourcing" are clearly sub-optimal- with primarily because of the workers self-coordinate to form the groups. Also, there is no machine assisted intelligence that maximizes quality or minimizes cost. In this work, we initiate the investigation of optimization opportunities in collaborative crowdsourcing. Many popular applications, such as collaborative document editing, sentence translation, or citizen science resort to this special form of human-based computing, where, crowd workers with appropriate skills and expertise are required to form *groups* to solve complex *tasks*. Central to any collaborative crowdsourcing process is the aspect of successful *collaboration* among the workers, which, for the first time, is *formalized and then optimized* in this work. Our formalism considers two main collaboration-related human factors, affinity and upper critical mass, appropriately adapted from organizational science and social theories. Our contributions are (a) proposing a comprehensive model for collaborative crowdsourcing optimization, (b) rigorous theoretical analyses to understand the hardness of the proposed problems, (c) an array of efficient exact and approximation algorithms with provable theoretical guarantees. Finally, we present a *detailed set of experimental results* stemming from two real-world collaborative crowdsourcing application using Amazon Mechanical Turk, as well as conduct synthetic data analyses on scalability and qualitative aspects of our proposed algorithms. Our experimental results successfully demonstrate the efficacy of our proposed solutions.

Integrating Task Assignment to Existing Platforms: Our aim is to integrate the task assignment algorithms in an existing Crowdsourcing platform. We present Crowd4U[10], an educational crowdsourcing platform capable of handling complex tasks. Currently, there exists no crowdsourcing platform today enables the end-to-end deployment of collaborative tasks. In addition to treating workers and tasks as rich entities, Crowd4U also provides an easy-to-use form-based task UI. Crowd4U implements worker-to-task assignment algorithms that are appropriate for each kind of task. Once workers are assigned to tasks, appropriate worker collaboration schemes are enforced to enable effective result coordination.

Worker Skill Estimation in Team-Based Tasks Estimating skills of workers is of paramount importance for complex tasks. Typically, completed tasks are evaluated either by human experts or by machine algorithms. Skill estimation models need to leverage the underlying skill aggregation function which determines how the skill of a team is aggregated from its constituent members. Many emerging applications such as collaborative editing, multi-player games, or fan-subbing require forming a team of experts to accomplish a task together. In this work, we investigate how to estimate individual worker's skill based on the outcome of the team-based tasks they have undertaken. We consider two popular skill aggregation functions and estimate the skill of the workers, where skill is either a deterministic value or a probability distribution. We propose efficient solutions for worker skill estimation using continuous and discrete optimization techniques. We present comprehensive experiments and validate the scalability and effectiveness of our proposed solutions using multiple real-world datasets.

Task Recommendation in Crowdsourcing with Implicit Observations: We initiate the study of task recommendation problem for crowdsourcing platforms, where we leverage both implicit feedback and explicit features of the tasks. We assume

that we are given a set of workers, a set of tasks, interactions (such as the number of times a worker has completed a particular task), and the presence of explicit features of each task (such as task location). We intend to recommend tasks to the workers by exploiting implicit interactions, and the presence or absence of explicit features in the tasks. We present two alternative optimization problems and propose effective solutions. We validate our proposed method using real-world dataset by comparing with several baseline methods.

Estimating Pairwise Distance through Crowdsourcing Researchers have been studying on how to use crowdsourcing to solve computational problems such as finding maximum [11], group by [12], top-k [12], reducing the uncertainty of the data [13] etc. Estimating all pairs of distances among a set of objects has wide applicability in various computational problems in databases, machine learning, and statistics. This work presents a *probabilistic framework for estimating all pair distances through crowdsourcing, where the human workers are involved in providing distance between some object pairs.* Since the workers are subject to error, their responses are considered with a probabilistic interpretation. In particular, the framework comprises of three problems: (1) Given multiple feedback on an object pair, how do we combine and aggregate those feedbacks and create a probability distribution of the distance? (2) Since the number of possible pairs is quadratic in the number of objects, how do we estimate, from the known feedback of small numbers of object pairs, the unknown distances among all other object pairs? For this problem, we leverage the metric property of distance, in particular, the triangle inequality property in a probabilistic setting. (3) Finally, how do we improve our estimate by soliciting additional feedback from the crowd? For all three problems, we present principled modeling and solutions. We experimentally evaluate our proposed framework by involving multiple

real-world and large-scale synthetic data, by enlisting workers from a crowdsourcing platform.

1.2 Dissertation Organization

In chapter 2 we describe the task assignment model for collaborative crowdsourcing. Our key contributions are threefold. First, we investigate the optimization opportunities in collaborative crowdsourcing. Second, we propose a comprehensive theoretical analysis of our overall problem and our staged solution `Grp&Splt`. Finally, we present a comprehensive set of experimental results that demonstrate the effectiveness of the proposed solution.

In chapter 3, we describe the deployment of our task assignment algorithm, `Grp`, in an existing crowdsourcing platform, Crowd4u.

We formalize the problem of skill estimation for team-based tasks in chapter 4. There, we describe both deterministic and probabilistic interpretation of skills. We define two different skill aggregation functions, `Sum` and `Max`. We propose principled solutions and theoretical analyses to estimate workers skill under both of these aggregation mechanisms. We conduct comprehensive experiments on NBA and DBLP datasets to show that our algorithms are accurate and efficient.

In chapter 5, we present the study of task recommendation in crowdsourcing exploiting both implicit feedback and the presence of the explicit task features. First, we present an explicit feature preference model for task recommendation. Second, we propose an optimization model based on the latent factors. We empirically validate our proposed methods using real-world datasets and we compare our results with the state of the art baseline solutions.

Finally, In chapter 6, we consider the novel problem of all-pairs distance estimation via crowdsourcing in a probabilistic setting. We identify three sub-components

of our frameworks and present solutions to each problem. Furthermore, we experimentally evaluate the framework using both real and synthetic dataset.

CHAPTER 2

Task Assignment Optimization in Collaborative Crowdsourcing

Many popular applications, such as collaborative document editing, sentence translation, or citizen science resort to collaborative crowdsourcing, a special form of human-based computing, where, crowd workers with appropriate skills and expertise are required to form *groups* to solve complex *tasks*. While there has been extensive research on workers' task assignment for traditional microtask based crowdsourcing, they often ignore the critical aspect of collaboration. Central to any collaborative crowdsourcing process is the aspect of solving collaborative tasks that requires successful *collaboration* among the workers. Our formalism considers two main collaboration-related factors - affinity and upper critical mass - appropriately adapted from organizational science and social theories. Our contributions are three fold. First, we formalize the notion of collaboration among crowd workers and propose a comprehensive optimization model for task assignment in a collaborative crowdsourcing environment. Next, we study the hardness of the task assignment optimization problem and propose a series of efficient exact and approximation algorithms with

provable theoretical guarantees. Finally, we present a detailed set of experimental results stemming from two real-world collaborative crowdsourcing application using Amazon Mechanical Turk.

2.1 Introduction

Crowdsourcing Complex Tasks: Micro task based crowdsourcing has been applied successfully in a number of domains such as collecting labeled data, fact checking, image recognition etc [14]. Here, the crowd workers can operate independently because of the simplicity of the tasks. However, such an individualistic approach will not work for many complex knowledge intensive tasks such as Citizen Science where crowdsourcing is increasingly being used. Collaborative crowdsourcing is an emerging paradigm where a set of workers with complementary skills form groups and collaborate to perform complex tasks. The synergistic effect of collaboration in group based activities is widely accepted in socio-psychological research and traditional team based activities [15, 16, 17]. A number of popular applications such as collaborative document editing, sentence translation, or citizen science could be modeled as collaborative crowdsourcing tasks. Despite its immense potential, the transformative effect of "collaboration" remains largely unexplored in crowdsourcing [9].

Task Assignment Optimization for Solving Collaborative Tasks: The optimization goals for task assignment is putatively similar between collaborative task and traditional micro-task - maximize the quality of the completed tasks while minimizing cost by assigning appropriate tasks to appropriate workers. Task assignment has been extensively studied for microtask based crowdsourcing. However, none of those algorithms are applicable for collaborative crowdsourcing as they ignore the critical aspect of *Collaboration*. Instead of working individually, workers collaboratively work on tasks and build on each others' contributions.

This collaborative aspect requires that a task assignment algorithm must take into account both the characteristics of individual workers and that of the group. Prior work has identified some key individual characteristics of the worker, dubbed as human factors, such as skill and wages. From prior work on socio-psychological research[15, 16], we have identified two key factors for group characteristics that entail successful collaboration. The first factor *worker-worker affinity* [18, 19] represents the comfort-level between workers in a group who work on the same task. It has been noted that successful teams have members with high affinity with each other. In contrast, teams with low affinity often suffer from low productivity and take longer to complete the tasks [**?**]. Social theories widely underscore the importance of *upper critical mass* [20] for group collaboration, which is a constraint on the size of groups beyond which the collaboration effectiveness diminishes [20, 21].

Overview of Technical Approach: Despite the importance of collaborative crowdsourcing, there has been a dearth of work that formalizes the notion of collaboration and the optimization objectives for task assignment for collaborative crowdsourcing tasks. Additionally, while key factors for successful collaboration such as worker affinity and critical mass has been identified in psycho-social theories, there has been no prior effort on formalizing these individual and group based human factors in a principled manner to optimize the outcome of a collaborative crowdsourcing environment. Hence, our first significant contribution lies in appropriately incorporating the interplay of this variety of complex human factors into a set of well-formulated optimization problems.

Intuitively, the objective for task assignment is to choose, for each task, a group of workers who collectively hold skills required for the task, collectively cost less than the task's budget and collaborate effectively. Using the notions of affinity and upper critical mass, we formalize the flat model of work coordination [22] in collaborative

crowdsourcing as a graph with nodes representing workers and edges labeled with pairwise affinities. A group of workers is a clique in the graph whose size does not surpass the critical mass imposed by a task. A large clique (group) may further be partitioned into subgroups (each is a clique of smaller size satisfying critical mass) to complete a task because of the task's magnitude. Each clique has an intra and an inter-affinity to measure respectively the level of cohesion that the clique has internally and with other cliques. A clique with high intra-affinity implies that its members collaborate well with one another. Two cliques with a high inter-affinity between them imply that these two groups of workers work well together. Our optimization problem reduces to finding a clique that maximizes intra-affinity, satisfies the skill threshold across multiple domains, satisfies the cost limit, and maximizes inter-affinity when partitioned into smaller cliques. We note that no existing work on team formation in social networks [23, 24] or collaborative crowdsourcing [9, 18, 19] has attempted similar formulations.

We show that solving the complex optimization problem explained above is prohibitively expensive and incurs very high machine latency. Such high latency is unacceptable for a real-time crowdsourcing platform. Therefore, we propose an alternative strategy `Grp&Splt` that decomposes the overall problem into two stages and is a natural alternative to our original problem formulation. Even though this two-stage formulation is also computationally intractable in the worst case, it allows us to design instance optimal exact algorithms that work well in the average case, as well as efficient approximation algorithms with provable bounds. In the first stage (referred to as `Grp`), we first form a single group of workers by maximizing intra-affinity, while satisfying the skill and cost thresholds. In the second stage, (referred to as `Splt`), we decompose this large group into smaller subgroups, such that each satisfies the group size constraint (imposed by critical mass) and the inter-affinity across sub-groups

is maximized. Despite being NP-hard [25], we propose an instance optimal *exact* algorithm `OptGrp` and a novel *2-approximation* algorithm `ApprxGrp` for the stage-1 problem. Similarly, we prove the NP-hardness and propose a *3-approximation* algorithm `Min-Star-Partition` for a variant of the stage-2 problem.

We conduct a comprehensive experimental study with two different applications (*sentence translation* and *collaborative document editing*) using real world data from Amazon Mechanical Turk and present rigorous scalability and quality analyses using synthetic data. Our experimental results demonstrate that our formalism is effective in aptly modeling the behavior of collaborative crowdsourcing and our proposed solutions are scalable.

In summary, this work makes the following contributions:

1. *Formalism*: We investigate the optimization opportunities in collaborative crowdsourcing. In section 2.4, we formally define our problem which incorporates a variety of human factors.
2. *Solutions*: We propose a comprehensive theoretical analysis of our problems and approaches. We analyze the computational complexity of our problems and propose a principled staged solution. We propose exact instance optimal algorithms as well as efficient approximation algorithms with provable approximation bounds.
3. *Experiments*: We present a comprehensive set of experimental results (two real applications as well as synthetic experiments) that demonstrate the effectiveness of our proposed solutions.

The chapter is organized as follows. Sections 2.2, 2.3, and 2.4 discuss a database application of collaborative crowdsourcing, our data model, problem formalization, and initial solutions. Sections 2.5 and 2.6 describe our theoretical analyses and pro-

posed algorithmic solutions. Experiments are described in 2.7, related work in Section 2.8, and conclusion are presented in Section 2.9.

2.2 An Application of Collaborative Task

Sentence translation [26, 18, 19] is a frequently encountered application of collaborative task, where the objective is to use the workers to build a translation database of sentences in different languages. Such databases later on serve as the "training dataset" for supervised machine learning algorithms for automated sentence translation purposes.

As a running example for this chapter, consider a translation task t designed for translating an English video clip to French. Typically, such translation tasks follows a 3-step process [18, 19]: English speakers first translate the video in English, professional editors edit the translation, and finally workers with proficiency in both English and French translate English to French. Consequently, such task requires skills in 3 different domains: English comprehension (d_1), English editing (d_2), and French Translation ability (d_3).

In our optimization setting, each task t has a requirement of minimum skill per domain and maximum cost budget, and workers should collaborate with each other (e.g., to correct each others' mistakes [18]), and the collaboration effectiveness is quantified as the *affinity* of the group. Some aspects of our formulation has similarities with team formation problems in social networks [23]. The notion of affinity has been identified in the related work on sentence translation tasks [18, 19], as well as team formation problems [23].

However, if the group is "too large", the effectiveness of collective actions diminishes [20, 21] while undertaking the translation task, as an unwieldy group of workers fail to find effective assistance from their peers [18, 19]. Therefore, each task

	$\mathbf{u_1}$	$\mathbf{u_2}$	$\mathbf{u_3}$	$\mathbf{u_4}$	$\mathbf{u_5}$	$\mathbf{u_6}$
d_1	0.66	1.0	0.53	0.0	0.13	0.0
d_2	0.0	0.0	0.66	0.73	0.66	0.13
d_3	0.0	0.33	0.53	0.0	0.8	0.93
Wage	0.4	0.3	0.7	0.8	0.5	0.8

Table 2.1: Workers skill and wage table

	$\mathbf{u_1}$	$\mathbf{u_2}$	$\mathbf{u_3}$	$\mathbf{u_4}$	$\mathbf{u_5}$	$\mathbf{u_6}$
$\mathbf{u_1}$	0.0	1.0	0.66	0.66	0.85	0.66
$\mathbf{u_2}$	1.0	0.0	0.66	0.85	0.66	0.85
$\mathbf{u_3}$	0.66	0.66	0.0	0.4	0.66	0.40
$\mathbf{u_4}$	0.66	0.85	0.4	0.0	0.4	0.0
$\mathbf{u_5}$	0.85	0.66	0.66	0.4	0.0	0.4
$\mathbf{u_6}$	0.66	0.85	0.4	0.0	0.4	0.0

Table 2.2: Workers Distance Matrix

t is associated with a corresponding upper critical mass constraint on the size of an effective group, i.e., a large group may need to be further decomposed into multiple subgroups in order to satisfy that constraint. A study of the importance of the upper critical mass constraint in the crowdsourcing context, as well as how to set its (application-specific) value, are important challenges that are best left to domain experts; however, we experimentally study this issue for sentence translation.

When this task arrives, imagine that there are 6 workers $u_1, u_2, \ldots, u_6$ available in the crowdsourcing platform. Each worker has a skill value on each of the three skill domains described above, and a wage they expect. Additionally, the workers' cohesiveness or affinity is also provided. These human factors of the workers are summarized in Tables 2.1 and 2.2, and the task requirements of t (including thresholds on aggregated skill for each domain, total cost, and critical mass) are presented in Table 2.3 and are further described in the next section. The objective is to form a "highly cohesive" group $\mathcal{G}$ of workers that satisfies the lower bound of skill of the task and upper bound of cost requirements. Due to the upper critical mass constraint,

Q_1	Q_2	Q_3	C	K
1.8	1.4	1.66	3.0	3

Table 2.3: Task Description

$\mathcal{G}$ may further be decomposed into multiple subgroups. After that, each sub-group undertakes a subset of sentences to translate. Once all the subgroups finish their respective efforts, their contributions are merged. Therefore, both the overall group and its subgroups must be cohesive. Incorporation of upper critical mass makes our problem significantly different from the body of prior works [23], as we may have to create a group further decomposed into mutiple subgroups, instead of a single group.

2.3 Data Model

We introduce our data model and preliminaries that will serve as a basis for our problem definition.

2.3.1 Preliminaries

Domains: We are given a set of domains $D = \{d_1, d_2, \ldots, d_m\}$ denoting knowledge topics. Using the running example in Section 2.2, there are 3 different domains - English comprehension (d_1), English editing (d_2), and French Translation ability(d_3).

Workers: We assume a set $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ of n workers available in the crowdsourcing platform. The example in Section 2.2 describes a crowdsourcing platform with 6 workers.

Worker Group: A worker group $\mathcal{G}$ consists of a subset of workers from $\mathcal{U}$ i.e. $\mathcal{G} \subseteq \mathcal{U}$.

Skills: A skill is the knowledge on a particular skill domain in D, quantified in a continuous $[0, 1]$ scale. It is associated with workers and tasks. The skill of a worker

represents the worker's expertise/ability on a topic. The skill of a topic represents the minimum knowledge requirement/quality for that task. A value of 0 for a skill reflects no expertise of a worker for that skill. For a task, 0 reflects no requirement for that skill.

How to learn the skill of the workers is an important and independent research problem in its own merit. Most related work has relied on learning skill of the workers from "gold-standard" or benchmark datasets using pre-qualification tests [27, 28].

Collaborative Tasks: A collaborative task t has the following characteristics :

- **Skill Threshold**: Each $Q_i \in R$ represents the minimum skill requirement that a task needs to achieve for domain d_i. A task is deemed complete, if it attains its skill requirement over all the domains.
- **Cost Threshold**: $C \in R$, the cost budget to hire workers for a particular task. This gives an upper bound on the aggregated cost of assigning the workers.
- **Critical Mass**: K is a positive integer (greater than 0) which denotes the maximum group size for a task. Tasks that require high skill threshold may need many workers and may violate the critical mass threshold. In that case, the workers should be splitted in subgroups (each satisfying the critical mass constraint) such that the workers across all the subgroups satisfy the skill and cost threshold.

Specifically, t is characterized by a vector, $\langle Q_1, Q_2, \ldots, Q_m, C, K \rangle$, of length $m + 2$. For the example in Section 2.2, there are 3 domains ($m = 3$) and their respective skill requirements, its cost C, and critical mass K of the task is described in Table 2.3. A task is considered complete if it attains its skill requirement over all domains and satisfies all the constraints.

2.3.2 Human Factors

A worker is described by a set of human factors. We consider two types of factors - factors that describe individual worker's characteristics and factors that characterize an individual's ability to work with fellow workers. Our contribution is in appropriately adapting these factors in collaborative crowdsourcing from multi-disciplinary prior works such as team formation [23, 24] and psychology research [20, 21].

2.3.2.1 Individual Human Factors: Skill and Wage

Individual workers in a crowdsourcing environment are characterized by their skill and wage.

Skill: For each knowledge domain d_i, $u_{d_i} \in [0, 1]$ is the expertise level of worker u in d_i. Skill expertise reflects the quality that the worker's contribution has on a task accomplished by that worker.

Wage: $w_u \in [0, 1]$ is the minimum amount of compensation for which a worker u is willing to complete a task. We choose a simple model where a worker specifies a single wage value independent of the task at-hand.

Table 2.1 presents the respective skill of the 6 workers in 3 different domains and their individual wages for the running example.

2.3.2.2 Group-based Human Factors: Affinities

Although related work in collaborative crowdsourcing acknowledges the importance of workers' affinity to enable effective collaboration [18, 19], there is no attempt to formalize the notion any further. A worker's effectiveness in collaborating with her fellow workers is measured as *affinity*. We adopt an affinity model similar to group

formation problems in social networks [29, 23], where the atomic unit of affinity is *pairwise*, i.e., a measure of cohesiveness between every pair of workers. After that, we propose different ways to capture *intra-group* and *inter-group* affinities.

Pairwise affinity: The affinity between two workers u_i and u_j, $\mathit{aff}(u_i, u_j)$, can be calculated by capturing the *similarity* between workers using simple socio-demographic attributes, such as region, age, gender, as done in previous work [18], as well as more complex psychological characteristics [30]. For our purpose, we normalize pairwise affinity values to fit in $[0, 1]$ and use a notion of worker-worker *distance* instead, i.e., where $\mathit{dist}(u_i, u_j) = 1 - \mathit{aff}(u_i, u_j)$. Thus a smaller distance between workers ensures a better collaboration. Table 2.2 presents the pair-wise distance of all 6 workers for running example in Section 2.2. As will be clear later, the notion of distance rathey than affinity enables the design of better algorithms for our purposes.

Intra-group affinity: For a group $\mathcal{G}$, its intra-group affinity measures the collaboration effectiveness among the workers in $\mathcal{G}$. Here again we use distance and compute intra-group distance in one of two natural ways: computing the diameter of $\mathcal{G}$ as the largest distance between any two workers in $\mathcal{G}$, or aggregating all-pair worker distances in $\mathcal{G}$:

$$\mathit{DiaDist}(\mathcal{G}) = Max_{\forall u_i, u_j \in \mathcal{G}}\, \mathit{dist}(u_i, u_j)$$

$$\mathit{SumDist}(\mathcal{G}) = \Sigma_{\forall u_i, u_j \in \mathcal{G}}\, \mathit{dist}(u_i, u_j)$$

For both definitions, smaller value is better.

Inter-group affinity: When a group violates the upper critical mass constraint [20], it needs to be decomposed into multiple smaller ones. The resulting subgroups need to work together to achieve the task. Given two subgroups G_1, G_2 split from a large group $\mathcal{G}$, their collaboration effectiveness is captured by computing their inter-group affinities. Here again, we use distance instead of affinity. More

concretely, the inter-group distance is defined in one of two natural ways: either the largest distance between any two workers across the sub-groups, or the aggregation of all pair-wise workers distances across subgroups:

$$DiaInterDist(G_1, G_2) = Max_{\forall u_i \in G_1, u_j \in G_2} dist(u_i, u_j)$$

$$SumInterDist(G_1, G_2) = \Sigma_{\forall u_i \in G_1, u_j \in G_2} dist(u_i, u_j)$$

This can be generalized to more than two subgroups: if there are x subgroups, overall inter-group affinity is the summation of inter-group affinity for all possible $\binom{x}{2}$ pairs.

2.4 Optimized Group Formation

Problem Settings: For each collaborative task, we intend to form *the most appropriate group of workers* from the available worker pool. A collaborative crowdsourcing task has skill requirements in multiple domains and a cost budget, which is similar to the requirements of collaborative tasks in team formation problems [29]. Then, we adapt the "flat-coordination" models of worker interactions, which is considered important in prior works in team formation [23] as the "coordination cost", or in collaborative crowdsourcing [18] itself, as 'the 'turker-turker" affinity model. However, unlike previous work, we attempt to fully explore the potential of "group synergy" [31] and how it yields the maximum qualitative effects in group based efforts by maximizing affinity among the workers (or minimizing distance). Finally, we intend to investigate the effect of upper critical mass in the context of collaborative crowdsourcing as a constraint on group size, beyond which the group must be decomposed into multiple subgroups that are cohesive inside and across. Indeed, our objective function is designed to form a group (or further decomposed into a set of subgroups) to undertake a specific task that achieves the highest qualitative effect, while satisfying the cost constraint.

(1) *Qualitative effect of a group*: Intuitively, the overall qualitative effect of a formed group to undertake a specific task is a function of the skill of the workers and their collaboration effectiveness. Learning this function itself is challenging, as it requires access to adequate training data and domain knowledge. In our initial effort, we therefore make a reasonable simplification, where we seek to maximize group affinity and pose quality as a hard constraint[1]. Existing literature (indicatively [31]) informs us that aggregation is a mechanism that turns private judgments (in our case individual workers' contributions) into a collective decision (in our case the final translated sentences), and is one of the four pillars for the wisdom of the crowds. For complex tasks like sentence translation or document editing, there is no widely accepted mathematical function of aggregation. We choose sum to aggregate the skill of the workers that must satisfy the lower bound of the quality of the task. This simplest and yet most intuitive functions for transforming individual contributions into a collective result has been adopted in many previous works [23, 29, 32]. Moreover, this simpler function allows us to design efficient algorithms. Exploring other complex functions (e.g., multiplicative function) or learning them is deferred to future work.

(2) *Upper critical mass*: Sociological theories widely support the notion of "critical mass"[20, 21] by reasoning that large groups are less likely to support collective action. However, whether the effect of "critical mass" should be imposed as a hard constraint, or it should have more of a gradual "diminishing return" effect, is itself a research question. For simplicity, we consider upper critical mass as a hard constraint and evaluate its effectiveness empirically for different values. Exploring more sophisticated function to capture critical mass is deferred to future work.

[1]Notice that posing affinity as a constraint does not fully exploit the effect of "group synergy".

Problem 1. AffAware-Crowd: *Given a collaborative task t, the objective is to* form a worker group $\mathcal{G}$*, further partitioned into a set of x subgroups $G_1, G_2, ...G_x$ (if needed) for the task t that minimizes the aggregated intra-distance of the workers, as well as the aggregated inter-distance across the subgroups of $\mathcal{G}$, and $\mathcal{G}$ must satisfy the skill and cost thresholds of t, where each subgroup G_i must satisfy the upper critical mass constraint of t. Of course, if the group $\mathcal{G}$ itself satisfies the critical mass constraint, no further partitioning in $\mathcal{G}$ is needed, giving rise to a single worker group. As explained above, quality of a task is defined as an aggregation (sum) of the skills of the workers [23, 29]. Similarly, cost of the task is the additive wage of all the workers in $\mathcal{G}$.*

2.4.1 Optimization Models

Given the definition of AffAware-Crowd above, we propose multiple optimization objective functions based on different inter- and intra-distance measures defined in Section 2.3.

For a group $\mathcal{G}$, we calculate intra-distance in one of the two possible ways: $DiaDist, SumDist$. If $\mathcal{G}$ is further partitioned to satisfy the upper critical mass constraint, then we also want to enable strong collaboration across the subgroups by minimizing inter-distance. For the latter, inter-distance is calculated using one of $DiaInterDist, SumInterDist$. Even though there may be many complex formulations to combine these two factors, in our initial effort our overall objective function is a simple sum of these two factors that we wish to minimize. This gives rise to 4 possible optimization objectives.

- *DiaDist*, *DiaInterDist*:

$$\text{Minimize } \{DiaDist(\mathcal{G}) \quad + \\ Max\{\forall G_i, G_j \in \mathcal{G} \quad DiaInterDist(G_i, G_j)\}\}$$

- *SumDist*, *DiaInterDist*:

$$\text{Minimize } \{SumDist(\mathcal{G}) \quad + \\ Max\{\forall G_i, G_j \in \mathcal{G} \quad DiaInterDist(G_i, G_j)\}\}$$

- *DiaDist*, *SumInterDist*:

$$\text{Minimize } \{DiaDist(\mathcal{G}) + \sum_{G_i, G_j \in \mathcal{G}} SumInterDist(G_i, G_j)\}$$

- *SumDist*, *SumInterDist*:

$$\text{Minimize } \{SumDist(\mathcal{G}) + \sum_{G_i, G_j \in \mathcal{G}} SumInterDist(G_i, G_j)\}$$

where, each of these objective function has to satisfy the following three constraints on skill, cost, and critical mass respectively, as described below:

$$\Sigma_{\forall_{u_i \in \mathcal{G}}} u_{d_i} \geq Q_i \quad \forall_{d_i}$$

$$\Sigma_{\forall_{u \in \mathcal{G}}} w_u \leq C$$

$$|G_i| \leq K \quad \forall i \in \{1, 2, \ldots, x\}$$

The rest of our discussion only considers *DiaDist* on intra-distance and *SumInterDist* on inter-distance. We refer to this variant of the problem as `AffAware-Crowd`. We note that our proposed optimal solution in Section 2.4 could be easily extended to other combinations as well.

Theorem 1. *Problem* `AffAware-Crowd` *is NP-hard [25].*

Proof. Given a collaborative task t and a set of users $\mathcal{U}$ and a real number value X, the decision version of the problem is, whether there is a group $\mathcal{G}$ (further partitioned into multiple subgroups) of users ($\mathcal{G} \subseteq \mathcal{U}$), such that the aggregated inter and intra distance value of $\mathcal{G}$ is X and skill, cost, and critical mass constraints of t are satisfied. The membership verification of the decision version of `AffAware-Crowd` is clearly polynomial.

To prove NP-hardness, we consider a variant of *compact location* [33] problem which is known to be NP-Complete. Given a complete graph G with N nodes, an integer $n \leq N$ and a real number X', the decision version of the problem is whether there is a complete sub-graph g' of size $n' \in N$, such that the maximum distance between between any pair of nodes in g' is X'. This variant of the compact location problem is known as `Min-DIA` in [33].

Our NP-hardness proof uses an instance of `Min-DIA` and reduces that to an instance of `AffAware-Crowd` problem in polynomial time. The reduction works as follows: each node in graph G represents a worker u, and the distance between any two nodes in G is the distance between a pair of workers for our problem. We assume that the number of skill domain is 1, i.e., $m = 1$. Additionally, we consider that each workers u has same skill value of 1 on that domain, i.e., $u_d = 1, \forall u$ and their cost is 0, i.e., $w_u = 0, \forall u$. Next, we describe the settings of the task t. For our problem, the task also has the quality requirement in only one domain, which is, Q_1. The skill, cost, and critical mass of t are, $\langle Q_1 = n', C = 0, K = \infty \rangle$. This exactly creates an instance of our problem in polynomial time. Now, the objective is to form a group $\mathcal{G}$ for task t such that all the constraints are satisfied and the objective function value

of `AffAware-Crowd` is X', such that there exists a solution to the `Min-DIA` problem, if and only if, a solution to our instance of `AffAware-Crowd` exists. □

2.4.2 Algorithms for AffAware-Crowd

Our optimization problem attempts to appropriately capture the complex interplay among various important factors. The proof of Theorem 1 shows that the simplest variant of the optimization problem is NP-hard. Despite the computational hardness, we attempt to stay as principled as possible in our technical contributions and algorithms design. Towards this end, we propose two alternative directions:

(I) **ILP**: We propose a Integer Linear Programming (ILP) [34] formulation to optimally solve our original overarching optimization problem. We note that even translating the problem to an ILP is non-trivial, because the subgroups inside the large group are unknown and are determined by the solution.

(II) **Staged Approach**: We propose an alternate strategy due to the fact that ILP is prohibitively expensive. We refer it as `Grp&Splt`. As the name suggests, it decomposes the original problem into two phases-

a) `Grp`: In this phase, a single group is formed that satisfies the skill and cost threshold but ignores the upper critical mass constraint. We briefly summarize the algorithms for `Grp` stage below:

- `ApprxGrp`: This is an approximation algorithm with approximation factor of 2. It invokes a subroutine, which uses branch and bound method, to find a group of workers who satisfy skill and cost constraint for the task. For efficiency, we rely on bucketing the cost values. We refer to this variant as `Cons-k-Cost-ApprxGrp`.

- `OptGrp`: This is an instance optimal algorithm that also uses branch and bound method. However, it iterates over all the valid solutions to find the optimal one.

b) `Splt`: In this phase, we partition the worker group (returned from the `Grp` phase) into smaller collaborative subgroups. First, we attempt to find the optimal number of subgroups and then find the assignment of workers into these subgroups. We propose `Min-Star-Parition`, an approximation algorithm for this problem.

Of course, this staged solution may not have any theoretical guarantees for our original problem formulation. However, our experimental results demonstrate that this formulation is efficient, as well as adequately effective.

2.4.2.1 ILP for AffAware-Crowd

We discuss the ILP next as shown in Equation 2.1. Let $e_{(i,i')}$ denote a boolean decision variable of whether a user pair u_i and u'_i would belong to same sub-group in group $\mathcal{G}$ or not. Also, imagine that a total of x groups $(G_1, G_2, \ldots, G_x)$ would be formed for task t, where $1 \leq x \leq n$ (i.e., at least the subgroup is $\mathcal{G}$ itself, or at most n singleton subgroups could be formed). Then, which subgroup the worker pair should be assigned must also be determined, where the number of subgroups is unknown in the first place. Note that translating the problem to an ILP is *non-trivial and challenging*, as the formulation deliberately makes the problem linear by translating each worker-pair as an atomic decision variable (as opposed to a single worker) in the formulation, and it also returns the subgroup where each pair should belong to. Once the ILP is formalized, we use a general-purpose solver to solve it. Although the *Max* operator in the objective function (expresses *DiaDist*) must be translated

appropriately further in the actual ILP implementation, in our formalism below, we preserve that abstraction for simplicity.

$$\begin{aligned}
\text{minimize} \quad & \mathcal{D} = \text{Max}\{e_{i,i'} \times dist(u_i, u_{i'})\} \quad + \\
& \sum_{\forall G_i, G_j \in \mathcal{G}} \quad \sum_{\forall u_i \in G_i, u_j \in G_j} e_{i,j} dist(u_i, u_j)
\end{aligned}$$

subject to

$$\begin{aligned}
& \sum_{i=1}^{n} \sum_{j=1}^{x} u_{(i,G_j)} \times u^i_{d_l} \geq Q_l \qquad \forall l \in [1, m] \\
& \sum_{i=1}^{n} \sum_{j=1}^{x} u_{(i,G_j)} \times w^i_u \leq C \\
& \sum_{i=1}^{n} u_{(i,G_j)} \leq K \qquad \forall j \in [1, x] \\
& \sum_{j=1}^{x} u_{(i,G_j)} \leq 1 \qquad \forall i \in [1, n] \\
& e_{i,i'} = \begin{cases} 1 & \exists j \in [1, x] \;\&\; u_{(i,G_j)} = 1 \;\&\; u_{(i',G_j)} = 1 \\ 0 & \text{otherwise} \end{cases} \\
& x \in \{0, 1, \ldots, n\} \\
& u_{(i,G_j)} \in \{0, 1\} \qquad \forall i \in [1, n], \forall j \in [1, x]
\end{aligned} \tag{2.1}$$

The objective function returns a group of subgroups that minimizes $DiaDist(\mathcal{G}) + \Sigma_{\forall_{G_i,G_j}} SumInterDist(G_i, G_j)$. The first three constraints ensure the skill, cost and upper critical mass thresholds, whereas the last four constraints ensure the disjointedness of the group and the integrality constraints on different Boolean decision variables.

When run on the example in Section 2.2, the ILP generates the optimal solution and creates group $\mathcal{G} = \{u_1, u_2, u_3, u_4, u_6\}$ with two subgroups, $G_1 = \{u_1, u_2, u_4\}$, and

$G_2 = \{u_3, u_6\}$. The distance value of the optimization objective is 4.23, which equals to $DiaDist(\mathcal{G}) + InterDist(G_1, G_2)$.

2.4.2.2 Grp&Splt : A Staged Approach

Our proposed alternative strategy `Grp&Splt` works as follows: in the `Grp` stage, we attempt to form a single worker group that minimizes $DiaDist(\mathcal{G})$, while satisfying the skill and cost constraints (and ignoring the upper critical mass constraint). Note that this may result in a large group, violating the upper critical mass constraints. Therefore, in the `Splt` phase, we partition this big group into multiple smaller sub-groups, each satisfying the upper critical mass constraint in such a way that the aggregated inter-distance between all pair of groups $\Sigma_{\forall_{G_i,G_j}} SumInterDist(G_i, G_j)$ is minimized. As mentioned earlier, there are three primary reasons for taking this alternative route: (a) In many cases we may not even need to execute `Splt`, because the solo group formed in `Grp` phase abides by the upper critical mass constraint leading to the solution of the original problem. (b) The original complex ILP is prohibitively expensive. Our experimental results demonstrate that the original ILP does not converge in hours for more than 20 workers. (c) Most importantly, `Grp&Splt` allows us to design efficient approximation algorithms with constant approximation factors as well as instance optimal exact algorithms that work well in practice, as long as the distance between the workers satisfies the *metric property* (triangle inequality in particular) [35, 36]. We underscore that the triangle inequality assumption is not an overstretch, rather many natural distance measures (Euclidean distance, Jaccard Distance) are metric and several other similarity measures, such as Cosine Similarity, Pearson and Spearman Correlations could be transformed to metric distance [37]. Furthermore, this assumption has been extensively used in distance computation in

the related literature [38, 23]. Without metric property assumptions, the problems remain largely inapproximable [36].

2.5 Enforcing Skill & Cost : GRP

In this section, we first formalize our proposed approach in `Grp` phase, discuss hardness results, and propose algorithms with theoretical guarantees. Recall that our objective is to form a single group $\mathcal{G}$ of workers that are cohesive (the diameter of that group is minimized), while satisfying the skill and the cost constraint.

Definition 1. `Grp`*: Given a task t, form a single group $\mathcal{G}$ of workers that minimizes DiaDist($\mathcal{G}$), while satisfying the skill and cost constraints, i.e., $\Sigma_{\forall u \in \mathcal{G}} u_{d_i} \geq Q_i, \forall_{d_i}$, & $\Sigma_{\forall u \in \mathcal{G}} w_u \leq C$.*

Theorem 2. *Problem* `Grp` *is NP-hard.*

Proof. Given a collaborative task t with critical mass constraint and a set of users $\mathcal{U}$ and a real number X, the decision version of the problem is, whether there is a group $\mathcal{G}$ of users ($\mathcal{G} \subseteq \mathcal{U}$), such that the diameter is X, and skill and cost constraints of t are satisfied.The membership verification of this decision version of `Grp` is clearly polynomial.

