

Optimized Group Formation for Solving Collaborative Tasks

Habibur Rahman · Senjuti Basu Roy · Saravanan Thirumuruganathan ·
Sihem Amer-Yahia · Gautam Das

Received: date / Accepted: date

Abstract 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 ap-

proximation 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.

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 [12]. 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.¹ The synergistic effect of collaboration in group based activities is widely accepted in socio-psychological research and traditional team based activities [22, 21, 5]. A number of popular applications such as collaborative document editing, sentence translation, or citizen science could be modelled as collaborative crowdsourcing tasks. Despite its immense potential, the transformative effect of “collaboration” remains largely unexplored in crowdsourcing [32].

Group Formation for Solving Collaborative Tasks: The optimization goals for task assignment is

¹ This work is the extension of our paper [46]. We extend our previous work by providing i) an additional technique for task assignment referred to as **Cons-cost-K-ApprxGrp**, ii) detail proofs of our algorithms and iii) additional experiments on both real and synthetic data

H. Rahman
UT Arlington
E-mail: habibur.rahman@mavs.uta.edu

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

S. Thirumuruganathan
QCRI, HBKU
E-mail: sthirumuruganathan@qf.org.qa

S. Amer-Yahia
CNRS, LIG
E-mail: sihem.amer-yahia@imag.fr

G. Das
UT Arlington
E-mail: gdas@uta.edu

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 [22, 21], we have identified two key factors for group characteristics that entails successful collaboration. The first factor *worker-worker affinity* [53, 33] 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 [34]. Social theories widely underscore the importance of *upper critical mass* [30] for group collaboration, which is a constraint on the size of groups beyond which the collaboration effectiveness diminishes [30, 43].

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 have 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 these 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 [29] in collaborative crowdsourcing as a graph with nodes representing workers and edges labeled with pair-wise 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 implies 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 [4, 37] or collaborative crowdsourcing [32, 53, 33] 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 staged 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 [17], 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 4, we formally define our problem which incorporates a variety of human factors.
2. *Solutions*: We propose 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 paper is organized as follows. Sections 2, 3, and 4 discuss a database application of collaborative crowdsourcing, our data model, problem formalization, and initial solutions. Sections 5 and 6 describe our theoretical analyses and proposed algorithmic solutions. Experiments are described in 7, related work in Section 8, and conclusion are presented in Section 9.

2 An Application of Collaborative Task

Sentence translation [9, 53, 33] 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 paper, consider a translation task t designed for translating an English video clip to French. Typically, such translation tasks follow a 3-step process [53, 33]: 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 [53]), 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 [4]. The notion of affinity has been identified in the related work on sentence translation tasks [53, 33], as well as team formation problems [4].

	u_1	u_2	u_3	u_4	u_5	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 1: Workers skill and wage table

	u_1	u_2	u_3	u_4	u_5	u_6
u_1	0.0	1.0	0.66	0.66	0.85	0.66
u_2	1.0	0.0	0.66	0.85	0.66	0.85
u_3	0.66	0.66	0.0	0.4	0.66	0.40
u_4	0.66	0.85	0.4	0.0	0.4	0.0
u_5	0.85	0.66	0.66	0.4	0.0	0.4
u_6	0.66	0.85	0.4	0.0	0.4	0.0

Table 2: Workers Distance Matrix

Q_1	Q_2	Q_3	C	K
1.8	1.4	1.66	3.0	3

Table 3: Task Description

However, if the group is “too large”, the effectiveness of collective actions diminishes [30, 43] while undertaking the translation task, as an unwieldy group of workers fail to find effective assistance from their peers [53, 33]. Therefore, each task 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, \dots, 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 1 and 2, and the task requirements of t (including thresholds on aggregated skill for each domain, total cost, and critical mass) are presented in Table 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, \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 co-

hesive. Incorporation of upper critical mass makes our problem significantly different from the body of prior works [4], as we may have to create a group further decomposed into multiple subgroups, instead of a single group.

3 Data Model

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

3.1 Preliminaries

Domains: We are given a set of domains $D = \{d_1, d_2, \dots, d_m\}$ denoting knowledge topics. Using the running example in Section 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 $\mathcal{U} = \{u_1, u_2, \dots, u_n\}$ of n workers available in the crowdsourcing platform. The example in Section 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 [13, 23]. It is also possible to use works such as [47] to learn the skill of workers for team based tasks.

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 split- ted in subgroups (each satisfying the critical mass constraint) such that the workers across all the sub- groups satisfy the skill and cost threshold.

