The VLDB Journal (2019) 28:1–23
https://doi.org/10.1007/s00778-018-0516-7

REGULAR PAPER

Optimized group formation for solving collaborative tasks

Habibur Rahman[1] · Senjuti Basu Roy[2] · Saravanan Thirumuruganathan[3] · Sihem Amer-Yahia[4] · Gautam Das[1]

Received: 30 March 2017 / Revised: 19 March 2018 / Accepted: 16 July 2018 / Published online: 26 July 2018

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 threefold. First, we formalize the notion of collaboration among crowd workers and propose a comprehensive optimization model for task assignment in a collaborative crowdsourcing environment. Next, we study the hardness of the task assignment optimization problem and propose a series of efficient exact and approximation algorithms with provable theoretical guarantees. Finally, we present a detailed set of experimental results stemming from two real-world collaborative crowdsourcing application using Amazon Mechanical Turk.

Keywords Crowdsourcing · Collaboration · Group formation · Algorithms

1 Introduction

Crowdsourcing complex tasks Microtask-based crowdsourcing has been applied successfully in a number of domains such as collecting labeled data, fact checking, and image recognition [14]. Here, the crowd workers can operate independently because of the simplicity of the tasks. However, such an individualistic approach will not work for many complex knowledge intensive tasks such as citizen science where crowdsourcing is increasingly being used. Collaborative crowdsourcing is an emerging paradigm where a set of workers with complementary skills form groups and collaborate to perform complex tasks.[1] The synergistic effect of collaboration in group-based activities is widely accepted in socio-psychological research and traditional team-based activities [5,24,25]. A number of popular applications such as collaborative document editing, sentence translation, or citizen science could be modeled as collaborative crowdsourcing tasks. Despite its immense potential, the transformative effect of "collaboration" remains largely unexplored in crowdsourcing [39].

Group formation for solving collaborative tasks The optimization goals for task assignment is putatively similar between collaborative task and traditional microtask—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

✉ Habibur Rahman
habibur.rahman@mavs.uta.edu

Senjuti Basu Roy
senjutib@njit.edu

Saravanan Thirumuruganathan
sthirumuruganathan@hbku.edu.qa

Sihem Amer-Yahia
sihem.amer-yahia@imag.fr

Gautam Das
gdas@uta.edu

1 UT Arlington, Arlington, USA

2 New Jersey Institute of Technology, Newark, USA

3 QCRI, HBKU, Doha, Qatar

4 CNRS, LIG, Grenoble, France

[1] This work is the extension of our paper [57]. 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

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 [60], such as skill and wages. From prior work on socio-psychological research[24,25], we have identified two key factors for group characteristics that entail successful collaboration. The first factor *worker–worker affinity* [40,70] 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 [41]. Social theories widely underscore the importance of *upper critical mass* [35] for group collaboration, which is a constraint on the size of groups beyond which the collaboration effectiveness diminishes [16,35,53,65].

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 upper critical mass has been identified in psycho-social theories, there has been no prior effort on formalizing these individual and group-based human factors in a principled manner to optimize the outcome of a collaborative crowdsourcing environment. Hence, our first significant contribution lies in appropriately incorporating the interplay of 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 [34] in collaborative crowdsourcing as a graph with nodes representing workers and edges labeled with pairwise affinities. A group of workers is a clique in the graph whose size does not surpass the upper critical mass imposed by a task. A large clique (group) may further be partitioned into subgroups (each is a clique of smaller size satisfying upper critical mass) to complete a task because of the task's magnitude. Each clique has an intra- and an inter-affinity to measure, respectively, the level of cohesion that the clique has internally and with other cliques. A clique with high intra-affinity implies that its members collaborate well with one another. Two cliques with a high inter-affinity between them imply that these two groups of workers work well together. Our optimization problem reduces to finding a clique that maximizes intra-affinity, satisfies the skill threshold across multiple domains, satisfies the cost limit, and maximizes inter-affinity when partitioned into smaller cliques. We note that no existing work on team formation in social networks [4,44] or collaborative crowdsourcing [39,40,70] 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 upper critical mass) and the inter-affinity across subgroups is maximized. Despite being NP-hard [20], 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 Sect. 4, we formally define our problem which incorporates a variety of human factors.
2. *Solutions* We propose a comprehensive theoretical analysis of our problems and approaches. We analyze the computational complexity of our problems and propose a principled staged solution. We propose exact instance optimal algorithms as well as efficient approximation algorithms with provable approximation bounds.
3. *Experiments* We present a comprehensive set of experimental results (two real applications as well as synthetic

experiments) that demonstrate the effectiveness of our proposed solutions.

The 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. We describe some of the limitations and possible extensions of our approach in Sect. 7. Experiments are described in Sect. 8, related work in Sect. 9, and conclusion is presented in Sect. 10.

2 An application of collaborative task

Sentence translation [11,40,70] 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 [40,70]: 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 [70]), and the collaboration effectiveness is quantified as the *affinity* of the group. Some aspects of our formulation have similarities with team formation problems in social networks [4]. The notion of affinity has been identified in the related work on sentence translation tasks [40,70], as well as team formation problems [4].

However, if the group is "too large," the effectiveness of collective actions diminishes [16,35,53,65] while undertaking the translation task, as an unwieldy group of workers fails to find effective assistance from their peers [40,70]. 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.

Table 1 Workers skill and wage table

	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 2 Workers distance matrix

	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 3 Task description

Q_1	Q_2	Q_3	C	K
1.8	1.4	1.66	3.0	3

When this task arrives, imagine that there are 6 workers $u_1, u_2, \ldots, u_6$ available on 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 upper 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 subgroup undertakes a subset of sentences to translate. Once all the subgroups finish their respective efforts, their contributions are merged. Therefore, both the overall group and its subgroups must be cohesive. Incorporation of upper critical mass makes our problem significantly different from the body of prior works [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 $D = \{d_1, d_2, \ldots, d_m\}$ denoting knowledge topics. Using the running example in Sect. 2, there are 3 different domains—English comprehension (d_1), English editing (d_2), and French Translation ability(d_3).