To prove NP-hardness, the follow the similar strategy as above. We use an instance of `Min-DIA` [33] and reduce that to an instance of `Grp`, as follows: each node in graph G of `Min-DIA` represents a worker u, and the distance between any two nodes in G is the distance between a pair of workers for our problem. We assume that the number of skill domain is 1, i.e., $m = 1$. Additionally, we consider that each workers u has the same skill value of 1 on that domain, i.e., $u_d = 1, \forall u$ and their cost is 0, i.e., $w_u = 0, \forall u$. Task t has quality requirement on only one domain, which is, Q_1. The skill requirement of t is $\langle Q_1 = n'$ and cost $C = 0 \rangle$. Now, the objective is to form

a group $\mathcal{G}$ for task t such that the skill and cost constraints are satisfied with the diameter of `Grp` as X', such that there exists a solution to the `Min-DIA` problem, if and only if, a solution to our instance of `Grp` exists. □

Proposed Algorithms for `Grp`: We discuss two algorithms at length - a) `OptGrp` is an instance optimal algorithm. b) `ApprxGrp` algorithm has a 2-*approximation factor*, as long as the distance satisfies the triangle inequality property. Of course, an additional optimal algorithm is the ILP formulation itself (referred to as `ILPGrp` in experiments), which could be easily adapted from Section 2.4. Both `OptGrp` and `ApprxGrp` invoke a subroutine inside, referred to as `GrpCandidateSet`. We describe a general framework for this subroutine next.

2.5.1 Subroutine GrpCandidateSet

Input to this subroutine is a set of n workers and a task t (in particular the skill and the cost constraints of t) and the output is a *worker group* that satisfies the skill and cost constraints. Notice that, if done naively, this computation takes 2^n time. However, Subroutine `GrpCandidateSet` uses effective pruning strategy to avoid unnecessary computations that is likely to terminate much faster. It *computes a binary tree representing the possible search space* considering the nodes in an arbitrary order, each node in the tree is a worker u and has two possible edges (1/0, respectively stands for whether u is included in the group or not). A root-to-leaf path in that tree represents a *worker group*.

At a given node u, it makes two estimated bound computation : a) it computes the lower bound of cost (LB_C) of that path (from the root upto that node), b) it computes the upper bound of skill of that path (UB_{d_i}) for each domain. It compares LB_C with C and compares UB_{d_i} with $Q_i, \forall d_i$. If $LB_C > C$ or $UB_{d_i} < Q_i$ for any of the

domains, that branch is fully pruned out. Otherwise, it continues the computation. Figure 2.1 has further details.

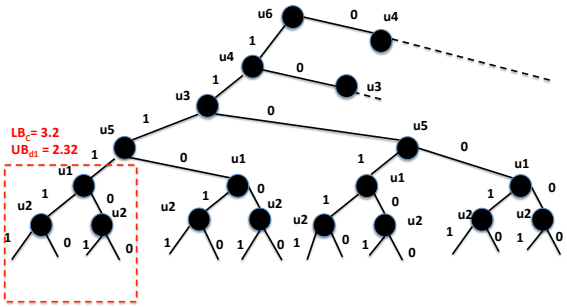


Figure 2.1: A partially constructed tree of GrpCandidateSet using the example in Section 2.2. At node $u_1 = 1$, $LB_C = w_{u_6} + w_{u_4} + w_{u_3} + w_{u_5} + w_{u_1} = 3.2$ and $UB_{d_1} = u^6_{d_1} + u^4_{d_1} + u^3_{d_1} + u^5_{d_1} + u^1_{d_1} + u^2_{d_1} = 2.32$. The entire subtree is pruned, since $LB_C(3.2) > C$.

`ApprxGrp` uses this subroutine to find the first valid answer, whereas, Algorithm `OptGrp` uses it to return all valid answers.

2.5.2 Further Search Space Optimization

When the skill and cost of the workers are arbitrary, a keen reader may notice that Subroutine `GrpCandidateSet` may still have to explore 2^n potential groups in the worst case. Instead, if we have only a constant number of costs and arbitrary skills, or a constant number of skill values and any arbitrary number of costs, interestingly, the search space becomes polynomial. Of course, the search space is polynomial when both are constants.

We describe the constant cost idea further. Instead of any arbitrary wage of the workers, we now can discretize workers wage apriori and create a constant number of k different buckets of wages (a worker belongs to one of these buckets) and build the search tree based on that. When there are m knowledge domains, this gives rise to

a total of mk buckets. For our running example in Section 2.2, for simplicity if we consider only one skill (d_1), this would mean that we discretize all 6 different wages in k (let us assume $k = 2$) buckets. Of course, depending on the granularity of the buckets this would introduce some approximation in the algorithm as now the workers actual wage would be replaced by a number which may be lesser or greater than the actual one. However, such a discretization is realistic, since many crowdsourcing platforms, such as AMT, allow only one cost per task.

For our running example, let us assume, bucket 1 represents wage 0.5 and below, bucket 2 represents wage between 0.5 and 0.8. Therefore, now workers u_3, u_4, u_6 will be part of bucket 2 and the three remaining workers will be part of bucket 1. After this, one may notice that the tree will neither be balanced nor exponential. Now, for a given bucket, the possible ways of worker selection is polynomial (they will always be selected from most skilled ones to the least skilled ones), making the overall search space polynomial for a constant number of buckets. In fact, as opposed to 2^6 possible branches, this modified tree can only have $(3+1)\times(3+1)$ possible branches. Figure 2.2 describes the idea further.

Once this tree is constructed, our previous pruning algorithm `GrpCandidateSet` could be applied to enable further efficiency.

2.5.3 Approximation Algorithm ApprxGrp

A popular variant of facility dispersion problem [35, 36] attempts to discover a set of nodes (that host the facilities) that are as far as possible, whereas, compact location problems [33] attempt to minimize the diameter. For us, the workers are the nodes, and `Grp` attempts to find a worker group that minimizes the diameter, while satisfying the multiple skills and a single cost constraint. We propose a 2-approximation algorithm for `Grp`, that is not studied before.

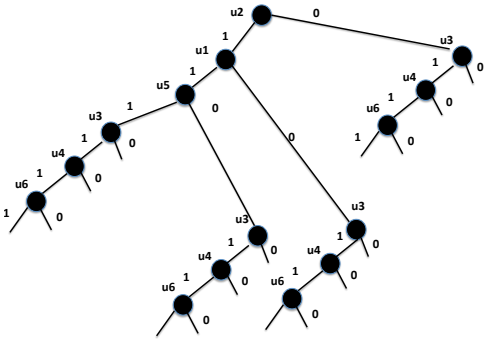


Figure 2.2: Possible search space using the example in Section 2.2, after the cost of the workers are discretized into $k = 2$ buckets, considering only one skill d_1. The tree is constructed in descending order of skill of the workers per bucket. For bucket 1, if the most skilled worker u_2 *is not selected, the other two workers* (u_1, u_5) *will never be selected.*

Algorithm `ApprxGrp` works as follows: The main algorithm considers a sorted (ascending) list $\mathcal{L}$ of distance values (this list represents all unique distances between the available worker pairs in the platform) and performs a binary search over that list. First, it calls a subroutine (`GrpDia`) with a distance value α that can run at the most n times. Inside the subroutine, it considers worker u_i in the i-th iteration to retrieve a *star graph*[2] centered around u_i that satisfies the distance α. The nodes of the star are the workers and the edges are the distances between each worker pair, such that no edge in that retrieved graph has an edge $> \alpha$. One such star graph is shown in Figure 2.3.

Next, given a star graph with a set of workers $\mathcal{U}'$, `GrpDia` invokes `GrpCandidateSet`$(\mathcal{U}', t)$ to select a subset of workers (if there is one) from $\mathcal{U}'$, who together satisfy the skill and cost thresholds. `GrpCandidateSet` constructs the tree in the best-first-search manner and terminates when the first valid solution is found, or no further search is possible. If the cost values are further discretized, then the tree

[2]Star graph is a tree on v nodes with one node having degree $v-1$ and other $v-1$ nodes with degree 1.

is constructed accordingly, as described in Section 2.5.2. This variant of `ApproxGrp` is referred to as `Cons-k-Cost-ApproxGrp`.

Upon returning a non-empty subset $\mathcal{U}''$ of $\mathcal{U}'$,

`GrpCandidateSet` terminates. Then, `ApprxGrp` stores that α and associated $\mathcal{U}''$ and continues its binary search over $\mathcal{L}$ for a different α. Once the binary search ends, it returns that $\mathcal{U}''$ which has the smallest α associated as the solution with the diameter upper-bounded by 2α, as long as the distance between the workers satisfy the triangle inequality[3]. In case `GrpDia` returns an empty worker set to the main function, the binary search continues, until there is no more option in $\mathcal{L}$. If there is no such $\mathcal{U}''$ that is returned by `GrpDia`, then obviously the attempt to find a worker group for the task t remains unsuccessful.

The pseudo-code of the algorithm `ApprxGrp` is presented in Algorithm 1. For the given task t using the example in Section 2.2, $\mathcal{L}$ is ordered as follows: $0, 0.4$, $0.66, 0.85, 1.0$. The binary search process in the first iteration considers $\alpha = 0.66$ and calls `GrpDia`$(\alpha, \{Q_i, \forall d_i\}, C)$. In the first iteration, `GrpDia` attempts to find a *star graph* (referred to Figure 2.3) with u_1 as the center of the star. This returned graph is taken as the input along with the skill threshold of t inside `GrpCandidateSet`next. For our running example, subroutine `GrpDia`$(0.66, 1.8\ , 1.66, 1.4, 2.5)$ returns u_1, u_3, u_4, u_6. Now notice, these 4 workers do not satisfy the skill threshold of task t (which are respectively $1.8, 1.66, 1.4$ for the 3 domains.). Therefore, `GrpCandidateSet`$(\mathcal{U}, t)$ returns false and `GrpDia` continues to check whether a star graph centered around u_2 satisfies the distance threshold 0.66. When run on the example in Section 2.2, `ApprxGrp` returns workers u_1, u_2, u_3, u_5, u_6 as the results with objective function value upper bounded by $\leq 2 \times 0.66$.

[3]Without triangle inequality assumption, no theoretical guarantee could be ensured [36].

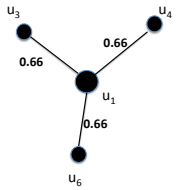


Figure 2.3: An instantiation of `GrpDia(0.66)` using the example in Section 2.2. A star graph centered u_1 is formed.

Algorithm 1 Approximation Algorithm `ApprxGrp`

Require: $\mathcal{U}$, human factors for $\mathcal{U}$ and task t

1: List $\mathcal{L}$ contains all unique distance values in increasing order
2: **repeat**
3: Perform binary search over $\mathcal{L}$
4: For a given distance α, $\mathcal{U}' = \texttt{GrpDia}(\alpha, \{\texttt{Q}_\texttt{i}, \forall \texttt{d}_\texttt{i}\}, \texttt{C})$
5: **if** $\mathcal{U}' \neq \emptyset$ **then**
6: Store worker group $\mathcal{U}'$ with diameter $d \leq 2\alpha$.
7: **end if**
8: **until** the search is complete
9: **return** $\mathcal{U}'$ with the smallest d

Theorem 3. *Algorithm* `ApprxGrp` *has a* 2*-approximation factor, as long as the distance satisfies triangle inequality.*

Proof. Algorithm `ApprxGrp` overall works as follows: it sorts the distance values in ascending fashion to create a list $\mathcal{L}$ and performs a binary search over it. For a given distance value α, it makes a call to `GrpDia(`α`)`. Recall Figure 2.3 that forms a star graph centered on u_1 with `GrpDia(0.66)` using the example in Section 2.2. Consider Figure 2.4 and notice that for a given distance value $=\alpha$, the complete graph induced by the star graph can not have any edge that is larger than $2 \times \alpha$, as long as the distance satisfies the triangle inequality property. Therefore, when

Algorithm 2 Subroutine `GrpDia`

Require: Distance matrix of the worker set $\mathcal{U}$, distance α, task t.

1: **repeat**
2: for each worker u
3: form a star graph centered at u, such that for each edge u, u_j, $dist(u,u_j) \leq \alpha$. Let $\mathcal{U}'$ be the set of workers in the star graph.
4: $\mathcal{U}'' =$ `GrpCandidateSet`$(\mathcal{U}', t)$
5: **if** $\mathcal{U}'' \neq \emptyset$ **then**
6: return $\mathcal{U}''$
7: **end if**
8: **until** all n workers have been fully exhausted
9: **return** $\mathcal{U}'' = \emptyset$

`GrpDia`(α) returns a non-empty worker set (that only happens when the skill and cost thresholds are satisfied), then, those workers satisfies the skill and cost threshold with the optimization objective value of $\leq 2\alpha$. Next, notice that algorithm `ApprxGrp` overall attempts to return the smallest distance α' for which `GrpDia(`α`')` returns a non-empty set, as it performs a binary search over the sorted list of distance values (where distance is sorted in smallest to largest). Therefore, any group of workers returned by `ApprxGrp` satisfies the skill and cost threshold value and $DiaDist(\mathcal{G})$ is at most 2-times worse than the optimal. Hence the approximation factor holds. □

Lemma 1. `Cons-k-Cost-ApproxGrp` *is polynomial.*

Proof. Under a constant number of k-costs, subroutine `GrpCandidateSet` will accept a polynomial computation time of $O(p+1)^{mk}$ at the worst case, where p is the maximum number of workers in one of the k buckets ($p = O(n)$). Subroutine `GrpDia` runs for all n workers at the worst case, and there is a maximum number of $log_2|\mathcal{L}|$

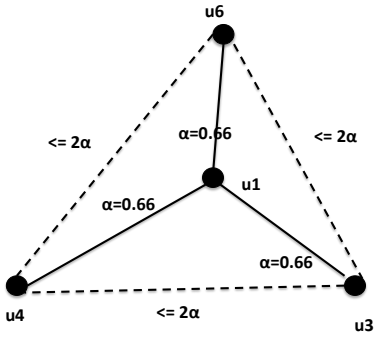


Figure 2.4: An instantiation of `GrpDia(0.66)` using the example in Section 2.2. The clique involving u_1, u_3, u_4, u_6 can not have an edge with distance $> 2 \times 0.66$, due to the triangle inequality property.

calls to `GrpDia` from the main function ($|\mathcal{L}| = O(n^2)$). Therefore, the asymptotic complexity of `Cons-k-ApproxGrp` is $O(n \times log_2|\mathcal{L}| \times (p+1)^{mk})$, which is polynomial.

□

2.5.4 Optimal Algorithm OptGrp

Subroutine `GrpCandidateSet` leaves enough intuition behind to design an instance optimal algorithm that works well in practice. It calls subroutine `GrpCandidateSet` with the actual worker set $\mathcal{U}$ and the task t. For `OptGrp`, the tree is constructed in depth-first-fashion inside `GrpCandidateSet` and all valid solutions from the subroutine are returned to the main function. The output of `OptGrp` is that candidate set of workers returned by `GrpCandidateSet` which has the smallest largest edge. When run on the example in Section 2.2, this `OptGrp` returns $\mathcal{G} = \{u_1, u_2, u_3, u_5, u_6\}$ with objective function value 1.0.

Furthermore, when workers wages are discretized into k buckets, `OptGrp` could be modified as described in Section 2.5.2 and is referred to as `Cons-k-Cost-OptGrp`.
Theorem 4. *Algorithm* `OptGrp` *returns optimal answer.*

Proof. Algorithm `OptGrp` invokes the subroutine `GrpCandidateSet`. Notice that `GrpCandidateSet` operates in the spirit of the branch-and-bound technique [39] to efficiently explore the search space and avoid unnecessary computations. `GrpCandidateSet` exploits the upper bound of cost and lower bound of skill to prune out all unnecessary branches of the search tree, as shown in Figure 2.1 and Figure 2.2. However, this subroutine returns all valid worker groups to `OptGrp`, and then, the main function selects the group with the smallest longest edge (i.e., smallest diameter value), and minimizes the objective function. Therefore, `OptGrp` is instance optimal, i.e., it returns the group of workers with the smallest diameter distance, while satisfying the skill and cost threshold. Therefore, `OptGrp` returns optimal answer. □

Lemma 2. `Cons-k-Cost-OptGrp` *is polynomial.*

Proof. Under a constant number of k-costs, subroutine `GrpCandidateSet` will accept a polynomial computation time of $O(n+1)^{mk}$ at the worst case. Once the subroutine returns all valid answers, the main function will select the one that has the smallest diameter. Therefore, the computation time of `Cons-k-Cost-OptGrp` is dominated by the computation time of the subroutine `GrpCandidateSet`. Therefore, Algorithm `Cons-k-OptGrp` runs in polynomial time of $O((p+1)^{mk}$. □

2.6 Enforcing Upper Critical Mass : SPLT

When `Grp` results in a large unwieldy group $\mathcal{G}$ that may struggle with collaboration, it needs to be partitioned further into a set of sub-groups in the `Splt` phase to satisfy the *upper critical mass* (K) constraint. At the same time, if needed, the workers across the subgroups should still be able to effectively collaborate. Precisely, these intuitions are further formalized in the `Splt` phase.

Definition 2. `Splt`: *Given a group $\mathcal{G}$, decompose it into a disjoint set of subgroups $(G_1, G_2, \ldots, G_x)$ such that $\forall_i |G_i| \leq K$, $\sum_i |G_i| = |\mathcal{G}|$ and the aggregated all pair inter group distance $\Sigma_{\forall_{G_i,G_j \in \mathcal{G}}} SumInterDist(G_i, G_j)$ is minimized.*

Theorem 5. *Problem* `Splt` *is NP-hard.*

Proof. Given a group $\mathcal{G}$, an upper critical mass constraint K, and a real number X, the decision version of the `Splt` is whether $\mathcal{G}$ can be decomposed to a set of subgroups such that the aggregated distances across the subgroups is X and the size of each subgroup is $\leq K$. The membership verification of `Splt` is clearly polynomial.

To prove NP-hardness, we reduce the `Minimum Bisection` [40] which is known to be NP-hard to an instance of `Splt` problem.

Given a graph $G(V, E)$ with non-negative edge weights the goal of Minimum Bisection problem is to create 2 non-overlapping partitions of equal size, such that the total weight of cut is minimized. The hardness of the problem remains, even when the graph is complete [40].

Given a complete graph with n' nodes, the decision version of the `Minimum Bisection` problem is to see whether there exists a 2 partitions of equal size, such that the total weight of the cut is X'. We reduce an instance of `Minimum Bisection` to an instance of `Splt` as follows: the complete graph represents the set of workers with non-negative edges as their distance and we wish to decompose this group to two sub-groups, where the upper critical mass is set to be $K = n'/2$. Now, the objective is to form the sub-groups with the aggregated inter-distance of X', such that there exists a solution to the `Minimum Bisection` problem, if and only if, a solution to our instance of `Splt` exists. □

Proposed Algorithm for Splt: Since the ILP for `Splt` can be very expensive, our primary effort remains in designing an alternative strategy that is more efficient,

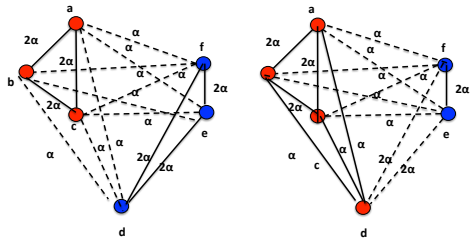


Figure 2.5: Balanced Partitioning in `SpltBOpt` when the distance satisfies triangle inequality for a graph with 6 modes. The left hand side figure has two partitions($\{a, b, c\}, \{d, e, f\}$) with 3-nodes in each (red nodes create one partition and blue nodes create another). The intra-partion edges are drawn solid, whereas, inter-partition edges are drawn as dashed. Assuming $K = 4$, in the right hand side figure, node d is moved with a, b, c. This increases the overall inter-partition weights, but is bounded by a factor of 2.

that allows provable bounds on the result quality. We take the following overall direction: imagine that the output of `Grp` gives rise to a large group $\mathcal{G}$ with n' workers, where $n' > K$. First, we determine the number of subgroups x and the number of workers in each subgroup G_i. Then, we attempt to find optimal partitioning of the n' workers across these x subgroups that minimizes the objective function. We refer to this as `SpltBOpt` which is the *optimal balanced partitioning* of $\mathcal{G}$. For the running example in Section 2.2, this would mean creating 2 subgroups, G_1 and G_2, with 3 workers in one and the remaining 2 in the second subgroup using the workers u_1, u_2, u_3, u_5, u_6, returned by `ApprxGrp`.

For the remainder of the section, we investigate how to find `SpltBOpt`. There are intuitive as well as logical reasons behind taking this direction. Intuitively, lower number of subgroups gives rise to overall smaller objective function value (note that the objective function is in fact 0 when $x = 1$). More importantly, as Lemma 3 suggests, under certain conditions, `SpltBOpt` gives rise to provable theoretical results for the `Splt` problem. Finding the approximation ratio of `SpltBOpt` for arbitrary number of partitions is deferred to future work.

Lemma 3. `SpltBOpt` *has 2-approximation for the* `Splt` *problem, if the distance satisfies triangle inequality, when* $x = \lceil \frac{n'}{K} \rceil = 2$.

Proof. Sketch: For the purpose of illustration, imagine that a graph with n' nodes is decomposed into two partitions. Without loss of generality, imagine partition-1 has n_1 nodes and partition-2 has n_2 nodes, where $n_1 + n_2 = n'$ with total weight of w'. Let K be the upper critical mass and assume that $K > n_1, K > n_2$. For such a scenario, `SpltBOpt` will move one or more nodes from the lighter partition to the heavier one, until the latter has exactly K nodes (if both partitions have same number of nodes then it will choose the one which gives rise to overall lower weight). Notice, the worst case happens, when some of the intra-edges with higher weights now become inter edges due to this balancing act. Of course, some inter-edges also gets knocked off and becomes intra-edges. It is easy to notice that the number of inter-edges that gets knocked off is always larger than that of the number of inter-edges added (because the move is always from the lighter partition to the heaver one). The next argument we make relies heavily on the triangle inequality property. At the worst case, every edge that gets added due to balancing, could at most be twice the weight of an edge that gets knocked off. Therefore, an optimal solution of `SpltBOpt` has 2-approximation factor for the `Splt` problem.

An example scenario of such a balancing has been illustrated in Figure 2.5, where $n_1 = n_2 = 3, K = 4$. Notice that after this balancing, three inter-edges get deleted (ad,bd,cd), each of weight α and two inter-edges get added, where each edge is of weight 2α. However, the approximation factor of 2 holds, due to the triangle inequality property. □

Even though the number of subgroups (aka partitions) is $\lceil \frac{n'}{K} \rceil$ with K workers in all but last subgroup, finding an optimal assignment of the n' workers across those

subgroups that minimizes the objective function is NP-hard. The proof uses an easy reduction from [41]. We start by showing how the solution to `SpltBOpt` problem could be bounded by the solution of a slightly different problem variant, known as `Min-Star` problem [41].

Definition 3. `Min-Star` *Problem: Given a group $\mathcal{G}$ with n' workers, out of which each of x workers $(u_1, u_2, \ldots, u_x)$, represents a center of a star sub-graph (each sub-graph stands for a subgroup), the objective is to partition the remaining $n' - x$ workers into one of these x subgroups $G_1, G_2, \ldots, G_x$ such that $\sum_{i=1}^{x} k_i dist(u_i, \cup_{j \neq i} G_j)$ $+ \sum_{i<j} k_i k_j dist(u_i, u_j)$ is minimized, where k_i is the total number of workers in subgroup G_i.*

Intuitively, `Min-Star` problem seeks to decompose the worker set into x subgroups, such that u_i is the center of a star graph for subgroup G_i, and for a fixed set of such workers $\{u_1, u_2, \ldots, u_x\}$, the contribution of u_i to the objective function is proportional to the sum of distances of a star subgraph rooted at u_i.

Solving `Min-Star`:Algorithm `Min-Star-Partition`: The pseudocode is listed in Algorithm 3 and additional details can be found in [41]. The key insight behind this algorithm is the fact that for a fixed set of workers $\{u_1, u_2, \ldots, u_x\}$, the second term of the objective function $\sum_{i<j} k_i k_j dist(u_i, u_j)$ is a constant. Furthermore, this expression could only take $\binom{n'}{x}$ distinct values corresponding to all possible combination of how the workers $\{u_1, u_2, \ldots, u_x\}$ are chosen from the group $\mathcal{G}$ with n' workers. Hence for a fixed set of workers, the objective now reduces to finding an optimal subgroups $G_1, \ldots, G_x$ that minimizes the first expression. Interestingly, this expression corresponds exactly to a special case of the popular *transportation problem* [42] that could be solved optimally with time complexity $O(n')$ [41]. We refer to [41] for further details.

Algorithm 3 Algorithm `Min-Star-Partition`

Require: Group $\mathcal{G}$ with n' workers and upper critical mass K

1: $x = \lceil \frac{n'}{K} \rceil$

2: **for all** subset $\{u_1, \ldots, u_x\} \subset \mathcal{G}$ **do**

3: Find optimal subgroups $\{G_1, \ldots, G_x\}$ for $\{u_1, \ldots, u_x\}$ by formulating it as transportation problem

4: Evaluate objective function for $\{G_1, \ldots, G_x\}$

5: **end for**

6: **return** subgroups $\{G_1, \ldots, G_x\}$ with least objective function

Finally, the objective function of the `SpltBOpt` is computed on the optimal partition of each instance of the transportation problem, and the one with the least value is returned as output. When run using $\mathcal{G} = \{u_1, u_2, u_3, u_5, u_6\}$ from `ApprxGrp`, this algorithm forms subgroups $G_1 = \{u_1, u_2, u_5\}$ and $G_2 = \{u_3, u_6\}$ with objective function value 3.89.

Theorem 6. *Algorithm for* `Min-Star-Partition` *has a 3-approximation for* `SpltBOpt` *problem.*

Proof. sketch: This result is a direct derivation of the previous work [41]. Previous work [41] shows that `Min-Star-Partition` obtains a 3-approximation factor for the `Minimum k-cut` problem. Recall that `SpltBOpt` is derived from `Minimum k-cut` by setting each partition size (possibly except the last one) to be equal with K nodes, giving rise to a total number of $\lceil \frac{n'}{K} \rceil$ partitions. After that, the result from [41] directly holds. □

Lemma 4. `Min-Star-Partition` *is polynomial.*

Proof. It can be shown that `Min-Star-Partition` takes $O(n'^{x+1})$ time, as there are $O(n'^{x})$ distinct transportation problem instances (corresponding to each one of $\binom{n'}{x}$

combinations), and each instance can be solved in $O(n')$ [41] time. Since, x is a constant, therefore, the overall running time is polynomial. □

2.7 Experiments

We describe our real and synthetic data experiments to evaluate our algorithms next. The real-data experiments are conducted on Amazon Mechanical Turk(AMT). The synthetic-data experiments are conducted using a parametrizable crowd simulator.

2.7.1 Real Data Experiments

Two different collaborative crowdsourcing applications are evaluated using AMT: i) Collaborative Sentence Translation (CST), ii) Collaborative Document Writing (CDW).

Workers: A pool of 120 workers participate in the sentence translation study, whereas, a different pool of 135 workers participate in the second one. Hired workers are directed to our website where the actual tasks are undertaken.

Pair-wise Affinity Calculation: Designing complex personality test [30] to compute affinity is beyond the scope of this work. We instead choose some simple factors to compute affinity that have been acknowledged to be indicative factors in prior works [18]. We calculate affinity in two ways - 1) `Affinity-Age`: age based calculation discretizes workers into different age buckets and assigns a value of 1 to a worker-pair, if they fall under the same bucket, 0 otherwise. 2) `Affinity-Region`: assigns a value of 1, when two workers are from the same country and 0 otherwise.

Evaluation Criteria: - The overall study is designed to evaluate: (1) Effectiveness of the proposed optimization model, (2) Effectiveness of affinity calculation techniques, and (3) Effect of different upper critical mass values.

Algorithms: We compare our proposed solution with other baselines: (1) To evaluate the first criteria, we use the ILP described in Section 2.4 against an alternative `Aff-Unaware` Algorithm [43]. The latter assigns workers to the tasks considering skill and cost but ignoring affinity. Since, ILP outputs optimal task assignment, we refer to this as `Optimal`(2) `Optimal-Affinity-Age` and `Optimal-Affinity-Region` are two variants of `Optimal` that use two different affinity calculation methods (`Affinity-Age` and `Affinity-Region` respectively) and are compared against each other to evaluate the second criteria. (3) `CrtMass-Optimal-K` assigns workers to tasks based on the optimization objective and varies different upper critical mass values K, which are also compared against each other for different K.

Overall user-study design: The overall study is conducted in 3-stages : (1) *Worker Profiling*: in stage-1, we hire workers and use pre-qualification tests using "gold-data" to learn their skills. We also learn other human factors as described next.(2) *Worker-to-task Assignment*: in stage-2, a subset of these hired workers are re-invited to participate, where the actual collaborative tasks are undertaken by them.(3) *Task Evaluation*: in stage-3, completed tasks are crowdsourced again to evaluate their quality.

Summary of Results: There are several key takeaways of our user study results. First and foremost, *effective collaboration is central to ensuring high quality results for collaborative complex tasks.* We evaluated 2 different affinity computation models and the results show that the people from same region collaborate more effectively than people in same age group. Interestingly, upper critical mass also has a significance in collaboration effectiveness, consequently, in the quality of the completed tasks. Quality increases from $K = 5$ to $K = 7$, but it decreases with statistical significance when $K = 10$ for `CrtMass-Optimal-`10.

2.7.1.1 Stage 1 - Worker Profiling

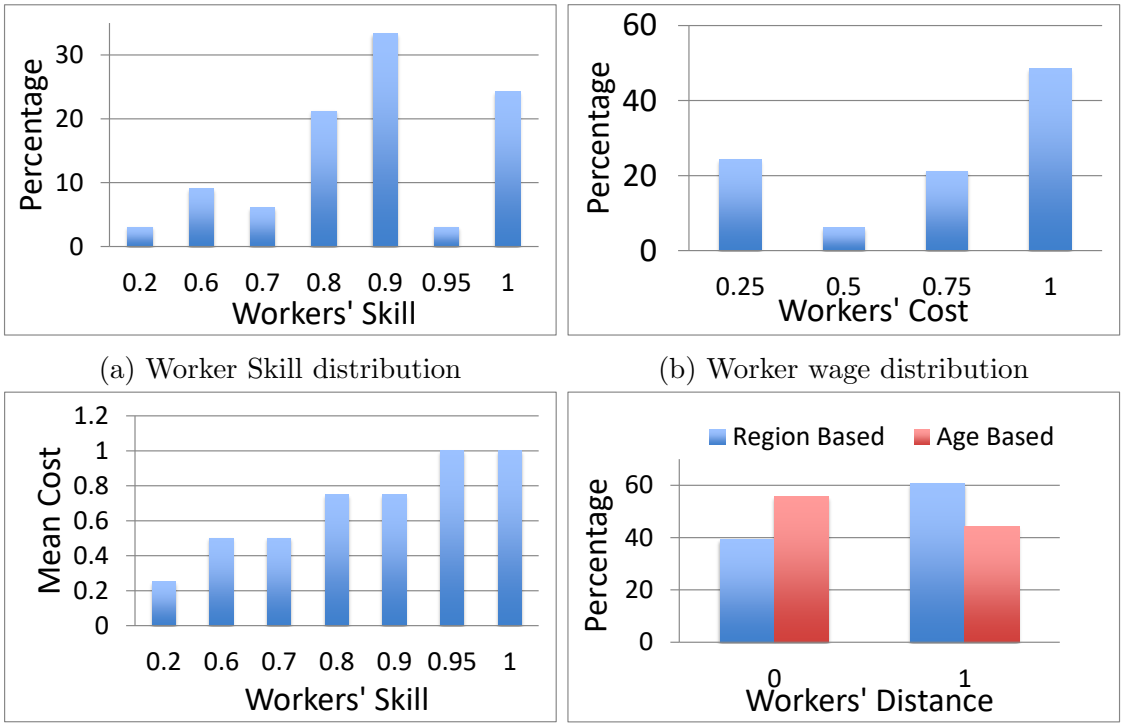
We hire two different sets of workers for sentence translation and document writing. The workers are informed that a subset of them will be invited (through email) to participate in the second stage of the study.

Skill learning for Sentence Translation: We hire 60 workers and present each worker with a 20 second English video clip, for which we have the ground truth translation in 4 different languages: English, French, Tamil, Bengali. We then ask them to create a translation in one of the languages (from the last three) that they are most proficient in. We measure each workers individual skill using Word Error Rate(WER) [44].

Skill learning for Document Writing: For the second study CDW , we hire a different set of 75 workers. We design a "gold-data" set that has 8 multiple choice questions per task, for which the answers are known (e.g. for the MOOCs topic in table 2.4 - one question was, *"Who founded Coursera?"*). The skill of each worker is then calculated as the percentage of her correct answers. For simplicity, we consider only one skill domain for both applications.

Wage Expectation of the worker: We explicitly ask a question to each worker on their expected monetary incentive, by giving them a high level description of the tasks that are conducted in the second stage of the study. Those inputs are recorded and used in the experiments.

Affinity of the workers: Hired workers are directed to our website, where they are asked to provide 4 simple socio-demographic information: gender, age, region, and highest education. Workers anonymity is fully preserved. From there, affinity between the worker is calculated using, `Affinity-Age` or `Affinity-Region`.



(a) Worker Skill distribution

(b) Worker wage distribution

(c) Distance Distribution-Region

(d) Distance Distribution-Age

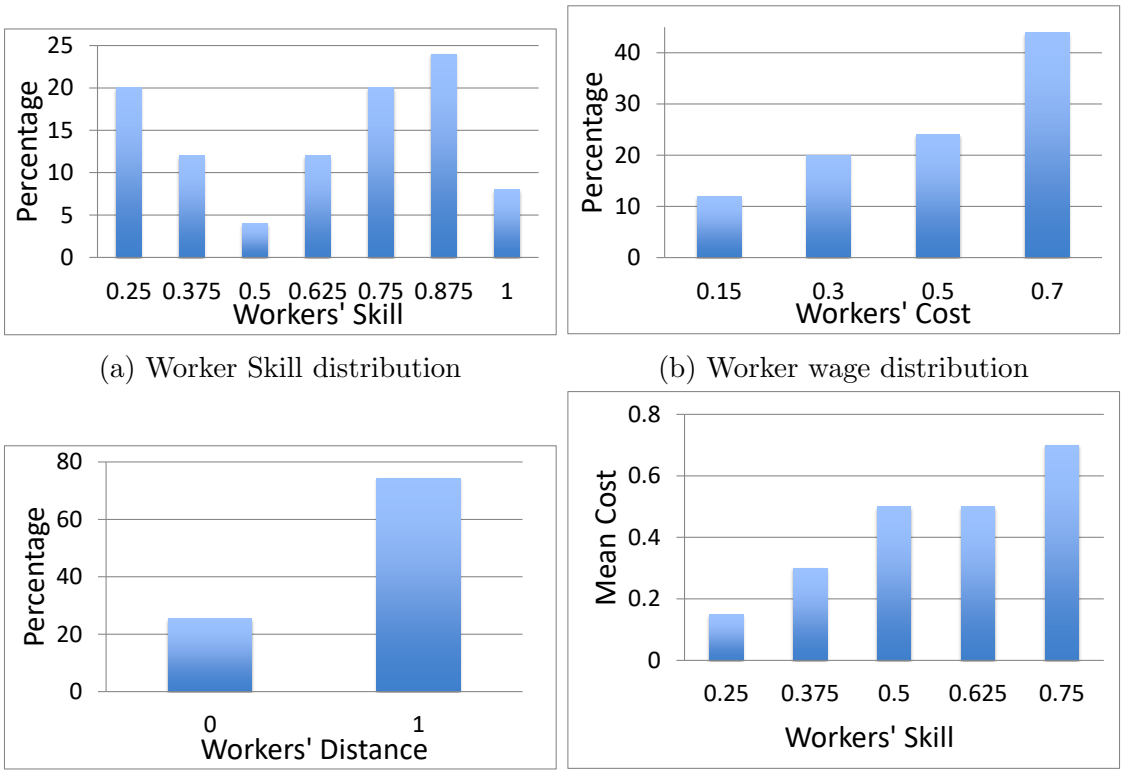
Figure 2.6: Worker profile distributions for the Sentence Translation Tasks in Section 2.7.1

Figure 2.6 and Figure 2.7 contain detailed workers profile distribution information.

2.7.1.2 Stage 2 - Worker-to-Task Assignment

Once the hired workers are profiled, we conduct the second and most important stage of this study, where the actual tasks are conducted collaboratively.

Collaborative Sentence Translation(CST): We carefully choose three English documentaries of suitable complexity and length of about 1 minute for creating subtitle in three different languages - French, Tamil, and Bengali. These videos are chosen from YouTube with titles: (1) Destroyer, (2) German Small Weapons, (3)British Aircraft TSR2.



(a) Worker Skill distribution

(b) Worker wage distribution

(c) Worker distance distribution

(d) Correlation between skill and wage

Figure 2.7: Worker profile distributions for the Collaborative Document Writing in Section 2.7.1

Collaborative Document Writing (CDW): Three different topics are chosen for this application: 1) MOOCs and its evolution, 2) Smart Phone and its evolution, 3) Top-10 places to visit in the world.

The skill and cost requirements of each tasks are described in the Table 2.4. These values are set by involving domain experts and discussing the complexity of the tasks with them.

Collaborative Task Assignment for CST: We set up 2 different worker groups per task and compare two algorithms `Optimal-CST` and `Aff-Unaware-CST` to evaluate the effectiveness of proposed optimization model. We set up additional 2 different worker groups for each task to compare `Optimal-Affinity-Region`

Task Name	Skill	Cost	Critical Mass
CST1- Destroyer	3.0	$5.0	5,7,10
CST2- German Weapons	4.0	$5.0	5,7,10
CST3 - British Aircraft	3	$4.5	5,7,10
CDW1- MOOCs	5	$3	5,7,10
CDW2- Smartphone	5	$3	5,7,10
CDW3- top-10 place	5	$3	5,7,10

Table 2.4: Description of different tasks; the default upper critical mass value is 5. Default affinity calculation is region based.

with `Optimal-Affinity-Age`. Finally, we set up 3 additional groups per task to compare the effectiveness of critical mass and compare `CrtMass-Optimal-5`, `CrtMass-Optimal-7`, `CrtMass-Optimal-10`. This way, a total of 15 groups are created. We instruct the workers to work incrementally using other group members contribution and also leave comment as they finish the work. These sets of tasks are kept active for 3 days.