Specifically, t is characterized by a vector, $\langle Q_1, Q_2, \dots, Q_m, C, K \rangle$, of length $m+2$. For the example in Section 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 3. A task is considered complete if it attains its skill requirement over all domains and satisfies all the constraints.

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 [4, 37] and psychology research [30, 43].

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

3.2.2 Group-based Human Factors: Affinities

Although related work in collaborative crowdsourcing acknowledges the importance of workers' affinity to enable effective collaboration [53, 33], 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 [38, 4], 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 , $\text{aff}(u_i, u_j)$, can be calculated by capturing the *similarity* between workers using simple socio-demographic attributes, such as region, age, gender, as done in previous work [53], as well as more complex psychological characteristics [44]. 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 $\text{dist}(u_i, u_j) = 1 - \text{aff}(u_i, u_j)$. Thus a smaller distance between workers ensures a better collaboration. Table 2 presents the pair-wise distance of all 6 workers for running example in Section 2. As will be clear later, the notion of distance rather 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} :

$$\text{DiaDist}(\mathcal{G}) = \text{Max}_{u_i, u_j \in \mathcal{G}} \text{dist}(u_i, u_j)$$

$$\text{SumDist}(\mathcal{G}) = \sum_{u_i, u_j \in \mathcal{G}} \text{dist}(u_i, u_j)$$

For both definitions, smaller value is better.

Inter-group affinity: When a group violates the upper critical mass constraint [30], 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:

$$\text{DiaInterDist}(G_1, G_2) = \text{Max}_{u_i \in G_1, u_j \in G_2} \text{dist}(u_i, u_j)$$

$$\text{SumInterDist}(G_1, G_2) = \sum_{u_i \in G_1, u_j \in G_2} \text{dist}(u_i, u_j)$$

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.

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 [38]. Then, we adapt the “flat-coordination” models of worker interactions, which is considered important in prior works in team formation [4] as the “coordination cost”, or in collaborative crowdsourcing [53] itself, as ‘the ‘turker-turker” affinity model. However, unlike previous work, we attempt to fully explore the potential of “group synergy” [51] 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 constraint². Existing literature (indicatively [51]) 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 [4, 38, 16]. 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” [30, 43] 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

² Notice that posing affinity as a constraint does not fully exploit the effect of “group synergy”.

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.

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, \dots, 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 [4,38]. Similarly, cost of the task is the additive wage of all the workers in \mathcal{G} .

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 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*:

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

– *SumDist*, *DiaInterDist*:

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

– *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:

$$\sum_{u_i \in \mathcal{G}} u_{d_i} \geq Q_i \quad \forall d_i$$

$$\sum_{u \in \mathcal{G}} w_u \leq C$$

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

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 4 could be easily extended to other combinations as well.

Theorem 1 *Problem AffAware-Crowd is NP-hard [17].*

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* [14] 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 any pair of nodes in g' is X' . This variant of the compact location problem is known as **Min-DIA** in [14].

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.

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) [50] 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-Partition, 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.

4.2.1 ILP for AffAware-Crowd

We discuss the ILP next as shown in Equation 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, \dots, 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.

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

subject to

$$\begin{aligned} \sum_{i=1}^n \sum_{j=1}^x u_{(i,G_j)} \times u_{d_l}^i &\geq Q_l & \forall l \in [1, m] \\ \sum_{i=1}^n \sum_{j=1}^x u_{(i,G_j)} \times w_u^i &\leq C \\ \sum_{i=1}^n u_{(i,G_j)} &\leq K & \forall j \in [1, x] \\ \sum_{j=1}^x u_{(i,G_j)} &\leq 1 & \forall i \in [1, n] \\ e_{i,i'} &= \begin{cases} 1 & \exists j \in [1, x] \& u_{(i,G_j)} = 1 \& u_{(i',G_j)} = 1 \\ 0 & \text{otherwise} \end{cases} \\ x &\in \{0, 1, \dots, n\} \\ u_{(i,G_j)} &\in \{0, 1\} & \forall i \in [1, n], \forall j \in [1, x] \end{aligned}$$

(1)

The objective function returns a group of subgroups that minimizes $DiaDist(\mathcal{G}) + \sum_{G_i, G_j} SumInterDist(G_i, G_j)$. 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, 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 $DiaDist(\mathcal{G}) + InterDist(G_1, G_2)$.

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 $\sum_{G_i, G_j} SumInterDist(G_i, G_j)$ 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) [15, 48]. 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 [52]. Furthermore, this assumption has been extensively used in distance computation in the related literature [3, 4]. Without metric property assumptions, the problems remain largely inapproximable [48].

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., $\sum_{u \in \mathcal{G}} u_{d_i} \geq Q_i, \forall d_i$, & $\sum_{u \in \mathcal{G}} w_u \leq C$.