Workers We assume a set $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ of n workers available in the crowdsourcing platform. The example in Sect. 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 of 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 on its own merit. Most related work has relied on learning the skill of the workers from "gold-standard" or benchmark datasets using pre-qualification tests [15,27]. It is also possible to use works such as [58] 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.
- *Upper 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 upper critical mass threshold. In that case, the workers should be split in subgroups (each satisfying the upper critical mass constraint) such that the workers across all the subgroups satisfy the skill and cost threshold. We discuss the impact of imposing a strict upper limit and how to relax it in Sect. 7.

Specifically, t is characterized by a vector, $\langle Q_1, Q_2, \ldots, Q_m, C, K\rangle$, of length $m+2$. For the example in Sect. 2, there are 3 domains ($m = 3$) and their respective skill requirements, its cost C, and upper critical mass K of the task are described in Table 3. A task is considered complete if it attains its skill requirement over all domains and satisfies all the constraints. For the ease of exposition, we assume that the domain of worker skills and task requirements are identical and that the contribution of a group for a given task can be computed by summing the skills of its workers. We discuss other aggregation functions in Sect. 7.

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 [4,44] and psychology research [16,35,53,65].

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 [40,70], 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 [4,45], where the atomic unit of affinity is *pairwise*, i.e., a measure of cohesiveness between every pair of workers. After that, we propose different ways to capture *intra-group* and *inter-group* affinities.

Pairwise affinity The affinity between two workers u_i and u_j, $\mathit{aff}(u_i, u_j)$, can be calculated by capturing the *similarity* between workers using simple socio-demographic attributes, such as region, age, gender, as done in previous work [70], as well as more complex psychological characteristics [54]. For our purpose, we normalize pairwise affinity values to fit in [0, 1] and use a notion of worker–worker *distance* instead, i.e., where $\mathit{dist}(u_i, u_j) = 1 - \mathit{aff}(u_i, u_j)$. Thus, a smaller distance between workers ensures a better collaboration. Table 2 presents the pairwise distance of all 6 workers for running example in Sect. 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 $\mathcal{G}$:

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

For both definitions, a smaller value is better.

Inter-group affinity When a group violates the upper critical mass constraint [35], 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 subgroups or the aggregation of all pairwise workers distances across subgroups:

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

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

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 [45]. Then, we adopt 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 [70] itself, as the "turker-turker" affinity model. However, unlike previous work, we attempt to fully explore the potential of "group synergy" [66] and how it yields the maximum qualitative effects in group-based efforts by maximizing affinity among the workers (or minimizing distance). Finally, we intend to investigate the effect of upper critical mass in the context of collaborative crowdsourcing as a constraint on group size, beyond which the group must be decomposed into multiple subgroups that are cohesive inside and across. Indeed, our objective function is designed to form a group (or further decomposed into a set of subgroups) to undertake a specific task that achieves the highest qualitative effect while satisfying the cost constraint.

1. *Qualitative effect of a group* Intuitively, the overall qualitative effect of a formed group to undertake a specific task is a function of the skill of the workers and their collaboration effectiveness. Learning this function itself is challenging, as it requires access to adequate training data and domain knowledge. In our initial effort, we, therefore, make a reasonable simplification, where we seek to maximize group affinity and pose quality as a hard constraint[2]. Existing literature (indicatively [66]) 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 function for transforming individual contributions into a collective result has been adopted in many previous works [4,17,45]. Moreover, this simpler function allows us to design efficient algorithms. For example, optimizing an optimization problem with Max in its constraint is a challenging problem [7,8,58]. Exploring other complex functions (e.g., max, min, and multiplicative functions) or learning them is deferred to future work.
2. *Upper critical mass* Sociological theories widely support the notion of "upper critical mass"[16,35,53,65] by reasoning that large groups are less likely to support collective action. However, whether the effect of "upper critical mass" should be imposed as a hard constraint, or it should have more of a gradual "diminishing return" effect, is itself a research question. For simplicity, we consider upper critical mass as a hard constraint specified by the domain expert and evaluate its effectiveness empirically for different values. Exploring more sophisticated functions to capture upper 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 $G_1, G_2, \ldots . G_x$ (if needed) for the task t

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

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 upper 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 [4,45]. 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 Sect. 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. As mentioned previously, exploration of other aggregation functions such as Max are left to future work. This gives rise to 4 possible optimization objectives.

- *DiaDist*, *DiaInterDist*:

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

- *SumDist*, *DiaInterDist*:

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

- *DiaDist*, *SumInterDist*:

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

- *SumDist*, *SumInterDist*:

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

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

$$\Sigma_{\forall_{u_i \in \mathcal{G}}} u_{d_i} \geq Q_i \quad \forall_{d_i}$$
$$\Sigma_{\forall_{u \in \mathcal{G}}} w_u \leq C$$
$$|G_i| \leq K \quad \forall i \in \{1, 2, \dots, x\}$$

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 Sect. 4 could be easily extended to other combinations as well.

Theorem 1 *Problem* `AffAware-Crowd` *is NP-hard [20].*

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 upper 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* [29] 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 subgraph g' of size $n' \in N$, such that the maximum distance between any pair of nodes in g' is $\leq X'$. This variant of the compact location problem is known as `Min-DIA` in [29].

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 upper 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. Toward this end, we propose two alternative directions:
(I) *ILP* We propose a integer linear programming (ILP) [63] 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 an approximation factor of 2. It invokes a subroutine, which uses the 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 the branch and bound method. However, it iterates over all the valid solutions to find the optimal one.