Collaborative Task Assignment for CDW: An similar strategy is adopted to collaboratively edit a document within 300 words, using the quality, cost, and critical mass values of the document editing tasks, described in Table 2.4.

2.7.1.3 Stage 3 - Task Evaluation

Collaborative tasks, such as knowledge synthesis, are often subjective. An appropriate technique to evaluate their quality is to leverage the *wisdom of the crowds*. This way a diverse and large enough group of individuals can accurately evaluate information to nullify individual biases and the herding effect. Therefore, in this stage we *crowdsource the task evaluation* for both of our applications.

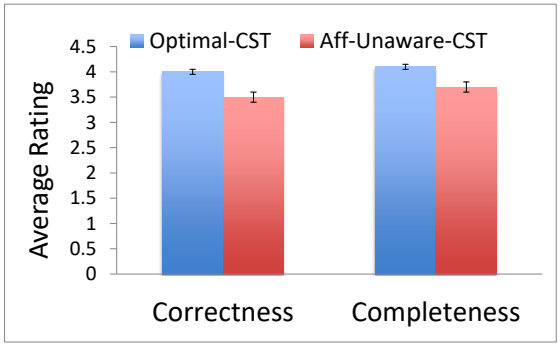
For the first study of Sentence Translation (CST), we have taken 15 final outcomes of the translation tasks as well as the original video clips and then set up as 3 different HITs in AMT. The first HIT is designed to evaluate the optimization model,

Average Rating							
Task	Algorithm	Compl.	Gram.	Neutra.	Clarity	Time	AV
MOOCs	`Optimal-CDW`	4.6	4.5	4.3	4.3	4.3	3.7
	`Aff-Unaware-CDW`	4.1	4.2	4.2	3.9	3.9	3.0
	`CrtMass-Optimal-10`	4.0	4.1	4.2	3.9	3.9	3.5
Smartphone	`Optimal`	4.8	4.6	4.7	4.1	4.2	4.2
	`Aff-Unaware`	4.1	4.1	4.2	4.2	3.9	3.3
	`CrtMass-Optimal-10`	4.0	3.9	3.8	4.1	3.9	3.3
Top-10 places	`Optimal`	4.4	4.2	4.3	4.2	4.3	4.3
	`Aff-Unaware`	3.9	3.8	3.7	3.6	3.3	2.9
	`CrtMass-Optimal-10`	3.9	4.0	4.1	4.0	3.9	3.9

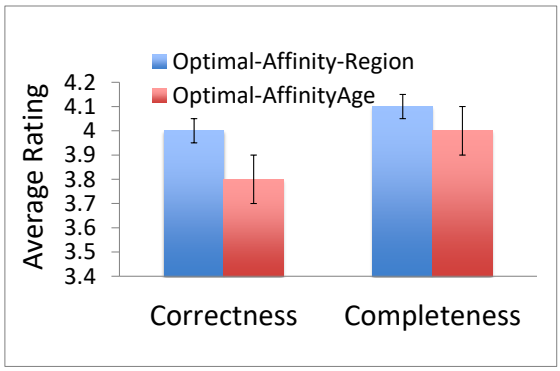
Table 2.5: **Stage 3 results of document writing application in Section 2.7.1:** Quality assessment on the completed tasks of Stage-2 is performed by a new set of 60 AMT workers on a scale of $1-5$. For all three tasks, the results clearly demonstrate that effective collaboration leads to better task quality. Even though all three groups (assigned to the same task) surpass the skill threhsold and satisfy the wage limit, however, our proposed formalism `Optimal` enables better team collaboration, resulting in higher quality of articles.

the second one to evaluate two different affinity computation models, and the final one to evaluate the effectiveness of upper critical mass. We assign 20 workers in each HIT, totaling 60 new workers. We evaluate the completed tasks in two quality dimensions, as identified by prior work [18] - 1. correctness of translation. 2.completeness of translation. The workers are asked to rate the quality in a scale of $1-5$ (higher is better) without knowing the underlying task production algorithm. Then, we average these ratings which is similar to obtaining the viewpoint of an average reader. The CST results of different evaluation dimensions are presented in Figure 2.8.

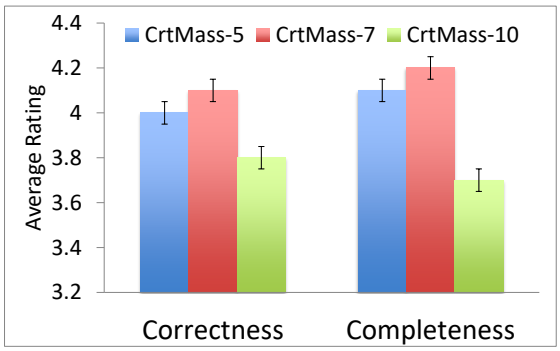
A similar strategy is undertaken for the CDW application, but the quality is assessed using 5 key different quality aspects, as proposed in prior work [45]. They are Completeness(Compl), Grammar, Neutrality(Neut), Clarity, Time and Added Value(AV). The results are summarized in Table 2.5. Both these results indicate that, indeed, our proposed model successfully incorporates different elements that are essential to ensure high quality in collaborative crowdsourcing tasks.



(a) Optimization Model



(b) Affinity Calculation



(c) Upper Critical Mass

Translation	Language
The destroyers are among the fastest and most deadly worships ever built. Mounting a powerful ? of offensive and defensive weapons, they can serve equally well as escorts for other vessels more in form of a ? in their own right.	English
Les destroyers sont parmi les plus rapides et les plus meurtrières jamais construits. Montage d'un ? puissant offensives et défensives, ils peuvent tout aussi bien servir d'escortes pour les autres navires. Au début, les navires étais conçus exclusivement pour détruire les bateaux ?.	French
Les destructeurs sont parmi les plus rapides et les plus meurtriers jamais construits. Montage d'un puissant arsenal d'armes défensives et offensives, ils peuvent tout aussi bien servir d'escorte aux autres navires, plus sous forme de (formidable navire d'attaque?) dans leur propre droit. Au début, les navires étaient conçus exclusivement pour détruire les bateaux Paxon.	French

(d) A French Translation Sample

Figure 2.8: **Stage 3 results of sentence translation:** Collected data with statistical significance (standard error) is presented. These results clearly corraborate that our affinity-aware optimization model `Optimal-CST` outperforms its affinity-unaware counterpart [43] with statistical significance across both quality dimensions.`Optimal-Affinity-Region` apperas to outeprform `Optimal-Affinity-Age` in "correctness". The results of `CrtMass-Optimal-10` clearly appers to be less effective than the other two, showing some anecdotal evidence that group size is important in collaborative crowdsourcing applications.

2.7.2 Synthetic Data Experiments

The purpose of this experiments is to show that our proposed algorithms perform well both qualitatively and efficiently. Besides evaluating the algorithms for our staged solution `Grp&Splt`, we also evaluate the algorithms for the `grp` stage. This will help us illustrate the fact that our algorithms for `Grp` create effective collaborative groups. This is also essential for the performance of `Splt` stage.

We conduct our synthetic data experiments on an Intel core I5 with 6 GB RAM. We use IBM CPLEX 12.5.1 for the ILP. A crowd simulator is implemented in Java to generate the crowdsourcing environment. All numbers are presented as the average of three runs.

Simulator Parameterization: The simulator parameters presented below are chosen akin to their respective distributions, observed in our real AMT populations.

1. *Simulation Period* - We simulate the system for a time period of 10 days, i.e. 14400 simulation units, with each simulation unit corresponding to 1 minutes. With one task arriving in every 10 minutes, our default setting runs 1 day and has 144 tasks.

2. *# of Workers* - default is 100, but we vary $|\mathcal{U}|$ upto 5000 workers.

3. *Workers skill and wage* - The variable u_{d_i} in skill d_i receives a random value from a normal distribution with the mean set to 0.8 and a variance 0.15. Worker's wages are also set using the same normal distribution.

4. *Task profile* - The task quality Q_i, as well as cost C is generated using normal distribution with specific mean 15 and variance 1 as default. Unless otherwise stated, each task has a skill.

5. *Distance* - Unless otherwise stated, we consider distance to be metric and generated using Euclidean distance.

6. *Critical Mass* - the default value is 7.

7. *Worker Arrival, Task Arrival* - By default, both workers and tasks arrive following a Poisson process, with an arrival rate of $\mu = 5$/minute 1/10 minute, respectively.

Implemented Algorithms: Here we first describe the algorithms for `Grp` stage.

1. `ApprxGrp:` We implement the algorithm `ApprxGrp`, described in Section 2.5.3.

2. `Cons-k-AG:` This is a variant of the algorithm `ApprxGrp` referred to as `Cons-k-cost-ApprxGrp`, described in Section 2.5.3. We set the number of cost buckets k to 15.

3. `GrpILP:` An ILP designed for `Grp` stage only.

3. `OptGrp:` This is an optimal algorithm that is similar to `GrpILP` both in terms of quality and efficiency. Hence, we decided to omit the results for `OptGrp`.

5. `RandGrp`: We also design an affinity unaware algorithm that finds a set of workers who satisfy skill and cost threshold, but does not optimize affinity.

Here are the list of algorithms for `Grp&Splt`

1. `Overall-ILP:` An ILP, as described in Section 2.4.

2. `Grp&Splt:` Uses `Cons-k- AG` for `Grp` and `Min-Star-Partition` for `Splt`.

3. `RandGrp&GrdSplt:` An alternative implementation. In phase-1, we use `RandGrp`. In phase-2, we partition users greedily into most similar subgroups satisfying critical mass constraint.

6. *No implementation of existing related work:* Due to critical mass constraint, we intend to form a group, further partitioned into a set of subgroups, whereas, no prior work has studied the problem of forming a group along with subgroups, thereby making our problem and solution unique.

Summary of Results: Our synthetic experiments also exhibit many interesting insights. First and foremost, `Grp&Splt` is a reasonable alternative formulation to solve `AffAware-Crowd`, both qualitatively and efficiency-wise, as `Overall-ILP` is not *scalable* and does not converge for more than 20 workers. Second, our proposed approximation algorithms for `Grp&Splt` are both efficient as well as effective, and they significantly outperform other competitors. Finally, our proposed formulation `AffAware-Crowd` is an effective way to optimize complex collaborative crowdsourcing tasks in a real world settings. We first present the overall quality and scalability of the combined `Grp&Splt`, followed by that of `Grp` individually.

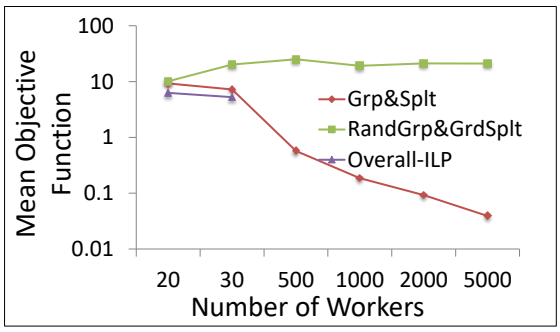


Figure 2.9: `Grp&Splt` : Objective Function varying Number of Workers

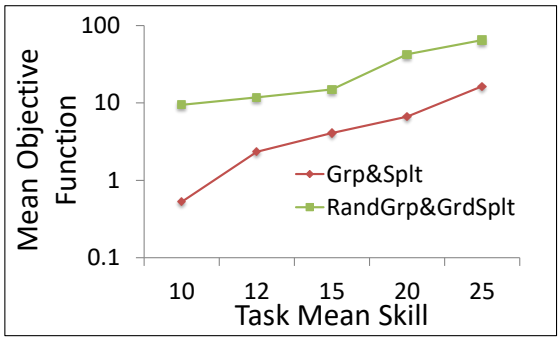


Figure 2.10: `Grp&Splt` : Objective Function varying Task Mean Skill

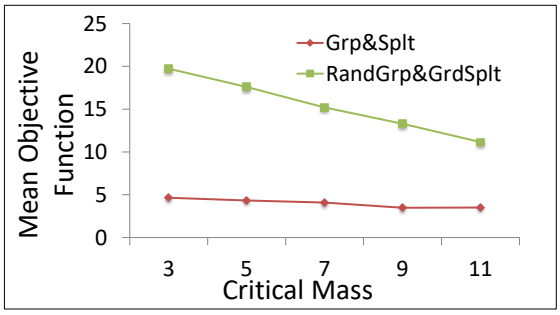


Figure 2.11: `Grp&Splt`: Objective Function varying Critical Mass

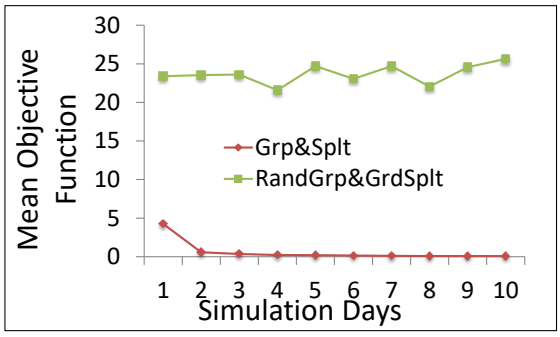


Figure 2.12: `Grp&Splt`:Objective function over Simulation Days

2.7.2.1 Quality Evaluation

We present the quality evaluations next.

2.7.2.1.1 Grp&Splt Quality: The average of overall objective function value, which is the sum of $DiaDist(G)$ and aggregated all pair $SumInterDist()$ across the subgroups, is evaluated and presented as mean objective function value for 144 tasks. `Overall-ILP` does not converge beyond 20 workers.

Varying # of Workers: Figure 2.9 has the results, with mean skill=15 and variance=1, demonstrates that `Grp&Splt` outperforms `RandGrp&GrdSplt` in all the cases, while being very comparable with `Overall-ILP`.

Varying Tasks Mean Skill: With varying mean skill (cost is proportional to skill),

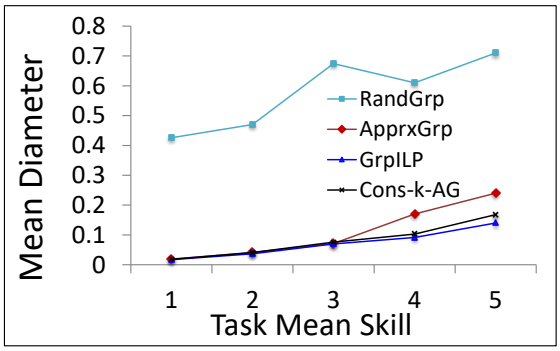


Figure 2.13: `Grp` : Mean Diameter varying Mean Skill

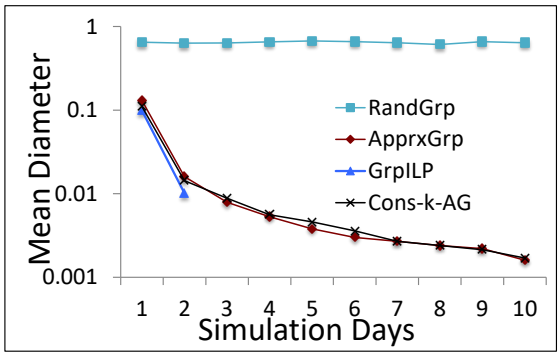


Figure 2.14: `Grp` :Mean Diamter varying Simulation Days

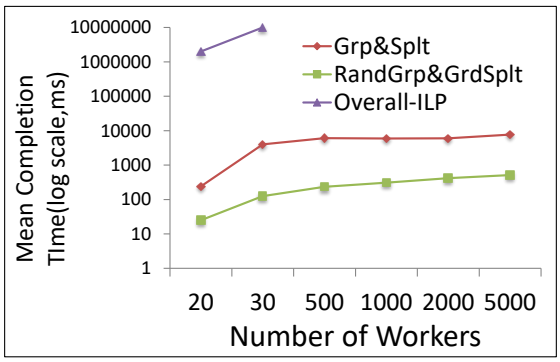


Figure 2.15: `Grp&Splt` : Mean Completion Time varying Number of Workers

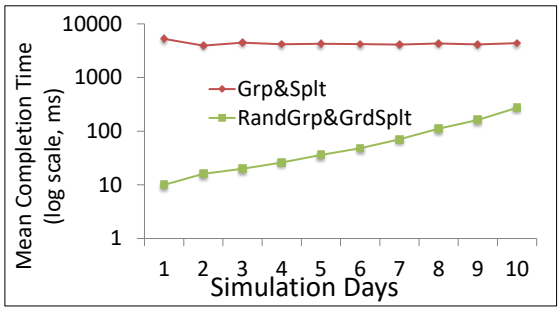


Figure 2.16: `Grp&Splt` : Mean Completion Time varying Simulation Days

Figure 2.10 demonstrates that the objective function gets higher (hence worse) for both the algorithms, as skill/cost requirement increases, while `Grp&Splt` outperforms `RandGrp&GrdSplt`. This intuitively is meaningful, as with increasing skill requirement, the generated group is large, which decreases the workers cohesiveness further.

Varying Critical Mass: As Figure 2.11 shows, with increasing critical mass, quality of both solutions increases, because the aggregated inter-distance across the partition gets smaller due to less number of edges across.

Varying Simulation Period: In Figure 2.12 simulation period is varied, where both workers and tasks arrive based on Poisson process. `Grp&Splt` convincingly out-

performs

`RandGrp&GrdSplt` in this experiment.

2.7.2.1.2 Grp Phase Quality: The objective function is the average `DiaDist()` value.

Varying Task Mean Skill: Figure 2.13 demonstrates that, although `ApprxGrp` and `Cons-k-AG` is 2-times worse than optimal theoretically, but in practice, it is as good as optimal. `GrpILP`.

Varying Simulation Period: Figure 2.14 demonstrates, that, as more workers are active in the system `GrpILP` cannot converge. Hence, we can not get the results for `GrpILP` beyond day-2. But, `ApprxGrp` and `Cons-k-AG` works fine and achieves almost optimal result.

2.7.2.2 Efficiency Evaluation

In this section, we demonstrate the scalability aspects of our proposed algorithms and compare them with other competitive methods by measuring the average completion time of a task. Like above, we first present the overall time for `Grp&Splt` phase, then followed by `Grp` phase.

2.7.2.2.1 Grp&Splt Efficiency: ***Varying # Workers:*** Figure 2.15 demonstrates that our solution `Grp&Splt` is highly scalable, whereas, `Overall-ILP` fails to converge beyond 20 workers. `RandGrp&GrdSplt` is also scalable (because of the simple algorithm in it), but clearly does not ensure high quality.

Varying Task Mean Skill: Akin to previous result, `Grp&Splt` and `RandGrp&GrdSplt` are both scalable,`Grp&Splt` achieves higher quality. We omit the chart for brevity.

Varying Critical Mass: As before, increasing critical mass leads to better effi-

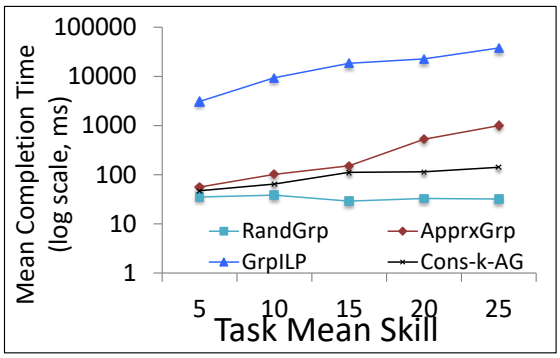


Figure 2.17: `Grp` : Mean Completion Time varying Mean Skill

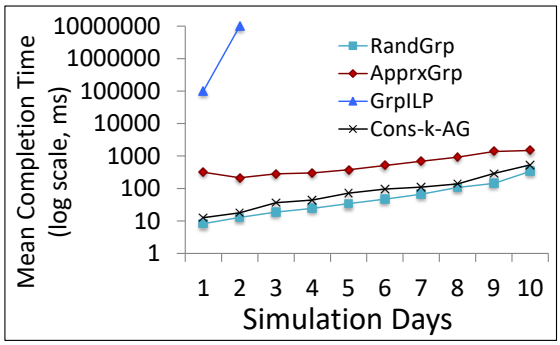


Figure 2.18: `Grp` :Mean Completion Time varying Simulation Days

ciency for the algorithms. We omit the chart for brevity.

Varying Simulation Period: Figure 2.16 demonstrates that `Grp&Splt` is highly scalable in a real crowdsourcing environment, where more and more workers are entering into the system. The results show that `RandGrp&GrdSplit` is also scalable (but significantly worse in quality). But as number of worker increases, efficiency decreases, for both, as expected.

2.7.2.2.2 Grp Phase Efficiency: We evaluate the efficiency of `ApprxGrp` by returning mean completion time for 144 tasks.

Varying Task Mean Skill: As Figure 2.17 demonstrates, `ApprxGrp` outperforms `GrpILP` significantly. As expected, `Cons-k-AG` is more efficient than `ApprxGrp` since it bucketize the cost values. With higher skill threshold, the difference between `RandGrp` and our algorithms becomes even more noticeable.

Varying Simulation Period: Figure 2.18 shows the average task completion time in each day for `ApprxGrp`, `Cons-k-AG`,`GrpILP`, `RandGrp`. Clearly, `GrpILP` is impractical to use as more workers arrive in the system.

++

2.8 Related Work

We discuss how our work is different from a few existing works that discuss the challenges in crowdsourcing complex tasks, as well as traditional team formation problems.

Crowdsourcing Complex Tasks: This type of human based computation [9, 46, 47] handles tasks related to knowledge production, such as article writing, sentence translation, citizen science, product design, etc. These tasks are conducted in groups, are less decomposable compared to micro-tasks (such as image tagging) [11, 48], and the quality is measured in a continuous, rather than binary scale.

A number of crowdsourcing tools are designed to solve application specific complex tasks. *Soylent* uses crowdsourcing inside a word processor to improve the quality of a written article [49]. *Legion*, a real time user interface, enables integration of multiple crowd workers input at the same time [50]. *Turkit* provides an interface to programmer to use human computation inside their programming model [51] and avoids redundancy by using a *crash and return model* which uses earlier results from the assigned tasks. *Jabberwocky* is another platform which leverages social network information to assign tasks to workers and provide an easy to use interface for the programmers [52]. *CrowdForge* divides complex task into smaller sub-tasks akin to map-reduce fashion [19]. *Turkomatic* introduces a framework in which workers aid requresters to break down the workflow of a complex task and thereby aiding to solve it using systematic steps [53].

The common aspects of these works is that they study the problem of decomposing a complex task into simpler tasks, which can be solved by independent workers. On the contrary, we focus on optimization based task assignment for complex task which may not indivisible. A preliminary work discusses modular team structures for complex crowdsourcing tasks, detailing however more on the application cases,

and not on the computational challenges[54]. One prior work investigates how to assign workers to the task for knowledge intensive crowdsourcing [43] and its computational challenges. However, this former work does not investigate the necessity nor the benefit of *collaboration*. Consequently, the problem formulation and the proposed solutions are substantially different from the one studied here.

Automated Team Formation: Although tangentially related with crowdsourcing, automated team formation is widely studied in computer assisted cooperative systems. [29] forms a team of experts in social networks with the focus of minimizing coordination cost among team members. Although their coordination cost is akin to our affinity, but unlike us, the former does not consider multiple skills. Team formation to balance workload with multiple skills is studied later on in [38] and multi-objective optimization on coordination cost and balancing workload is also proposed [23, 55], where coordination cost is posed as a constraint. Density based coordination is introduced in [32], where multiple workers with similar skill are required in a team, such as ours. Formation of team with a leader (moderator) is studied in [56]. Minimizing both communication cost and budget while forming a team is first considered in [57, 58]. The concept of pareto optimal groups related to the skyline research is studied in [57].

While several elements of our optimization model are actually adapted from these related work, there are many stark differences that precludes any easy adaptation of the team formation research to our problem. Unlike us, none of these works considers *upper critical mass* as a group size constraint, that forms a group multiple subgroups, which makes the former algorithms inapplicable in our settings. Additionally, none of these prior work studies our problem with the objective to maximize affinity with multiple skills and cost constraints. In [59], authors demonstrate empirically that the utility is decreased for larger teams which validates our approach

to divide group into multiple sub-groups obeying *upper critical mass*. However, no optimization is proposed to solve the problem.

In summary, principled optimization opportunities for complex collaborative tasks to maximize collaborative effectiveness under quality and budget constraints is studied for the first time in this work.

2.9 Conclusion and Future Work

We borrow our motivation from the fact that the aspect of collaboration naturally fits into solving many complex tasks. To that end, we develop a framework which aims to find the optimal group of workers for collaborative tasks. We identify both individual and group based human factors (i.e. affinity, critical mass) that are significant for successful completion of collaborative tasks. We propose a set of optimization objectives, which maximize the collaboration, while appropriately considering the complex interplay of human factors. We show that our overall problem is NP-complete, and then provide a two-staged solution to our problem. Furthermore, we show that the problem at each individual stage is also NP-Complete. This prompts us to design efficient approximation algorithm for both of the stages. Our extensive experiments on real data collected from Amazon Mechanical Turk show the superiority of our algorithms on their respective baseline counterparts. In the next chapter, we show the implementation of our task assignment algorithm to an existing crowdsourcing platform.

CHAPTER 3

Collaborative Crowdsourcing with Crowd4U

3.1 Introduction

We propose to demonstrate Crowd4U, a prototype system for the deployment of collaborative tasks. Unlike other crowdsourcing frameworks, collaboration is a central tenet of Crowd4U and permeates all its features. Crowd4U provides support for end-to-end deployment of collaborative tasks and enables task decomposition, worker-to-task assignment, and effective worker collaboration during task completion. Requesters can specify collaborative tasks in a declarative manner. Tasks are then assigned to groups of workers taking human factors [60, 61] and worker-to-worker affinity into account. Crowd4U incorporates diverse task decomposition approaches and handles various worker collaboration schemes that ensure effective result coordination. We will demonstrate 3 scenarios that represent best emerging collaborative crowdsourcing: text translation, citizen journalism, and surveillance in the aftermath of a disaster.

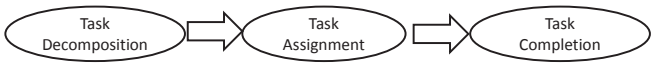


Figure 3.1: Deployment process for complex collaborative tasks. Result coordination is achieved via worker collaboration schemes in task completion.

Comparison with Other Frameworks: While existing research has investigated task assignment for crowdsourcing in diverse contexts, it often ignored the aspect of collaboration among workers, that is central to the success of complex tasks. For example, well-known crowdsourcing systems such as Deco [62], CrowdDB [63] that focus on enabling important primitives in database query processing through micro-tasks that are performed by individual workers whose responses are then aggregated. Other frameworks such as PyBossa [64] or Hive [65] are more generic and can be used for a variety of tasks but are still bound by fixed workflows and micro-tasks with no distinct notion of collaboration. Frameworks such as Argonaut [66] enables complex context-heavy data processing tasks (dubbed "macrotasks") but are still performed by individual workers and not by teams. In contrast, Crowd4U is *declarative, generic* and *collaboration-aware*.

Collaboration Types: To illustrate the range of applications that can be enabled by Crowd4U, we organize our demonstration based on the three collaboration types specified above through three different demonstration scenarios. We use the application of video subtitle generation and translation to highlight sequential collaboration where workers improve the contribution of each other. Support for simultaneous collaboration is exemplified by a citizen journalism application where workers generate report on a specific topic by working in parallel. Hybrid collaboration is showcased by surveillance tasks where some workers contribute to fact collection in a sequence, and correcting each others' observations, while others provide testimonials separately and simultaneously.

The rest of the chapter is organized as follows. We first describe how the architecture of Crowd4U was revisited to enable collaboration (Section 3.2.1). We then describe how task assignment is enabled under different worker coordination schemes (Section 3.2.2).

3.2 Crowd4U for Collaboration

Crowd4U is a non-profit all-academic open crowdsourcing platform that has been used in a variety of domains such as libraries, natural disasters, digital archives, cognitive science, and health informatics. Crowd4U was initially designed to support micro-tasks wherein a requester posts a task that is achieved by one worker at a time [10]. In this section, we describe how the platform has been revisited to support collaboration. We first describe the extended architecture, then we explain how tasks are assigned to workers, and finally, show how workers complete tasks.

3.2.1 Collaboration Architecture

Figure 3.2 shows the major components of Crowd4U. A requester who wants to register tasks into Crowd4U writes a *project description* in CyLog, a rule-based declarative language for crowdsourcing applications with complex data flows [67]. Crowd4U also provides tools to help requesters generate CyLog rules by allowing them to define tasks with a form-based user interface and spreadsheets. The rules describing a task are interpreted and executed by the CyLog processor, which dynamically generates and registers tasks into the task pool. We now describe how tasks are processed.

Crowd4U can use any task decomposition algorithm to break a complex task into micro-tasks. Its key innovations are the reliance of a task-specific assignment algorithm to find the best group of workers to complete a task, and the implementation

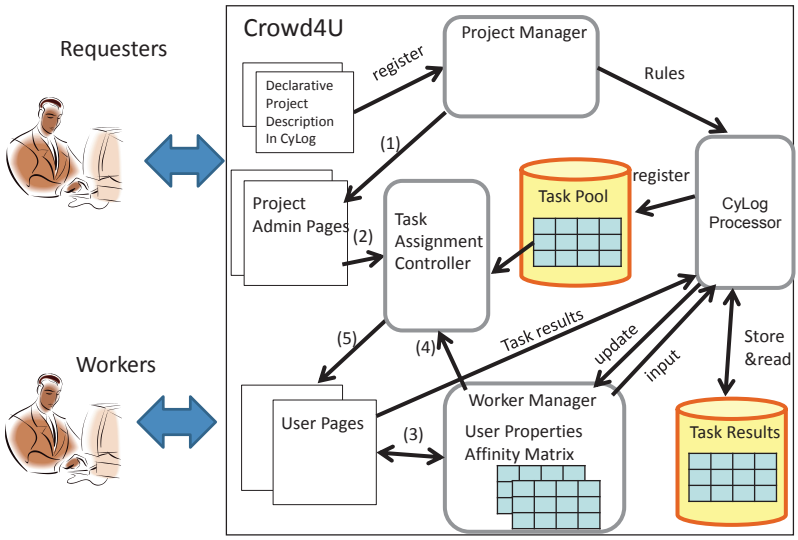


Figure 3.2: Crowd4U architecture and workflow for collaborative task assignment

of sequential, simultaneous and hybrid worker collaboration, to ensure effective result coordination. We describe how task assignment and result coordination are achieved.

3.2.2 Task assignment

A feature of Crowd4U is that the task assignment is conducted in a declarative manner. To make that possible, Crowd4U manages three types of relationships between workers and tasks explicitly. (1) *Eligible* means that a worker is eligible for performing a task. This is computed by the CyLog processor using the project description and worker human factors. For example, in a project description a task requester may specify that only workers who log in to Crowd4U and speak English as a native language are eligible for their tasks. (2) *InterestedIn* means that a worker is interested in performing a task. This is declared by each worker when she is shown a list of eligible tasks. (3) *Undertakes* means that a worker confirms that she performs a task. A (worker,task) pair can go into this relationship status only when the worker is *Eligible* for that task.

3.2.2.1 Task assignment workflow

Task assignment for collaborative tasks is performed as follows (Figure 3.2): (1) For each submitted project description, an administration page for the project is generated. The page has a form to enter desired human factors for collaborative task assignment. (2) The entered factors are sent to the *task assignment controller*. (3) User pages show workers the list of collaborative tasks for which they are eligible, and ask them to specify their interest in tasks. The *InterestedIn* relationship is recorded in the worker human factors table maintained by the worker manager. (4) The worker manager supplies the task assignment controller with desired human factors and a worker affinity matrix. (5) The assignment controller chooses a team of workers that satisfies the desired human factors, out of the workers who are eligible and interested in the task. Then, the controller outputs the suggested team and each worker in the team is asked to join the collaborative task.

Depending on the nature of the task, the assignment algorithm combines different human factor such as workers' skills, worker-to-worker affinity and upper critical mass [61], to find the right team of workers to complete a task. Skills are used to filter out unqualified workers. Worker-to-worker affinity corresponds to the "comfort-level" of workers who are part of a team solving a collaborative task. Upper critical mass is a constraint on the size of the team beyond which the collaboration effectiveness diminishes. The worker manager supplies the CyLog processor and the task assignment manager with worker human factors (e.g., languages, countries, and application-specific human factors) and the relationship among workers encoded in the *worker affinity matrix*, which maintains the information on how a pair of workers is expected to work well. For example, in the case of surveillance tasks, if workers live in the same geographic area, their affinity value is larger. The assignment controller

waits for a sufficient number of workers to show interest in the task before building a team satisfying the requester's desired human factors. Then, the suggested team members are asked to perform the task. Unless all suggested workers start to perform the collaborative task (i.e., to go into the *undertakes* status) by the specified deadline, task assignment is re-executed to find a new team. In addition, if none of the possible teams satisfying human factors accepts the task, Crowd4U suggests to the requester to update her input. Once workers undertake a task, Crowd4U monitors their collaboration to ensure effective result coordination on behalf of requesters.

3.2.2.2 Illustration of Task assignment algorithm

Task assignment depends on the nature of the task. We adapt the task assignment algorithm that we proposed in [61] to each kind of task. In a nutshell, that algorithm is based on a 2-stage approach, `Grp&Splt`. that decomposes the task assignment into two phases. In the `Grp` phase, the algorithm forms a single team (possibly larger than upper critical mass) that satisfies the knowledge, budgetary and other constraints. In the `Splt` phase, the large group is partitioned into smaller groups (of size at most upper critical mass) such that the total pairwise distance (i.e. complement of affinity) is minimized.

In the text translation case (sequential tasks), task assignment only uses the `Grp` phase to find a single team of relevant workers, i.e., those with appropriate language skills. The task is equally divided among the team members. For citizen journalism (simultaneous tasks), task assignment relies on `Grp` and a variant of `Splt`, where the task is decomposed into a set of independent sub-tasks (such as, independent sections of a document to draft together). The sub-group members edit simultaneously on their allocated section, and at the end, collaboration across the sub-groups is needed to effectively merge the sections and prepare the overall document. The surveillance

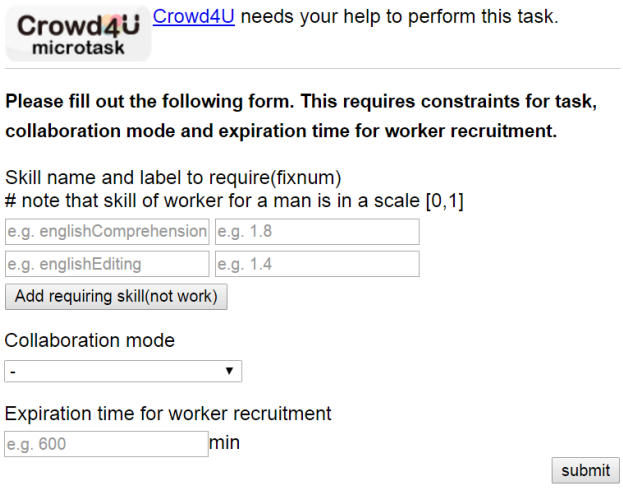


Figure 3.3: Constraint entry form in an project administration page

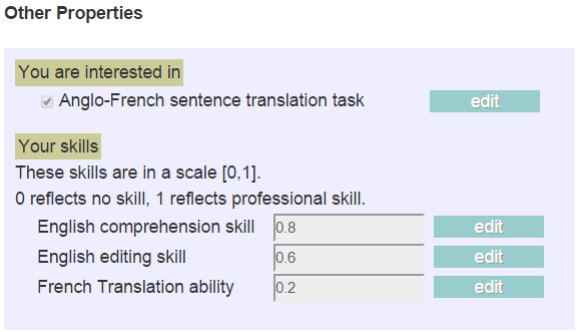


Figure 3.4: Worker human factors on a worker page

task, on the other hand, uses a hybrid collaboration pattern. The task is first decomposed into a set of geographic regions. Workers from the same region (as a substitute measure of affinity) are considered to form sub-groups and they can work both sequentially and simultaneously. The number of sub-groups is equal to the number of regions. `Splt` phase is not needed explicitly, as the sub-groups across the regions may not need to collaborate.

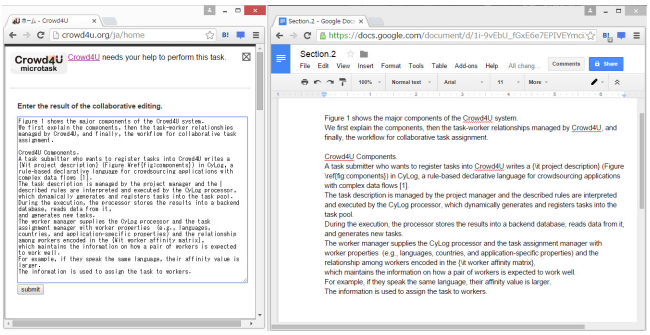


Figure 3.5: Conducting simultaneous collaboration task

3.2.3 Result coordination

Once workers have been assigned to tasks, they must be monitored to enforce appropriate collaboration. Result coordination is ensured via the following worker collaboration schemes:

Sequential Collaboration: In this mode, the team members collaborate with each other through the tasks dynamically generated based on other members' task results. For example, after a worker translates a sentence into another language, a task for checking the result is dynamically generated, and the result is sent to another team members.

Simultaneous Collaboration: In this mode, Crowd4U first assigns the task to solicit her SNS ID (e.g., Google account) to communicate with other members in the team. After all the members are in the "undertakes" status, the collaborative task is generated and assigned to all the members with the list of obtained IDs. The members work together with any collaboration tool (e.g., Google docs). The result of the collaborative task is submitted by one of the team members, but recorded as the result produced by the team. This collaboration scheme is most appropriate for citizen journalism where workers with complementary skills co-edit a report.