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, we follow the similar strategy as above. We use an instance of **Min-DIA** [14] 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 $\langle Q_1 = n' \text{ and cost } C = 0 \rangle$. 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 4. Both **OptGrp** and **ApprxGrp** invoke a subroutine inside, referred to as **GrpCandidateSet**. We describe a general framework for this subroutine next.

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 1 has further details.

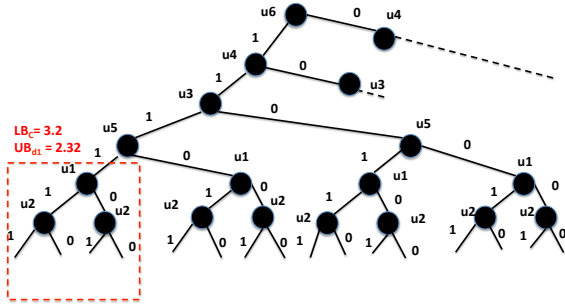


Fig. 1: A partially constructed tree of **GrpCandidateSet** using the example in Section 2. At node $u_1 = 1$, $LB_C = w_{u_6} + w_{u_4} + w_{u_3} + w_{u_5} + w_{u_1} = 3.2$ and $UB_{d_1} = u_{d_1}^6 + u_{d_1}^4 + u_{d_1}^3 + u_{d_1}^5 + u_{d_1}^1 + u_{d_1}^2 = 2.32$. The entire subtree is pruned, since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

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

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, 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 describes the idea further.

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

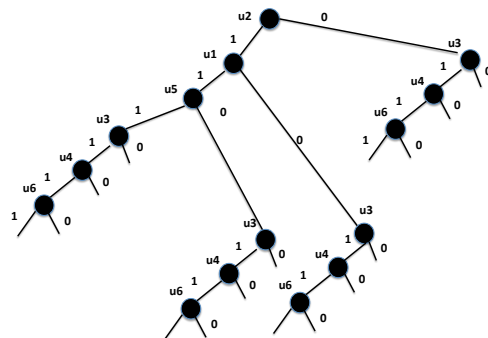


Fig. 2: Possible search space using the example in Section 2, after the cost of the workers are discretized into $k = 2$ buckets, considering only one skill d_1 . The tree is constructed in descending order of skill of the workers per bucket. For bucket 1, if the most skilled worker u_2 is not selected,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

5.3 Approximation Algorithm ApprxGrp

A popular variant of facility dispersion problem [15, 48] attempts to discover a set of nodes (that host the facilities) that are as far as possible, whereas, compact location problems [14] 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.

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*³ 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 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 is constructed accordingly, as described in Section 5.2. This variant of ApprxGrp is referred to as Cons-k-Cost-ApprxGrp.

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 inequality⁴. 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, \mathcal{L} is ordered as fol-

³ Star graph is a tree on v nodes with one node having degree $v - 1$ and other $v - 1$ nodes with degree 1.

⁴ Without triangle inequality assumption, no theoretical guarantee could be ensured [48].

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 , $\mathcal{U}' = \text{GrpDia}(\alpha, \{Q_i, \forall d_i\}, C)$
 - 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
-

lows: 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 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 GrpCandidateSetnext. 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, ApprxGrp returns workers u_1, u_2, u_3, u_5, u_6 as the results with objective function value upper bounded by $\leq 2 \times 0.66$.

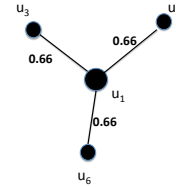


Fig. 3: An instantiation of GrpDia(0.66) using the example in Section 2. A star graph centered u_1 is formed.

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 3 that forms a star graph centered on u_1 with GrpDia(0.66) using the example in Section 2. Consider Figure 4 and notice that for a given distance value $= \alpha$, the complete graph induced by the star graph