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

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

4.2.1 ILP for AffAware-Crowd

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

$$
\begin{aligned}
\text{minimize} \quad & \mathcal{D} = \text{Max}\{e_{i,i'} \times dist(u_i, u_{i'})\} \\
& \quad + \sum_{\forall G_i, G_j \in \mathcal{G}} \; \sum_{\forall u_i \in G_i, u_j \in G_j} e_{i,j} dist(u_i, u_j) \\
\text{subject to} \quad & \\
& \sum_{i=1}^{n} \sum_{j=1}^{x} u_{(i,G_j)} \times u^i_{d_l} \geq Q_l \qquad \forall l \in [1, m] \\
& \sum_{i=1}^{n} \sum_{j=1}^{x} u_{(i,G_j)} \times w^i_u \leq C \\
& \sum_{i=1}^{n} u_{(i,G_j)} \leq K \qquad \forall j \in [1, x] \\
& \sum_{j=1}^{x} u_{(i,G_j)} \leq 1 \qquad \forall i \in [1, n] \\
& e_{i,i'} = \begin{cases} 1 & \exists j \in [1, x] \,\&\, u_{(i,G_j)} = 1 \,\&\, u_{(i',G_j)} = 1 \\ 0 & \text{otherwise} \end{cases} \\
& x \in \{0, 1, \ldots, n\} \\
& u_{(i,G_j)} \in \{0, 1\} \qquad \forall i \in [1, n], \forall j \in [1, x]
\end{aligned}
\tag{1}
$$

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

When run on the example in Sect. 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 subgroups, each satisfying the upper critical mass constraint in such a way that the aggregated inter-distance between all pair of groups $\Sigma_{\forall_{G_i,G_j}} SumInterDist(G_i, G_j)$ is minimized. Clearly, one could employ other alternative approaches such as ignoring the cost and/or skill constraints followed by a systematic merging procedure. Indeed, we considered such an approach but settled on the `Grp&Splt` as it has the following properties:

- *Solution feasibility* We observed from a number of real-world and synthetic experiments, it is often feasible to identify a coherent group of workers that satisfy the task requirements without exceeding the critical mass criteria. Only in a small fraction of time—such as when the task is very challenging and/or worker pool is not very competent—does one need to even employ the second stage of `Splt`. Furthermore, if no group could solve the task, it could be easily identified by our approach whereas the merging-based approach might evaluate many combinations.
- *Solution guarantees* `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) [56,59]. On the other hand, we were unable to derive analogous guarantee for the merging-based approach. 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 [69]. 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 [59].
- *Solution scalability* The optimal solution based on ILP is prohibitively expensive. Our experimental results demonstrate that the original ILP does not converge in hours for more than 20 workers, whereas our `Grp&Splt` scales well for thousands of workers.

5 Enforcing skill and Cost: GRP

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

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

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

Proof Given a collaborative task t with upper 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, follow the similar strategy as above. We use an instance of `Min-DIA` [29] and reduce that to an instance of `Grp`, as follows: each node in graph G of `Min-DIA` represents a worker u, and the distance between any two nodes in G is the distance between a pair of workers for our problem. We assume that the number of skill domain is 1, i.e., $m = 1$. Additionally, we consider that each workers u has the same skill value of 1 on that domain, i.e., $u_d = 1, \forall u$ and their cost is 0, i.e., $w_u = 0, \forall u$. Task t has quality requirement on only one domain, which is, Q_1. The skill requirement of t is $\langle Q_1 = n'$ and cost $C = 0\rangle$. Now, the objective is to form a group $\mathcal{G}$ for task t such that the skill and cost constraints are satisfied with the diameter of `Grp` as X', such that there exists a solution to the `Min-DIA` problem, if and only if, a solution to our instance of `Grp` exists. □

Proposed algorithms for `Grp` We discuss two algorithms at length—a) `OptGrp` is an instance optimal algorithm. b) `ApprxGrp` algorithm has a 2-*approximation factor*, as long as the distance satisfies the triangle inequality property. Of course, an additional optimal algorithm is the ILP formulation itself (referred to as `ILPGrp` in experiments), which could be easily adapted from Sect. 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

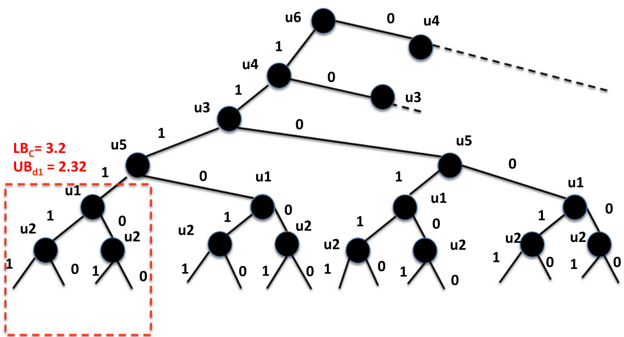

Fig. 1 A partially constructed tree of GrpCandidateSet using the example in Sect. 2. At node $u_1 = 1$, $LB_C = w_{u_6} + w_{u_4} + w_{u_3} + w_{u_5} + w_{u_1} = 3.2$ and $UB_{d_1} = u^6_{d_1} + u^4_{d_1} + u^3_{d_1} + u^5_{d_1} + u^1_{d_1} + u^2_{d_1} = 2.32$. The entire subtree is pruned, since $LB_C(3.2)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

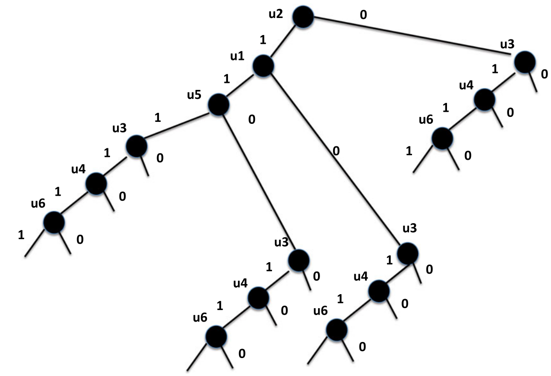

Fig. 2 Possible search space using the example in Sect. 2, after the cost of the workers are discretized into $k = 2$ buckets, considering only one skill d_1. The tree is constructed in descending order of skill of the workers per bucket. For bucket 1, if the most skilled worker u_2 *is not selected,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

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 up to 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 shows further details.