Hybrid Collaboration: Crowd4U allows to interleave the two result coordination schemes in a complex data flow. For example, surveillance and correction tasks are executed as a sequential collaboration while the testimonials are provided simultaneously.

3.2.4 Interface Design and Worker Interaction

Figure 3.3 is part of screen-shot of the project administration page where a requester specifies the desired human factors for task assignment. The requester also specifies an expiration time for worker recruitment.

Figure 3.4 shows the set of human factors that can be updated by each worker. Those factors are either provided by the worker when creating an Crowd4U account (e.g., native languages, location) or computed by the system based on previously performed tasks (e.g., via qualification tests, or by learning workers' profiles as in [68].

Figure 3.5 is a worker's screen in a simultaneous collaboration where she communicates with other workers using Google doc and submits the result for a Crowd4U task. Crowd4U could be combined with any collaboration tool. While delegating communication methods to other collaboration tools, Crowd4U controls task generation and assignment; it manages relationships between workers and tasks, builds teams, generates new tasks based on the intermediate task results, and assigns tasks to workers in any type of collaboration.

3.3 Conclusion

We demonstrate Crowd4U, a declarative and collaboration-aware crowdsourcing framework. We also show that Crowd4U can elegantly handle different worker collaboration schemes and thereby deploy tasks in diverse domains, such as, sentence translation, citizen journalism, and surveillance. Crowd4U's declarative and extensi-

ble architecture can easily be leveraged to incorporate additional worker collaboration schemes and other task assignment algorithms. In the next chapter, we present how to estimate skill for collaborative tasks.

CHAPTER 4

Worker Skill Estimation in Team-Based Tasks

4.1 Introduction

Automated team formation is widely studied in computer-assisted cooperative systems [69, 70, 71, 72, 57, 58]. This body of work assumes that a team of experts is to be formed to undertake a task that requires expertise in one or more domains. The formed team is assumed to have the expertise or skills required to meet the expected quality of the task (as well as other constraints such as coordination cost). Naturally, the formulation of this team formation problem assumes that the skills of individual workers are known a priori. We seek to investigate an orthogonal question: *Given a set of completed tasks undertaken by a team of workers, estimate the skills of the individual workers.* We refer to this as the skill estimation problem for team-based tasks.

A number of applications rely on team-based work. Examples are researchers co-authoring a paper, experts reviewing scientific papers, athletes playing team-based

games, as well as some emerging crowdsourcing applications, such as Galaxy Zoo[1] or Foldit[2].

Skill Estimation Problem: Estimating the skills of individual workers for team-based tasks is acknowledged to be an important open problem [73] in this space. We borrow the settings of the team formation problem [69, 70, 71, 72, 57, 58]. Inputs to our skill estimation problem are a set of teams (each team is a set of workers), that has completed one (or more) tasks. Each task requires a skill that is also known a-priori. Each completed task gets evaluated quality-wise and a numeric score is assigned to it. A worker may participate in different tasks with different teams. A worker skill is a *deterministic value*, or a *probability distribution (pdf)* that we wish to estimate as accurately as possible from the quality feedback assigned to the tasks in which her team participated. Modeling skill as a pdf can capture the fact that some workers have large variance in their skill levels when performing tasks, whereas others have smaller variances. For example, two players may have the same average points per game, but one has greater variance over the other.

Skill Aggregation Functions: To be able to effectively estimate the skills of workers involved in team-based efforts, it is critical to formulate how a team's skill is computed by aggregating the skills of individual workers in the team. Prior work [73] indicates that there exists several skill aggregation functions: (1) `Sum` where the skill of a team is the sum of skills of individual workers. As an example, the number of blocks that a basketball team makes in a game is the *sum* of the defense skills of the defenders. (2) `Max` where the skill of a team corresponds to its most skilled worker. The quality of a research paper may be dominated by the expertise of the most skilled author. (3) A `complex` function defined over workers skills, as well

[1]http://www.galaxyzoo.org/

[2]http://fold.it/portal/

as other aspects, such as collaboration effectiveness, is another alternative. Such a complex function might not assume independence between different workers' skills. We explore Sum and Max in depth and discuss extending our algorithms to handle complex aggregation functions in section 4.6.

Task Quality: In general, measuring task quality depends on the application. We assume that we are provided with a quality evaluation (as a numeric score) for each task. The applications we describe above can indeed be evaluated in many ways: for example, the number of citations of a paper reflects its (quality) impact, a team has an offense or a defense score in a particular basketball game.

Team Skills and Task Quality: Typically, workers are evaluated based on their skills (where skill is deterministic or probabilistic) while tasks are evaluated based on quality. It is apparent that the skills of the workers in a team contribute to the quality outcome of the task they undertake together. We assume that there is a known, *one-to-one correspondence* between worker skills and task quality. In the basketball example, defense skills of workers provide defense score for the game, while offense skills give rise to an offense score.

Challenges: Even when the relationship between skill and quality is injective, i.e. one-to-one, there are number of challenges in solving the skill estimation problem. The key challenge comes from the fact that the quality evaluation reflects the aggregated skill of the entire team, while we seek to estimate the skill of individual workers. Proportionally allocating the final quality of a task among its constituent workers to estimate the skill of every worker, considering different tasks that she has undertaken, is non-trivial.

Our Approach: In this chapter, we primarily focus on learning individual worker skills under `Sum` and `Max` aggregation functions. Our methods could be trivially extended to `Min` aggregation. Moreover, if dependency between the workers can be

expressed in a linear fashion, our deterministic solutions can be extended to handle those scenarios as well. We defer more detailed discussion on this to Section 4.6. At a high level, our approach is based on computing the "distance" between the estimated skills of the individuals and the known quality of the completed tasks that they have undertaken, assuming a given skill aggregation function. We refer to this distance as error and quantify it using the ℓ_2 function, a common distance measure. Thereby, we formalize skill estimation as an optimization problem with the goal of minimizing error.

We start by considering deterministic skills. The `Sum` variant, `Sum-Skill-D`, is formalized as a continuous optimization problem while the `Max` variant, `Max-Skill-D`, is posed as a discrete optimization problems. We propose quadratic programming-based solutions for `Sum` and max-algebra[74, 75] based solutions for `Max`. We employ a similar optimization framework for skills described as a probability distribution function (pdf). Both variants, `Sum-Skill-P` for `Sum` and `Max-Skill-P` for `Max`, are designed to estimate the skill pdf of each worker such that the aggregated ℓ_2 error between the joint pdf, i.e., the team's skill, and that of the individual pdfs is minimized. For `Sum-Skill-P` (resp., `Max-Skill-P`), the pdf that represents the team skill is a joint pdf computed by taking the *sum convolution* (resp., *max convolution*) [76] of individual skill pdfs of the participants. The key challenge here is to be able to *deconvolve* the joint pdfs to estimate the individual pdfs accurately.

Finally, we present a comprehensive evaluation of our solutions using two real-world datasets and demonstrate that they indeed estimate the true skills of individual workers effectively and compare with several appropriate baseline algorithms. Additionally, we demonstrate that our solutions are scalable using large-scale synthetic data. In summary, we make the following contributions:

	u_1	u_2	u_3	quality ($\overrightarrow{Q}$)
$\mathbf{t_1}$	1	1	0	15
$\mathbf{t_2}$	0	1	1	10

Table 4.1: Task Assignment Matrix and Quality Evaluation Vector

- **Formalism:** We formalize the problem of skill estimation for team-based tasks for different skill aggregations (Section 4.3).
- **Solutions:** We propose a comprehensive optimization-based formulation considering both deterministic and probabilistic interpretations of skills. We propose principled solutions and present theoretical analyses (Sections 4.4 & 4.5).
- **Discussion & Experimental results:** We conduct comprehensive experiments on multiple real-world datasets and a synthetic one that show that our algorithms are accurate and efficient (Section 6.6). We discuss the extensions of the problems in Section 4.6.

4.2 Applications of Team-Based Tasks

We now present a generic running example which will be used throughout the chapter and motivate the two skill aggregation functions studied in here.

Example 1. *Running Example: Imagine a specific instance of the skill estimation problem where the following input is provided: A Boolean matrix, that represents which worker worked on which tasks (i.e., worker to task assignment matrix) and a vector that represents the evaluated quality of the two tasks (see Table-4.1). Our objective is to learn the skills of workers* u_1, u_2, u_3.

Sum Skill Aggregation - Team-based sports: Consider a team-based activity such as Basketball where each player contributes to a game in multiple ways: attack players who together contribute to *scores* and defense players who together

contribute to *blocks*. Naturally, the *scores* and *blocks* of a team are the sum of its individual worker's scores and blocks. Given available history of past games and their respective outcomes (scores and blocks), we intend to learn the skill of individual players in *scores* and in *blocks*.

Maximum Skill Aggregation - Research paper co-authorship: For a team of researchers co-authoring a paper, the qualitative outcome of the work is often driven by the most skilled (or experienced) researcher. Similarly, the quality evaluation of a paper could be modeled simply as the number of citations it gets within a given time period. As a concrete scenario, the number of citations that a database paper gets is an indicator of its technical quality. In this example, we intend to learn each co-author's skill (expertise) in the database area.

These two aggregation functions represent a wide range of team formation application scenarios [77, 78]. Further discussions on other skill aggregation is deferred to Section 4.6.

4.3 Data Model and Formalism

4.3.1 Data Model

Workers: We have a set $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ of n available workers. For the NBA application, workers are players, whereas in the co-authorship application, workers are the authors.

Domains & Skills: We are given a set of skill domains $D = \{d_1, d_2, \ldots, d_m\}$. Skill domains are associated with both tasks and workers. We assume that each domain is independent and focus on estimating workers' skill per domain. For the NBA application, domains could be defense or attack and each player has a value for each domain (0 denotes no skill in a domain).

Our problem now simplifies to estimating a single skill per worker and invoking the estimation for each domain independently. We represent a worker skill as s^u. The skills of all n workers are presented by a vector $\overrightarrow{S}$. s^u is either a *deterministic value, or a probability distribution*. The latter scenario assumes that some workers have large variance in their skill levels when performing tasks, whereas others have smaller variances. For example, in NBA two players may have the same average points per game, but one has greater variance over the other. Thus, s^u is a random variable and is represented by a probability distribution function (pdf). To simplify exposition, we assume that the skill pdf of a worker u is discretized across w possible range of values (i.e.,buckets), where $\sum_w Pr(s^u = w) = 1$. Using Example 2, if the skill pdf of worker u_1 (in the range of $[0 - 15]$) is discretized using 3 equi-width buckets, the buckets may represent skill $[0 - 5], [5 - 10], [10 - 15]$.

Team Based Tasks: We assume a set $\mathcal{T}$ of l completed team-based tasks. Each task involves a team of workers. For the NBA application, each game is a task, whereas for co-authorship, each research paper is a task.

Teams or Groups: A team or group $\mathcal{G} \subseteq \mathcal{U}$ comprises of a set of workers from $\mathcal{U}$ who participate in a task together. A team $\mathcal{G}$ undertaking a task t is referred to as $\mathcal{G}^t$.

Task Assignment Matrix: For each completed task t, we know the workers in $\mathcal{G}^t$ who undertook t. This gives rise to the task assignment matrix $\mathcal{A}_{l \times n}$ (n workers and l tasks). Each cell $a_{ij} \in \{0, 1\}$ contains a binary value, where a 1 indicates that the i^{th} task was undertaken by the j^{th} worker, and 0 otherwise.

Task Quality Evaluation Vector: Each completed task t is assigned a continuous quality score. For all l tasks, we obtain a vector of length l, $\overrightarrow{Q}$. For example, the quality of a team $\mathcal{G}$ in a game t could be measured as the total *scores* in that

game. The number of citations could automatically reflect the quality of a published research paper.

4.3.2 Formalism

Next, we formalize different skill aggregation functions for team-based applications.

Additive (Sum) Skill Aggregation Model: In this model [73], the performance of a team for a task t is computed as the sum of skills of the workers who undertook task t together. Formally, the skill of a team $\mathcal{G}^t$ for task t, can be computed as:

$$q^t = \sum_{u \in \mathcal{G}^t} s^u \tag{4.1}$$

Team-based sports are popular examples of the additive skill aggregation model, where more workers add more value to the task. In the running example, the skill of team of workers u_1 and u_2 working on task t_1 is $(s^{u_1} + s^{u_2}) = 15$.

Maximum (Max) Skill Aggregation Model: In this model [73], the team skill is dominated by the skill of the most skilled worker in the team. Formally, we have:

$$q^t = \max_{u \in \mathcal{G}^t} s^u \tag{4.2}$$

This model fits closely with creative tasks[77]. For example, a research work may require forming a team, where the quality is primarily dominated by the highest skilled researcher. In the running example, the skill of team (u_1, u_2) working on task t_1 can be computed as $\max(s^{u_1}, s^{u_2}) = 15$.

4.3.2.1 Problem Definition

Worker Skill Estimation: For each worker u, s^u needs to be computed considering all the tasks undertaken by u. As a simple example, for the *additive aggregation model*, this gives rise to a system of linear equations, satisfying $\mathcal{A} \times \overrightarrow{S} = \overrightarrow{Q}$, where the objective is to estimate $\overrightarrow{S}$.

In many scenarios, there may not be any feasible solution to a given problem instance. Consider our running example again under `Max` skill and assume that worker u_1 also participated in task t_2. Now we can assign either skill value of 15 or 10 to her; either way, this does not produce a feasible solution (because $\mathcal{A} \times max(\overrightarrow{S})$ and $\overrightarrow{Q}$ are not same). Therefore, we must estimate skill accuracy by measuring some error.

We relax our formulation into an inequality - i.e $\mathcal{A} \times \overrightarrow{S} \preceq \overrightarrow{Q}$. Our objective is to estimate an optimal value for $\overrightarrow{S}$ that satisfies all the inequalities, and has a small *reconstruction error*. For every task t, the *reconstruction error* is the difference between the estimated skills of $\mathcal{G}^t$ and the given quality of t. The overall reconstruction error across all l tasks is denoted by $\mathcal{E}(\overrightarrow{Q}, \mathcal{A} \otimes \overrightarrow{S})$. Our optimization, therefore, is to

$$\text{Minimize } \overrightarrow{Q} - \mathcal{A} \otimes \overrightarrow{S} \tag{4.3}$$

The operator $\otimes$ is $\times$ for additive skill model and max for maximum skill aggregation model.

Reconstruction Error: Our problem is most aptly represented with *one sided error* [79] which assumes that the actual quality value of task t (i.e., q^t) is never smaller than that of the estimated skills of the team that undertook t, for a given skill aggregation model. While this conservative approach may underestimate the true skill of a worker, it in turn provides better assignment of workers to future tasks, where the assigned workers will necessarily surpass the minimum skill requirement of the tasks. On the other hand, *two sided error* may overestimate worker's skill, which

may lead to poor task assignment, because the true skill of a worker is actually smaller than what is estimated. Formally, one sided error could be specified as $\overrightarrow{Q} - \mathcal{A} \times \overrightarrow{S}$ and we require this expression to be non-negative. Considering one sided error, operator $\preceq$ only represents $\leq$ between the left and the right hand side of the above equation.

Error Functions: Recall that we compute the reconstruction error $\mathcal{E}(\overrightarrow{Q}, \mathcal{A} \otimes \overrightarrow{S})$ between two vectors, $\overrightarrow{Q}$ and $\mathcal{A} \otimes \overrightarrow{S}$ by measuring their distance or norm. The distance between two vectors $\overrightarrow{V}$ and $\overrightarrow{V'}$ could naturally be computed using several norms. We focus on ℓ_2 and note that our solution framework requires simple adaptation for ℓ_1 and L_∞.

ℓ_2 norm: $||\overrightarrow{V} - \overrightarrow{V'}||_2 = \sqrt{\Sigma_k (v_k - v'_k)^2}$. As an example, for our running example, if $\mathcal{A} \otimes \overrightarrow{S}$ is a vector $(9, 10)^T$ for two tasks and $\overrightarrow{Q}$ is $(15, 10)^T$, the reconstruction error is $\mathcal{E}(\overrightarrow{Q}, \mathcal{A} \otimes \overrightarrow{S}) = \sqrt{(9-15)^2 + (10-10)^2} = 6$

Selecting Optimal $\overrightarrow{S}$: In an over-determined system [80], where there are more tasks than workers, there may not be any feasible solution. Our objective in this case is to identify a solution that has the smallest ℓ_2 reconstruction error. For an under-determined system [80], there are more workers than tasks. In this scenario, there may be many feasible solutions and the objective is to select one of them that minimizes some prior function. The most common approach is to use `MaxEnt` or principle of Maximum Entropy [81] for such scenarios. We explore the former in depth (which is realistic for our applications) and defer the latter to future work.

Optimization Problems: Formally, given a task assignment matrix $\mathcal{A}$ and task quality estimate $\overrightarrow{Q}$, where, $\mathcal{A} \times \overrightarrow{S} \leq \overrightarrow{Q}$, estimate $\overrightarrow{S}$ (where s^u is the skill of worker u in this vector) that minimize:

Problem 2. `Sum-Skill-D` $\mathcal{E}(\overrightarrow{Q}, \mathcal{A} \times \overrightarrow{S})$, *where* s^u *is deterministic.*

Problem 3. `Sum-Skill-P` $\mathcal{E}(\overrightarrow{Q}, \mathcal{A} \times \overrightarrow{S})$, *where* s^u *is a discrete pdf.*

Problem 4. `Max-Skill-D` $\mathcal{E}(\overrightarrow{Q}, \mathcal{A} \times max(\overrightarrow{S}))$, *where* s^u *is deterministic.*

Problem 5. `Max-Skill-P` $\mathcal{E}(\overrightarrow{Q}, \mathcal{A} \times max(\overrightarrow{S}))$, *where* s^u *is a discrete pdf.*

4.4 Sum-Skill

We consider the first skill aggregation function - `Sum` - where the team skill corresponds to the *sum* of its members.

4.4.1 Sum-Skill-D

For the deterministic case, the skill of each worker u (i.e. s^u) corresponds to an unknown variable. Given a task t, the skill of a team is the sum of its individual workers who undertook it - $\sum_{u \in t} s^u$. This expression is upper-bounded by the qualitative skill assigned to the task. Formally, each task t completed by team $\mathcal{G}^t$ corresponds to an inequality

$$\sum_{u \in \mathcal{G}^t} s^u \leq q^t \tag{4.4}$$

`Sum` naturally lends itself to formulating skill estimation as a system of linear inequalities as follows:

$$\mathcal{A} \times \overrightarrow{S} \leq \overrightarrow{Q} \tag{4.5}$$

Running Example: The example from Section 4.3.1 can be formalized as a set of constraints, such as:

$$s^{u_1} + s^{u_2} \leq 15; \qquad s^{u_2} + s^{u_3} \leq 10; \qquad lb \leq s^u \leq ub$$

where lb and ub are problem specific lower and upper bounds for the skill of workers. The above inequalities are trivially satisfiable by setting all entries of $\overrightarrow{S}$ to 0. In order to obtain realistic values, we need to design an optimization formulation based on how close the current assignment is to the qualitative assignment provided by the domain expert, i.e., by minimizing the reconstruction error.

ℓ_2 **Reconstruction Error:** We use ℓ_2 measure that computes the Euclidean distance between $\overrightarrow{Q}$ and $\mathcal{A} \times \overrightarrow{S}$. If the linear inequalities are indeed equalities, this would reduce to linear least squares [82]. Due to inequalities and additional constraints this becomes a constrained least square problem. Specifically, our formulation has a quadratic objective and linear constraints - which we model as a quadratic programming problem with linear constraints. This variant is a known instance of convex optimization [83].

The corresponding optimization problem is formalized as

$$\begin{aligned} \text{minimize} \quad & \sqrt{\sum_t {e_t}^2} \\ \text{subject to} \quad & q^t - (\overrightarrow{A_t} \times s^u) \geq e_t, \forall u \in \mathcal{G}^t \\ & lb \leq s^u \leq ub \end{aligned} \tag{4.6}$$

where lb and ub are problem specific lower and upper bounds for the skill of workers. Specifically, our problem corresponds to a *box-constrained* least squares 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 methods. 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 least square optimization with equality constraints [83]. The interior-point methods encode the convex set (of solutions) as a barrier function. Quasi-Newton methods are then used to optimize this function. Using our running example, the estimated skill vector of workers are $\langle 9.0014, 5.994, 4.0031 \rangle$ with ℓ_2 error of 0.

Complexity: Both the active-set and interior-point methods are very efficient and run in polynomial time [80]. The worst case complexity for computing constrained least squares (by using generalized singular value decomposition) is $O(n^2 l + n^3)$ [80].

In practice, iterative algorithms often return the solution within a small number of iterations [80].

4.4.2 Sum-Skill-P

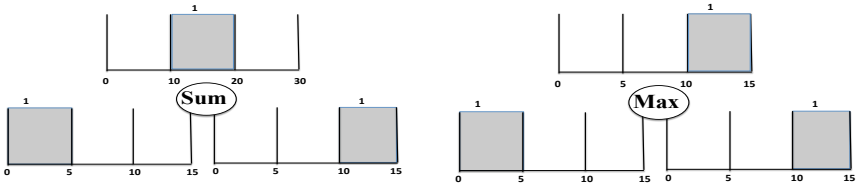
We now describe the skill estimation problem considering skill of a worker u, i.e., s^u as a probability distribution function (or simply a pdf) that is unknown to us. It can be any arbitrary distribution, which is discretized over a set of w possible range of values (each range is a bucket) that the pdf can take. While such discretization may introduce error in the overall calculation, there are no efficient alternatives that can handle any arbitrary pdf.

If there is only a single task in the task assignment matrix $\mathcal{A}$, we intend to produce the skill pdfs of the workers such that the *joint pdf of quality of the assigned team is as close as possible to the obtained quality.* However, the task assignment matrix contains many tasks with (possibly) different quality and a worker has typically undertaken many tasks. Thus, we need to estimate the skill pdfs of the workers such that the ℓ_2 error across all the tasks is minimized. The quality of each completed task (q^t) by a team is known and performed by taking the `Sum` of individual worker's skill pdfs. As we describe below, this step is akin to taking the `Sum`-*convolution of the individual skill pdfs of the workers to compute the joint skill pdf of the team.* However, we do not know these individual skill pdfs - rather, only the quality of each team as a whole (i.e., the q^t's) are available at our disposal. The challenge is to be able to estimate the individual skill pdfs from these q^t's (in other words, deconvolve the q^ts to generate the individual skill pdfs) such that the error is minimized. Moreover, the one-sided error constraints must also be respected. More specifically, we need to perform the following three necessary transformations for that.

(1) **Computing skill pdf of a team:** When each worker's skill in a team $\mathcal{G}^t$ who undertakes task t is a pdf, the quality of the team $\mathcal{G}^t$ is also a pdf. We assume the independence of workers' skill, i.e., the skill of a worker does not improve/degrade due to the presence of certain fellow workers. For `Sum` skill, the joint pdf of the team's skill (or quality) (e.g., multiparty online games) could be computed using *Sum-Convolution* of the individual skill pdfs. The definition of *Sum-Convolution* of two pdfs is adapted from prior work [76] and is given below. A simple example of the joint skill distribution using `Sum`-aggregation (i.e., Sum Convolution) is presented in Figure 4.1(a) (further described in (2)), considering two skill pdfs after appropriate discretization. In general, Sum-Convolution of an arbitrary number of M pdfs can be computed by performing a sequence of $M-1$ Sum-convolutions, first convolving the first and the second pdfs, then convolving the resultant pdf with the third pdf, and so on.

Definition 4 (Sum-Convolution of Distributions)**.** *Assume that $f(x)$, $g(x)$ are the pdfs of two independent random variables X and Y respectively. The pdf of the random variable $X+Y$ (the sum of the two random variables) is the convolution of the two pdfs: $*(\{f,g\})(x) = \int_0^x f(z)g(x-z)\,dz$.*

(2) **Representing q^t as a pdf:** The quality of task t, q^t, a deterministic value, should also be represented as a pdf. Consider Example 2 (Section 4.2) and notice that $q^{t_1} = 15$. If u_1 and u_2 have worked in task t_1, without any prior information, the skill of both workers s^{u_1} and s^{u_2} can range between $[0, 15]$. While the task quality can be between $[0-30]$, we know that it has a skill of 15. Therefore, the resultant pdf is between $[0, 30]$, yet only the skill value of 15 has a probability of 1, and all other skill values have probability of 0. We translate the deterministic quality of each completed task to a pdf and discretize involving w equi-width buckets: for t_1, these



(a) (b)

Figure 4.1: An example joint pdf after, (a) `Sum` aggregation using Sum-Convolution (b) `Maximum` aggregation using Max-Convolutions. In fact, our objective is to learn the individual skill pdfs, given q^t.

buckets are $[0-10], [10-20], [20-30]$, where only the second bucket is associated with a probability of 1 (as it contains 15).

(3) **One sided error:** Unlike the deterministic case, where one can easily specify one sided error constraints, such as, $s^{u_1} + s^{u_2} \leq 15$, there is no obvious easy way to specify such *hard constraints*, when each s^u is a pdf. Therefore, we ensure that the *probability that the joint pdf* $s^{u_1} + s^{u_2}$ *is larger than* 15 is smaller than a predefined threshold, λ. i.e., $Pr(s^{u_1} + s^{u_2} > 15) \leq \lambda$. By controlling the value of λ, we can tune these constraints in a flexible manner. Since the skill pdf of each worker is discretized over a set of w different ranges, we can still use ℓ_2 distance to compute the difference or error between the joint pdf (represented using Sum-Convolution) and the pdf that represents the obtained quality [84].The corresponding optimization problem is,

$$\text{minimize} \quad \sqrt{\sum_t (q^t - (\overrightarrow{A_t} \times s^u))^2} \tag{4.7}$$

The constraints are to be set up such that the pdfs of the workers satisfy the *probability axioms* as well as the one-sided error constraints, such as.

$$Pr(\sum_{u \in t} s^u > q^t) \leq \lambda \qquad \forall t$$

$$\sum_w Pr(s^u = w) = 1 \qquad \forall u.$$

If skill pdf a worker is represented involving w buckets, then, each skill pdf is associated with w unknown variables. Table 4.2 shows the various pdfs that are associated with Example 2, if each of the pdfs are discretized using 3 ($w = 3$) buckets. Without loss of generality, imagine variables (unknowns) p_i^u represent the probability of the i-th skill bucket of s^u (for skill of u_1, $p_1^{u1}, p_2^{u1}, p_3^{u1}$ are $[0-5], [5-10], [10-15]$, respectively). Also, let p_i^t (known) represent the probability of the i-th skill bucket of q^t (for q^{t_1}, $[0-10], [10-20], [20-30]$ are represented using $p_1^{t1}, p_2^{t1}, p_3^{t1}$ respectively, where, $p_2^{t1} = 1$).

The challenge in solving the optimization problem is estimating these variables (in other words, deconvolve the q^ts to generate the individual skill pdfs) such that they minimize the ℓ_2 error. For our running example, just considering the second bucket of q^{t_1} ($[10-20]$, where the probability mass is 1), we need to set up the variables such that the probability that the sum of $s^{u_1} + s^{u_2}$ to be in $[10-20]$ is as high as possible. This can be done by computing the probability, when $s^{u_1} = [5-10]$& $s^{u_2} = [5-10]$, or $s^{u_1} = [0-5]$& $s^{u_2} = [10-15]$, or $s^{u_1} = [10-15]$& $s^{u_2} = [0-5]$. Therefore, the corresponding formulation is to solve and minimize for

$$\left(p_2^{t1} - \{p_2^{u1} * p_2^{u2} + p_1^{u1} * p_3^{u2} + p_3^{u1} * p_1^{u2}\}\right)^2$$

It is easy to notice that even for the toy example that involves only two workers per task, such a formulation gives rise to a quartic polynomial (degree of 4). For the general case, the degree of the resultant polynomial could be of the order n. Solving such functions optimally is thus prohibitively expensive. We resort to a hill climbing based efficient heuristic solution, as a viable alternative.

Heuristic Algorithm: We design a hill climbing based heuristic algorithm that uses random restarts. We start with a pdf for each s^u (uniform in our case in lack of any prior knowledge) and the overall objective function value (i.e., ℓ_2 error) is

pdf	range	known/unknown
s^{u_1}	$[0-5], [5-10], [10-15]$	all unknown
s^{u_2}	$[0-5], [5-10], [10-15]$	all unknown
s^{u_3}	$[0-5], [5-10], [10-15]$	all unknown
q^{t_1}	$[0-10], [10-20], [20-30]$	all known
q^{t_2}	$[0-10], [10-20], [20-30]$	all known

Table 4.2: Discretized pdfs using 3-equi-width histograms for Example 2

computed. In a single iteration, this algorithm selects one of the workers u at random and updates its pdf by a small value δ. Notice that, since the pdf of each worker is discretized using w buckets, this step corresponds to randomly choosing one of the w buckets and increasing (or decreasing) the probability of it by δ, while readjusting the other buckets uniformly to keep the probability mass to 1. As an example, for u_1, if $w = 3, \delta = 0.2$, and the first skill bucket (p_1^{u1}) of the initial uniform distribution ($p_1^{u1} = 0.33, p_2^{u1} = .33, p_2^{u1} = .34$) of s^{u_1} is being increased, then the adjusted pdf of s^{u_1} will be, $p_1^{u1} = 0.53, p_2^{u1} = .23, p_2^{u1} = .24$. With the modified pdf of u, it recomputes the objective function value and takes this change, if the error is further reduced. This process continues until no change can be found to improve the error. The solution is then said to be "locally optimal". With random restart, the algorithm performs hill-climbing iteratively, each time with a random initial condition and the best solution is kept at the end as the final solution. The various restarts increase the likelihood of finding "global optima". Our algorithm discovers the global optima on Example 2 and produces the following 3 distributions, $s^{u_1} = [0, 0, 1], s^{u_2} = [1, 0, 0], s^{u_3} = [0, 1, 0]$.

Complexity: The exact asymptotic form is hard to derive, as that depends on how fast it reaches the local optima in a given iteration. Our experimental results indicate that the solution converges within a few minutes most of the time even for a large scale dataset.

4.5 Max-Skill

We now describe our solution for the maximum skill aggregation function `Maximum` (or simply `Max`). Under `Max`, the skill of a team corresponds to that of its most skilled member and the problem of estimation individual skills becomes a discrete optimization problem. As before, we first describe the deterministic solution, and then illustrate the probabilistic case.

4.5.1 Max-Skill-D

Before we describe our solution, we explain the correspondence between our problem and a mathematical algebraic theory *Max-Plus Algebra* [74, 75], which has been developed to solve a number of discrete optimization problems.

4.5.1.1 Overview of Max-Algebra

Traditionally, Max (or Max-Plus) Algebra provides techniques for solving non-linear problems that could be specified in the form of linear problems, when arithmetic *addition is replaced by a maximum operation*, and arithmetic multiplication is replaced by addition [74, 75]. Further, the inverse of a number is equivalent to its negation and ∞ is denoted by ϵ. Using mathematical notations, the key max-algebraic equations are given below:

$$a \oplus b = \max(a, b) \qquad a \otimes b = a + b \qquad a^{-1} = -a$$

$$a \oplus \epsilon = a \qquad a \otimes \epsilon = \epsilon$$

Almost all linear algebraic operations could be derived in the context of max algebra. Specifically, the matrix-vector multiplication required for solving a system of (in)equalities can be specified as:

$$A \otimes b = \sum_{k}^{\otimes} a_{i,k} \otimes b_k = \max_{k}(a_{i,k} + b_k) \tag{4.8}$$

Intuitively, the system of inequalities $A \otimes x \leq b$ can be interpreted as

$$(a_{11} \otimes x_1) \oplus (a_{12} \otimes x_2) \oplus \ldots \oplus (a_{1n} \otimes x_n) \leq b_1$$

$$\ldots$$

$$(a_{l1} \otimes x_1) \oplus (a_{l2} \otimes x_2) \oplus \ldots \oplus (a_{ln} \otimes x_n) \leq b_l$$

Using the standard linear algebraic notation, this is equivalent to solving the system of linear inequalities:

$$\max\{(a_{11} + x_1), (a_{12} + x_2), \ldots, (a_{1n} + x_n)\} \leq b_1$$

$$\ldots$$

$$\max\{(a_{l1} + x_1), (a_{l2} + x_2), \ldots, (a_{ln} + x_n)\} \leq b_l$$

If this system of inequalities has a solution, then we can see that it must satisfy the following set of inequalities:

$$x_1 \leq min\{(b_1 - a_{11}), (b_2 - a_{21}), \ldots, (b_l - a_{l1})\}$$

$$\ldots$$

$$x_n \leq min\{(b_l - a_{1n}), (b_2 - a_{2n}), \ldots, (b_l - a_{ln})\}$$

The candidate solution that we derive this way is called the *principal solution* [74, 75], using Equation 4.9.

$$\overline{x_i} = \left(\max A_{i,j} \otimes (b_i)^{-1}\right)^{-1} = \min\{b_i \otimes a_{ij}^{-1}\} \tag{4.9}$$

4.5.1.2 Proposed Solution

Considering `Max`, we however have slightly different formulations than in Max-Algebra. Given a task t which is undertaken by group $\mathcal{G}^t$, we intend to estimate the skills of the workers to minimize, $q^t - \max(\overrightarrow{A_t} \times s^u) \geq e_t, \forall u \in \mathcal{G}^t$. For our running example, this gives rise to the following set of constraints:

$$\max(s^{u_1}, s^{u_2}) \leq 15; \quad \max(s^{u_2} + s^{u_3}) \leq 10; \quad lb \leq s^u \leq ub$$

where lb and ub are problem specific lower and upper bounds for the skill of workers. Even though, the operator inside $\max(\overrightarrow{A_t} \times s^u)$ is a multiplication (instead of an addition in the traditional max-algebra), the techniques proposed in Max-Algebra leaves enough intuition behind to design a solution for our problem considering one sided error. Algorithm 4 presents the pseudo-code.

Algorithm 4 Algorithm for `Max-Skill-D`

1: **Input:** $\mathcal{A}$, $\overrightarrow{Q}$
2: Replace 1 and 0 in $\mathcal{A}$ to 0 and ϵ respectively
3: Construct system of linear inequalities $\mathcal{A} \otimes \overrightarrow{S} \leq \overrightarrow{Q}$
4: Compute principal solution vector $\overrightarrow{S}$ using Equation 4.9
5: **return** $\overrightarrow{S}$

In particular, consider our running example and notice that we can adapt the principal solution technique to form the following inequalities, based on the aforementioned constraints.

$$s^{u_1} \leq \min(15) = 15 \qquad s^{u_2} \leq \min(15, 10) = 10$$
$$s^{u_3} \leq \min(10) = 10$$

`Max` becomes computationally much harder when two sided error is considered. Exploration of two sided error for max skill aggregation is deferred to future work.

Lemma 5. *The principal solution vector is a valid solution to the system of inequalities for the max skill aggregation problem* `Max`.

Proof. (Sketch): Once we express the maximum skill aggregation using the system of inequalities, the proof directly follows from [74, 75]. □

Lemma 6. *The principal solution vector minimizes the reconstruction error for ℓ_2 for one sided error.*

Proof. (sketch): The principal solution vector outputs a feasible region for each variable (each variable corresponds to a worker's skill in the given domain). A detailed proof in [85] shows that how the proposed solution minimizes ℓ_∞ norm. Extending it to ℓ_2 is trivial. □

Time Complexity: The principal solution could be computed by a single pass over the matrix $\mathcal{A}$ and $\overrightarrow{Q}$. The running time is dominated by the dimension of matrix $\mathcal{A}$ which is $l \times n$. Therefore, the time complexity is $O(nl)$.

4.5.2 Max-Skill-P

Next, we describe the probabilistic skill estimation under `Max`. Akin to Section 4.4.2, we first translate the quality of each completed task, i.e., q^t as a pdf and

intend to estimate the skill pdf of each worker u, represented as s^u, while satisfying the one-sided error constraints using ℓ_2. However, unlike `Sum`, the skill aggregation function is now *maximum* (i.e., `Max`), reflected by taking the *maximum* skill among the participating workers.

To compute the joint probability distribution of two independent random variables under `Max`, one has to compute the *Max-Convolution* [76] of two random variables, that we formally define below. The joint distribution of M random variables under `Max` could be computed by performing a sequence of $M-1$ pairwise *Max-Convolution*. Consider Figure 4.1(b), where the joint pdf of two random variables under `Max` skill aggregation is presented.

Definition 5 (Max-Convolution of Distributions)**.** *Assume that $f(x)$, $g(x)$ are the pdfs of the two independent random variables X, Y respectively. The pdf of the random variable $Max(X,Y)$ (the maximum of the two random variables) is the max convolution of the two pdfs:* $_{max} * (\{f,g\})(x) = f(x)\int_0^x g(z)\,dz + g(x)\int_0^x f(z)\,dz$.

The main challenge is to estimate the individual skill pdfs of the workers as accurately as possible, such that, the distance between the obtained skills (based on Max-Convolution) and the assigned quality is as small as possible. The optimization problem exploits the same framework and is now restated as,

$$\text{minimize} \quad \sqrt{\sum_t (q^t - (\overrightarrow{A_t} \times max(s^u)))^2}$$

The constraints are to be set up such that the pdfs of the workers satisfy the *probability axioms*, as well as one sided constraints, akin to Section 4.4.2. Similar to the `Sum` counterpart, we discretize the skill pdf of each worker using w buckets and and assign a variable per worker per bucket that we intend to estimate.

Similar to the sum problem, the optimization problem gives rise to solving a polynomial of degree of n, if a task has n number of workers. Unfortunately, solving the problem is prohibitively expensive. Therefore, we design a hill climbing based efficient heuristic algorithm.

Heuristic Algorithm: Algorithm for `Max-Skill-P` runs is a greedy fashion and performs hill climbing with random restarts. It is very similar with the algorithm for `Sum-Skill-P` in flavor, except the fact that now it has to perform Max-Convolution to compute the joint pdf of skill of the workers. We omit the details for brevity. Our algorithm discovers the global optima on Example 2 and produces the following 3 distributions, $s^{u_1} = [0, 0, 1], s^{u_2} = [1, 0, 0], s^{u_3} = [0, 1, 0]$.