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  $u, u_j$ ,  $\text{dist}(u, u_j) \leq \alpha$ . Let  $\mathcal{U}'$  be the set of workers in the star graph.
4:      $\mathcal{U}'' = \text{GrpCandidateSet}(\mathcal{U}', t)$ 
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  $\mathcal{U}'' = \emptyset$ 

```

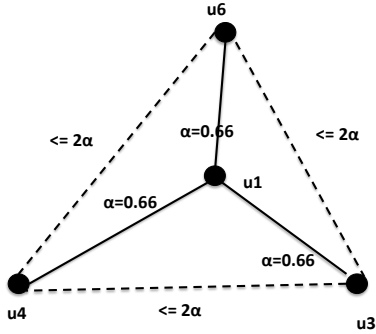


Fig. 4: An instantiation of $\text{GrpDia}(0.66)$ using the example in Section 2. The clique involving u_1, u_3, u_4, u_6 can not have an edge with distance $> 2 \times 0.66$, due to the triangle inequality property.

can not have any edge that is larger than $2 \times \alpha$, as long as the distance satisfies the triangle inequality property. Therefore, when $\text{GrpDia}(\alpha)$ 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 $\text{GrpDia}(\alpha')$ 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 $\text{DiaDist}(\mathcal{G})$ is at most 2-times worse than the optimal. Hence the approximation factor holds.

Lemma 1 $\text{Cons-k-Cost-ApprxGrp}$ 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 $\log_2 |\mathcal{L}|$ calls to GrpDia from the main function

($|\mathcal{L}| = O(n^2)$). Therefore, the asymptotic complexity of Cons-k-ApprxGrp is $O(n \times \log_2 |\mathcal{L}| \times (p+1)^{mk})$, which is polynomial.

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, 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 5.2 and is referred to as $\text{Cons-k-Cost-OptGrp}$.

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 [40] 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 1 and Figure 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 $\text{Cons-k-Cost-OptGrp}$ 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 $\text{Cons-k-Cost-OptGrp}$ is dominated by the computation time of the subroutine GrpCandidateSet . Therefore, Algorithm Cons-k-OptGrp runs in polynomial time of $O((p+1)^{mk})$.

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, \dots, G_x) such that $\forall_i |G_i| \leq K$, $\sum_i |G_i| = |\mathcal{G}|$ and the aggregated all pair inter group distance $\sum_{G_i, G_j \in \mathcal{G}} \text{SumInterDist}(G_i, G_j)$ 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** [28] 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 [28].

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, 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

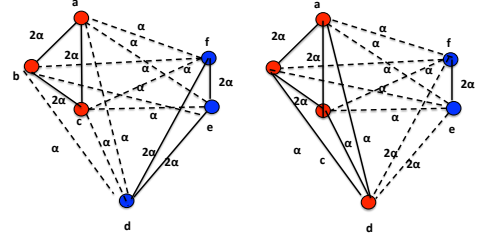


Fig. 5: Balanced Partitioning in **SpltB0pt** when the distance satisfies triangle inequality for a graph with 6 modes. The left hand side figure has two partitions ($\{a, b, c\}, \{d, e, f\}$) with 3-nodes in each (red nodes create one partition and blue nodes create another). The intra-partition edges 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.

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 **SpltB0pt** which is the *optimal balanced partitioning* of \mathcal{G} . For the running example in Section 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 **SpltB0pt**. 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, **SpltB0pt** gives rise to provable theoretical results for the **Splt** problem. Finding the approximation ratio of **SpltB0pt** for arbitrary number of partitions is deferred to future work.

Lemma 3 SpltB0pt 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, **SpltB0pt** 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 heavier 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 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 [20]. 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 [20].

Definition 3 Min-Star Problem: Given a group \mathcal{G} with n' workers, out of which each of x workers (u_1, u_2, \dots, 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, \dots, G_x such that $\sum_{i=1}^x k_i \text{dist}(u_i, \cup_{j \neq i} G_j) + \sum_{i < j} k_i k_j \text{dist}(u_i, u_j)$ 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 $\{u_1, u_2, \dots, u_x\}$, 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 [20]. The key insight behind this algorithm is the fact that for a fixed set of workers $\{u_1, u_2, \dots, u_x\}$, the second term of the objective function $\sum_{i < j} k_i k_j \text{dist}(u_i, u_j)$ is a constant. Furthermore, this expression could only take $\binom{n'}{x}$ distinct values corresponding to all possible combination of how the workers $\{u_1, u_2, \dots, u_x\}$ are chosen from the group \mathcal{G} with n' workers. Hence for a fixed set of workers,