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 a priori 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 Sect. 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.

5.3 Approximation algorithm ApprxGrp

A popular variant of facility dispersion problem [56,59] attempts to discover a set of nodes (that host the facilities) that are as far as possible, whereas, compact location problems [29] 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

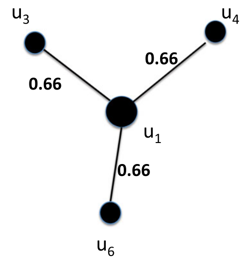

Fig. 3 An instantiation of `GrpDia(0.66)` using the example in Sect. 2. A star graph centered u_1 is formed

Algorithm 1 Approximation Algorithm `ApprxGrp`

Require: $\mathcal{U}$, human factors for $\mathcal{U}$ and task t
List $\mathcal{L}$ contains all unique distance values in increasing order
repeat
  Perform binary search over $\mathcal{L}$
  For a given distance α, $\mathcal{U}' =$ `GrpDia(`, $\{Q_i, \forall d_i\}$, `C)`
  if $\mathcal{U}' \neq \emptyset$ **then**
    Store worker group $\mathcal{U}'$ with diameter $d \leq 2\alpha$.
  end if
until the search is complete
return $\mathcal{U}'$ with the smallest d

Algorithm 2 Subroutine `GrpDia`

Require: Distance matrix of the worker set $\mathcal{U}$, distance α, task t.
repeat
  for each worker u
  form a star graph centered at u, such that for each edge u, u_j, $dist(u, u_j) \leq \alpha$. Let $\mathcal{U}'$ be the set of workers in the star graph.
  $\mathcal{U}''$ = `GrpCandidateSet`$(\mathcal{U}', t)$
  if $\mathcal{U}'' \neq \emptyset$ **then**
    return $\mathcal{U}''$
  end if
until all n workers have been fully exhausted
return $\mathcal{U}'' = \emptyset$

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 Fig. 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 Sect. 5.2. This variant of `ApproxGrp` is referred to as `Cons-k-Cost-ApproxGrp`.

Upon returning a non-empty subset $\mathcal{U}''$ of $\mathcal{U}'$, `GrpCandidateSet` terminates. Then, `ApprxGrp` stores that α and associated $\mathcal{U}''$ and continues its binary search over $\mathcal{L}$ for a different α. Once the binary search ends, it returns that $\mathcal{U}''$ which has the smallest α associated as the solution with the diameter upper bounded by 2α, as long as the distance between the workers satisfy the triangle inequality[4]. 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 Sect. 2, $\mathcal{L}$ is ordered as follows: 0, 0.4, 0.66, 0.85, 1.0. The binary search process in the first iteration considers $\alpha = 0.66$ and calls `GrpDia`$(\alpha, \{Q_i, \forall d_i\}, C)$. In the first iteration, `GrpDia` attempts to find a *star graph* (referred to Fig. 3) with u_1 as the center of the star. This returned graph is taken as the input along with the skill threshold of t inside `GrpCandidateSet`next. For our running example, subroutine `GrpDia`(0.66, 1.8, 1.66, 1.4, 2.5) returns u_1, u_3, u_4, u_6. Now notice, these 4 workers do not satisfy the skill threshold of task t (which are, respectively, 1.8, 1.66, 1.4 for the 3 domains.). Therefore, `GrpCandidateSet`$(\mathcal{U}, t)$ returns false and `GrpDia` continues to check whether a star graph centered around u_2 satisfies the distance threshold 0.66. When run on the example in Sect. 2, `ApprxGrp` returns workers u_1, u_2, u_3, u_5, u_6 as the results with objective function value upper bounded by $\leq 2 \times 0.66$.

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 Fig. 3 that forms a star graph centered on u_1 with `GrpDia(0.66)` using the example in Sect. 2. Consider Fig. 4 and notice that for a given distance value $=\alpha$, the complete graph induced by the star graph cannot have any edge that is larger than $2 \times \alpha$, as long as the distance satisfies the triangle inequality property. Therefore, when `GrpDia`(α) returns a non-empty worker set (that only happens when the skill and cost thresholds are satisfied), then, those workers satisfy the skill and cost threshold with the optimization objective value of $\leq 2\alpha$. Next, notice that algorithm `ApprxGrp` overall attempts to return the smallest distance α' for which `GrpDia`(α') returns a non-empty set, as it performs a binary search over the sorted list of distance values (where distance is sorted in smallest to largest). Therefore, any group of workers returned by `ApprxGrp` satisfies the skill and cost threshold value and

[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 [59].

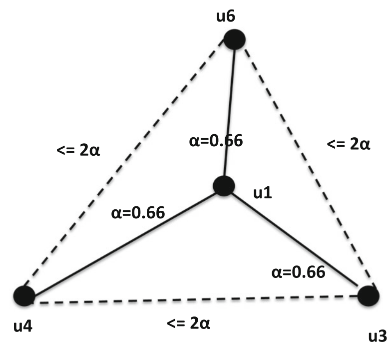

Fig. 4 An instantiation of `GrpDia(0.66)` using the example in Sect. 2. The clique involving u_1, u_3, u_4, u_6 cannot have an edge with distance $> 2 \times 0.66$, due to the triangle inequality property

DiaDist$(\mathcal{G})$ is at most 2-times worse than the optimal. Hence the approximation factor holds. □

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

Proof Under a constant number of k-costs, subroutine `GrpCandidateSet` will accept a polynomial computation time of $O(p+1)^{mk}$ at the worst case, where p is the maximum number of workers in one of the k buckets ($p = O(n)$). Subroutine `GrpDia` runs for all n workers at the worst case, and there is a maximum number of $log_2|\mathcal{L}|$ calls to `GrpDia` from the main function ($|\mathcal{L}| = O(n^2)$). Therefore, the asymptotic complexity of `Cons-k-ApproxGrp` is $O(n \times log_2|\mathcal{L}| \times (p+1)^{mk})$, which is polynomial. □

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 Sect. 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 Sect. 5.2 and is referred to as `Cons-k-Cost-OptGrp`.