Complexity: The running time complexity is similar to that of `Sum-Skill-P`.

4.6 Discussion

Min Skill Aggregation: It is easy to see that the techniques developed in Section 4.5 can easily be extended to handle `Min` skill, should the application fit that aggregation model. A well studied body of research, `Min-Plus` (or Tropical) algebra [74] has been developed to study the algebraic operations with the `Min` operator. The fundamental operations can be specified by the identities $a \oplus b = a + b$ and $a \otimes b = min\{a, b\}$. Most results from Max-Plus algebra are directly applicable just by switching the `Max` operator with `Min`. Specifically, we can rewrite Equation 4.9 specifying the principal solution as,

$$\overline{x_i} = \left(MinA_{i,j} \otimes (b_i)^{-1}\right)^{-1} = Max\{b_i \otimes a_{ij}^{-1}\} \tag{4.10}$$

The only change required in Algorithm 4 is to replace Equation 4.9 with 4.10.

Similarly, the probabilistic skill learning algorithms could be adapted by deconvolving `Min`-functions [76].

Complex Skill Aggregations: In this chapter, we have initiated the first ever formal treatment to estimate the skills of the workers for team based tasks. Our chosen functions share some attractive properties: (a) They are representative of the skill aggregation functions commonly used in a number of real-world team based tasks [77, 78]. (b) They are supported by a well developed body of research (such as linear, max-plus algebra, or min-algebra) that allows the development of efficient polynomial time algorithms.

Our proposed optimization framework could be used to represent any arbitrary skill aggregation function that takes the skills of a set of workers and outputs a scalar score for the team. Specifically, the optimization objective for a complex skill aggregation function f can be formulated as:

$$\begin{aligned} \text{minimize} \quad & \mathcal{E}(\overrightarrow{Q}, f(\mathcal{A}, \overrightarrow{S})) \\ \text{subject to} \quad & q^t - f(\overrightarrow{A_t}, s^u) \geq 0, \forall u \in \mathcal{G}^t \\ & lb \leq s^u \leq ub \end{aligned} \tag{4.11}$$

Under certain constraints (such as convexity), such a formulation might even be solved efficiently. Investigation into their (in)tractability and design of efficient solutions remain open problems at this point.

Independence and Collaboration: The independence assumption is indeed true in several applications, such as online multi-party games and sentence translation by fans (fan-subbing). It also allows us designing efficient solutions.

Our deterministic approaches could be extended to incorporate collaborations, as long as, the aggregation functions are linearly expressible. Similarly, our probabilistic algorithms could also be adapted by representing worker-dependence with higher order histograms [86] to compute their joint distributions.

In team productivity literature [77, 78], it is known that some individuals act as "multipliers" or "enablers". Similarly, affinity between team members also plays a role. We describe two popular models next.

(1) Additive Factor Model [77, 78]: The skill of a worker is composed of two factors; a baseline skill that the worker exhibits in all tasks, and a constant factor that depends on the team and the task. Thus the same worker could have different levels of performance for each task. If the additive factor is computed through a known function, then it could be added into the model as a constant factor with a unit weight. The problem then becomes finding the baseline skills of the workers, and our algorithms are applicable. *(2) Pairwise Affinity Model* [77, 78]: Alternatively, the skill of a worker in a task could be computed as the sum of the individual worker and the pairwise affinity that she has with other members of the team. Our model could be extended to handle this scenario through "linearization", where, we add a new variable for each pairwise interaction. Under this, our proposed deterministic algorithms extend. *(3) Non-Linear Affinity Models:* Notice that both the models described above could be described by a linear function. It is possible to design non-linear affinity functions (such as based on a clique). These scenarios fall under the broader category of complex aggregation functions.

4.7 Experimental Evaluation

Our development and test environment uses Python 2.7 on a Linux Ubuntu 14.04 machine, with Intel Core i5 2.3 GHz processor and a 6-GB RAM. We use an existing convex optimization package [3] for solving `Sum-Skill`. All numbers are presented as the average of three runs.

[3] http://cvxopt.org/userguide/solvers.html

4.7.1 Dataset Descriptions

1) NBA: We collected 317, 371 tuples of NBA scores from 1991-2004 regular season. We pre-process this dataset and generate the worker to task assignment matrix $\mathcal{A}$ by matching players with games. We consider two independent skill dimensions, i) Number of points, ii) Number of Assists where the team skill is the computed by the additive skill model. Our final dataset contains 21000 matches and 1200 players.

Ground truth in NBA dataset: The ground truth consists of the number of points and the number of assists of a player played in a particular game. If a player has played several games (which is really the case always), this gives rise to a distribution, as opposed to a single skill value per worker.

2) DBLP: We use a subset of DBLP[4] considering the papers that are published from year 2000. We primarily consider authors who publish in database conferences (SIGMOD, VLDB, CIKM, ICDE, EDBT), in the area of query processing. Each publication is a completed task that is undertaken by a set of authors; we consider the number of citations as its quality. This dataset consists of 20123 publications and 22700 unique authors.

Ground truth in DBLP dataset: Unlike the NBA dataset, there is no ground truth available per worker (i.e., no truth is known about an author's expertise). Therefore, we neither have a skill pdf nor a single skill value per worker. As we describe in Section 4.7.3, we take up a cross-validation type of approach for evaluation here.

3) Synthetic Dataset: We generate a synthetic dataset for evaluating the scalability of our proposed solutions for the deterministic and probabilistic algorithms. The total number of workers varies between 5000 and 20000 and the total number of

[4]http://dblp.uni-trier.de/xml/

tasks varies between 5000 and 250000. The quality vector is an uniform distribution $[0-1]$.

4.7.2 Implemented Algorithms

Since no prior work has studied the skill estimation problem for team based tasks, we ourselves design multiple baseline solutions for comparative evaluation.

4.7.2.1 Sum Skill

Deterministic Algorithms:

(1) `BL-Sum-Regression-D`: We treat the problem as a multivariate regression problem, with the presence or absence of each individual in a team as the independent variables. The quality of a completed task is the dependent variable. The objective is to learn the co-efficients (skills) of the workers, such that the ℓ_2 error between the estimated quality and actual quality is minimized. This is equivalent to solving a least squares regression problem that minimizes $\|\mathcal{A} \times \overrightarrow{S} - \overrightarrow{Q}\|^2$. Notice that this baseline does not necessarily satisfy the one sided error constraints.

(2) `BL-Sum-Uniform-Avg-D`: For each task with quality value q^t, we uniformly distribute the skill value among its constituent workers. Thus, any worker u, who undertake t receives a skill score $s^u = \frac{q^t}{|\mathcal{U}'|}$, where $|\mathcal{U}'|$ is the set of workers who undertake t. The final skill of u is calculated by taking the average of her received scores across all the tasks. This baseline does not optimize the error value or satisfy the one sided error constraints.

(3) `BL-Sum-Uniform-Min-D`: This baseline is similar to the the previous one, except that we choose more conservative assignment of skill for each worker such that the one sided error constraints are fully satisfied. First, we uniformly distribute the obtained quality of each task among its constituent workers. The final skill of u is calculated

by taking the *minimum* of her received scores across all the tasks she has undertaken.
(4) `Sum-Skill-D` : Our proposed solution in Section 4.4.1 is compared with the baseline solutions, whenever appropriate.

Probabilistic Algorithms: To the best of our knowledge, there does not exist a regression based model which treats the independent variables in a probabilistic manner.

(1) `BL-Sum-Uniform-P`: This algorithm is similar to
`BL-Sum-Uniform-Avg-D`, but produces a pdf per worker at the end. For a task with quality q^t, we uniformly distribute q^t among its constituent workers and generate a discrete pdf per worker, after considering all the tasks she participated.
(2) `Sum-Skill-P` : Our proposed solution in Section 4.4.2 is compared with the baseline solution, whenever appropriate.

4.7.2.2 Max-Skill

Deterministic Algorithms:

(1) `BL-Max-D`: In this baseline solution, we assume that the quality of a task reflects the skill of only one of the constituent workers. For each task, in step 1, we first choose a worker uniformly at random and assign her skill s^u as the quality of the task q^t. Each of the remaining workers u' who undertook the same task receives a $s^{u'}$ smaller than s^u, using a uniform random distribution. In step 2, finally these obtained skills are averaged per worker to compute the final skill value.
(2) `Max-Skill-D` : Our proposed solution in Section 4.5.1 is compared with the baseline solution, whenever appropriate.

Probabilistic Algorithms:

(1) `BL-MAX-P`: This algorithm is designed in the same spirit as that of `BL-MAX-D`. The step-1 of this algorithm is similar to its deterministic counterpart; In step 2, instead

of averaging the skill of each worker, we generate a pdf.

(2) `Max-Skill-P` : Our proposed solution in Section 4.5.2 is compared with the baseline solution, whenever appropriate.

4.7.3 Experimental Analyses Setup

4.7.3.1 Measures & Parameter Setup: Quality

Deterministic Scenario: We adopt a classical cross-validation based set up [87] to evaluate our deterministic algorithms. We divide the dataset in train and test and perform 3-fold cross validation. In particular, each record in a test set is a task that is undertaken by a set of workers. For each such worker, we estimate their respective skill considering only the train dataset (those who do not appear in train get a skill of 0). For each task in test set, the ground truth (i.e., the true quality) is the associated quality value. We compare the estimated quality of a task with that of the ground truth for that task.

We compare and contrast different algorithms by presenting *average absolute error* and *normalized relative error*. Relative error is computed as $\frac{\sqrt{\sum_{\forall t} e_t \times e_t}}{\sqrt{\sum_{\forall t} q^t \times q^t}}$. We present the percentage of tasks in our test set which overestimates the task quality (compared to the ground truth), thus violating the one-sided error constraints (Section 4.3.2.1).

Probabilistic Scenario: For the probabilistic variant of the NBA dataset, we compute the ℓ_2 error between the *estimated skill pdfs and that of the ground truth distributions*. We measure *relative error* over the test dataset (assuming the evaluated quality of each task in the test set as ground truth). We need to transform the estimated pdf of each worker to a single value by computing *expected skill*, i.e.,

$ExpectedSkill(u) = \sum_w (Pr(s^u = w) \times w)$ and measuring the ℓ_2 error between the actual quality and the expected quality of the tasks.

We discretize the pdfs using w equi-width histograms (buckets), where each bucket is a skill range. To compute the expected skill of a worker, we consider the upper limit of the skill range per bucket: for example, if the pdf of a worker is $Pr([0-5]) = .7, Pr([5-10]) = .3$, then the expected skill is $.7 \times 5 + .3 \times 10 = .65$.

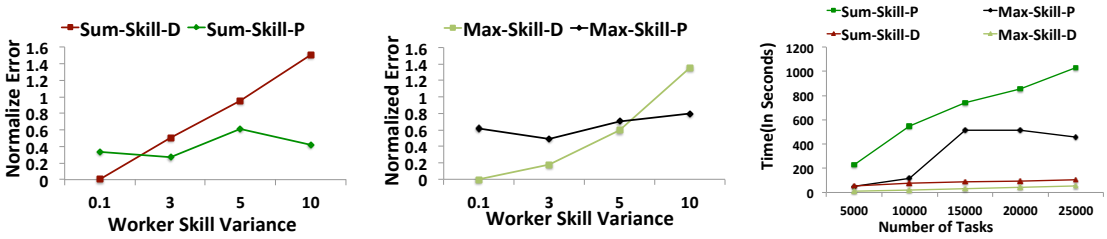
Hill climbing algorithms have many parameters, w - # buckets to approximate the equi-width pdfs, δ - the amount by which we modify the pdfs in each step, and α - # failed iteration for the convergence of the hill climbing algorithm. Our default set up is $w = 10$, $\delta = 0.05$, $\lambda = 0.01$ (one sided error threshold), $\alpha = 1000$, # random restarts=5.

4.7.4 Summary of Results

Quality Experiments: Our first set of results (considering relative error) strongly corroborate our hypothesis on the underlying skill aggregation model for a given application - i.e., NBA dataset follows `Sum-Skill`, whereas, DBLP dataset follows `Max-Skill`. After this result, we present the rest of the experiments by considering the most appropriate dataset for it (i.e.,NBA for `Sum`, DBLP for `Max`). Our deterministic algorithms demonstrate that we consistently outperform all the baseline algorithms (including the regression based one) in *minimizing the error value*, as well as *consistently obeying the one sided error constraints*. Same observation holds for the probabilistic algorithms, where `Sum-Skill-P` and `Max-Skill-P` significantly outperform their respective baseline counterparts.

Scalability Experiments: For the deterministic scenario, our results indicate that our proposed solutions are scalable. Even when the task assignment matrix is very large, they only take a few minutes to complete. `Max-Skill-D` is more scalable

than `Sum-Skill-D`. This is consistent with our theoretical analyses in Section 4.5.1. For the probabilistic scenario, the heuristic algorithms are more efficient than the optimal variants and often converge within few minutes and scale well. For lack of space, we only present a subset of results. The presented results are representative.



(a) `Sum-Skill Quality` (b) `Sum-Skill Quality` (c) `Scalability`

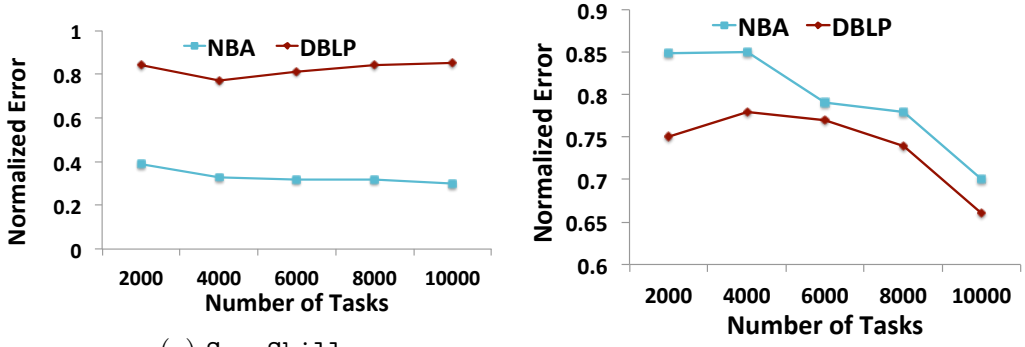
Figure 4.2: **Quality and Scalability trade-off: Deterministic/Probabilistic Models**

Deterministic/Probabilistic Skill Models: To highlight the tradeoffs between deterministic and probabilistic models, we conduct three experiments. We generate a synthetic dataset where each worker is associated with a skill pdf where we vary the variance for a fixed task assignment matrix. When the variance is low, then the workers are more consistent in their performance across different tasks. When the worker has high variance, her performance might have high deviation from the expected value. The results in Figure 4.2a and Figure 4.2b show that the deterministic variants are preferable for lower variance of worker's skill, while the probabilistic variants are preferable for higher variance. In our third experiment, we calculate the runtime of our deterministic and probabilistic algorithms with varying task size. Unsurprisingly, Figure 4.2c shows that deterministic algorithms are more scalable than probabilistic algorithms. This is quite expected as probabilistic algorithm takes longer time to converge due to greater number of unknowns than its deterministic counterparts.

4.7.5 Qualitative Experiments

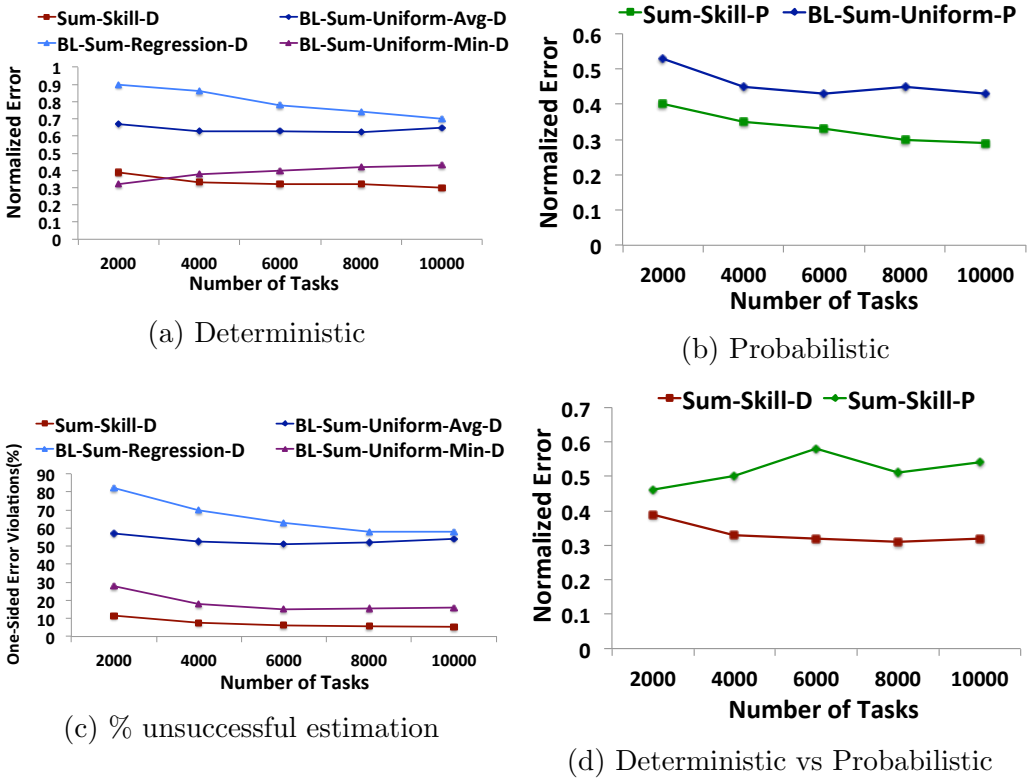
4.7.5.1 Hypothesis Validation

In this set of experiments, we vary the number of tasks and measure the average relative ℓ_2 error of both `Sum-Skill` and `Max-Skill` based skill estimation algorithms using both DBLP and NBA dataset. We present these results using `Sum-Skill-D` and `Max-Skill-D` algorithms. Figures 4.3a and 4.3b present the results which strongly corroborate our hypothesis: i.e., NBA dataset follows `Sum-Skill`, ensuring lower error compared to DBLP dataset. On the contrary, `Max-Skill` is better estimated using DBLP dataset (with smaller error compared to that of NBA). Clearly, in case of `Sum-Skill` error remains low in NBA dataset, as the training size gets higher where as DBLP dataset did not show any of this pattern. These results show that the formalized skill aggregation functions are indeed appropriate to capture real world applications.



(a) `Sum-Skill` (b) `Sum-Skill`

Figure 4.3: **Experiments for hypothesis validation:** Average relative ℓ_2 error for `Sum-Skill-D` and `Max-Skill-D` considering NBA and DBLP datasets, with varying # tasks. For `Sum-Skill-D` , NBA has significantly lower error, and for `Max-Skill-D` DBLP dataset outputs smaller relative error.



(a) Deterministic

(b) Probabilistic

(c) % unsuccessful estimation

(d) Deterministic vs Probabilistic

Figure 4.4: **Experiments to validate `Sum-Skill` aggregation**: These results clearly demonstrate the our solutions consistently outperform the baseline for both the measures we present in the Y-axis. The underlying dataset that is used is NBA.

4.7.5.2 Sum-Skill

Here we vary the training dataset size to evaluate our proposed algorithms `Sum-Skill-D` and `Sum-Skill-P`.

Normalized Error - `Sum-Skill-D` : Figure 4.4a presents the results and clearly demonstrates that our proposed algorithm consistently outperforms all the baseline solutions, including the regression based baseline. The error decreases with increasing number of tasks, which corroborates that with increasing training set size, the skill estimation becomes more accurate.

One-sided Error Constraints - `Sum-Skill-D` : Figure 4.4c clearly demonstrates that our proposed algorithm consistently outperforms the three other baselines

in one sided error constraints. BL-Sum-Uniform-Avg-D heavily overestimates workers' skill. BL-Sum-Uniform-Min-D is the best baseline, as it is designed to obey such constraints. Same observation holds for Sum-Skill-P and BL-Sum-Uniform-P.

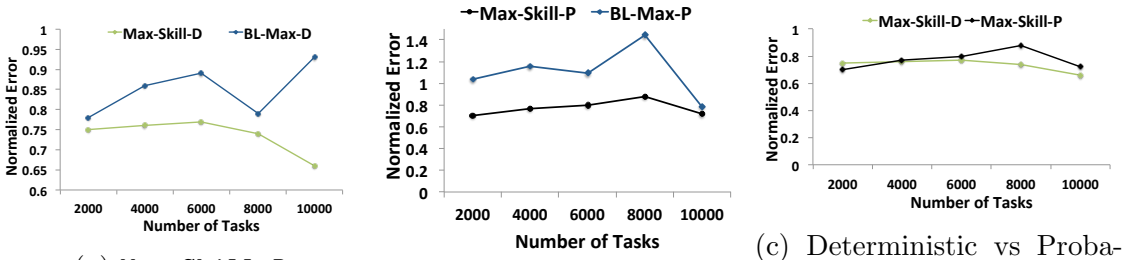
Normalized Error - Sum-Skill-P **:** We present the normalized error of our proposed solution with that of the baseline algorithm, BL-Sum-Uniform-P. Figure **??** corroborates that Sum-Skill-P is significantly more accurate.

Normalized Error -Sum-Skill-D **vs** Sum-Skill-P **:** Figure 4.4d presents the comparison between them. Although Sum-Skill-D performs better, Sum-Skill-P provides more granular information on worker's skill.

4.7.5.3 Max-Skill

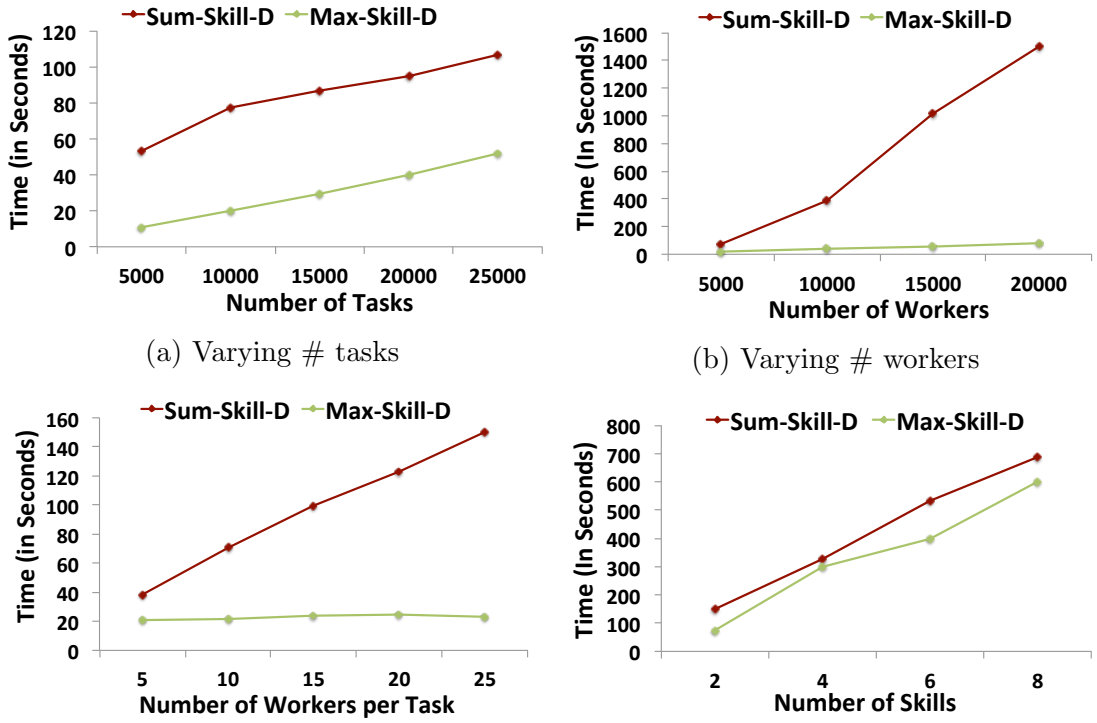
We use the DLBP dataset to perform further experiments on Max-Skill. We vary the number of tasks and measure the average absolute ℓ_2 error.

Figure 4.5a shows that Max-Skill-D outperforms BL-Max-D algorithm. With increasing training set, the algorithm learns better, hence the error of Max-Skill-D decreases. Figure 4.5b shows that Max-Skill-P outperforms the probabilistic baseline BL-Max-P consistently. However, error varies with the task size using Max-Skill-P , because the algorithm cannot always find the global optima. Figure 4.5c shows the comparison between Max-Skill-D and Max-Skill-P. We observe that Max-Skill-D performs better for larger training data. However, Max-Skill-P provides the distribution of worker skill which can be of importance for a particular application.



(a) `Max-Skill-D` (b) `Max-Skill-P` (c) Deterministic vs Probabilistic

Figure 4.5: **Experiments to validate `Max-Skill` aggregation** : In these experiments, we compute the average error by varying the number of tasks. Clearly, our proposed solutions `Max-Skill-D` outperforms the baseline algorithm `BL-Max-D` and `Max-Skill-P` outperforms `BL-Max-P`. The underlying dataset is DBLP.



(a) Varying # tasks (b) Varying # workers

(c) Varying workers-to-tasks ratio (d) Varying # skill domains

Figure 4.6: **Experiments to validate the scalability of the deterministic skill estimation algorithms:** The following default settings is considered: # workers=5000, # tasks=10000, # workers/task=10, # domains= 1. These results clearly demonstrate that our proposed solutions are scalable.

4.7.6 Scalability Experiments

4.7.6.1 Deterministic Estimation: Sum and Max Skill

For the deterministic scenario, we vary number of workers, tasks, workers/task, and domains. Our run-time experiments have the following default settings: #workers=5000, #tasks=10000, #workers/task=10, #domains= 1.

Varying Number of Tasks: We vary the number of tasks in this experiment. Figure 4.6a shows that both `Sum-Skill-D` and `Max-Skill-D` scale well with increasing number of tasks. Quite unsurprisingly, the latter outperforms the former algorithm scalability-wise. This result is expected and is consistent with our theoretical analyses of the algorithms.

Varying Number of Workers: Our observation here is akin to the previous experiment. Both algorithms scale well. Figure 4.6b shows the result.

Varying Number of (Workers/Task): The objective of this experiment is to observe the influence of the number of workers per task in the running time analyses. From figure 4.6c, it is apparent that while `Max-Skill-D` scales very well due to its linear time complexity. However, `Sum-Skill-D` also performs reasonably with larger data size.

Varying Number of Domains: In this experiment, we vary number of domains and measure the scalability of the proposed algorithms. Figure 4.6d presents the results and demonstrates that our proposed solutions are scalable.

4.7.6.2 Probabilistic Estimation: Sum and Max Skill

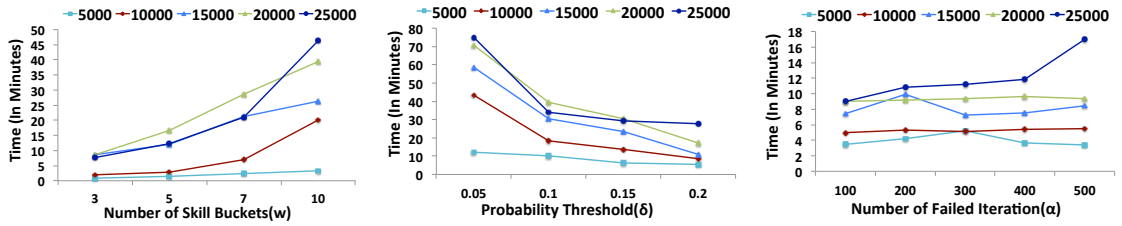
For the probabilistic skill, for brevity, we only present a subset of results. As stated before, we vary the parameters that influence the efficiency of the hill climbing algorithms.

Varying w - `Max`: As we increase the number of skill buckets, the number of unknown to solve increases, hence it takes longer to converge. Figure 4.7a demonstrates that `Max-Skill-P` scales well with varying w for five different size datasets. `Sum-Skill-P` takes about 60% more time than `Max-Skill-P` as the joint pdf has larger ranges (e.g., two pdfs in the range of $[0-15]$ gives a joint pdf between $[0-30]$ for sum, whereas, it is still $[0-15]$ for max) for sum, thereby requiring more computations. Although efficiency decreases with increasing w, but we get the distribution of worker skill with more granularity.

Varying δ - `Sum`: With higher step size (δ) in the hill climbing, the algorithm converges faster but with lower accuracy. `Sum-Skill-P` results are presented in Figure 4.7b. `Max-Skill-P` has similar trend, but takes about 60% less time. We omit the chart for brevity.

Varying # failed iterations α - `Sum`: This parameter α dictates after how many failed iterations a random restart takes place inside the hill climbing algorithms. Figure 4.7c shows that our solution scales well with increasing α. As usual, `Max-Skill-P` takes about 60% less time always.

Parameter Tuning: This is evident from Figure 4.7a, 4.7b and 4.7c that with the increase of w and α and decrease of δ, latency will be higher. However, high value of w and α and low value of δ ensure better results in terms of quality. Additionally, with higher # random restart, the solutions likely get better qualitatively, although the running time also increases (1.1 minute per restart on an average). With smaller



(a) `Max-Skill-P` : Varying w (b) `Sum-Skill-P` : Varying δ (c) `Sum-Skill-P` : Varying α

Figure 4.7: **Experiments to validate the scalability of the probabilistic skill estimation algorithms**: default settings: # domains=1, $n = 1000, w = 3, \delta = 0.2$, # failed iterations=200, # random restarts=5; despite having to solve a polynomial of degree 20, our solutions scale well and terminate within a few minutes. We only present a subset of results for brevity.

λ (one sided error threshold), we get solutions that better satisfy the one-sided error constraint, but that improvement comes with an increasing computation time. Clearly, we need to consider trade-offs while choosing these parameter values. Empirically, with $w = 10$, $\delta = 0.05$, $\lambda = 0.01$, $\alpha = 1000$, # random restarts=5, we get the best trade-off between quality and scalability.

4.8 Related Work

While no prior work has solved skill estimation problem for team based tasks, we present existing work that are tangentially related.

Team Formation: A tangential problem is the team formation problem [69, 88]. Formation of team considering a social network is first studied in [69, 70, 71]. The objective of these body of work is to form a team of experts to solve a particular task, which assumes that the skill of the experts are known and given as inputs. On the contrary, we intend to estimate the skill of the workers by investigating the quality of the completed tasks they have undertaken. No prior work, however, studies any formalism or solution to estimate worker's skill for team based tasks.

Skill Estimation in Micro-task based Crowdsourcing: Crowdsourcing has gained significant traction in the research community for solving problems, such as, image tagging, annotating labels, or looking up addresses of people [89]. These applications are primarily designed on micro-tasks, where the task is completed by an individual at its entirety (e.g., a worker tags an image all by herself). While skill estimation or evaluating the quality of the workers in crowdsourcing has gained recent research attention [1, 2, 3], the focus is entirely on micro-task based applications. We however consider team based tasks [90, 91].

Disaggregation Methods: Disaggregation methods are studied to disaggregate weather data to find hourly rainfall, temperature, or wind speed from daily maxima or minima [92, 93]. These methods do not lend any extension to solve our problem either, as their dis-aggregation happens "locally", i.e., per day, as opposed to our problem, where a worker can undertake many tasks and the skill estimation must consider all of them to minimize the error.

Regression Based Models: The `Sum`-skill problem bears resemblance with the *least square regression [87]* that we consider as a baseline, without having to satisfy the one-sided error constraints. Quantile regression [87] models the relationship between *the independent variables and the conditional quantile of the dependent variable*. Maximum quantile regression will learn the relationship between the maximum value of the dependent variable given the independent variables. Unlike that, our `Max` aggregation problem intends to learn the dependent variable which is the maximum of the independent variables. These are fundamentally different.

4.9 Conclusion

We initiate the study of estimating skill of the individual workers in team based tasks for various applications. We formalize it as an *optimization problem considering*

multiple skill aggregation functions, where skill of a worker is either deterministic or a probability distribution. We propose principled solutions to all the studied variants of the problem and provide in-depth analyses. We run a comprehensive set of experiments considering two real world datasets that demonstrate the effectiveness of our proposed skill estimation algorithms. We also conduct large scale synthetic experiments to validate the scalability of our proposed solutions. In the next chapter, we present two task recommendation algorithm using implicit observations.

CHAPTER 5

Feature Based Task Recommendation in Crowdsourcing with Implicit Observations

Existing research in crowdsourcing has investigated how to recommend tasks to workers based on which task the workers have already completed, referred to as *implicit feedback*. We, on the other hand, investigate the task recommendation problem, where we leverage both implicit feedback and explicit features of the task. We assume that we are given a set of workers, a set of tasks, interactions (such as the number of times a worker has completed a particular task), and the presence of explicit features of each task (such as task location). We intend to recommend tasks to the workers by exploiting the implicit interactions, and the presence or absence of explicit features in the tasks. We formalize the problem as an optimization problem, propose two alternative problem formulations and respective solutions that exploit implicit feedback, explicit features, as well as the similarity between the tasks. We compare the efficacy of our proposed solutions against multiple state-of-the-art techniques using two large scale real world datasets.

5.1 Introduction

Crowdsourcing platforms, such as Amazon's Mechanical Turk or Crowdflower, have recently gained immense popularity due to their elegant framework, where a task requester can get work done by numerous virtual workers for very low compensation. One common problem in these platforms is that the workers have to suffer a huge latency to find suitable tasks. This creates unhappiness and eventually leads to the abandonment of the platform. To that end, task recommendation problems are proposed in the crowdsourcing context, where the objective is to recommend a set of tasks to each worker such that these tasks are best suited for the workers [94, 95]. However, to the best of our knowledge, there does not exist any related work that focuses on the task recommendation problem by considering *the explicit characteristics of the tasks themselves*, we refer to this as *explicit task features*. For example, if the crowdsourcing task is from a citizen science application for identifying species, then, the explicit features of the tasks may come from those respective locations of observation, or a given species taxonomy to describe the relationship between the species. The novelty of our work is to leverage historical task completion of the workers (that is referred to as *implicit feedback*) augmented with explicit task characteristics or features to recommend tasks to the workers.

Our focus of the investigation is limited to citizen science applications that are designed towards volunteer based crowdsourcing, where the importance of effective task recommendation is pivotal. Currently, a volunteer, upon identifying a species, uploads the information to the server specifying the details of the identification. A common problem which frequently occurs in this scenario is *incorrect identification*. A reliable task recommender system can alleviate the problem. If we have historical data on how many tasks a volunteer has successfully performed and those observa-

tions are on what species and from which locations, then the probability of incorrect identification is much lower.

Concretely, we focus on solving the problem of task recommendation for crowdsourcing platforms, considering the implicit feedback of the workers (primarily considering what tasks workers have undertaken in the past) and exploiting the presence of explicit task feature information (such as task locations). Notice that the *implicit feedback* is only reasonable as opposed to exploring other alternative formulations, such as, *explicit feedback*, because, the latter assumes what the workers rate a task after completing that task. For a given set of tasks and a pool of workers, this information could be presented in the form of a matrix, where the rows are the workers, and the columns are the tasks. A particular row-column value may describe how many times a specific worker has undertaken a given task in the past. We refer to this as the **Worker Task Completion matrix**. Additionally, explicit features of the tasks are also available. We define the task features as the presence of a set of attributes which describe a task. For instance, we can think of a task as identifying a particular species. Then, we can use structural information such as the taxonomy of the Species nomenclature as task feature or the previous locations where the task is identified as task features. We show that incorporating explicit feature improves task recommendation.

Based on the description above, we propose two optimization models based on explicit task features. The high-level idea is to exploit the *worker task completion matrix* and task feature information. In particular, we propose two alternative problem formulations -**(1) Feature Preference Model:** We exploit the explicit task feature preference information and formalize the problem as solving linear equations. We use constrained least square to solve the problem and obtain user's feature vector which is used for recommending tasks to the worker. **(2) Latent Factor Model:**

Since, the explicit task features are known to the recommender system, this additional information is fed to the matrix factorization model. We obtain two latent factor matrices which are exploited for further recommendation.

Finally, we present a set of experimental results using a real world dataset. Our dataset, **Ebird**, extracted from the eBird project website[1] contains bird observation data of 1767 observations and 5000 workers. We evaluate our proposed task recommendation models using two different evaluation metrics, *Mean Percentile Ranking and Precision Recall Curve*. In addition, we implement a state-of-the-art solution as baseline solutions [97].

In summary, we make the following contributions-

(1) We initiate the study of task recommendation in crowdsourcing platform exploiting both implicit feedback and the presence of explicit task features.

(2)We propose two optimization models and elegant solutions for task recommendation that rely on both least squares and matrix factorization based techniques. First, we present a feature preference model to acknowledge explicit task features. Additionally, we exploit the explicit features of the task and propose an optimization model that constraints the latent factors based on the task similarity.

(3) We empirically validate our proposed methods with a large scale real world dataset and compare them against the state-of-the-art solution for task recommendations.

In the next section, we describe the data model and preliminaries. We then present our proposed problem formulations and algorithms, which is followed by experimental evaluations and related work.

[1]ebird.org

5.2 Data Model and Preliminaries

We first describe our data models and preliminaries.

Set of workers, W: We consider a set of workers/users as, $\mathcal{W} = \langle w_1, w_2, w_3 \dots w_{n_w} \rangle$ for which we want to provide recommendation based on their task preference.

Set of Tasks, T: We consider a set of tasks $T = \langle t_1, t_2, t_3, \dots t_{n_t} \rangle$.

Features, F: We consider a set of task features as -

$F = \langle f_1, f_2, f_3, \dots f_{n_l} \rangle$

Worker Task Completion Matrix, $C_{n_w \times n_t}$: $c_{wi} \in \mathbb{N}$ represents the number of times worker w has successfully completed a task i. This matrix is very sparse, which is very common in recommender systems. However, this represents the implicit signal as opposed to the explicit signal. In case of explicit feedback, each entry represents preference which is normally in the scale of 1(bad) to 5(excellent). However, this value here represents the frequency of action, or how many times worker has done the task, which also represents the confidence worker w have on selecting task i.