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 $\{u_1, \dots, u_x\} \subset \mathcal{G}$ **do**
- 3: Find optimal subgroups $\{G_1, \dots, G_x\}$ for $\{u_1, \dots, u_x\}$ by formulating it as transportation problem
- 4: Evaluate objective function for $\{G_1, \dots, G_x\}$
- 5: **end for**
- 6: **return** subgroups $\{G_1, \dots, G_x\}$ with least objective function

the objective now reduces to finding an optimal subgroups G_1, \dots, G_x that minimizes the first expression. Interestingly, this expression corresponds exactly to a special case of the popular *transportation problem* [18] that could be solved optimally with time complexity $O(n')$ [20]. We refer to [20] for further details.

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 [20]. Previous work [20] 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 [20] 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 $O(n')$ [20] time. Since, x is a constant, therefore, the overall running time is polynomial.

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.

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 [44] 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 [53]. 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 4 against an alternative **Aff-Unaware** Algorithm [49]. 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 take-aways 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**.

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) [35].

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 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**.

Figure 6 and Figure 7 contain detailed workers profile distribution information.

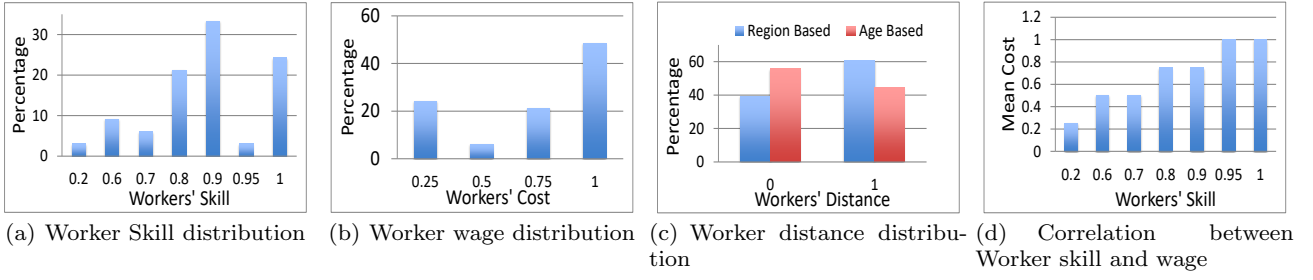


Fig. 6: Worker profile distributions for the Sentence Translation Tasks in Section 7.1

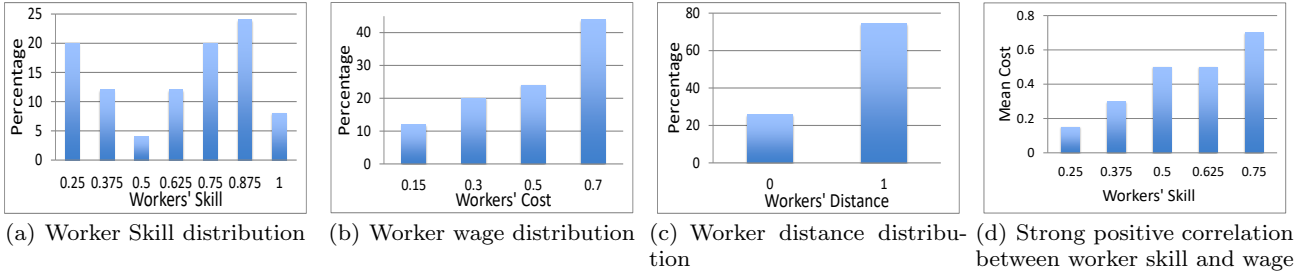


Fig. 7: Worker profile distributions for the Collaborative Document Writing in Section 7.1

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

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.

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 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** 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 4.

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, 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 [53] - 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 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 [8]. The results are summarized in Table 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.

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 $\mu = 5/\text{minute}$ 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 5.3.
2. **Cons-k-AG**: This is a variant of the algorithm **ApprxGrp** referred to as **Cons-k-cost-ApprxGrp**, described in Section 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 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

Task	Algorithm	Average Rating					
		Completeness	Grammar	Neutrality	Clarity	Timeliness	Added-value
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 5: **Stage 3 results of document writing application in Section 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 threshold and satisfy the wage limit, however, our proposed formalism **Optimal** enables better team collaboration, resulting in higher quality of articles.

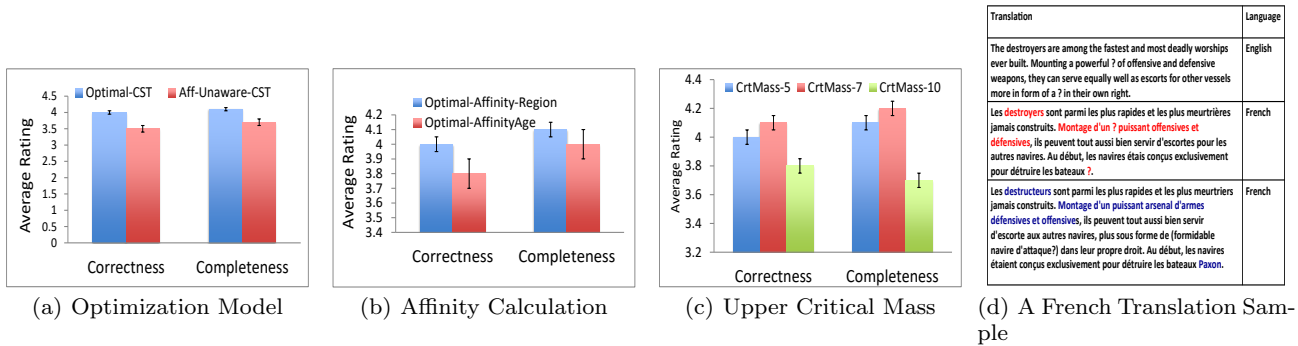


Fig. 8: **Stage 3 results of sentence translation:** Collected data with statistical significance (standard error) is presented. These results clearly corroborate that our affinity-aware optimization model **Optimal-CST** outperforms its affinity-unaware counterpart [43] with statistical significance across both quality dimensions. **Optimal-Affinity-Region** appears to outperform **Optimal-Affinity-Age** in “correctness”. The results of **CrtMass-Optimal-10** clearly appears to be less effective than the other two, showing some anecdotal evidence that group size is important in collaborative crowdsourcing applications.