Theorem 4 *Algorithm* `OptGrp` *returns optimal answer.*

Proof Algorithm `OptGrp` invokes the subroutine `GrpCandidateSet`. Notice that `GrpCandidateSet` operates in the spirit of the branch-and-bound technique [47] 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 Figs. 1 and 2. However, this subroutine returns all valid worker groups to `OptGrp`, and then, the main function selects the group with the smallest longest edge (i.e., smallest diameter value), and minimizes the objective function. Therefore, `OptGrp` is instance optimal, i.e., it returns the group of workers with the smallest diameter distance, while satisfying the skill and cost threshold. Therefore, `OptGrp` returns optimal answer. □

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

Proof Under a constant number of k-costs, subroutine `GrpCandidateSet` will accept a polynomial computation time of $O(n+1)^{mk}$ at the worst case. Once the subroutine returns all valid answers, the main function will select the one that has the smallest diameter. Therefore, the computation time of `Cons-k-Cost-OptGrp` is dominated by the computation time of the subroutine `GrpCandidateSet`. Therefore, Algorithm $\mathtt{Cons-k-OptGrp}$ 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 subgroups in the `Splt` phase to satisfy the *upper critical mass* (K) constraint. At the same time, if needed, the workers across the subgroups should still be able to effectively collaborate. Precisely, these intuitions are further formalized in the `Splt` phase.

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

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

Proof Given a group $\mathcal{G}$, an upper critical mass constraint K, and a real number X, the decision version of the `Splt` is whether $\mathcal{G}$ can be decomposed into 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` [33] which is known to be NP-hard to an instance of `Splt` problem.

Given a graph $G(V, E)$ with nonnegative 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 [33].

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 nonnegative edges as their distance and we wish to decompose this group to two subgroups, where the upper critical mass is set to be $K = n'/2$. Now, the objective is to form the subgroups 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 subgroup G_i. Then, we attempt to find an optimal partitioning of the n' workers across these x subgroups that minimizes the objective function. We refer to this as `SpltBOpt` which is the *optimal balanced partitioning* of $\mathcal{G}$. For the running example in Sect. 2, this would mean creating 2 subgroups, G_1 and G_2, with 3 workers in one and the remaining 2 in the second subgroup using the workers u_1, u_2, u_3, u_5, u_6, returned by `ApprxGrp`.

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

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

Proof Sketch: For the purpose of illustration, imagine that a graph with n' nodes is decomposed into two partitions. Without loss of generality, imagine partition-1 has n_1 nodes and partition-2 has n_2 nodes, where $n_1 + n_2 = n'$ with total weight of w'. Let K be the upper critical mass and assume that $K > n_1, K > n_2$. For such a scenario, `SpltBOpt` will move one or more nodes from the lighter partition to the heavier one, until the latter has exactly K nodes. (If both partitions have the 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 get knocked off and becomes intra-edges. It is easy to notice that the number of inter-edges that get knocked off is always larger than 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.

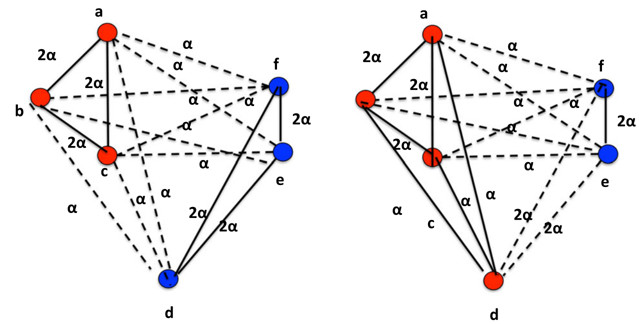

Fig. 5 Balanced partitioning in `SpltBOpt` when the distance satisfies triangle inequality for a graph with 6 modes. The left-hand side figure has two partitions($\{a, b, c\}$, $\{d, e, f\}$) with 3-nodes in each (red nodes create one partition and blue nodes create another). The intra-partition edges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

An example scenario of such a balancing is illustrated in Fig. 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 minimize the objective function is NP-hard. The proof uses an easy reduction from [23]. We start by showing how the solution to the `SpltBOpt` problem could be bounded by the solution of a slightly different problem variant, known as `Min-Star` problem [23].

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

Algorithm 3 Algorithm `Min-Star-Partition`

Require: Group $\mathcal{G}$ with n' workers and upper critical mass K
1: $x = \lceil \frac{n'}{K} \rceil$
2: **for all** subset $\{u_1, \ldots, u_x\} \subset \mathcal{G}$ **do**
3: Find optimal subgroups $\{G_1, \ldots, G_x\}$ for $\{u_1, \ldots, u_x\}$ by formulating it as transportation problem
4: Evaluate objective function for $\{G_1, \ldots, G_x\}$
5: **end for**
6: **return** subgroups $\{G_1, \ldots, G_x\}$ with least objective function

is minimized, where k_i is the total number of workers in subgroup G_i.