Task Feature Matrix, $Y_{n_t \times n_l}$: $y_{il} \in \{0, 1\}$, represents the presence of feature l on task i. The value of y_{il} depends on the application, it can be 0 or 1 indicating the presence or absence of the features. Or, it can be a real value which indicates the weight of the feature for that task.

Worker Feature Matrix, $X_{n_w \times n_l}$: $x_{wl} \in \mathbb{R}$: This is similar to the task feature matrix except that it represents the relation of each feature to user. Each row in this matrix indicates the user's preference to the features. We estimate this matrix in order to calculate the user to task recommendation.

Notation	Description
W,T,F	Set of workers, tasks, and features
C	Worker-Task Completion Matrix
X	Worker-Feature Matrix
Y	Task-Feature Matrix
P	Boolean Preference Matrix
U	Worker-Latent Factor Matrix
V	Task-Latent Factor Matrix
Sim	Task Similarity Matrix

Table 5.1: Notations & Interpretations

Preference matrix, $P_{n_w \times n_t}$ **:** $p_{wi} \in \{0, 1\}$ This is a boolean matrix indicates the preference of user to task, where "0" indicates no preference and "1" indicates the preference of user to task. This matrix is derived from the confidence matrix C -

$$p(x) = \begin{cases} 1, & \text{if } c(x) \geq 1 \\ 0, & \text{otherwise} \end{cases}$$

Worker to Latent Factor Matrix, $U_{n_w \times n_f}$**:** This represents the worker to latent factor matrix.

Task to Latent Factor Matrix, $V_{n_t \times n_f}$**:** This represents the task to latent factor matrix.

Task Similarity Matrix, Sim Task similarity is calculated based on the number of attributes shared by two tasks. Similarity between two tasks t_i and t_j is calculated as -$sim(t_i, t_j) = \frac{1}{1+e^{-Y_i^t Y_j}}$. For a task we normalize the similarity values such that it sums up to 1.

Table 5.1 summarizes the above mentioned notations-

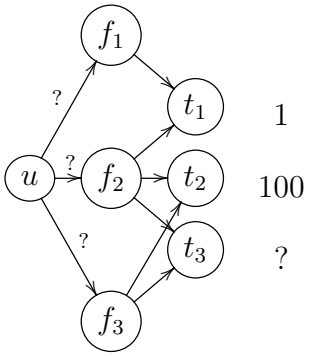


Figure 5.1: An example with 3 features and 2 tasks

5.3 Problem Formulation

We present two alternative formulations of our proposed task recommendation model. In our first formulation, we directly make use of the task feature matrix to estimate worker's feature preference. The second formulation is traditional latent factorization based model with an added factor based on task similarity calculated from using task feature matrix. The difference between the two formulation is that the first formulation explicitly uses task feature matrix whereas the second formulation implicitly uses the task feature matrix to put constraints on the latent factors.

5.3.1 Feature Preference Model

We assume that the reason that a particular worker has completed a particular task is that the worker has a hidden preference over the task features which we want to uncover. In the traditional low rank matrix factorization model, the task is to find U and V in the latent space. However, in our case we have the explicit knowledge of the feature space Y, hence by learning the preference of each worker in the feature space, X we can recommend new task.

Illustrative Example: Consider the scenario illustrated in Figure 6.1a. Here We consider 3 tasks and 3 features, where task t_1 consists of feature f_1 and f_2. Both

task t_2 and t_3 consists of f_2 and f_3. We also know that worker u has completed the task t_2 100 times, task t_1 only once and hasn't completed task $t3$. It is clear from the facts that worker prefers f_3 more than f_2, otherwise task t_1 would have been completed more times. Hence, our method will find out the feature preference for f_1, f_2 and f_3 and recommend task t_3 to the worker u which hasn't been completed by the worker.

Problem Definition: We are given a set of workers W, and a set of Tasks T and the implicit interaction information between them are given in C. Additionally, we assume the knowledge over the task feature matrix Y. Our aim is to compute the matrix X as accurately as possible such that the dot product $D = XY$ produces meaningful recommendation.

Formally, we want to minimize the following objective function M -

$$M = \sum_{w,i} q_{wi}(p_{wi} - x_w y_i)^2 + \lambda(\|X\|^2) \tag{5.1}$$

where the non-negativity constraint must be satisfied. Such that,

$$X \geq 0$$

and,

$$q_{wi} = 1 + \alpha \times c_{wi}$$

Here, q_{wi} is designed such that, the weight of positive signals is amplified. In other words, if a particular observation has high confidence the system will choose x_w such that $x_w y_i$ becomes close to 1. α is set to a positive value and indicates the confidence for the positive signals over negative signals.

5.3.2 Latent Factor Model

First,we describe the intuition behind the latent factor models for implicit recommendation.

Let's consider a simple recommender system where $r_{ij} \in R$ represents the explicit ranking for workers to item. A typical model to find U and V is to minimize the following loss function -

$$H = \sum_{i,j}(r_{ij} - U_i V_j)^2 + \lambda(\|U\|^2 + \|V\|^2)$$

Here, λ is the regularization parameter. The goal is to find U and V such that it minimizes the error. For any new item and new task the predicted recommendation score is calculated by multiplying U_i with V_j. Here, the important thing to notice is that the optimization procedure only minimizes the error for which ratings are present.

Now, in the context of task recommendation we are dealing with implicit feedback, where the worker does not provide any explicit preference over tasks. Hence, the number of times a worker has completed any particular task or the frequency of interaction with the worker to task is deemed as a confidence value for that observation [96, 97]. The cost function is to minimize -

$$M = \sum_{i,j} q_{wi}(p_{wi} - u_w v_i)^2 + \lambda(\|U\|^2 + \|V\|^2) \tag{5.2}$$

q_{wi} is the confidence value associated with each entry or how much confidence we have on that entry, p_{ij} is the preference as described in section 5.2. In other words, q_{wi} reflects how much confidence we have on p_{wi}. In [96], q_{wi} is defined as -

$$q_{wi} = 1 + \alpha \times c_{wi} \tag{5.3}$$

Here, α is a positive value indicates the confidence boost for the positive signals over the negative signals.

To incorporate the task similarity into the latent factor based formulation, we add a penalty term in the equation. Our intuition is that if the similarity between any two tasks is high, then they should also be similar in the latent factor space. Our notion of similarity is defined in the previous section. The problem formulation is given below.

Problem Definition: Given the worker Task Completion matrix C, we seek to find latent factor matrix U and V by minimizing the following objective function.

$$M = \sum_{i,j} q_{wi}(p_{wi} - u_w v_i)^2 + \lambda(\|U\|^2 + \|V\|^2 - \sum_{i,i'} v_i^t v_i' Sim(i, i')) \tag{5.4}$$

The last term in the objective function implies that if the similarity between two items are high, two item latent vectors should also be close to each other. q_{wi} is defined exactly as Equation 5.3

5.4 Algorithms for Task Recommendations

We present our solutions based on the two alternative formulations described in previous section.

5.4.1 Solution using Feature Preference Model

We present the techniques to solve the objective function described in Equation 5.1. Throughout the following sections we refer to our algorithms as

`Feature-Based-NNLS` or Feature Based Non-Negative-Least Square. Here, we have a fixed Y matrix and the non-negativity of matrix X needs to be respected. Since Y matrix is fixed in our case, the objective function is quadratic.

In order to solve for X by minimizing the objective function described in Equation 5.1, first we need to take the derivative against each user vector x_w. Let's introduce another notation W_w which denotes a diagonal matrix, where $W^w_{ii} = q_{wi}$.

$$\begin{aligned}\frac{\partial M}{\partial x_w} &= -2\sum_i q_{wi}(p_{wi} - x_w^T y_i)y_i + 2\lambda x_w \\ &= -2Y^t W^w p_w + 2Y^t W^w Y x_w + 2\lambda x_w\end{aligned}$$

Now, by putting $\frac{\partial M}{\partial x_w} = 0$ we get the analytical solution for x_w as -

$$x_w = (Y^t W^w Y + \lambda I)^{-1} Y^t W^w P_w \tag{5.5}$$

However, we cannot solve exactly due to the non-negativity constraint. Hence we want to minimize the L-2 error for each user $\|(Y^t W^w Y + \lambda I)x_w - Y^t W^w P_w\|^2$. There exists two ways to solve this problem - i) the constrained version of this problem can be transformed using a new formulation to an unconstrained version and then solve it ii) or, the problem can be solved by treating as a Generalized Singular Value Decomposition problem [80]

Complexity : The worst case complexity for solving this constrained version of the least square problem for each user is $O(n_l^3)$ since the size of the coefficient matrix in our case is $n_l \times n_l$ [80]. Hence for each user the complexity becomes $O(n_w n_l^3)$.

Presence of User-Feature: The first variant appears where we know the user's feature preference instead of task feature information. Then, we can use the sim-

ilar formulation to find out the Task Feature-matrix and thereby solving the task recommendation problem.

Presence of both User-Feature and Task-Feature: Another variant of this problem appears, when we know only the zero entries of both Task-Feature matrix Y and User-Feature matrix X, then the non-zero entries of both matrices are subject to optimization. This problem is similar to solving a sparse constrained matrix factorization problem. This can be computationally expensive if the number of explicit features are too many.

5.4.2 Solution using Latent Factor Model

Here, we describe the techniques to solve the objective function described in Equation 5.4. We call this algorithm as Implicit Factorization with Task Similarity, `IFTS`. We apply alternating least square based approach to solve this problem. This is an iterative approach where we partially differentiate objective M with respect to both users and items. At each iteration, we fix the item's latent factor matrix V in order to solve for U and vice versa. If we differentiate the objective function with respect to U, we get the similar solution as equation 5.5 for solving U,

$$u_w = (V^t W^w V + \lambda I)^{-1} V^t W^w P_w \tag{5.6}$$

However, for solving V we need to consider the penalty term while differentiating against V

$$\begin{aligned}\frac{\partial M}{\partial v_i} &= -2\sum_i q_{wi}(p_{wi} - u_w^T v_i)v_i + 2\lambda v_i - \lambda \sum_{i'=1}^{n_t} Sim(i,i')v'_i \\ &= -2V^t W^i p_i + 2U^t W^i U v_i + 2\lambda v_i - \lambda \sum_{i'=1}^{n_t} Sim(i,i')v'_i\end{aligned}$$

By setting $\frac{\partial M}{\partial v_i} = 0$, we get the following equation for solving v_i

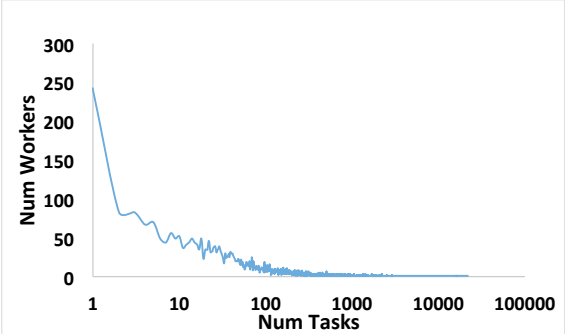
$$v_i = (U^t W^i U + \lambda I)^{-1}(U^t W^i P_i + \lambda * 0.5 * \sum_{i'=1}^{n_t} Sim(i, i')v'_i) \tag{5.7}$$

We achieve reasonable speed-up by using the fact that $U^t W_i U = U^t U + U^t(W_i - I)U$ [96] . We can compute $U^t U$ once and reuse it over all the iterations and number of non-zero elements in $W_i - I$ is exactly the number of users for which $q_{wi} > 0$, let's call that m_i, is much smaller than the total number of users. Then, $m << n_u$. The running time of this algorithm is $O(n_f^2 \mathbf{N} + n_f^3)$ with the additional cost of $\sum_{i'=1}^{n_t} Sim(i, i')v'_i$, where $\mathbf{N}$ is the total number of non-zero entries, $\sum_{i=1}^{i=n_t} m_i = \mathbf{N}$. So the overall running time of this algorithm is $O(n_f^2 \mathbf{N} + n_f^3 + n_w * n_t)$.

5.5 Experiments

System: Our development and test environment uses Python 2.7 on a linux Ubuntu 14.04 machine, with Intel Core i5 2.3 GHz processor and a 6-GB Ram. All numbers are presented as the average of three runs.

5.5.1 Dataset Descriptions



(a) `Ebird`

Figure 5.2: **Worker Task Distribution**

We collect data from a popular citizen science platform named *Ebird* [2]

Ebird: Ebird is a popular citizen science platform for bird observations. We crawl all the observations from year 2012 and randomly choose a set of 5000 workers for our experiments, number of tasks 1767, with a total number of 2.5 million observations. We use 294 locations as task features. Worker task distribution for this dataset is given in Figure 5.2a

Evaluation: We evaluate our method using a hold out test set. We randomly choose 90 percent of our data as the training set and remaining 10 percent we choose as the test set.

5.5.2 Implemented Baseline Algorithms

i)`Implicit-ALS-Negative`: This algorithm is implemented according to [97].This is a modification of the previous algorithm where they introduce the idea of negative signals. If a worker has not completed a task then the total number of times that task has been completed by other users is considered as the weight of the negative signals.

ii) `Feature-Based Regression`: In this algorithm, we assume that the task-feature matrix V is given to us. We solve the regularized regression [98] problem $(C_{ij} - x_i y_j)^2 + \lambda \|X\|^2$ to find X.

Our Proposed Solutions: We refer to our algorithms as follows: Feature Preference Model as `Feature-Based-NNLS` and Implicit Factorization with Task Similarity as `IFTS`. For the latter approach, we run 30 iterations of our algorithms.

[2]Ebird.org

5.5.3 Evaluation Metrics

We use two evaluation metrics to justify our proposed method. Instead of Root Mean Square Error(RMSE)[98], which is common in explicit recommendation system, [96] proposes Mean Percentile Ranking(MPR) as a viable metric for implicit feedback. We also use Precision-Recall curve introduced in [97] for evaluation. Our recommendation is based on the estimated Worker-Task Preference matrix, $\hat{P}$. For `Feature-Based-NNLS`, $\hat{P} = XY$, where X is Worker-Feature matrix and Y is Task-Feature matrix. For `IFTS`, $\hat{P} = UV$. We experimented with different values of α and choose $\alpha = 50$, which gives us reasonable results.

Mean Percentile Ranking(MPR): The mathematical formula to calculate MPR is $\frac{\sum_{ij} c_{ij}\rho_{ij}}{\sum_{ij} c_{ij}}$. Here, c_{ij} indicates the number of task t_j performed by worker u_i. ρ_{ij} is the percentile ranking of the task j for worker i. For instance, if a task t_j is recommended as the first task, then it will be on the 0^{th} percentile, hence $\rho_{ij} = 0$, or if it is the last task it is on the 100^{th} percentile, then $\rho_{ij} = 100$. If the tasks are recommended at random then the process has an expected MPR of 50%. MPR values become high, if the tasks completed by the worker higher number of times, are at the top of the sorted list. **Precision-Recall(PR)Curve**: Precision is defined as the percentage of recommended tasks that are relevant, whereas, Recall means the percentage of relevant tasks that are retrieved. In this method, we want to evaluate our method based on how many task in the test set we can correctly predict by taking only (t%) of the top-tasks. We vary t (in an increment of 1%) in a continuous manner and obtain PR curve.

Summary of Results: The objective of our empirical study is to see how effective our proposed task recommendation models are in comparison with the baseline models. Our proposed algorithm `Feature-Based-NNLS` convincingly outperforms the baseline algorithms in both MPR and PR-Curve. The reason behind the worse per-

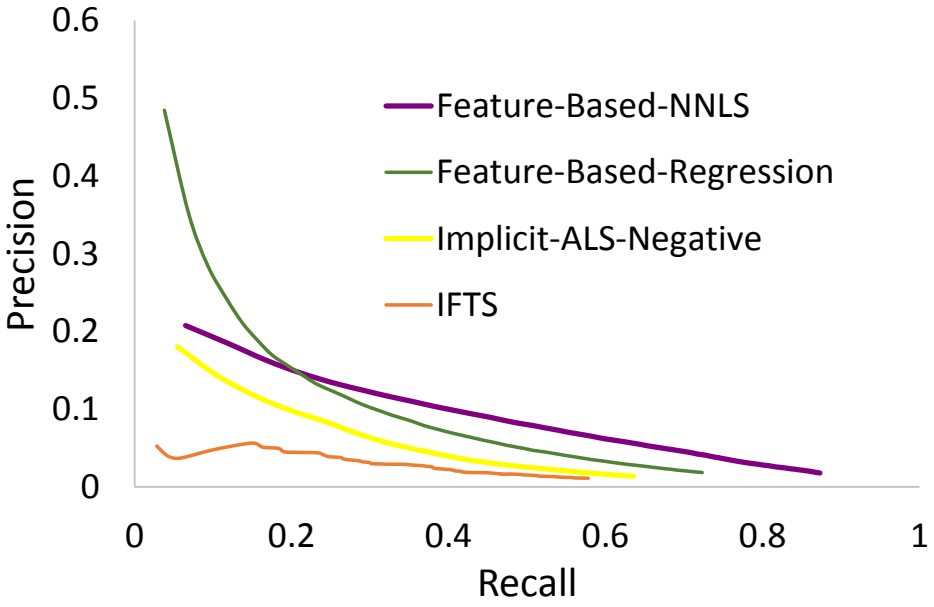
Algorithm	MPR Value
`Implicit-ALS-Negative`	17.3
`Feature-Based-Regression`	13.706
`Feature-Based-NNLS`	**5.68**
`IFTS`	6.87

Table 5.2: MPR Results

formance of `Implicit-ALS-Negative` is that the worker does not choose tasks from a list of available task list, so a task that hasn't been attempted by the user really has no preference rather than "negative preference". `IFTS` also performs reasonably well compare to other methods.

Results: MPR: The First results we present here is a comparison of different algorithms with MPR. `Feature-Based-Regression` and `Implicit-ALS-Negative` perform worse than other methods. `Feature-Based-Regression` does not perform well, because it overfits the training data with negative feature values even after we apply regularization. The reason behind worse performance of `Implicit-ALS-Negative` is twofold - i) Since worker does not choose tasks from a list of available task list, so task that hasn't been attempted by the user really has no preference rather than "negative preference". ii) Substituting task availability as "negative preference" as done by [97] may not be an ideal solution for the problem. `IFTS` performs better than the other baselines. `IFTS` performs better in a small correlated feature space, which explains the better performance in our dataset.

PR curve: Figure 5.3a show the precision recall results for Ebird dataset. While this result mostly corroborate the MPR results, `IFTS` does not perform as well as expected.



(a) `Ebird`

Figure 5.3: **PR Curve**

5.6 Related Work

No existing work has studied the implicit feedback based task recommendation problem considering explicit features for crowdsourcing applications. However, here we describe some of the existing work which are of similar interest.

Recommender systems: Researchers have studied both content based recommender systems and collaborative filtering for personalized recommendation. The former creates a user profile based only on her individual preferences and suffers from sparsity and low quality recommendation [99, 100]. Collaborative filtering exploits user-user similarity or task-task similarity for recommending items to user. They are broadly categorized into two categories - i) Neighborhood based and ii) Latent factor based [101, 102] models. Several efforts have been made to exploit the benefits of both of this approach by combining them together [103, 104, 105]. However, our study suggests that their techniques mostly rely on explicit feedback or content

based feedback, whereas our model relies on implicit feedback. This precludes direct adaptation.

With the growing popularity of websites like Quora[3] and StackExchange[4], an orthogonal problem is to find a set of experts for a given task. This is orthogonal to our work, as these methods try to find the right worker given a task, whereas, we focus on the opposite problem of finding the right tasks for workers.

Task Recommendation in Crowdsourcing: Task recommendation with explicit observation is studied in [95], where they employ probabilistic matrix factorization to recommend task in Mechanical Turk. However, they assume the presence of worker's search history and task accept/reject data, from which they get explicit signal whether user likes or dislikes a task. [5] proposes classification based task recommender system, where the authors first create a user profile based on user meta-data using explicit feedback, then train a binary classifier to determine the likelihood of user selecting that particular task. Pick a crowd [4] uses social network as well as worker's information for task recommendation. We are the first to treat worker-task completion history as implicit observations and incorporate task feature information for recommendation.

5.7 Conclusion

In this paper, we study the task recommendation problem in crowdsourcing applications considering implicit feedback and explicit task features. We formalize the problem as an optimization problem and propose two alternative formulations. We design two elegant solutions that exploit implicit feedback, explicit features, as well as similarity between the tasks in constraining the latent factors of the tasks. We

[3]Quora.com

[4]www.stackexchange.com

formally analyze the complexity and present the efficacy of our proposed solutions by comparing against multiple state-of-the-art techniques.

CHAPTER 6

A Probabilistic Framework for Estimating Pairwise Distances Through Crowdsourcing

Estimating all pairs of distances among a set of objects has wide applicability in various computational problems in databases, machine learning, and statistics. This work presents a *probabilistic framework for estimating all pair distances through crowdsourcing, where the human workers are involved to provide distance between some object pairs*. Since the workers are subject to error, their responses are considered with a probabilistic interpretation. In particular, the framework comprises of three problems : (1) Given multiple feedback on an object pair, how do we combine and aggregate those feedback and create a probability distribution of the distance? (2) Since the number of possible pairs is quadratic in the number of objects, how do we estimate, from the known feedback for a small numbers of object pairs, the unknown distances among all other object pairs? For this problem, we leverage the metric property of distance, in particular, the triangle inequality property in a probabilistic settings. (3) Finally, how do we improve our estimate by soliciting additional feedback from the crowd? For all three problems, we present principled modeling and

solutions. We experimentally evaluate our proposed framework by involving multiple real-world and large scale synthetic data, by enlisting workers from a crowdsourcing platform.

6.1 Introduction

In this chapter, we investigate the following problem: *how to obtain pairwise distance values between a given set of objects by using feedback from a crowdsourcing platform?* This problem lies at the core of a plethora of computational problems in databases, machine learning, and statistics, such as top-k query processing, indexing, clustering, and classification problems. We consider an approach where feedback from the crowd is solicited in the form of simple pair-wise comparison questions. As an example, given two images (a, b), workers are asked to rate (in a scale of $[0, 1]$) how dissimilar these two images are. The worker response may be interpreted as the *distance* between the two images. Although the number of pairwise questions is quadratic in the number of objects, the main idea in this chapter is to only involve the workers in answering a small number of key pair-wise questions, and to estimate the remaining pair-wise distances using the metric properties of the distance function, in particular the *triangle inequality property* [106] - a property that is true for distance functions that arise in many common applications.

Our iterative crowdsourcing distance estimation framework has three key probabilistic components. When we solicit distance information for a specific object pair from multiple workers, we recognize that due to the subjectivity of the task involved, workers may disagree on their feedback, or may even be uncertain about their own estimate. Thus we develop a probabilistic model for *aggregating* multiple workers feedback to create a single probability distribution of the distance learned about that object pair. Next, given that we have learned the distance distributions of several

object pairs from the crowd, we *estimate* the probability distributions of the remaining pairwise distances by leveraging the triangle inequality property of the distances. Finally, if there is still considerable "uncertainty" in the learned/estimated distances and we have an opportunity to solicit additional feedback, we investigate *which object pair* should we choose to solicit the next feedback on. This iterative procedure is continued until all pair-wise distances have been learned/estimated with a desired target certainty (or alternatively, the budget for soliciting feedback from the crowd has been exhausted).

Novelty: There have been a few prior works that have studied computational problems using crowdsourcing that require distance computations. For example, entity resolution [107, 7, 108] problems investigate entity disambiguation, and [8] study top-k and clustering problems in a crowdsourced settings. However, these works have developed their formalism and solutions tightly knit to their specific applications of interest, and do not offer any obvious extension to solve other distance-based applications. For example,the work in [109] is focused on determining whether two objects are the same or not, and not on the broader notion of quantifying the amount of distance between them. In contrast, our proposed framework offers a unified solution to all these computational problems, as they all can leverage our distance estimation framework to obtain the distance between any pair of objects. Please note that once all pair distances are computed, finding the top-k objects, or finding the clusters of the objects is easier to compute. Hence, our problem is more general than the above mentioned body of works. We discuss related work more thoroughly in Section 6.7.

Challenges and Technical Highlights: There are substantial challenges in formalizing and solving the key problems that arise in our three probabilistic components. Perhaps the most straightforward is the first component, i.e., how to aggregate

the feedbacks received from multiple workers into a single pdf that describes the distance between two objects. There has been several prior works on reconciling the answers from multiple workers, which range from simple majority voting to sophisticated matrix factorization techniques [110, 111] on binary data, or opinion pooling [112, 113, 114] on categorical data. However these methods are largely focused on aggregating Boolean/categorical feedback (e.g., "are these two entities the same?"), whereas in our case we need to merge the potentially diverging and uncertain numeric (distance) feedback from multiple workers into a single probability distribution.

The most challenging aspect of our framework is the second component. Knowing distance distributions of some of the object pairs from the crowd, we have to estimate the probability distributions of the distances of the remaining object pairs, by leveraging the metric property of the distance. While the intuitive idea is simple (e.g., "if a is close to b, and b is close to c, then a and c cannot be too far apart"), the problem is challenging because (a) the known distances themselves are distributions rather than deterministic quantities, and (b) the metric property imposes interdependence between all the pairwise distances in a complex manner. In fact, since there are $n(n-1)/2$ pair-wise distances (where n is the number of objects), each such distance can be assumed to be a random variable such that all distances are jointly distributed in a high dimensional $(n(n-1)/2)$ space with interdependencies governed by the triangle inequality. In principle, this joint distribution must be first computed, and then the (marginal) pdfs computed as estimates of the unknown distances. The unknown pairs cannot be estimated in isolation, as a small change in one pdf is likely to disrupt the joint distribution and the triangle inequality property impacting the other pdfs.

We argue that in certain cases, computing the joint distribution may require us to solve a mixture of *over and under-constrained* nonlinear optimization system,

whereas in other cases it may reduce to solving an under-constrained system with many feasible solutions ([115]). For the former cases, we formalize the optimization problem as a combination of *least squares and maximum entropy* formulation and present algorithm `LS-MaxEnt-CG` that adopts a conjugate gradient approach [116, 117] to iteratively compute the joint distribution. For the latter cases, the problem reduces to that of maximizing entropy, and we present an algorithm `MaxEnt-IPS` that leverages the idea of *iterative proportional scaling* [118, 119] to efficiently converge to an optimal solution. Both of these solutions, while ideal, only work for small to moderate problem instances since they are exponential in the dimensionality of the joint distribution being estimated. Consequently, we also present a heuristic solution `Tri-Exp` that scales much better and can handle larger problem instances.

In the third component, our task is to decide, from among the remaining unknown object pairs, which one to select for soliciting distance feedback from the crowd. Intuitively, the selected object pair should be the one whose distance (after being learned from the crowd) is likely to reduce the "overall" uncertainty of the remaining unknown distance pdfs the most, i.e., minimize the *aggregated variance* of the remaining pdfs. To solve this problem in a meaningful way, it is critical to be able to model how workers are likely to respond to a solicitation, because their anticipated feedback needs to be taken into account for selecting the most effective pair. Finally, we also recognize that this approach of resolving one object pair at a time by the crowd may be sub-optimal and slow to converge. Thus, we also describe an extension where we "look ahead" and select multiple promising unresolved object pairs, and engage the crowd in simultaneously providing feedback for these pairs.

Summary of Contributions: In summary, we make the following contributions in this chapter:

- We consider the novel problem of all-pairs distance estimation via crowdsourcing in a probabilistic settings.
- We identify three key sub-components of our iterative framework, and present formal definitions of problems and the solutions for each of the component (Sections 6.2,6.3,6.4,6.5).
- We experimentally evaluate our framework using both real world and synthetic datasets to demonstrate its effectiveness (Section 6.6).

6.2 Data Model and Problem Formulations

We first describe the data model and then formalize the problems considered in this chapter.

6.2.1 Data Model

Objects and Actual Distances: We are given a set $\mathcal{O}$ of n objects, with no two objects being the same. Objects could be images, restaurants, movies, etc. Let $d(i, j)$ be the actual distance between objects i and j. Assume that all distances are normalized within the interval $[0, 1]$, where larger values denote larger distances, and that metric properties are satisfied, in particular the *triangle inequality* [106] or *relaxed triangle inequality* [120] property, as we define below. We are interested in using this property for learning all the $\binom{n}{2}$ pairs of distances.

Triangle Inequality Property: For every three objects (i, j, k) that comprise a triangle $\triangle_{i,j,k}$, $d(i, j) \leq d(i, k) + d(k, j)$ and $d(i, j) \geq \big|d(i, k) - d(k, j)\big|$.

To lift the strict notion of triangle inequality, one can consider *relaxed triangle inequality*, that assumes $d(i, j) \leq c.(d(i, k) + d(k, j))$, where c is a known constant that is not too large. Indeed, the *relaxed triangle inequality* [120] property allows us to effectively incorporate subjective human feedback from crowd workers.

Question: A question $Q(i, j)$ to a worker requests feedback on her estimate of $d(i, j)$. The same question Q is directed to m different workers in the available workers pool, in order to gather multiple feedback.

Feedback: Let $f(i, j)$ represents a worker's feedback for the distance. The worker could either give a single value, or a range/distribution of values (if she is uncertain about the distance).[1] Even if the worker gives a single value, if it is known from past history of her performance that this worker is prone to making errors and is only correct with a certain probability p (say, 80%) (referred to as correctness probability), then her single-value feedback can be converted to a more general probability distribution (pdf) over the range $[0, 1]$ (e.g.,using techniques described in Section 6.3). We henceforth assume that the "raw" feedback of the worker has been appropriately processed into a pdf over $[0, 1]$.

Known and Unknown Distances: Once a distance question $Q(i, j)$ has been answered by multiple workers, their respective feedbacks needs to be *aggregated* into a single pdf representing how the crowd has estimated the distance between i and j. Exactly how this aggregation is done is the first of the three key challenges of this chapter, and is described in detail in Section 6.3. We denote the random variable described by this pdf as $d^k(i, j)$, where the superscript k denotes that the distance is now "known". Note that it still may not be the *actual* deterministic distance $d(i, j)$, unless the crowd's responses are completely error free, which is often not the case in practice.

Of the $\binom{n}{2}$ distances, let D_k represent the set of known distances, i.e., the ones for which feedback has been obtained from the crowd. Let D_u represent the

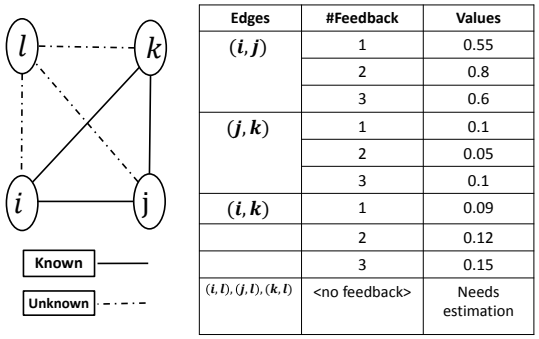
[1]The latter type of feedback is common in experts opinion aggregation problems [121], where a worker has partial knowledge on a particular topic and her answer reflects that with a distribution over the possible answers.

Notation	Interpretation
$\mathcal{O}$	set of n objects
$d(i,j)$	distance between objects i and j
$Q(i,j)$	asking distance on an object pair (i,j) in $[0-1]$ scale
$f(i,j)$	feedback on object pair (i,j)
$\triangle_{i,j,k}$	triangle formed by objects (i,j,k)
$d^k(i,j), d^u(i,j)$	known and unknown distance between an object pair (i,j), respectively
D^k, D^u	known and unknown set of distances, respectively
$\mathbf{D}$	distance vector
$Pr(\mathbf{D})$	joint probability distribution of $\mathbf{D}$
$\mathbf{W}$	vector representing all buckets of the multi-dimensional histogram of $Pr(\mathbf{D})$
m	m different feedbacks on the same question
$\mathbf{A}$	a Boolean matrix of constraints

Table 6.1: Notations

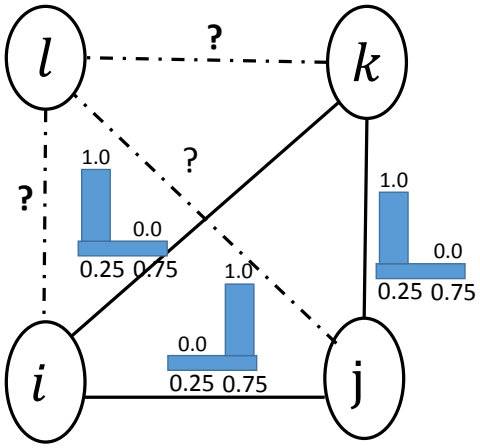
remaining set of "unknown" distances, i.e., distances between those pair of objects for which feedback has not been explicitly obtained from the crowd. Consider $d^u(i,j) \in D_u$. Even though no information about this distance has yet been solicited, some distributional information about this distance can be derived since it depends on other pairwise distances in a complex manner (due to the triangle inequality property). We discuss this issue next.

Joint Distribution of All Pairs Distances: Consider the set of all distances $D_k \cup D_u$. We may view this set as a *distance vector* $\mathbf{D}$ of length $\binom{n}{2}$, whose every entry is a random variable representing the distance between the respective two objects



Edges	#Feedback	Values
(i,j)	1	0.55
	2	0.8
	3	0.6
(j,k)	1	0.1
	2	0.05
	3	0.1
(i,k)	1	0.09
	2	0.12
	3	0.15
$(i,l),(j,l),(k,l)$	<no feedback>	Needs estimation

(a) Illustration of Example 2



(b) Distances as Distributions (Histograms with $\rho = 0.5$)

Figure 6.1: **Illustrative Example.**

(i, j). The space of all instances of $\mathbf{D}$ is $[0, 1]^{\binom{n}{2}}$, however since the $\binom{n}{2}$ distances are interdependent upon each other due to the metric properties, the *valid instances* are those that satisfy the triangle property, i.e., for the triangle $\triangle_{i,j,k}$ defined by any three objects (i, j, k), the three corresponding distances should satisfy the triangle inequality.

Let $Pr(\mathbf{D})$ represent the joint probability distribution of $\mathbf{D}$. Our task is to estimate $Pr(\mathbf{D})$ such that the marginal distribution for a known random variable $d^k(i, j)$ should correspond to the pdf learned from the crowd. We note that once we have an accurate estimation of $Pr(\mathbf{D})$, we can get estimates of the distributions of the unknown random variables $d^u(i, j)$ by computing their marginals. In the next subsection we formalize the problems considered in this chapter.

Table 6.1 summarizes the notations used in the chapter.

Example 2. *Image indexing for K-nearest neighbor queries: Our proposed framework is apt to process K-nearest neighbor queries over an image database, where, given a query image, the objective is to obtain an ordered list of images from the database, ordered by how closely they match the query image. To handle such queries faster,*

one potential avenue is to pre-process the image database and create an index that will cluster the images according to their distance among themselves. Then, as an example, if we have found that a query image $\mathcal{I}$ is far from a database image i and and if the indexes inform us that another image j is close enough to i, then, we may never need to actually compute the distance between $\mathcal{I}$ and j.

With such an application in mind, consider a toy image database in Figure 6.1a with $n = 4$ images (i, j, k, l), where our eventual goal is to find the distances between all pairs of images. Assume that out of six possible pairs of distances, three are known: (i, j), (j, k), and (i, k). I.e., for each of these pairs, we have solicited feedback from several workers in the crowd, and aggregated the feedbacks to obtain a single probability distribution to describe the distance. The distances of the remaining three pairs are unknown and need to be estimated, again as probability distributions. Furthermore, if we need to solicit further feedback on a question, i.e., get the crowd to provide distance for an unknown pair, we intend to find what is the best question (best pair) to ask.

6.2.2 Problem Formulations

Recall from Section 1 that the iterative distance estimation framework involves three probabilistic components, which gives rise to three problems that need to be solved: (a) how to aggregate feedbacks from multiple workers for a specific distance question, (b) given some of the learned distances, how to estimate the remaining unknown distances, and (c) which object pair to select next for soliciting feedback from the crowd. In the remainder of this section, we provide formal definitions of these problems, and offer some insights into their complexities.

6.2.2.1 Problem 1: Aggregation of Workers Feedback for a Specific Object Pair

The first problem may be specified as follows:

Problem 6. *Given a set of m feedbacks for the distance question $Q(i,j)$, where each feedback could be a pdf, aggregate those feedback to create a single pdf for the random variable $d^k(i,j)$.*

Using Example 2, this is akin to aggregating three different feedbacks from three different workers to compute $d^k(i,j)$.

6.2.2.2 Problem 2: Estimation of Unknown Distances

In this problem we need to leverage the known aggregated distances in D_k to estimate the remaining unknown distances D_u. Obviously, if the distances are completely arbitrary, the unknown distances cannot be computed from the known distances. However, if the distances are *metrics*, in particular satisfying the triangle inequality property, then this property can be leveraged in making better estimates of the unknown distances. Many well known distances are metric, such as, $\ell_2, \ell_1, \ell_\infty$, while other popular distances such as, Jaccard distance and Cosine distance could be transformed to metrics. For us, the challenge is to investigate how this property can be used in the case when the distances are probability distributions rather than fixed deterministic values.

Recall that $\mathbf{D}$ is a random vector representing all the $\binom{n}{2}$ distances, and $Pr(\mathbf{D})$ represents the joint distribution of $\mathbf{D}$. We now describe some important properties that $Pr(\mathbf{D})$ should possess.

The space of all instances of $\mathbf{D}$, i.e., $[0,1]^{\binom{n}{2}}$, may be divided into two as follows: (a) *Valid instances*, i.e., any instance of $\mathbf{D}$ such that all triangles $\triangle_{i,j,k}$ satisfy the triangle inequality, and (b) *Invalid instances*, i.e., any instance of $\mathbf{D}$ such that there exists a triangle $\triangle_{i,j,k}$ that does not satisfy the triangle inequality. Thus $Pr(\mathbf{D})$ should be a function constrained such that the cumulative probability mass over all valid (respectively invalid) instances of $\mathbf{D}$ should be 1 (respectively 0).