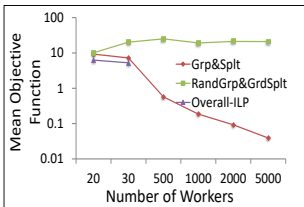


Fig. 9: **Grp&SplT** : Objective Function varying Number of Workers

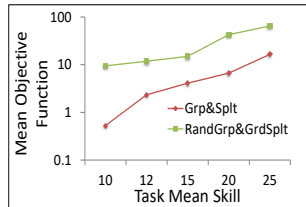


Fig. 10: **Grp&SplT** : Objective Function varying Task Mean Skill

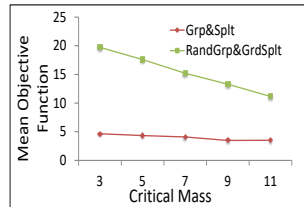


Fig. 11: **Grp&SplT**: Objective Function varying Critical Mass

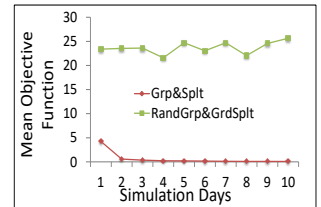


Fig. 12: **Grp&SplT**: Objective function over Simulation Days

real world settings. We first present the overall quality and scalability of the combined **Grp&SplT**, followed by that of **Grp** individually.

7.2.1 Quality Evaluation

We present the quality evaluations next.

Grp&SplT Quality: The average of overall objective function value, which is the sum of $Dist(G)$ 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 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 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

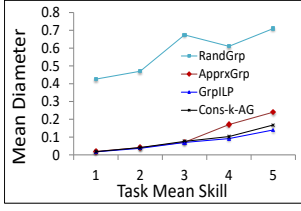


Fig. 13: Grp : Mean Diameter varying Mean Skill

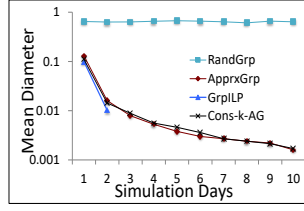


Fig. 14: Grp : Mean Diameter varying Simulation Days

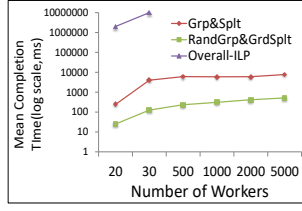


Fig. 15: Grp&SplT : Mean Completion Time varying Number of Workers

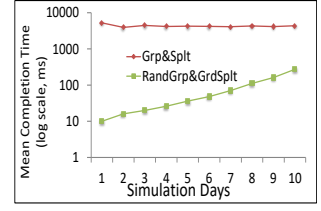


Fig. 16: Grp&SplT : Mean Completion Time varying Simulation Days

intuitively is meaningful, as with increasing skill requirement, the generated group is large, which decreases the workers cohesiveness further.

Varying Critical Mass: As Figure 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 12 simulation period is varied, where both workers and tasks arrive based on Poisson process. Grp&SplT convincingly outperforms

RandGrp&GrdSplT in this experiment.

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

Varying Task Mean Skill: Figure 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 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.

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.

Grp&SplT Efficiency: Varying # Workers: Figure 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.

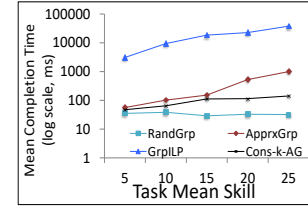


Fig. 17: Grp : Mean Completion Time varying Mean Skill

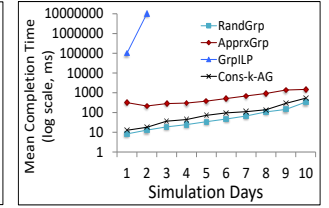


Fig. 18: Grp : Mean Completion Time varying Simulation Days

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

Varying Simulation Period: Figure 16 demonstrates that Grp&SplT is highly scalable in a real crowd-sourcing environment, where more and more workers are entering into the system. The results show that RandGrp&GrdSplT is also scalable (but significantly worse in quality). But as number of worker increases, efficiency decreases, for both, as expected.

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

Varying Task Mean Skill: As Figure 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 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.

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 [32,31,2] 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) [19,24], 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 [6]. *Legion*, a real time user interface, enables integration of multiple crowd workers input at the same time [39]. *Turkit* provides an interface to programmer to use human computation inside their programming model [41] 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 [1]. *CrowdForge* divides complex task into smaller sub-tasks akin to map-reduce fashion [33]. *Turkomatic* introduces a framework in which workers aid requesters to break down the workflow of a complex task and thereby aiding to solve it using systematic steps [36].

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 [11]. One prior work investigates how to assign workers to the task for knowledge intensive crowdsourcing [49] 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. We initiate the study of task assignment optimization in collaborative tasks in [46]. Crowd4u, an academic crowdsourcing platform, effectively integrate our previous work in their framework citekeda2016collaborative. Effectiveness of collaborative teams drawn from the online labor marketplaces for solving innovative tasks is studied in [7]. Es-