Intuitively, `Min-Star` problem seeks to decompose the worker set into x subgroups, such that u_i is the center of a star graph for subgroup G_i, and for a fixed set of such workers $\{u_1, u_2, \ldots, u_x\}$, the contribution of u_i to the objective function is proportional to the sum of distances of a star subgraph rooted at u_i.
Solving `Min-Star` *Algorithm* `Min-Star-Partition` The pseudo-code is listed in Algorithm 3 and additional details can be found in [23]. The key insight behind this algorithm is the fact that for a fixed set of workers $\{u_1, u_2, \ldots, u_x\}$, the second term of the objective function $\sum_{i<j} k_i k_j dist(u_i, u_j)$ is a constant. Furthermore, this expression could only take $\binom{n'}{x}$ distinct values corresponding to all possible combination of how the workers $\{u_1, u_2, \ldots, u_x\}$ are chosen from the group $\mathcal{G}$ with n' workers. Hence for a fixed set of workers, the objective now reduces to finding optimal subgroup $G_1, \ldots, G_x$ that minimizes the first expression. Interestingly, this expression corresponds exactly to a special case of the popular *transportation problem* [21] that could be solved optimally with time complexity $O(n')$ [23]. We refer to [23] 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 [23]. Previous work [23] 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 [23] 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 $O(n')$ [23] time. Since x is a constant, therefore, the overall running time is polynomial. □

7 Discussion

Skill aggregation functions A skill aggregation function computes the overall contribution of a group from the skills of its individual workers. In this paper, we primarily focused on Sum aggregation function where the contribution of the group is simply the sum of the skills of its members. While Sum is one of the most widely used aggregation function, it is not the only one. Other options include Max, Min, and many other complex aggregation functions.

Sum is typically used when the contribution of the group can be approximated by the sum of the contributions of individual workers. For example, the score of a team in most sports is often the sum of the score of individual players (shoots, blocks, runs, etc.). The number of images labeled by a group is the sum of images labeled by individual members and so on. *Max* is often a good approximation for complex problem-solving tasks. In many creative tasks such as research, the insight from the most creative person often sets the overall quality of the task. For example, often it is not possible for multiple high school students to solve a grad school level problem. *Min* is often used for competitive games/tasks. For example, in a team-based game where the score of the team is proportional to its slowest member, Min is often a good aggregation function.

Submodular aggregation functions could be used to model the diminishing returns property inherent in many real-world tasks. Often, adding a new team member has a higher marginal utility when added to a smaller team than to a larger team. Common examples include inverse and exponential decay functions. Of course, one could also use many other arbitrarily complex functions.

Note that our staged approach can be readily extended to different types of the aggregation function, primarily because we optimize only on affinity and keep skill and cost as constraints. In other words, the algorithms and their corresponding theoretical guarantees rely on whether the intra- and inter-group distances satisfy the metric property. One could use any mechanism to verify if the skill of a worker group satisfies a task's quality constraints.

We describe how our methods can be adapted for Max aggregation. Consider the instance optimal algorithm `OptGrp`. It works by computing the lower bound of cost and upper bound of skill for a given path and uses it for systematic pruning. This does not require any change to handle Max aggregation. The approximate algorithm ApprxGrp repeatedly calls subroutine GrpCandidateSet (used for solving OptGrp) with different bounds on the maximum intra-group distance, thereby requiring no change. Finally, consider the Splt algorithm that assumes that the chosen group already satisfies the skill constraints. The algorithm operates by only considering the inter-group distances that are needed for transforming it into a transportation problem. Once again, no change is necessary for handling Max aggregation.

Task quality and worker skills In this paper, we considered a simple setting where the relationship is one-to-one and the *Sum* aggregation function is used to measure the contribution of a group. In other words, task quality is computed as the sum of the workers' skills. However, there could be a complex relationship between worker skills and task quality. One could use arbitrary functions that take into account other worker/task characteristics. These could be worker centric (such as motivation, autonomy, etc.), task centric (such as novelty, diversity, meaningfulness) or group centric (such as affinity with other workers). One could use graphical models, such as Bayesian Networks, to represent such relationship. It is an open problem to study the impact of these factors on task assignment for collaborative crowdsourcing. Note that algorithms from the staged approach are oblivious to how one determines if the skill constraints of a task are satisfied.

Critical mass In our paper, we consider upper critical mass as a hard constraint that must be satisfied. There are a number of real-world applications where this constraint is meaningful. For example, citizen science applications, typically constrain the group size to limit the infrastructure cost and other resource usages. Group travel applications would like to limit the group size as the number of available seats is limited. Finally, in many games (educational or otherwise), the number of players is usually put as hard constraints. As we showed in our complexity analysis, task assignment even for this restricted setting is challenging.

There are a number of ways to relax this constraint. One relatively straightforward way is to design a skill aggregation function that is aware of the group size. Aggregation functions that exhibit diminishing returns are a natural possibility where adding a new member to a smaller group provides higher marginal utility than adding to a larger group. Under this setting, we can readily remove the size constraint from our optimization formulation and simply use just the Grp stage to identify the group that minimizes the intra-group distance that also satisfies the task requirements.

8 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 parameterizable crowd simulator.

8.1 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.

Pairwise affinity calculation Designing complex personality test [54] to compute affinity is beyond the scope of this work. We instead choose some simple factors to compute affinity that has been acknowledged to be indicative factors in prior works [70]. 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 Sect. 4 against an alternative Aff-Unaware Algorithm [61]. The latter assigns workers to the tasks considering skill and cost but ignoring affinity. Since, ILP outputs optimal task assignment, we refer to this as Optimal(2) Optimal-Affinity-Age and Optimal-Affinity-Region are two variants of Optimal that use two different affinity calculation methods (Affinity-Age and Affinity-Region, respectively) and are compared against each other to evaluate the second criteria. (3) CrtMass-Optimal-K assigns workers to tasks based on the optimization objective and varies different upper critical mass values K, which are also compared against each other for different K.

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

Evaluation: in stage-3, completed tasks are crowdsourced again to evaluate their quality.

Summary of results There are several key takeaways from 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 the 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`.

8.1.1 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 worker's individual skill using Word Error Rate(WER) [42].

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 first smart phone was manufactured by:* with possible answers: (a) Nokia, (b) Samsung, (c) Ericsson). 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, the affinity between the worker is calculated using, `Affinity-Age` or `Affinity-Region`.

Figures 6 and 7 contain detailed workers profile distribution information.

Table 4 Description of different tasks; the default upper critical mass value is 5

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 places	5	\$3	5, 7, 10

Default affinity calculation is region based

8.1.2 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 task are described in 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 the 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 upper 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 upper critical mass values of the document editing tasks, as described in Table 4.