Additionally, $Pr(\mathbf{D})$ should be constrained such that the marginal distributions corresponding to the individual random variables in D_k (i.e. the known distances) should agree with the corresponding distance pdfs learned from the crowd. However, this constraint may not be always possible to satisfy, as crowd feedback is inherently an error-prone human activity, which can result in inconsistent feedback that violates the triangle inequality. Thus our task will be to estimate $Pr(\mathbf{D})$ such that the marginal distributions corresponding to individual random variables in D_k are "as close as possible" to the pdfs learned from the crowd.

Once such a $Pr(\mathbf{D})$ has been constructed, the pdfs of the unknown distances can estimated by computing the marginal distributions of each variable in D_u.

In the rest of this subsection, we provide more details of the problem formulation.

Discretization of the pdfs using Histograms: For computational convenience, for the rest of the chapter we assume that (single or multi-dimensional) probability distributions are represented as discrete histograms, as is common in databases [122]. In particular, we assume that the $[0, 1]$ interval is discretized into equi-width intervals of width ρ (where ρ is a predefined parameter). A r-dimensional pdf is thus represented by a r-dimensional histogram with $(\frac{1}{\rho})^r$ buckets. Each bucket contains a probability mass representing the probability of occurrence of its center value, and the sum of the probabilities of all buckets equals 1.

For the running example in Figure 6.1a, we use $\rho = 0.5$. Thus a one-dimensional pdf is represented by a 2-bucket histogram, where the first bucket is between $[0 - 0.5]$ with center at 0.25 and the second bucket is $[0.5 - 1.0]$ with center at 0.75. Figure 6.1b of the running example shows how each known distance (known edge) is represented as a one-dimensional histogram after discretizing and aggregating inputs from multi-

ple users, where the feedback values are replaced by the corresponding bucket centers (we describe details of our techniques for input aggregation, i.e., Problem 1, in Section 6.3).

Estimating $Pr(\mathbf{D})$: Once we have the histograms for each individual known edge, the joint distribution $Pr(\mathbf{D})$ needs to be estimated as a multi-dimensional histogram with $(\frac{1}{\rho})^{\binom{n}{2}}$ buckets. Our task is to estimate the probability mass of each of these buckets. Using the running example, there are 2^6 buckets, whose centers range from [0.25, 0.25, 0.25, 0.25, 0.25, 0.25] to [0.75, 0.75, 0.75, 0.75, 0.75, 0.75]. Computing the probability mass of a specific bucket, e.g.,
$Pr(0.25, 0.27, 0.25, 0.25, 0.25, 0.75)$, is equivalent of computing the probability of the simultaneous events $d(i,j) = 0.25$ & $d(j,k) = 0.27$ & $d(i,k) = 0.25$ & $d(i,l) = 0.25$ & $d(k,l) = 0.25$ & $d(j,l) = 0.75$. The computation of $Pr(\mathbf{D})$ can be modeled as a linear system with $(\frac{1}{\rho})^{\binom{n}{2}}$ unknowns, where each unknown represents the probability mass of a bucket. These unknowns have to satisfy three types of linear constraints:

(1) *Constraints imposed by the known pdfs*: $Pr(\mathbf{D})$ should be such that its marginal for any known distance $d^k(i,j)$ should satisfy the corresponding one-dimensional pdf learned from the crowd. Thus, each bucket of each known marginal pdf will generate a linear constraint. In our running example, a one-dimensional bucket such as $Pr(d(i,k) = 0.25)$ will generate a linear equation of the form $\sum Pr(*, *, 0.25, *, *, *) = Pr(d(i,k) = 0.25)$.

(2) *Constraints due to triangle inequality*: Some of the buckets in the joint distribution must have zero probability mass if they violate triangle inequality constraints. In our running example, consider any of the 8 bucket of the form $(0.75, 0.25, 0.25, *, *, *)$. The probability mass of each such bucket has to be set to 0, since $d(i,j) = 0.75$, $d(j,k) = 0.25$ and $d(i,k) = 0.25$ does not satisfy the triangle inequality (this happens

irrespective of any combination of the values for the remaining three edges, hence they are represented as '*').

(3) *Probability axiom constraint*: A final constraint requires that the sum of all the buckets of the joint distribution adds up to 1. In our running example, this implies that $Pr(0.25, 0.25, 0.25, 0.25, 0.25, 0.25) + Pr(0.25, 0.25, 0.25, 0.25,$
$0.25, 0.75) + \ldots + Pr(0.75, 0.75, 0.75, 0.75, 0.75, 0.75) = 1$.

If $\mathbf{W}$ represents the vector of $(\frac{1}{\rho})^{\binom{n}{2}}$ unknowns, and M represents the set of constraints, then the linear system may be expressed as $\mathbf{AW} = \mathbf{b}$, where $\mathbf{A}$ is a Boolean matrix of size $|M| \times (\frac{1}{\rho})^{\binom{n}{2}}$, and $\mathbf{b}$ is a vector of length $|M|$. Interestingly, as the following discussion shows, solving this linear system is not a straightforward task.

Scenario 1: Over-Constrained Case: In general, an *over-constrained linear system* $\mathbf{AW} = \mathbf{b}$ is one which has no feasible solution [123]. In our case, it is indeed possible that the marginal distributions corresponding to the individual random variables in D_k (i.e. the known distances) that are learned from the crowd gives rise to an over-constrained scenario. This is because crowd feedback is inherently an error-prone human activity, which can result in inconsistent feedback that violates the triangle inequality. For example, $\triangle_{i,j,k}$ in Example 2 has only one deterministic instance with edge weights $d(i,j) = 0.75$, $d(j,k) = 0.25$ and $d(i,k) = 0.25$. Clearly, $\triangle_{i,j,k}$ does not satisfy the triangle inequality, since $d(i,j) > d(i,k) + d(j,k)$. Hence, there is no valid joint distribution $Pr(\mathbf{D})$ which can estimate the known pdfs.In such cases, we estimate $Pr(\mathbf{D})$ such that the marginal distributions corresponding to individual random variables in D_k are "as close as possible" (using *least squares* principle) to the pdfs learned from the crowd. More formally, given $\mathbf{A}$ and $\mathbf{b}$, we estimate $\mathbf{W}$ such that $||\mathbf{AW} - \mathbf{b}||^2$ is *minimized*.

Scenario 2: Under-Constrained Case: In general, an *under-constrained linear system* **AW**=**b** is one which has multiple feasible solutions [123]. In our case, while estimating **W**, we may also encounter under-constrained scenarios. Using Example 2 and considering triangle $\triangle_{i,j,l}$, we note that any of the following solutions are feasible: $d(i,l) = 0.75$, $d(l,j) = 0.75$, or $d(i,l) = 0.75$, $d(l,j) = 0.25$, or $d(i,l) = 0.25$, $d(l,j) = 0.75$. In such cases, *maximum entropy* principles [118] are used to choose a solution that is consistent with all the constraints but otherwise is as uniform as possible. More formally, the objective is to solve the linear system **AW**=**b** that maximizes the entropy of the joint distribution $-\sum_{w \in \mathbf{W}} Pr(w) \log Pr(w)$.

Scenario 3: Combined Case: Since our problem instances may involve both over and under-constrained scenarios, we unify both into a single minimization problem using a weighted linear combination, where the weight λ can be used to tune the solution to ensure better least square or higher uniformity. Our final problem is described as follows:

Problem 7. *Estimate the joint distribution vector* $\mathbf{W}$ *such that* $f(\mathbf{W}) = \lambda \times ||\mathbf{AW} - \mathbf{b}||^2 + (1-\lambda) \times \sum_{w \in \mathbf{W}} Pr(w) \log Pr(w)$ *is* minimized.

Before we move to our next problem definition, we point out an interesting issue. The exponential size of Problem 2 (the number of buckets in the multi-dimensional histogram is intractably large for most real-world instances) suggests that a complete solution of Problem 2 is prohibitive. Fortunately, we observe that computing the joint distribution is merely a intermediate (and not strictly necessary) objective - our eventual objective is to estimate the one-dimensional pdfs of the unknown distances $d^u(i,j)$. This issue is discussed in more detail in Section 6.4, and in particular we present heuristics to directly compute the unknown one-dimensional pdfs without having to compute the intermediate joint distribution.

6.2.2.3 Problem 3: Asking the Next Best Question

Recall that our overall approach is an iterative process. If we have the need to solicit further feedback from the crowd, we have to select an object pair from D_u, as human workers have not yet been involved in providing feedback about such pairs. Our objective is to select the most promising pair, i.e., that is most likely to reduce the "uncertainty" of the remaining unknown distances the most. We measure uncertainty by aggregating the *variances* of the remaining unknown distance pdfs (the variance of $d^u(i,j)$ with mean μ is measured as $\sigma^2_{d^u(i,j)} = \sum_{\forall q} p_q * (q-\mu)^2$).

Problem 8. *From the set D_u of the candidate object pairs, choose the next best question $Q(i,j)$ to solicit feedback from the human workers, such that, upon receiving the feedback, the* aggregated variance *over the remaining unknown distances is minimized.*

Aggregated variance, `AggrVar` is formalized in one of the two natural ways, *average variance or largest variance*:

(1) Average variance over the remaining unknown distances:

$$\frac{\sum \sigma^2_{d^u(i',j')}}{|D_u| - 1}, d^u(i',j') \in \{D_u - d^u(i,j)\}. \tag{6.1}$$

(2) Largest variance over the remaining unknown distances:

$$\max_{d^u(i',j')} \sigma^2_{d^u(i',j')}, d^u(i',j') \in \{D_u - d^u(i,j)\}. \tag{6.2}$$

Considering Example 2, this problem will seek to choose the next best question (i.e., edge or object pair) from $D_u = \{(i,l),(j,l),(k,l)\}$.

6.3 Problem 1: Aggregation of Workers Feedback

In this section, we describe our proposed solution `Conv-Inp-Aggr` of aggregating multiple feedbacks on a single object pair (i.e., an edge).

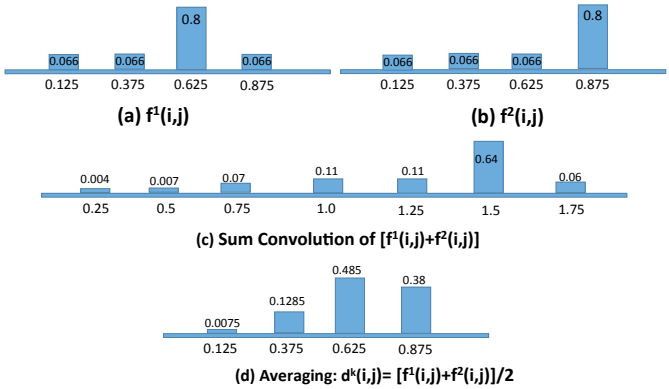


Figure 6.2: Worker Feedback Aggregation

In general, given a set of m different feedbacks $f^1(i,j), f^2(i,j), \ldots f^m(i,j)$, where each feedback is a random variable describing distance on an object pair (i,j), such that the set of random variables are *independently distributed*, our objective is to define a new random variable whose distribution represents the *average* of the underlying input pdfs, i.e., pdf of $\frac{f^1(i,j)+f^2(i,j)+\ldots+f^m(i,j)}{m}$. The independence assumption allows us to use the prior technique of *sum-convolution* [124] to obtain the sum of the input pdfs and then averaging that convolved pdf to obtain the average.

Algorithm 5 `Conv-Inp-Aggr`

1: **Input:** Set of m feedbacks for (i,j).
2: Perform a sequence of $m-1$ Sum-convolutions over the feedback pdfs.
3: Calculate $d^k(i,j)$ by re-calibrating the resultant pdf of previous step into pre-specified adjusted range. This step require averaging over the bucket values and reallocate the probability masses accordingly.
4: **return** $d^k(i,j)$

We illustrate this approach using the first two feedbacks for the pair (i, j) in our running example in Figure 6.1a. The first worker's feedback (denoted as $f^1(i, j)$) of 0.55 is converted into a pdf. This is shown in Figure 6.2(a) as a 4-bucket histogram (i.e., with $\rho = 0.25$, buckets with boundaries $[0-0.25], [0.25-0.5], [0.5-0.75], [0.75-1.0]$, and centers at 0.125, 0.375, 0.625, 0.875 respectively). As the feedback value 0.55 is in $[0.5-0.75]$, we can assign a probability mass of 1 to this bucket, and 0 to all other buckets. However, if we have prior information that the worker is only correct 80% of the time (correctness probability $p = 0.8$), we can assign a probability mass of 0.8 to the bucket $[0.5-0.75]$, and distribute the remaining probability mass uniformly among the remaining three buckets. This latter approach is used to generate the pdf illustrated in Figure 6.2(a). Similarly, Figure 6.2(b) shows the pdf for feedback 2 of (i, j).

The sum-convolution of these two pdfs is presented in Figure 6.2(c). Since the centers of the buckets of each of the individual pdf are between $[0.125, 0.875]$, their sum can be any value between $[0.25, 1.75]$. For each discrete value x between $[0.25, 1.75]$, the probability of $f^1(i, j) + f(^2(i, j)$ equal to x is calculated by computing the joint probability of $f^1(i, j) = x'$ and $f^2(i, j) = x$", such that, $x' + x" = x$.

With $m = 2$ feedbacks, the bucket values are then reassigned to the centers as follows: $0.25 \rightarrow 0.125$, $0.5 \rightarrow 0.25$, ..., $1.75 \rightarrow 0.875$. After this is done, if we have a transformed bucket center with non-zero probability that does not correspond to any of the input buckets, then the mass of that bucket is redistributed to its closest bucket. When two buckets are equally close, the mass is equally divided between the two buckets. As an example, since $1.0 \rightarrow 0.5$ after averaging, but 0.5 does not correspond to any bucket center , the probability mass of $Pr(f^1(i, j) + f^2(i, j) = 1.0)$ gets uniformly split between its two closest centers 0.375 and 0.625. The resultant distribution is given in Figure 6.2(d).

Figure 6.1b shows the aggregation results for (i, j) of Figure 6.1a with worker being completely accurate ($p = 1.0$) and with $\rho = 0.5$.

Running Time: If each pdf is approximated using an equi-width histogram of width ρ, the time to perform average convolution involving m different pdfs is $O(m \times 1/\rho^2)$ [124].

6.4 Problem 2: Estimation of Unknown Distances

In this section, we present our proposed solutions of the problem 7- i.e., how to estimate the distance of the unknown object pairs from the given known distances. Using Example 2, this step is to estimate three unknown distances $D_u = \{(i, l), (j, l), (k, l)\}$, by leveraging the three known distances. We present two alternative solutions - an optimal solution by computing joint distribution that is exponential to the number of object pair $\binom{n}{2}$, and a much faster heuristic alternative.

6.4.1 Algorithms for Optimal Solution

Recall our proposed formulation in Section 6.2.2 and note that the optimal solution of computing the unknown distances is to first produce a joint distribution $Pr(\mathbf{D})$ on a high-dimensional space over all $\binom{n}{2}$ object pairs. This is due to our underlying abstraction that assumes that all objects are connected to each other which gives rise to a complete graph - hence the distribution of an unknown edge can not simply be learned in isolation. Once the joint distribution is obtained, the unknown pdfs are to be computed as marginals from the joint distribution. We investigate and design algorithms for the following two scenarios:

(1) As demonstrated in Example 2, our problem can unfortunately be both *over as well as under-constrained*. In fact, when the known pdfs are inconsistent (i.e., do not

satisfy triangle inequality), there may not be any *feasible solution* to compute $Pr(\mathbf{D})$ that satisfies all the known pdfs. At the same time, a part of our solution space may still be under-constrained, especially the part that involves the unknown pdfs where multiple feasible solutions may exist.

(2) For the special case when the known pdfs are consistent, the scenario is merely under-constrained and may have multiple feasible solutions, as we describe in Section 6.4.1.2.

6.4.1.1 Combined Case

For this scenario, the problem of computing the joint distribution is formalized as an optimization problem (Problem 7) with the objective to minimize a weighted linear combination of least square and negative entropy (notice $-Pr(w)\log Pr(w)$ is the entropy), i.e., $f(\mathbf{W}) = \alpha \times ||\mathbf{AW} - \mathbf{b}||^2 + \beta \times \sum_{w \in \mathbf{W}} Pr(w)\log Pr(w)$ is to be *minimized*. The first part of the formulation is designed for the over-constrained settings, i.e., we satisfy the known pdfs as closely as possible if there is no feasible solution, whereas the second part of the formulation is to handle under-constrained nature of the problem through maximum entropy modeling that will choose the joint distribution model that is consistent with all the constraints but otherwise is as uniform as possible. From the joint distribution $Pr(\mathbf{D})$, we obtain the unknown distance pdfs by computing appropriate marginals.

Lemma 7. *$f(\mathbf{W})$ is convex.*

Proof. (Sketch) It can be shown that the linear aggregation of two convex functions is always convex [125], which proves the above lemma. □

Algorithm `LS-MaxEnt-CG`: Based on Lemma 7 $f(\mathbf{W})$ is convex. We propose Algorithm `LS-MaxEnt-CG`, by appropriately adapting nonlinear conjugate gradient algo-

rithms [116, 117] that are popular iterative algorithms to solve such non-linear convex optimization problems. The overall pseudo-code is presented below in Algorithm 6.

Algorithm 6 `LS-MaxEnt-CG`

1: **Input:** matrix A, constraint vector b, vector $\mathbf{W}$ with $\frac{1}{\rho}^{\binom{n}{2}}$ unknown variables, tolerance error η.

2: Initialize $\mathbf{W}$ with the steepest direction in the first iteration $\Delta\mathbf{W}_0 = -\nabla_{\mathbf{W}} f(\mathbf{W}_0)$

3: In the i-th iteration, compute β'_i using Fetcher-Reeves method [126].

4: Update the conjugate direction: $s_i = \Delta\mathbf{W}_i + \beta'_i s_{i-1}$.

5: Perform a line search to obtain α'_i, $\alpha'_i = \arg\min_{\alpha'} f(\mathbf{W}_i + \alpha' s_i)$.

6: Update the position: $\mathbf{W}_{i+1} = \mathbf{W}_i + \alpha'_i s_i$

7: Repeat Steps $3 - 7$ to until the $error \leq \eta$.

8: **return** $f(\mathbf{W})$

Using Example 2 with $\rho = 0.5$, the joint distribution produces the probability for each of the 2^6 buckets that sum up to 1. From this joint distribution, the marginal distributions can be computed for the three unknown edges. $(i, l) : [0.25 : 0.366, 0.75 : 0.634]$,$(j, l) : [0.25 : 0.366, 0.75 : 0.634]$, $(k, l) : [0.25 : 0.366, 0.75 : 0.634]$.

Running Time: It has been shown in [117] that conjugate gradient has a running time complexity of $O(m'\sqrt{\kappa})$, where m' is the number of non-zero entries in the matrix $\mathbf{A}$ and κ is the number of iteration before convergence. However, in our case, as described in Section 6.2.2, the size of the input matrix $\mathbf{A}$ itself is exponential to the number of object pairs.

6.4.1.2 Under-Constrained Case

For the under-constrained settings, the optimization function becomes simpler, with the objective to maximize entropy $f(\mathbf{W}) = -\sum_{w \in \mathbf{W}} Pr(w) \log Pr(w)$, while satisfying the known constraints. Each constraint C_i is a restriction on some subset of these possible $(\frac{1}{\rho})^{\binom{n}{2}}$ cells to sum up to some observed value $p(C_i)$. More specifically, each $C_i = \sum(w_i \times I_{i,j})$, where $I_{i,j} = 1$ if j-th cell is included in the constraint C_i, and 0 otherwise.

Algorithm `MaxEnt-IPS`: It has been shown that the objective function always has a unique solution as long as the constraints are consistent [119]. Of course, this problem can be solved using a general purpose optimization algorithm. However, we propose `MaxEnt-IPS`, an *iterative proportional scaling (or IPS)* algorithm [118, 119] that exploits the structural property of the objective function and uses the observation that the optimal w_i values can be expressed in the following product form.

$$w_j^{\mu} = \mu_0 \Pi_{C_i} {\mu_i}^{I_{i,j}}$$

For each constraint C_i, there is a constraint μ_i that gets updated inside the IPS algorithm and μ_0 is a normalization constant to ensure that all cells add up to 1. This algorithm iteratively updates the μ_i's and the cell values w_i's. It is guaranteed to converge to the optimal solution as long as all constraints are consistent. Once the histogram buckets W and hence the joint distribution $Pr(\mathbf{D})$ is computed, the unknown marginals are obtained similarly as before. We omit further details and the pseudo-code for brevity but refer to [118, 119] for for more information on the IPS method.

`MaxEnt-IPS` does not converge for the input presented in Example 2 (b), as it is over-constrained. However, if we modify the example such that the aggregated feedback for (j, k) is 0.75 instead of 0.25, then the following outputs are obtained for

the three edges: $(i,l) : [0.25 : 0.333, 0.75 : 0.667]$,$(j,l) : [0.25 : 0.333, 0.75 : 0.667]$, $(k,l) : [0.25 : 0.333, 0.75 : 0.667]$.

Running Time: The maximum entropy modeling is known to be NP-hard [127]. The `MaxEnt-IPS` algorithm terminates based on the convergence of all the μ's. In each iteration it makes updates to all the buckets in the joint distribution, which is exponential in size $(O(\frac{1}{\rho})^{\binom{n}{2}})$. If `MaxEnt-IPS` requires κ iterations to converse, the asymptotic complexity of this algorithm is exponential, i.e., $O(\kappa \times (\frac{1}{\rho})^{\binom{n}{2}})$.

6.4.2 Efficient Heuristic Algorithm

Both the problem variants and their respective solutions studied in Sections 6.4.1.1 and 6.4.1.2 first compute the joint distribution over an $\binom{n}{2}$-dimensional space as optimization problems. After that, the unknown distributions are computed from the joint distribution. Even with $n = 5$ objects and $\rho = 0.5$, the joint distribution is to be computed on an $2^{\binom{5}{2}} = 2^{10}$ dimensional space. Due to its exponential nature, computing the joint distribution is practically impossible as n increases. As a realistic alternative, we next present `Tri-Exp`, an *efficient heuristic algorithm that avoids computing the entire joint distribution, but explores the individual triangles in a greedy manner to estimate the pdfs of the unknown edges.* The pseudo-code is presented in Algorithm 7.

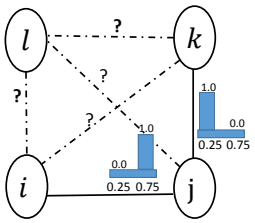


Figure 6.3: Example to Illustrate `Tri-Exp`

While Algorithm `Tri-Exp` avoids computing the joint distribution and instead performs a greedy exploration over the individual triangles one-by-one, there are still considerable challenges - each unknown object pair (edge) is involved in $n-2$ different triangles (with different triangle inequality constraints) and the algorithm must be adapted to estimate the pdf of the unknown edge such that it satisfies all the triangles. In particular, it encounters two scenarios.

Scenario 1: During execution, the algorithm may encounter some triangles which have two edges already known and only the third edge is to be estimated. For such cases, the algorithm *will greedily select that unknown edge that completes the highest number of triangles*, once estimated. When an unknown edge is involved in multiple triangles with two edges known for each triangle, then the final estimated pdf must satisfy the triangle inequality property of all the triangles. We first estimate the pdf of the unknown edge considering each triangle, following which the final pdf is computed by performing the sum-convolution and averaging the convolved pdf (recall Section 6.3), such that the triangle inequality property is satisfied for all the triangles.

Scenario 2: Another scenario that is likely to occur is when there only exists triangles with two unknown edges. In such cases, both of the unknown edges are jointly estimated, by relying on the known edge.

Solution Considering Scenario 1: As an example, consider Figure 6.3 and note that based on this greedy selection, at the very first iteration, it will select (i, k) for estimation, as that will complete at least one triangle $\triangle_{i,k,l}$ (because, the two edges of this triangle are already known and the third edge is to be estimated), whereas none of the other unknown edges will complete any triangle. Considering triangle $\triangle_{i,j,k}$, the algorithm will have to apply the triangle inequality property to select the possible ranges of values that (i, j) is allowed to take.

Our method will estimate the pdf of (i,k) as $Pr((i,k) = 0.25) = 0.0$, $Pr((i,k) = 0.75) = 1.0$ considering $\triangle_{i,j,k}$. After that, (i,k) should also be estimated considering another triangle $\triangle_{i,l,k}$. The final pdf of (i,k) must satisfy the triangle inequality property of both of these triangles.

Algorithm 7 `Tri-Exp`: heuristic distance estimation algorithm

1: **Input:** known and unknown distance edges.
2: **if** There exists triangles with one unknown and two known edges **then**
3: Greedily select that unknown edge and estimate it such that it results in the maximum number of triangles with all known edges
4: **else**
5: When no such triangle is found, consider a triangle and estimate two unknown edges jointly
6: **end if**
7: Perform sum convolution and averaging for all associated triangles such that triangle inequality is satisfied
8: Repeat steps $2-7$ until all edges are estimated
9: **return** distance edges

Solution Considering Scenario 2: Consider Figure 6.3 again and assume that (i,k) is estimated in iteration one. Even after that, both $\triangle_{i,j,l}$ and $\triangle_{j,k,l}$ have two unknown edges.

In $\triangle_{j,k,l}$, where both (k,l) and (j,l) are unknowns and are to be estimated using the pdf of the known edge (j,k). Without further knowledge, we calculate the joint distribution for (j,l) and (k,l) by assigning uniform probability to each of these possible values. Once, we get the joint distribution, we calculate the pdfs for both

(j,k) and (j,l) which will be exactly equal to each other, which is $\{0.25 : 0.5, 0.75 : 0.5\}$. As before, when multiple triangles are involved with an unknown edge, the pdf of that edge needs to be estimated considering triangle inequality property of all the involved triangles.

`Tri-Exp` outputs the following pdfs for the example in Figure 6.3 $(i,k) : [0.25 : 0.5, 0.75 : 0.5], (k,l) : [0.25 : 0.61, 0.75 : 0.39], (j,l) : [0.25 : 0.43, 0.75 : 0.57], (i,l) = [0.25 : 0.4, 0.75 : 0.6]$

Running Time: Time complexity of `Tri-Exp` is $O(|D_u|(n \times \frac{1}{\rho}^2 + log(|D_u|))$, where $|D_u|$ is the number of unknown pairs, ρ is the histogram-width, and n is the number of objects. At worst case, $|D_u| = O(n^2)$; in such cases, the algorithm takes cubic time to run. Nevertheless, this analysis shows that the running time of `Tri-Exp` is substantially superior than its exponential counterparts, `LS-MaxEnt-CG` or `MaxEnt-IPS`.

6.5 Problem 3: Asking the Next Best Question

If there is still considerable "uncertainty" in the learned / estimated distances and we have an opportunity to solicit additional feedback, we investigate (in this third problem) which object pair should we choose to solicit the next feedback on. There are several variants of this problem. In the *online* variant, we have the liberty of asking one question at a time and continue the process until all initially unknown pdfs converges "satisfactorily", or a budget B expires. The budget could be used to specify a limit on the number of questions to be asked, or the maximum number of workers to be involved. In the *offline* variant, we need to decide all questions ahead of time so that the fixed budget expired. In the *hybrid* variant, we could solicit workers feedbacks for several batches of say k questions per iteration. In this chapter we

mainly focus on the online variant, but also present a simple extension to solve the offline problem.

Modeling Possible Worker feedback: Recall the definition of Problem 8 and note that from a given candidate set of questions D_u (where each question is on an object pair), the problem is to select that question which minimizes the aggregated variance `AggrVar` most. The challenge, however, is to be able to *anticipate* possible workers responses that is currently unknown, to be able to guide the optimization problem. A question $Q(i,j) \in D_u$ is essentially a random variable whose distribution has been estimated already by solving Problem 7. Without any further information, the framework has the following limited options to make guesses about future responses of the workers:

(1) The response pdf from the m workers, when aggregated, will be the same as the current estimated pdf of $d^u(i,j)$. Under this scenario, the framework does not learn anything new about $d(i,j)$ and hence `AggrVar` remains unchanged. We therefore do not use this option in our algorithm.

(2) The aggregated response of the worker will be identical to some measures of the current pdf that dictates its central tendency; for example the *mean* μ of the current pdf can used as the anticipated value of the future aggregated feedback.

In this latter case, the pdf of $d^u(i,j)$ changes (its variance becomes 0), and it is also likely to affect the pdfs of other edges (i.e., the joint distribution changes). More intuitively, when a pdf is represented by its mean, the other pdfs (edges) involved with it are likely to demonstrate lower divergence, hence tighter distribution. As described later, this option is used in our algorithm for selecting the next best question.

Consider a very simple example with 3 objects (i,j,k) that satisfy triangle inequality such that $(i,j) : Pr(d(i,j) = 0.125) = 1$; $(i,k) : Pr(d(i,k) = 0.125) =$

$0.9, Pr(d(i,k) = 0.375) = 0.1$. To satisfy triangle inequality, the pdf of the third edge (j,k) must be between $[0.0, 0.5]$. However, if we substitute (i,k) with its mean 0.15 (considering it as a candidate question), the pdf of (j,k) becomes tighter and only between $[0, 0.275]$. It is easy to notice that the latter pdf of (j,k) will result in a smaller variance in comparison with the former one.

Algorithm `Next-Best-Tri-Exp`**:** The algorithm for computing the next best question runs in iteration and considers each candidate question $Q(i,j)$ in turn. Then, it considers the impact of changing the current pdf of the object pair to its mean (to emulate workers' feedback). This is done by re-estimating the pdfs in $D_u - d^u(i,j)$. For that, it uses a sub-routine to solve Problem 7, described in Section 6.4 using any of `LS-MaxEnt-CG`, `MaxEnt-IPS`, or `Tri-Exp` algorithms. Once the unknown pdfs in $\{D_u - d^u(i,j)\}$ are re-estimated, it computes `AggrVar` using either Equation 6.1 or 6.2 and maintains the so-far best question by choosing the minimum. Once all the candidates are evaluated, the best candidate is the one that results in the smallest `AggrVar`. The pseudo-code is presented in Algorithm 8. Using Example 2, this returns (i,l) as

Algorithm 8 `Next-Best-Tri-Exp`: Selecting the next best question

1: **Input:** known and estimated distance edges.
2: **for** $d^u(i,j) \in D_u$ **do**
3: Replace the distribution of $d^u(i,j)$ by its mean
4: Select $d^u(i,j) = \text{argmax}_{\forall d^u(i,j) \in D_u}$ `AggrVar`$(d^u(i,j))$ as the candidate question
5: **end for**
6: **return** $d^u(i,j)$

the next best question, as that minimizes the `AggrVar` based on both formulation of aggregated variance. **Running time:** To choose the next best question, this algo-

rithms has to evaluate each candidate question in D_u. The primary computation time in each candidate question is taken to invoke an algorithm to solve Problem 7 as a subroutine. Therefore, the running time of this algorithm is asymptotically $O(|D_u| \times$ running time of the sub-routine).

Extension to the Offline Problem: If we need to decide how to spend all the budget B ahead of time, we need to decide all the questions offline, we note that the problem becomes computationally more challenging, as there will be an exponential number of possible choices ($\binom{|D_u|}{B}$, assuming the budget allows for B questions) and the ordering of the questions also matters in reducing aggregate variance. However, a simple extension to our current algorithm can effectively solve this offline problem, where we run our online solution B times to select the best B questions greedily. We present experiments on this regard and show that our proposed solution can be effective in solving the offline problem.

6.6 Experimental Evaluation

Our development and test environment uses python 2.7 on a Linux Ubuntu 14.04 machine, with Intel core i5 2.3 GHz processor and a 6-GB RAM. All values are calculated as the average of three runs.

6.6.1 Datasets Description

We use three real world datasets and one synthetic dataset for our experiments. (1) **Image**: The real world dataset is obtained from the PASCAL database[2]. A total of 24 images of 3 different categories are extracted. We generate 3 subsets of size 10, 5, 5 for which we have solicited all pair distance information. Each pair is set up as a HIT (human intelligence task) in Amazon Mechanical Turk (AMT) and

[2]http://host.robots.ox.ac.uk/pascal/VOC/databases.html

we solicit 10 different workers' feedback on the similarity of the images. A total of 50 different workers are involved in this study. (3) **SanFrancisco**: We choose 72 locations from the city of San Francisco and crawl traveling distances (both-ways) among all pair of locations (2556 pairs) using google api[3]. The purpose this dataset is to validate the scalability of our algorithms. Here, we use the traveling distances as worker feedback instead of explicitly soliciting the workers' feedback. (2) **Cora**: This is a real world publication dataset of 1838 records, 190 real world entities. We use this dataset to compare our algorithms with Entity Resolution algorithms in [109]. We choose 3 random instances of this dataset with 20 records, which constitutes of 190 edges. We apply our algorithms in these instances and present our results. (4) **Synthetic**: We generate a large scale synthetic dataset for performing efficiency experiments. Here, we vary from 100 to 400 objects which gives rise from 4950 to 79800 object pairs.Additionally, another small synthetic dataset of 5 objects with 10 edges is generated.

6.6.2 Implemented Algorithms

(1) Worker Feedback Aggregation: We consider the following algorithms:

(i) `Conv-Inp-Aggr`: This is our proposed convolution based solution to aggregate workers feedback that is described in Section 6.3.

(ii) `BL-Inp-Aggr`: We implement a baseline algorithm that creates aggregated pdf by calculating the average probability over each discrete bucket center of the input pdfs. Here we ignore the ordinal nature of the feedback scale and treat each bucket as a categorical value.

(2) Estimation of Unknown Edges: We are unaware of any related works that study distance estimation in probabilistic settings.

[3]https://developers.google.com/maps/

(i) `Tri-Exp`: This algorithm is described in Section 6.4.2.

(ii) `LS-MaxEnt-CG`: This algorithm is designed to estimate the unknown edges considering both over and under constrained settings, described in section 6.4.1.1.

(iii) `MaxEnt-IPS`: This algorithm, described in section 6.4.1.2, refers to the optimal estimation of unknown edges considering only under-constrained settings.

(iv) `BL-Random`: We design a baseline algorithm that is similar to `Tri-Exp`. It estimates the unknown edges considering triangles; however, unlike `Tri-Exp` (which first attempts to consider the edges that complete the highest number of triangles), `BL-Random` arbitrarily chooses unknown edges and estimates them.

(3) Asking the Next Best Question: These algorithms are designed to demonstrate the effectiveness of the next best question in reducing `AggrVar`, as described in Section 6.5. As `LS-MaxEnt-CG` and `Maxent-IPS` are computationally prohibitive, we implement `Tri-Exp` and `BL-Random` as subroutines to decide the next best questions. We divide these algorithms into two parts - *Online* and *Offline*.

Online Algorithms: Here we solicit one question at a time to the crowd (i) `Next-Best-Tri-Exp`: This is our proposed solution in Section 6.5 that uses `Tri-Exp` at each iteration as the subroutine to re-estimate the unknown edges. (ii) `Next-Best-BL-Random`: This is again our proposed solution in Section 6.5 that uses `BL-Random` at each iteration as the subroutine.

Offline Algorithms: Here we solicit a set of questions ahead of time. (i) `Offline-Tri-Exp`: This is the offline variant of `Next-Best-Tri-Exp` described in Section 6.5.

(4) Entity Resolution(ER): As discussed in Section 6.7 on related works, under certain circumstances the problem of *entity resolution*, in particular the techniques proposed in [109], may be considered a special case of the distance estimation

problem considered in this chapter. Consequently, we experiment with the following algorithms:

(i) `Next-Best-Tri-Exp-ER`: This is a modified vesion of `Next-Best-Tri-Exp` algorithm where we find the number of questions that need to be asked so that `Aggr-Var` is zero.

(ii) `Rand-ER` : We implement the *Random* algorithm from [109]. We call this algorithm `Rand-ER`. This algorithm has a proven complexity of $O(nk)$, where n denotes the number of objects and k denotes the number of clusters/similar entities.

6.6.3 Experimental Set up

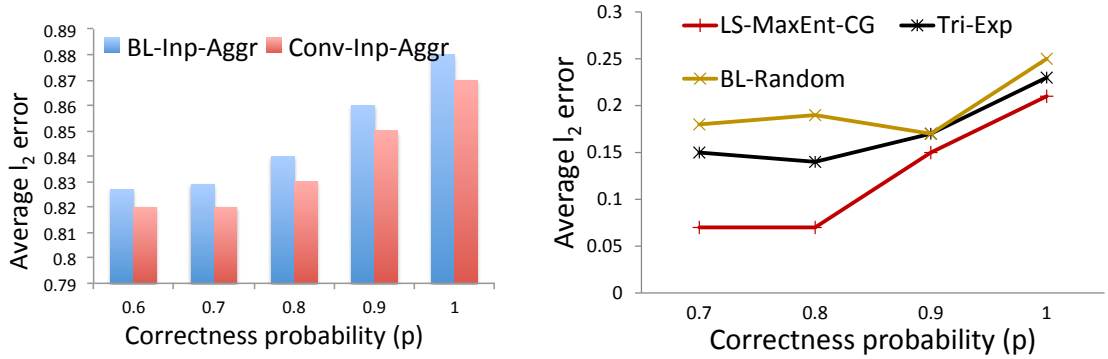
Parameter Settings: Unless otherwise mentioned, we assume $\rho = 0.25$. In other words, there are 4 equi-width buckets with bucket range $[0.0 - 0.25)$, $[0.25 - 0.5)$, $[0.5 - 0.75)$, $[0.75 - 1.0)$ with centers at 0.125, 0.375, 0.625 and 0.875. Depending on the value of p (worker correctness), the distribution of the known edges are created. For example, if a worker provides a feedback of 0.8, with $p = 60\%$, that edge is created by assigning probability of 60% on distance 0.875, and the remaining 40% probability is uniformly assigned to the other buckets. In practice, correctness probability can be obtained by asking a set of screening questions and then by averaging their accuracy. The weight of λ is set to 0.5 (unless otherwise stated) for Problem 7.