timization of human factors (such as skill or expertise) in complex tasks is studied in [47,45]. These works justify our modeling for considering social interaction variable such as affinity and individual human factors. s

Automated Team Formation: Although tangentially related with crowdsourcing, automated team formation is widely studied in computer assisted cooperative systems. [38] 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 [3] and multi-objective optimization on coordination cost and balancing workload is also proposed [4,42], where coordination cost is posed as a constraint. Density based coordination is introduced in [16], where multiple workers with similar skill are required in a team, such as ours. Formation of team with a leader (moderator) is studied in [25]. Minimizing both communication cost and budget while forming a team is first considered in [26,27]. The concept of pareto optimal groups related to the skyline research is studied in [26].

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 [10], 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.

9 Conclusion and Future Work

In this paper, 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 comple-

tion 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 future, we plan to explore alternative collaboration frameworks. An example of such framework can be a star-shaped framework, where the task assignment module assigns both managers and the workers for a task. We also plan to estimate the worker to worker affinity more accurately since it plays a very important role in collaborative task assignment process. We would like to leverage the task assignment framework and task evaluation score to estimate the affinity of the workers.

References

1. 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*, pages 53–64. ACM, 2011.
2. 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*, pages 1531–1532. ACM, 2015.
3. A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi. Power in unity: Forming teams in large-scale community systems. *CIKM '10*, 2010.
4. A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi. Online team formation in social networks. *WWW '12*, 2012.
5. H. P. Andres. Team cognition using collaborative technology: a behavioral analysis. *Journal of Managerial Psychology*, 2013.
6. 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 23rd annual ACM symposium on User interface software and technology*, pages 313–322. ACM, 2010.
7. K. Boudreau, P. Gaule, K. R. Lakhani, C. Riedl, and A. W. Woolley. From crowds to collaborators: Initiating effort & catalyzing interactions among online creative workers. 2014.
8. K. Chai, V. Potdar, and T. Dillon. Content quality assessment related frameworks for social media. In *ICCSA 2009*.
9. D. L. Chen and W. B. Dolan. Building a persistent workforce on mechanical turk for multilingual data collection. In *HCOMP*, 2011.
10. M. Chhabra, S. Das, and B. Szymanski. Team formation in social networks. In *Computer and Information Sciences III*, pages 291–299. Springer, 2013.
11. A. T. M. B. Daniela Retelny, Sbastien Robaszkiewicz. Expert crowdsourcing with flash teams. In *CrowdConf 2013 poster*.
12. 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*, pages 238–247. ACM, 2015.
13. J. S. Downs, M. B. Holbrook, S. Sheng, and L. F. Cranor. Are your participants gaming the system?: screening mechanical turk workers. *CHI '10*.
14. S. K. et. al. Compact location problems. *Th. Comp. Sci*, 1996.
15. S. S. R. et. al. Facility dispersion problems: Heuristics and special cases. In *WADS*, 1991.
16. A. Gajewar and A. D. Sarma. Multi-skill collaborative teams based on densest subgraphs. In *SDM*, pages 165–176. SIAM, 2012.
17. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. 1979.
18. M. Grotschel and L. Lovász. Combinatorial optimization. *Handbook of combinatorics*, 2:1541–1597, 1995.
19. S. Guo, A. G. Parameswaran, and H. Garcia-Molina. So who won?: dynamic max discovery with the crowd. In *SIGMOD Conference*, pages 385–396, 2012.
20. N. Guttmann-Beck and R. Hassin. Approximation algorithms for minimum k-cut. *Algorithmica*, 2000.