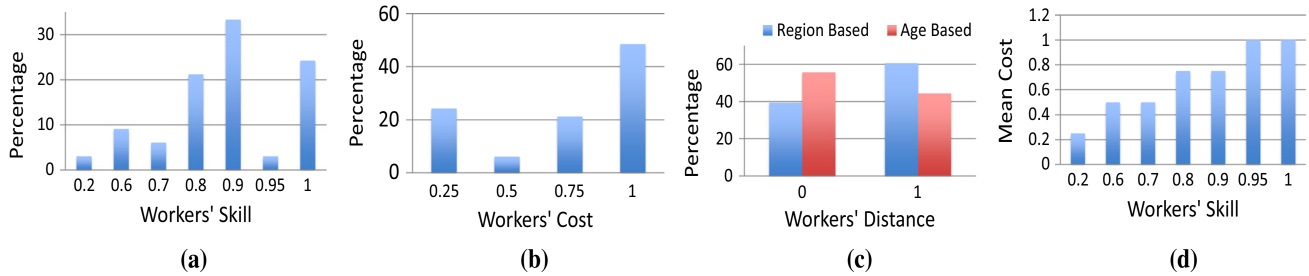

Fig. 6 Worker profile distributions for the sentence translation tasks in Sect. 8.1. **a** Worker skill distribution, **b** worker wage distribution, **c** worker distance distribution, **d** correlation between worker skill and wage

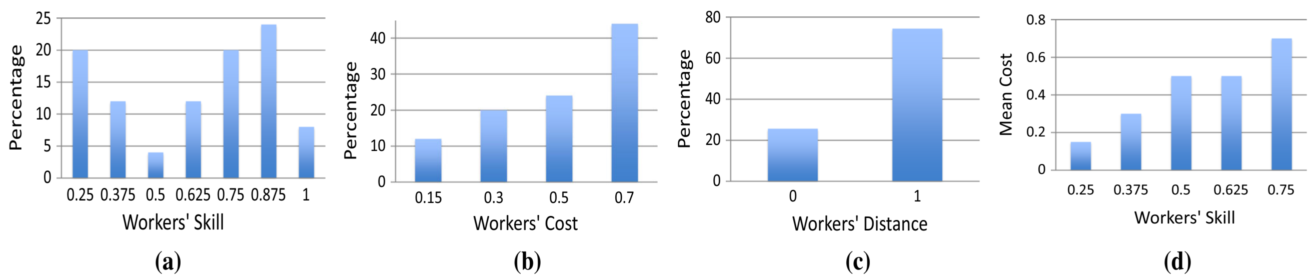

Fig. 7 Worker profile distributions for the collaborative document writing in Sect. 8.1. **a** Worker skill distribution, **b** worker wage distribution, **c** worker distance distribution, **d** strong positive correlation between worker skill and wage

8.1.3 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 [70]—1. Correctness of the translation. 2. Completeness of the translation. The workers are asked to rate the quality on a scale of $1-5$ (higher is better) without knowing the underlying task production algorithm. Then, we average these ratings which are similar to obtaining the viewpoint of an average reader. The CST results of different evaluation dimensions computed by human raters are presented in Fig. 8. We also evaluated our approach using automated measures such as BLEU score whose results are found in Table 5.

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 [10]. The results are summarized in Table 6. Both these results indicate that, indeed, our proposed model successfully incorporates different elements that are essential to ensure high quality in collaborative crowdsourcing tasks.

8.2 Synthetic data experiments

The purpose of this experiment 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 minute. With one task arriving in every 10 minutes, our default setting runs 1 day and has 144 tasks.

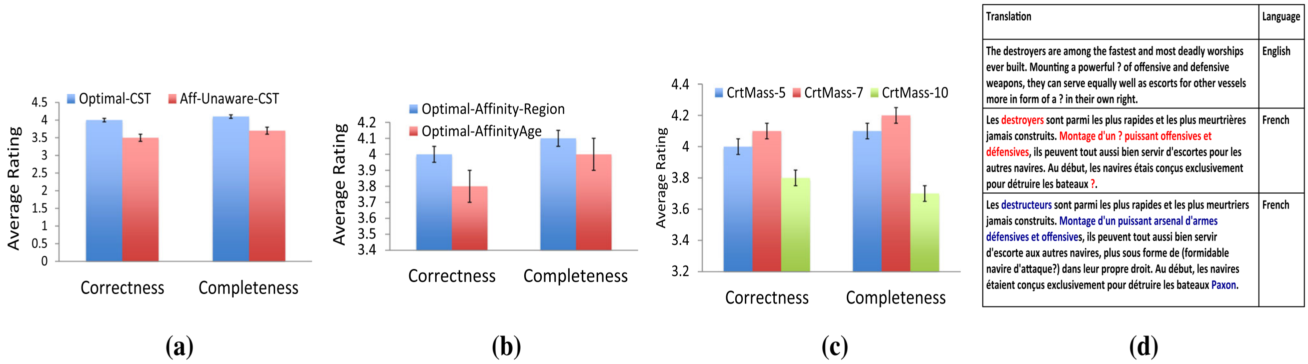

Fig. 8 *Stage 3 results of sentence translation* Collected data with statistical significance (standard error) is presented. These results clearly corroborate that our affinity-aware optimization model `Optimal-CST` outperforms its affinity-unaware counterpart [42] with statistical significance across 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. **a** Optimization model, **b** affinity calculation, **c** upper critical mass, **d** a French translation sample

Table 5 BLEU score of sentence translation average BLEU score of `Optimal-Affinity-Age` and `Optimal-Affinity-Region` considering three different sentence translation tasks in Bengali

Algorithm	BLEU score
`Optimal-Affinity-Age`	0.30
`Optimal-Affinity-Region`	0.28
`Aff-Unaware`	0.26

These results are comparable to each other, and the difference is not statistically significant

2. *# of Workers*—default is 100, but we vary $|\mathcal{U}|$ up to 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. *Upper 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 $\mu = 5$/min 1/10 min, respectively.