Quality Experiments:(i) *Worker Feedback Aggregation*: We use real data for this experiment as this dataset contains multiple workers feedback. We consider each triangle in isolation where all the edge distances are known. Hence, for each edge with 10 different feebacks, we know the ground truth distribution. We use `Conv-Inp-Aggr` and `BL-Inp-Aggr` for aggregating two out of the three edges. Based on our respective algorithm, we estimate the third edge. We then compute the ℓ_2 error of our

estimated edge from the ground truth distribution for the third edge. (ii) *Unknown Edge Estimation*: Since `LS-MaxEnt-CG`, `MaxEnt-IPS` are exponential in the number of object pairs (i.e., $S^{^nC_2}$), we have to limit our settings to a very small dataset with $n = 5$ nodes and 10 edges. We use the small *Synthetic* dataset, as well as a subset of real world dataset for this experiment. For the *Synthetic* dataset, we consider `MaxEnt-IPS` as the optimal solution, and compare the effectiveness of the other three algorithms by calculating the average ℓ_2 error over the unknown edges, compared to the optimal. Out of the 10 edges, we randomly mark 4 edges as known (and create their distribution as described before), and estimate the remaining 6 unknown edges. For the *Image* dataset, all ground truth distributions are known for the selected 5 objects. Like above, we mark 4 randomly chosen edges to be known and estimate the remaining 6 edges by considering the 4 different algorithms. As before, we present the average ℓ_2 error - but this time in comparison with the ground truth. (iii)*Asking the Next Best Question*: We use the *SanFrancisco* dataset for which we have all pair of ground truth information. At each step, we replace the step of asking a question to the crowd by the ground truth information. The default value of p is 1.0 and the default budget $B = 20$ questions. Number of known edges is is set to 90% of the total edges.

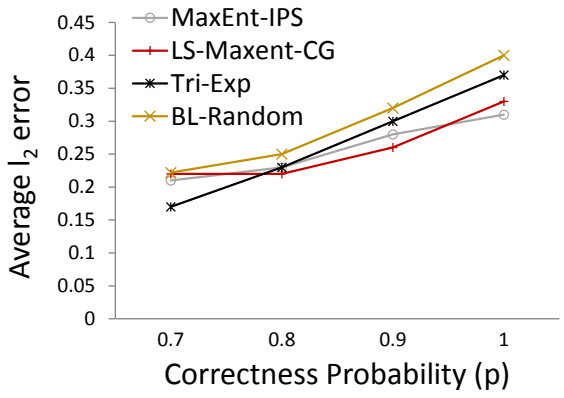
Application to ER: We use *Cora* dataset to perform comparison with ER methods. We assume that each edge is described by a pdf with two ordinal buckets 0 (duplicate) and 1 (not duplicate). We use *number of questions* as our metric which is widely used in ER literature. This value describes the number of questions to be asked before all the entities are resolved. We use 3 random smaller instances of size 20 *Cora* dataset to evaluate our algorithm.

Scalability Experiments: We use the large scale synthetic dataset for the scalability experiments. We vary the following 4 parameters: (i) number of objects n. (ii) number of buckets b' to approximate the pdfs. (iii) number of unknown edges $|D_u|$. (iv) worker correctness p.When one of these aforementioned parameters is varied, the other three are kept constant. The default values for these 4 parameters are, $n = 100$, $|D_u| = 40\%$ of all edges, $b' = 4$, $p = 0.8$. Please note that we primarily present the scalability results for `Tri-Exp` and `BL-Random`, as `LS-MaxEnt-CG` and `MaxEnt-IPS` takes 1.5 days to converge even when $n = 6$.



(a) Worker feedback aggregation real data

(b) Error vs p synthetic Data



(c) Error vs p real Data

Figure 6.4: **Quality Experiments:** i)Worker Feedback Aggregation ii) Unknown Edge Estimation

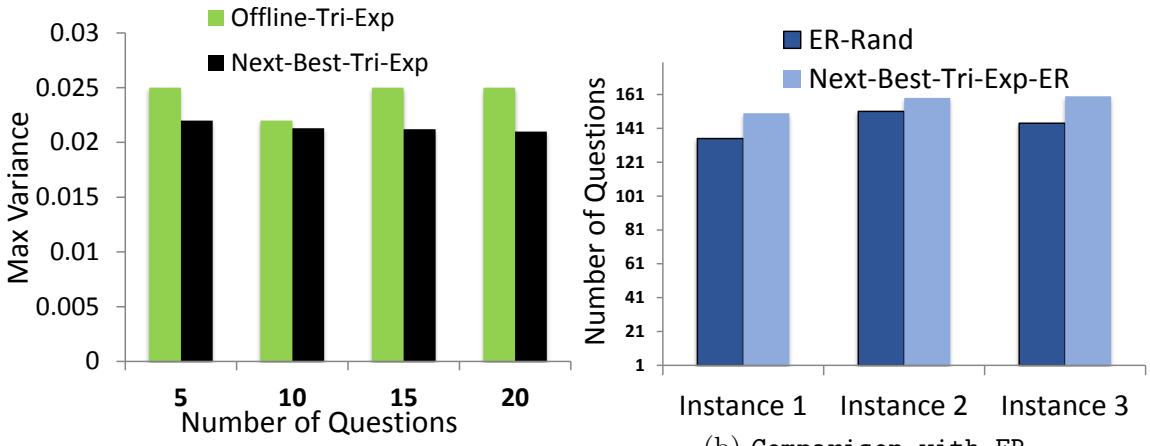
6.6.4 Results

6.6.4.1 Summary of Results

Quality Experiments: Our first experiment on aggregating feedback suggests the superiority of `Conv-Inp-Aggr` over `BL-Inp-Aggr`. For unknown edge estimation, the results indicate that both `Tri-Exp` and `LS-MaxEnt-CG` perform better than the baseline `BL-Random`. For both of them, we observe that with higher worker accuracy (correctness) p, the error increases for all these competing algorithms. While this may appear counter-intuitive, our post-analysis indicates that this is due to the probabilistic nature of our proposed framework and the algorithms, which are most effective, when the workers responses are truly probabilistic. For the third problem, with more questions asked, the `AggrVar` reduces. In both of these aforementioned scenarios, `Next-Best-Tri-Exp` convincingly outperforms `Next-Best-BL-Random`.

Application to ER: Our result demonstrates that `Rand-ER` outperforms `Next-Best-Tri-Exp-ER`. This is expected since our method is designed to solve a more general problem than ER methods - the ER method assumes no worker uncertainty (i.e., workers are always 100% accurate), and it is dependent on the notion of transitive closure, which is a very special case of triangle inequality.

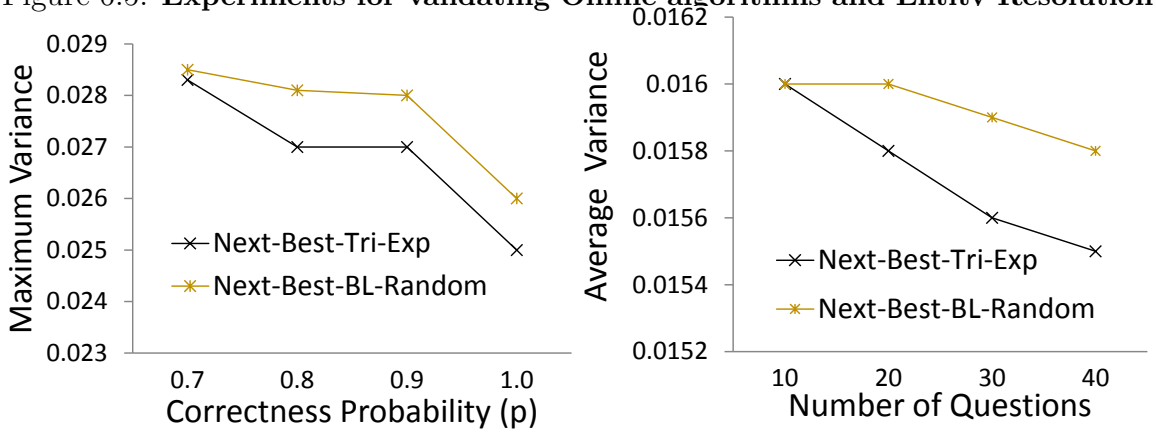
Scalability Experiments: We show that `Tri-Exp` performs reasonably well with the increasing number of objects, buckets, known edges, or worker correctness. The computation time of `BL-Random` is similar to that of `Tri-Exp`, while `Tri-Exp` is qualitatively superior. Therefore, we only present the results of `Tri-Exp` in these experiments. The algorithms that rely on computing joint distribution `LS-MaxEnt-CG`, `MaxEnt-IPS` do not converge beyond a very small number of objects ($n = 5$) even in days.



(a) `Online vs Offline`

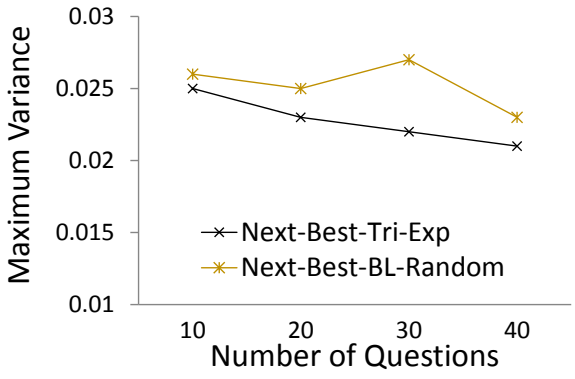
(b) `Comparison with ER`

Figure 6.5: **Experiments for validating Offline algorithms and Entity Resolution**



(a) Maximum variance with correctness

(b) Average variance with number of questions



(c) Maximum variance with number of questions

Figure 6.6: **Quality Experiments**: Asking the Next Best Question

6.6.4.2 Quality Experiments

(i) *Worker feedback aggregation*: Figure 6.4a shows that `Conv-Inp-Aggr` consistently outperforms the baseline.

(ii) *Estimating Unknown Edges:* We present the results for estimating unknown edges in Figure 6.4b and 6.4c. For the synthetic data, `LS-MaxEnt-CG` is superior to the other two methods, while `Tri-Exp` outperforms `BL-Random`. The pattern remains the same for the real data as both `LS-MaxEnt-CG` and `MaxEnt-IPS` exhibit superiority over `BL-Random`. `Tri-Exp` peforms reasonably well for real data. The fact that `LS-MaxEnt-CG` is the best performing algorithm for the real data demonstrates that, in reality, workers may indeed provide inconsistent feedback that do not obey triangle inequality, hence our proposed optimization model is appropriate to capture that settings.

(iii)*Asking the Next Best Question:* We first compare our online algorithms `Next-Best-Tri-Exp` and `Next-Best-BL-Random`.

(a) Varying p: We vary p and present `AggrVar` considering maximum variance. Figure 6.6a presents the results for this experiment. While the maximum variance for `Next-Best-BL-Random` and `Next-Best-Tri-Exp` decreases with increasing worker accuracy, latter performs better than the former. For average variance, we encounter the same pattern. Hence, we omit the results for brevity.

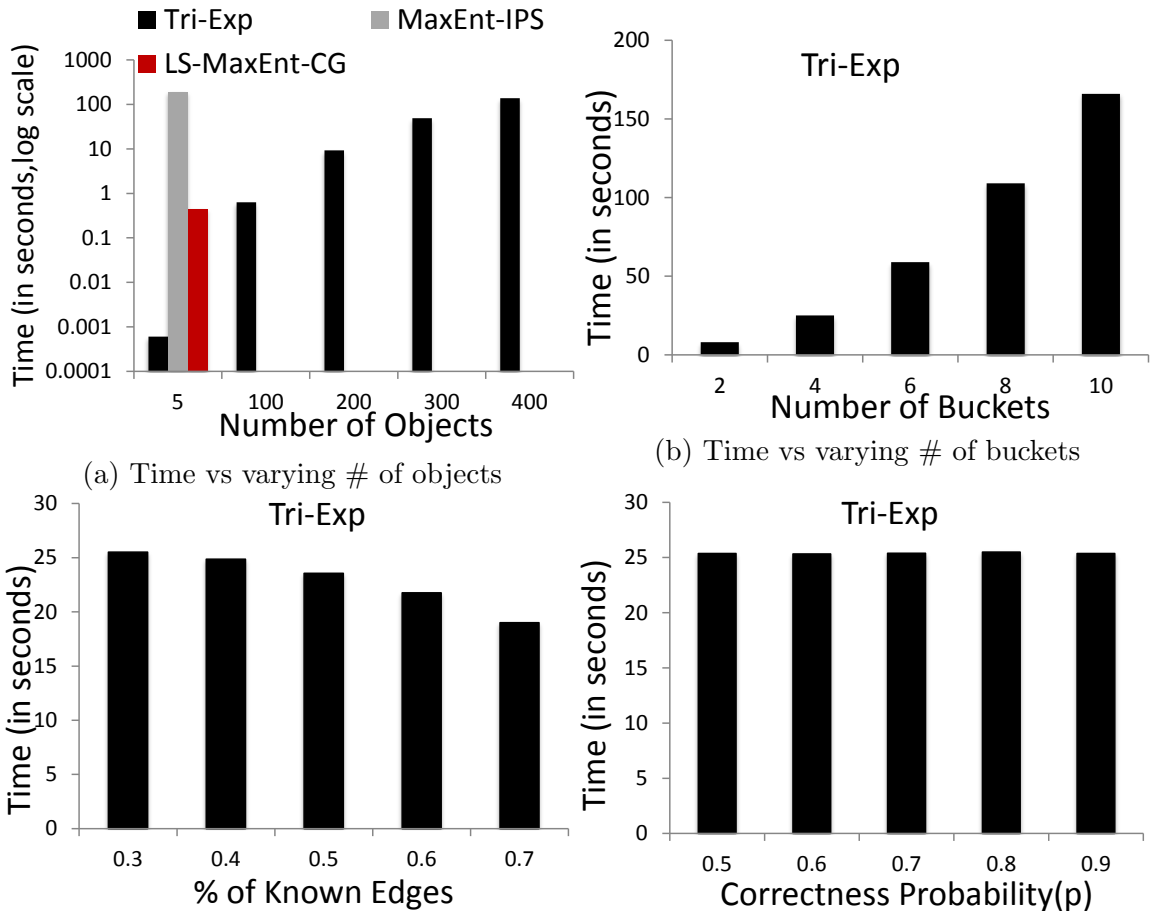
(b) Varying B: Our goal here is to test how `AggrVar` reduces with the increasing number of questions (budget B). Figure 6.6b and Figure 6.6c present the outcome of these experiments. It is interesting to observe that with a fairly small number of questions, the `AggrVar` reduces drastically and the system reaches a stable state.

(c) Online vs Offline Experiment: Figure 6.5a presents the result. As expected, `Next-Best-Tri-Exp` performs better than the `Offline-Tri-Exp`, but with very small

margin. This result proves that `Offline-Tri-Exp` is very suitable for traditional crowdsourcing framework as online algorithms have high latency.

iv) *Entity Resolution:* Figure 6.5b shows the results for Entity Resolution. Although `Next-Best-Tri-Exp-ER` performs a little worse than `Rand-ER`, we argue that our method is not optimized for finding duplicate entities. Please notice that our method can be applied to find duplicate entities while it is not possible vice versa.

6.6.4.3 Scalability Experiments



(a) Time vs varying # of objects

(b) Time vs varying # of buckets

(c) Time vs varying # of unknown edges

(d) Time vs varying # of unknown edges

Figure 6.7: **Scalability Experiments**: 4 different parameters are varied. Our default settings is $n = 100$, $p = 0.8$, $|D_u| = 50\%$, $b' = 4$.

(i) *Worker feedback aggregation:* We observe that the time to aggregate workers feedback is akin to the triangle computation time of `Tri-Exp`. For brevity, we omit the details.

(ii) *Unknown Edge Estimation:* We observe that both heuristic algorithms are equally efficient. Hence, we just present the results of `Tri-Exp`. (a) Varying n: Figure 6.7a presents these results and indicates that `Tri-Exp` converges in a reasonable time, even for higher values of n.

(b) Varying b': Figure 6.7b presents these results and indicates that `Tri-Exp` scales well with increasing b'.

(c) Varying $|D_k|$: Figure 6.7c presents these results and shows that `Tri-Exp` is scalable with increasing number of unknown edges and takes lesser time, as $|D_k|$ increases.

(d) Varying p: Figure 6.7d indicates that the running time of `Tri-Exp` is not affected by p.

(iii)*Asking the Next Best Question:* The running time of `Next-Best-Tri-Exp` and `Next-Best-BL-Random` are similar and dominated by the size of $|D_u|$. These results are similar to that of Figure 6.7c and omitted for brevity.

6.7 Related Work

User Input Aggregation: Aggregation of opinions is studied in several prior works in AI [112, 113, 114]. An opinion is described as a pdf over a set of categorical values. Since, their methods do not consider the notion of distance, they do not offer an easy extension to our problem. Aggregation of binary feedback(Yes/No) in crowdsourcing is studied in [110, 111]. Their proposed models estimate both worker accuracy and the true answer considering a bipartite graph of workers and tasks. They do not extend beyond binary feedback while we assume a numeric feedback model. [128] study how to find the ranking of a tuple, where tuple scores are given by probability

distributions. While this problem is fundamentally different from our first problem, their proposed approach nevertheless justifies our proposed way of convolving multiple pdfs for aggregation.

Distance Estimation: *Distance estimation using crowdsourcing* has gained a significant interest recently for solving a variety of computational problems that require distance estimation, such as top-k, clustering, entity resolution (ER), etc [6, 7, 8]. In most of these works, the dependency on distances is only indirect, as these works are based on asking users to resolve Boolean similarity or ranking questions, e.g., whether two objects are similar or not, or whether one object should be ranked higher than the other. In contrast, our work is the first to directly solicit, from the crowd, the broader notion of numeric distances between objects. In [6], the authors propose a crowdsourced clustering method by leveraging matrix completion techniques, where human workers are involved to annotate objects in a deterministic settings. Entity resolution using crowdsourcing have been studied in [107, 7, 109]. The closest related work is that of [109]. The main differences between this work and ours are: (a) the are only concerned with the Boolean notion of objects equivalency, whereas we try to learn numeric distances between objects, (b) they assume that the crowd can make no mistake, which is unrealistic for distance computations, and (c) they leverage the notion of transitive closure, which is a much simpler notion compared to that of triangle inequality. Therefore their main focus has been on determining the optimal set of questions to ask the crowd, whereas our focus has been on even more basic issues such as how to aggregate uncertain user feedbacks and update the probabilistic distribution models of the distances.

Asking Next Best Question: Our third problem formulation borrows motivation from [129, 7, 130]. [129] describes the problem of finding the maximum

item from paiwise comparisons, [7] tackles entity resolution, and [130] studies top-k queries in uncertain database. They all designed algorithms for finding the next best question which maximize the expected accuracy for their respective problems. Both [129] and [7] prove that finding next best question is NP-Complete. In [130], authors construct a Tree of Possible Ordering(TPO) in order to find the next best question. Although we employ the similar settings, our unique problem formulation requires us to design novel solutions.

6.8 Conclusion

We present a probabilistic distance estimation framework in crowdsourcing platforms that has wide applicability in different domains. One of the novel contributions of the work is to consider worker feedback with probabilistic interpretation and describe the overall framework with three key components.The effectiveness of our proposed solutions are validated empirically using both real and synthetic data.

REFERENCES

[1] P. G. Ipeirotis *et al.*, "Quality management on amazon mechanical turk," in *KDD workshop*, 2010.

[2] M. Kokkodis *et al.*, "Have you done anything like that?: Predicting performance using inter-category reputation," in *WSDM*, 2013.

[3] M. Joglekar *et al.*, "Evaluating the crowd with confidence," in *SIGKDD*, 2013.

[4] D. E. Difallah, G. Demartini, and P. Cudré-Mauroux, "Pick-a-crowd: tell me what you like, and i'll tell you what to do," in *Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2013, pp. 367–374.

[5] V. Ambati, S. Vogel, and J. G. Carbonell, "Towards task recommendation in micro-task markets." in *Human computation*. Citeseer, 2011, pp. 1–4.

[6] J. Yi *et al.*, "Semi-crowdsourced clustering: Generalizing crowd labeling by robust distance metric learning," in *NIPS*, 2012, pp. 1772–1780.

[7] S. E. Whang *et al.*, "Question selection for crowd entity resolution," *PVLDB*, 2013.

[8] V. Polychronopoulos *et al.*, "Human-powered top-k lists," in *WebDB*, 2013.

[9] A. Kittur, J. V. Nickerson, M. Bernstein, E. Gerber, A. Shaw, J. Zimmerman, M. Lease, and J. Horton, "The future of crowd work," in *CSCW '13*, 2013.

[10] A. Morishima *et al.*, "Crowd4u: An initiative for constructing an open academic crowdsourcing network," in *HCOMP*, 2014.

[11] S. Guo, A. G. Parameswaran, and H. Garcia-Molina, "So who won?: dynamic max discovery with the crowd," in *SIGMOD Conference*, 2012, pp. 385–396.

[12] S. B. Davidson *et al.*, "Using the crowd for top-k and group-by queries," in *TODS*, 2013.

[13] C. J. Zhang, L. Chen, Y. Tong, and Z. Liu, "Cleaning uncertain data with a noisy crowd," in *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*. IEEE, 2015, pp. 6–17.

[14] D. E. Difallah, M. Catasta, G. Demartini, P. G. Ipeirotis, and P. Cudré-Mauroux, "The dynamics of micro-task crowdsourcing: The case of amazon mturk," in *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015, pp. 238–247.

[15] J. Hffmeier and G. Hertel, "When the whole is more than the sum of its parts: Group motivation gains in the wild," *Journal of Experimental Social Psychology*, 2011.

[16] G. Hertel, "Synergetic effects in working teams," *Journal of Managerial Psychology*, 2011.

[17] H. P. Andres, "Team cognition using collaborative technology: a behavioral analysis," *Journal of Managerial Psychology*, 2013.

[18] R. Yan, M. Gao, E. Pavlick, and C. Callison-Burch, "Are two heads better than one? crowdsourced translation via a two-step collaboration of non-professional translators and editors."

[19] A. Kittur, B. Smus, S. Khamkar, and R. E. Kraut, "Crowdforge: Crowdsourcing complex work," in *UIST*, 2011.

[20] R. Kenna and B. Berche, "Managing research quality: critical mass and optimal academic research group size," *IMA Journal of Management Mathematics*.

[21] G. Marwell, P. E. Oliver, and R. Prahl, "Social networks and collective action: A theory of the critical mass," *American Journal of Sociology*, 1988.

[22] D. Katz and R. L. Kahn, "The social psychology of organizations," 1978.

[23] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi, "Online team formation in social networks," ser. WWW '12, 2012.

[24] T. Lappas, K. Liu, and E. Terzi, "Finding a team of experts in social networks," ser. KDD '09.

[25] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, 1979.

[26] D. L. Chen and W. B. Dolan, "Building a persistent workforce on mechanical turk for multilingual data collection," in *HCOMP*, 2011.

[27] J. S. Downs, M. B. Holbrook, S. Sheng, and L. F. Cranor, "Are your participants gaming the system?: screening mechanical turk workers," ser. CHI '10.

[28] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decis. Support Syst.*, vol. 43, no. 2, pp. 618–644, Mar. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.dss.2005.05.019

[29] T. Lappas, K. Liu, and E. Terzi, "Finding a team of experts in social networks," in *SIGKDD*, 2009, pp. 467–476.

[30] I. B. Myers and M. H. McCaulley, *Myers-Briggs Type Indicator: MBTI.* Consulting Psychologists Press, 1988.

[31] J. Surowiecki, "The wisdom of crowds: Why the many are smarter than the few and how collective wisdom shapes business," *Economies, Societies and Nations*, 2004.

[32] A. Gajewar and A. D. Sarma, "Multi-skill collaborative teams based on densest subgraphs." in *SDM*. SIAM, 2012, pp. 165–176.

[33] S. K. et. al., "Compact location problems," *Th. Comp. Sci*, 1996.

[34] A. Schrijver, *Theory of Linear and Integer Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1986.

[35] S. S. R. et. al, "Facility dispersion problems: Heuristics and special cases," in *WADS*, 1991.

[36] D. J. Rosenkrantz, G. K. Tayi, and S. S. Ravi, "Facility dispersion problems under capacity and cost constraints," *J. Comb. Optim.*, 2000.

[37] S. van Dongen and A. J. Enright, "Metric distances derived from cosine similarity and pearson and spearman correlations," *CoRR*, vol. abs/1208.3145, 2012.

[38] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi, "Power in unity: Forming teams in large-scale community systems," ser. CIKM '10, 2010.

[39] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Operations research*, vol. 14, no. 4, pp. 699–719, 1966.

[40] M. Karpinski, "Approximability of the minimum bisection problem: an algorithmic challenge," in *Mathematical Foundations of Computer Science 2002*.

[41] N. Guttmann-Beck and R. Hassin, "Approximation algorithms for minimum k-cut," *Algorithmica*, 2000.

[42] M. Grotschel and L. Lovász, "Combinatorial optimization," *Handbook of combinatorics*, vol. 2, pp. 1541–1597, 1995.

[43] S. B. Roy, I. Lykourentzou, S. Thirumuruganathan, S. Amer-Yahia, and G. Das, "Optimization in knowledge-intensive crowdsourcing," *CoRR*, vol. abs/1401.1302, 2014.

[44] D. Klakow and J. Peters, "Testing the correlation of word error rate and perplexity," *Speech Commun.*, vol. 38, no. 1, pp. 19–28, Sept. 2002. [Online]. Available: http://dx.doi.org/10.1016/S0167-6393(01)00041-3

[45] K. Chai, V. Potdar, and T. Dillon, "Content quality assessment related frameworks for social media," in *ICCSA 2009*.

[46] A. Kittur and R. E. Kraut, "Harnessing the wisdom of crowds in wikipedia: quality through coordination," in *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, ser. CSCW '08. New York, NY, USA: ACM, 2008, pp. 37–46. [Online]. Available: http://doi.acm.org/10.1145/1460563.1460572

[47] S. Amer-Yahia and S. Basu Roy, "From complex object exploration to complex crowdsourcing." in *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015, pp. 1531–1532.

[48] H. Kaplan, I. Lotosh, T. Milo, and S. Novgorodov, "Answering planning queries with the crowd," *PVLDB*, vol. 6, no. 9, pp. 697–708, 2013.

[49] M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich, "Soylent: a word processor with a crowd inside," in *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. ACM, 2010, pp. 313–322.

[50] W. S. Lasecki, K. I. Murray, S. White, R. C. Miller, and J. P. Bigham, "Real-time crowd control of existing interfaces," in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, ser.

UIST '11. New York, NY, USA: ACM, 2011, pp. 23–32. [Online]. Available: http://doi.acm.org/10.1145/2047196.2047200

[51] G. Little, L. B. Chilton, M. Goldman, and R. C. Miller, "Turkit: human computation algorithms on mechanical turk," in *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. ACM, 2010, pp. 57–66.

[52] S. Ahmad, A. Battle, Z. Malkani, and S. Kamvar, "The jabberwocky programming environment for structured social computing," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 2011, pp. 53–64.

[53] A. Kulkarni, M. Can, and B. Hartmann, "Collaboratively crowdsourcing workflows with turkomatic," in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*. ACM, 2012, pp. 1003–1012.

[54] A. T. M. B. Daniela Retelny, Sbastien Robaszkiewicz, "Expert crowdsourcing with flash teams," in *CrowdConf 2013 poster*.

[55] A. Majumder, S. Datta, and K. Naidu, "Capacitated team formation problem on social networks," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '12. New York, NY, USA: ACM, 2012, pp. 1005–1013. [Online]. Available: http://doi.acm.org/10.1145/2339530.2339690

[56] M. Kargar and A. An, "Discovering top-k teams of experts with/without a leader in social networks," ser. CIKM '11, 2011.

[57] M. Kargar, A. An, and M. Zihayat, "Efficient bi-objective team formation in social networks," in *Machine Learning and Knowledge Discovery in Databases*, ser. Lecture Notes in Computer Science, P. Flach, T. Bie, and N. Cristianini,

Eds. Springer Berlin Heidelberg, 2012, vol. 7524, pp. 483–498. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33486-3_31

[58] M. Kargar, M. Zihayat, and A. An, "Finding affordable and collaborative teams from a network of experts."

[59] M. Chhabra, S. Das, and B. Szymanski, "Team formation in social networks," in *Computer and Information Sciences III*. Springer, 2013, pp. 291–299.

[60] S. B. Roy *et al.*, "Task assignment optimization in knowledge-intensive crowdsourcing," *VLDB J.*, 2015.

[61] H. Rahman *et al.*, "Task assignment optimization in collaborative crowdsourcing," in *ICDM*, 2015.

[62] H. Park *et al.*, "Deco: A system for declarative crowdsourcing," *Proceedings of the VLDB Endowment*, 2012.

[63] A. Feng *et al.*, "Crowddb: Query processing with the vldb crowd," 2011.

[64] R. M. Borromeo *et al.*, "Automatic vs. crowdsourced sentiment analysis," in *IDEA*, 2015.

[65] N. Gibson and J. Talburt, "Hive: crowdsourcing education data," *Journal of Computing Sciences in Colleges*, vol. 25, no. 5, pp. 72–78, 2010.

[66] D. Haas *et al.*, "Argonaut: macrotask crowdsourcing for complex data processing," 2015.

[67] A. Morishima *et al.*, "Cylog/game aspect: An approach to separation of concerns in crowdsourced data management," *Information Systems*, 2016, (to appear).

[68] H. Rahman *et al.*, "Worker skill estimation in team-based tasks," *PVLDB*, 2015.

[69] T. Lappas *et al.*, "Finding a team of experts in social networks," in *SIGKDD*, 2009.

[70] A. Anagnostopoulos *et al.*, "Power in unity: forming teams in large-scale community systems," in *CIKM*, 2010.

[71] A. Anagnostopoulos *et al.*, "Online team formation in social networks," in *WWW*, 2012.

[72] A. Majumder *et al.*, "Capacitated team formation problem on social networks," in *SIGKDD*, 2012.

[73] S. Leonardi, http://research.microsoft.com/en-us/events/bda2013/team-formation.pdf, 2013.

[74] P. Butkovic, *Max-linear systems: theory and algorithms*. Springer, 2010.

[75] M. H. Andersen, "Max-plus algebra: Properties and applications," 2011.

[76] B. Arai *et al.*, "Anytime measures for top-k algorithms on exact and fuzzy data sets," *The VLDB Journal*, 2009.

[77] R. Reagans and E. W. Zuckerman, "Networks, diversity, and productivity: The social capital of corporate r&d teams," *Organization science*, vol. 12, no. 4, pp. 502–517, 2001.

[78] S. A. Haslam, *Psychology in organizations*. Sage, 2004.

[79] H. Buhrman *et al.*, "One-sided versus two-sided error in probabilistic computation," in *STACS 99*, 1999.

[80] A. Björck, *Numerical methods for least squares problems*. Siam, 1996.

[81] E. T. Jaynes, "On the rationale of maximum-entropy methods," *Proceedings of the IEEE*, vol. 70, no. 9, pp. 939–952, 1982.

[82] R. I. Jennrich, "Asymptotic properties of non-linear least squares estimators," *The Annals of Math. Statistics*, 1969.

[83] P. B. Stark *et al.*, "Bounded-variable least-squares: an algorithm and applications," *Computational Statistics*, 1995.

[84] S.-H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," *City*, 2007.

[85] P. Butkovič, "One-sided max-linear systems and max-algebraic subspaces," in *Max-linear Systems: Theory and Algorithms*. Springer, 2010, pp. 53–70.

[86] V. Poosala *et al.*, "Selectivity estimation without the attribute value independence assumption," in *VLDB*, 1997.

[87] J. Han *et al.*, *Data Mining: Concepts and Techniques*, 2000.

[88] A. Zzkarian and A. Kusiak, "Forming teams: an analytical approach," *IIE transactions*, vol. 31, no. 1, pp. 85–97, 1999.

[89] A. Kittur *et al.*, "Crowdsourcing for usability: Using micro-task markets for rapid, remote, and low-cost user measurements," *CHI*, 2008.

[90] A. Kittur, "Crowdsourcing, collaboration and creativity." *ACM Crossroads*, 2010.

[91] S. B. Roy *et al.*, "Task assignment optimization in knowledge-intensive crowdsourcing," *VLDB Journal*, pp. 1–25, 2015.

[92] B. Debele *et al.*, "Accuracy evaluation of weather data generation and disaggregation methods at finer timescales," *Advances in Water Resources*, 2007.

[93] T. A. Garrett, "Aggregated versus disaggregated data in regression analysis: implications for inference," *Economics Letters*, 2003.

[94] D. Geiger and M. Schader, "Personalized task recommendation in crowdsourcing information systemscurrent state of the art," *Decision Support Systems*, vol. 65, pp. 3–16, 2014.

[95] M.-C. Yuen, I. King, and K.-S. Leung, "Taskrec: probabilistic matrix factorization in task recommendation in crowdsourcing systems," in *Neural Information Processing*. Springer, 2012, pp. 516–525.

[96] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on.* IEEE, 2008, pp. 263–272.

[97] C. H. Lin, E. Kamar, and E. Horvitz, "Signals in the silence: Models of implicit feedback in a recommendation system for crowdsourcing," 2014.

[98] Q. Wu, Y. Ying, and D.-X. Zhou, "Learning rates of least-square regularized regression," *Foundations of Computational Mathematics*, vol. 6, no. 2, pp. 171–192, 2006.

[99] C.-H. Lee, Y.-H. Kim, and P.-K. Rhee, "Web personalization expert with combining collaborative filtering and association rule mining technique," *Expert Systems with Applications*, vol. 21, no. 3, pp. 131–137, 2001.

[100] M. Kwak and D.-S. Cho, "Collaborative filtering with automatic rating for recommendation," in *Industrial Electronics, 2001. Proceedings. ISIE 2001. IEEE International Symposium on*, vol. 1. IEEE, 2001, pp. 625–628.

[101] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence*, vol. 2009, p. 4, 2009.

[102] M. Gjoka and F. Soldo, "Exploring collaborative filters: Neighborhood-based approach," Tech. Rep.

[103] P. Forbes and M. Zhu, "Content-boosted matrix factorization for recommender systems: experiments with recipe recommendation," in *Proceedings of the fifth ACM conference on Recommender systems.* ACM, 2011, pp. 261–264.

[104] J. Nguyen and M. Zhu, "Content-boosted matrix factorization techniques for recommender systems," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 6, no. 4, pp. 286–301, 2013.

[105] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 426–434.

[106] F. W. Lawvere, "Metric spaces, generalized logic, and closed categories," *Rendiconti del seminario matématico e fisico di Milano*, 1973.

[107] J. Wang *et al.*, "Crowder: Crowdsourcing entity resolution," *PVLDB*, 2012.

[108] C. Chai, G. Li, J. Li, D. Deng, and J. Feng, "Cost-effective crowdsourced entity resolution: A partial-order approach," in *Proceedings of the 2016 International Conference on Management of Data*, ser. SIGMOD '16. New York, NY, USA: ACM, 2016, pp. 969–984. [Online]. Available: http://doi.acm.org/10.1145/2882903.2915252

[109] N. Vesdapunt *et al.*, "Crowdsourcing algorithms for entity resolution," *PVLDB*, 2014.

[110] N. Dalvi *et al.*, "Aggregating crowdsourced binary ratings," in *WWW*, 2013, pp. 285–294.

[111] A. Ghosh *et al.*, "Who moderates the moderators?: crowdsourcing abuse detection in user-generated content," in *EC*, 2011.

[112] A. Garg, T. Jayram, S. Vaithyanathan, and H. Zhu, "Generalized opinion pooling." in *AMAI*, 2004.

[113] F. Dietrich and C. List, "Probabilistic opinion pooling," 2014.

[114] A. Carvalho and K. Larson, "A consensual linear opinion pool," *arXiv preprint arXiv:1204.5399*, 2012.

[115] Å. Björck, "Numerical methods for least squares problems," *Pressure Rate Deconvolution Methods for Well Test Analysis*, 1996.

[116] C. Xie *et al.*, "A maximum entropy-least squares estimator for elastic origin–destination trip matrix estimation," *Transportation Research Part B: Methodological*, 2011.

[117] R. Fletcher, *Practical methods of optimization*. John Wiley & Sons, 2013.

[118] S. Sarawagi, "User-adaptive exploration of multidimensional data." in *VLDB*, 2000.

[119] H. Mannila *et al.*, "Prediction with local patterns using cross-entropy," in *KDD*, 1999.

[120] R. Fagin and L. Stockmeyer, "Relaxing the triangle inequality in pattern matching," *International Journal of Computer Vision*, vol. 30, pp. 219–231, 1998.

[121] D. Gerardi *et al.*, "Aggregation of expert opinions," *Games and Economic Behavior*, 2009.

[122] Y. Ioannidis, "The history of histograms (abridged)," in *VLDB*, 2003.

[123] G. H. Golub *et al.*, "An analysis of the total least squares problem," *SIAM Journal on Numerical Analysis*, 1980.

[124] B. Arai *et al.*, "Anytime measures for top-k algorithms," in *PVLDB*, 2007.

[125] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[126] R. Fletcher *et al.*, "Function minimization by conjugate gradients," *The computer journal*, 1964.

[127] C.-W. Ko *et al.*, "An exact algorithm for maximum entropy sampling," *Operations Research*, 1995.

[128] J. Li and A. Deshpande, "Ranking continuous probabilistic datasets," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 638–649, 2010.

[129] S. Guo *et al.*, "So who won?: dynamic max discovery with the crowd," in *SIGMOD*, 2012.

[130] E. Ciceri, P. Fraternali, D. Martinenghi, and M. Tagliasacchi, "Crowdsourcing for top-k query processing over uncertain data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 28, no. 1, pp. 41–53, 2016.

BIOGRAPHICAL STATEMENT

Habibur Rahman was born in the district of Munshigonj, Bangladesh. He received his B.Sc in Computer Science and Engineering in 2009 from Bangladesh University of Engineering and Technology, Dhaka. His research interests include Crowdsourcing, Social Analytics and Recommender Systems. He has interned in Microsoft Research (Redmond), Qatar Computing Research Institute and WalmartLabs. He has published in several premier conferences such as VLDB, ICDE, CIKM, ICDM etc.