21. G. Hertel. Synergetic effects in working teams. *Journal of Managerial Psychology*, 2011.
22. 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.
23. A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decis. Support Syst.*, 43(2):618–644, Mar. 2007.
24. H. Kaplan, I. Lotosh, T. Milo, and S. Novgorodov. Answering planning queries with the crowd. *PVLDB*, 6(9):697–708, 2013.
25. M. Kargar and A. An. Discovering top-k teams of experts with/without a leader in social networks. *CIKM '11*, 2011.
26. M. Kargar, A. An, and M. Zihayat. Efficient bi-objective team formation in social networks. In P. Flach, T. Bie, and N. Cristianini, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 7524 of *Lecture Notes in Computer Science*, pages 483–498. Springer Berlin Heidelberg, 2012.
27. M. Kargar, M. Zihayat, and A. An. Finding affordable and collaborative teams from a network of experts.
28. M. Karpinski. Approximability of the minimum bisection problem: an algorithmic challenge. In *Mathematical Foundations of Computer Science 2002*.
29. D. Katz and R. L. Kahn. The social psychology of organizations. 1978.
30. R. Kenna and B. Berche. Managing research quality: critical mass and optimal academic research group size. *IMA Journal of Management Mathematics*.
31. 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, CSCW '08*, pages 37–46, New York, NY, USA, 2008. ACM.
32. 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.

33. A. Kittur, B. Smus, S. Khamkar, and R. E. Kraut. Crowdforge: Crowdsourcing complex work. In *UIST*, 2011.
34. A. Kittur, B. Suh, B. A. Pendleton, and E. H. Chi. He says, she says: conflict and coordination in wikipedia. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 453–462. ACM, 2007.
35. D. Klakow and J. Peters. Testing the correlation of word error rate and perplexity. *Speech Commun.*, 38(1):19–28, Sept. 2002.
36. A. Kulkarni, M. Can, and B. Hartmann. Collaboratively crowdsourcing workflows with turkomatic. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1003–1012. ACM, 2012.
37. T. Lappas, K. Liu, and E. Terzi. Finding a team of experts in social networks. *KDD '09*.
38. T. Lappas, K. Liu, and E. Terzi. Finding a team of experts in social networks. In *SIGKDD*, pages 467–476, 2009.
39. 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*, *UIST '11*, pages 23–32, New York, NY, USA, 2011. ACM.
40. E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719, 1966.
41. G. Little, L. B. Chilton, M. Goldman, and R. C. Miller. TurkIt: human computation algorithms on mechanical turk. In *Proceedings of the 23rd annual ACM symposium on User interface software and technology*, pages 57–66. ACM, 2010.
42. 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*, *KDD '12*, pages 1005–1013, New York, NY, USA, 2012. ACM.
43. G. Marwell, P. E. Oliver, and R. Prael. Social networks and collective action: A theory of the critical mass. *American Journal of Sociology*, 1988.
44. I. B. Myers and M. H. McCaulley. *Myers-Briggs Type Indicator: MBTI*. Consulting Psychologists Press, 1988.
45. P. Ojha and P. Talukdar. Quality estimation of workers in collaborative crowdsourcing using group testing. 2016.
46. H. Rahman, S. B. Roy, S. Thirumuruganathan, S. Amer-Yahia, and G. Das. Task assignment optimization in collaborative crowdsourcing. In *Data Mining (ICDM), 2015 IEEE International Conference on*, pages 949–954. IEEE, 2015.
47. H. Rahman, S. Thirumuruganathan, S. B. Roy, S. Amer-Yahia, and G. Das. Worker skill estimation in team-based tasks. *Proceedings of the VLDB Endowment*, 8(11):1142–1153, 2015.
48. D. J. Rosenkrantz, G. K. Tayi, and S. S. Ravi. Facility dispersion problems under capacity and cost constraints. *J. Comb. Optim.*, 2000.
49. S. B. Roy, I. Lykourantzou, S. Thirumuruganathan, S. Amer-Yahia, and G. Das. Optimization in knowledge-intensive crowdsourcing. *CoRR*, abs/1401.1302, 2014.
50. A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986.
51. 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.
52. S. van Dongen and A. J. Enright. Metric distances derived from cosine similarity and pearson and spearman correlations. *CoRR*, abs/1208.3145, 2012.
53. 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.