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

Table 6 Stage 3 results of document writing application in Sect. 8.1: 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

Average rating							
Task	Algorithm	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

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

1. `ApprxGrp` We implement the algorithm `ApprxGrp`, described in Sect. 5.3.
2. `Cons-k-AG` This is a variant of the algorithm `Apprx Grp` referred to as `Cons-k-cost-ApprxGrp`, described in Sect. 5.3. We set the number of cost buckets k to 15.
3. `GrpILP` An ILP designed for `Grp` stage only.
4. `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 Sect. 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 upper critical mass constraint.

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

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

8.2.1 Quality evaluation

We present the quality evaluations next.

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

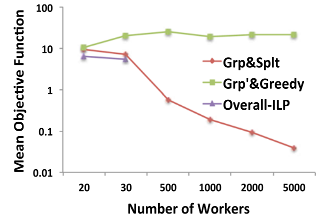

Fig. 9 `Grp&Splt`: objective function varying number of workers

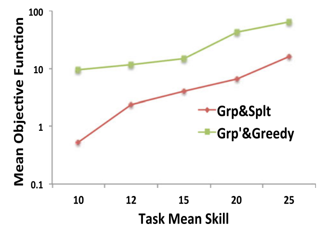

Fig. 10 `Grp&Splt`: objective function varying task mean skill

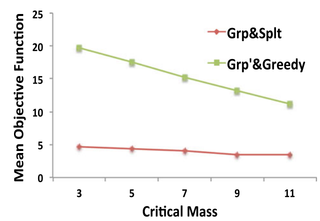

Fig. 11 `Grp&Splt`: objective function varying upper critical mass

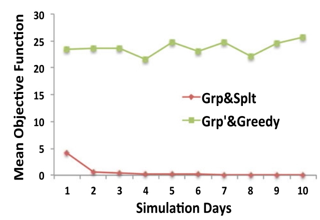

Fig. 12 `Grp&Splt`: objective function over simulation days

Varying # of Workers Figure 9 shows the results, with mean skill=15 and variance=1, demonstrating 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), Fig. 10 demonstrates that the objective function gets higher (hence worse) for both the algorithms, as skill/cost requirement increases, while `Grp&Splt` outperforms `RandGrp&GrdSplt`. This intuitively is meaningful, as with increasing skill requirement, the generated group is large, which decreases the workers cohesiveness further.
Varying critical mass As Fig. 11 shows, with increasing upper 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 Fig. 12 simulation period is varied, where both workers and tasks arrive based

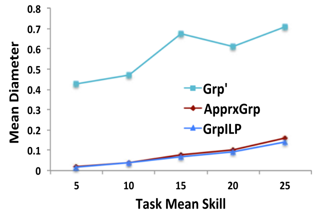

Fig. 13 `Grp`: mean diameter varying mean skill

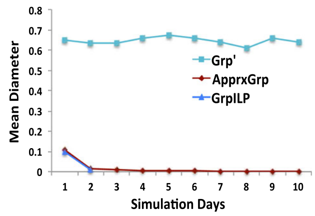

Fig. 14 `Grp`: mean diameter varying simulation days

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` are 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 cannot get the results for `GrpILP` beyond day-2. But, `ApprxGrp` and `Cons-k-AG` works fine and achieves almost optimal result.

8.2.2 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.
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.

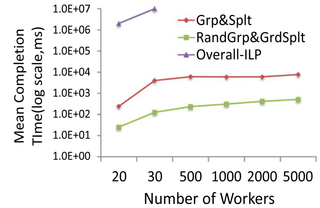

Fig. 15 `Grp&Splt`: mean completion time varying number of workers

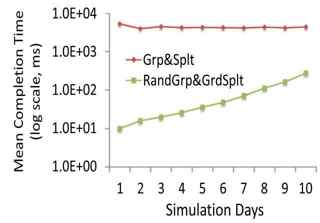

Fig. 16 `Grp&Splt`: mean completion time varying simulation days

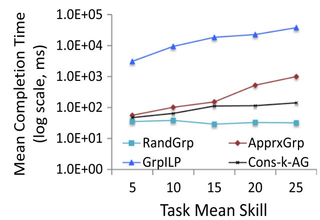

Fig. 17 `Grp`: mean completion time-varying mean skill

Varying upper critical mass As before, increasing upper 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 crowdsourcing environment, where more and more workers are entering into the system. The results show that `RandGrp&GrdSplit` is also scalable (but significantly worse in quality). But as number of worker increases, efficiency decreases, for both, as expected.

Grp phase efficiency We evaluate the efficiency of `ApprxGrp` by returning mean completion time for 144 tasks.
Varying task mean skill As Fig. 17 demonstrates, `ApprxGrp` outperforms `GrpILP` significantly. As expected, `Cons-k-AG` is more efficient than `ApprxGrp` since it bucketizes 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.

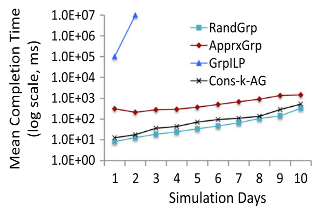

Fig. 18 `Grp`: mean completion time-varying simulation days

9 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.

There has been extensive work in the database community to tackle many of the data management challenges in crowdsourcing [48]. Many key tasks in databases such as data collection, filtering, top-K, skyline and other analytic tasks have been done through crowdsourcing. For an exhaustive list, please refer to [48,49,52].

Crowdsourcing complex tasks This type of human-based computation [2,38,39] handles tasks related to knowledge production, such as article writing, sentence translation, citizen science, and product design. These tasks are conducted in groups, are less decomposable compared to microtasks (such as image tagging) [22,28], 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 [46]. *Turkit* provides an interface to a programmer to use human computation inside their programming model [50] 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 provides an easy-to-use interface for the programmers [1]. *CrowdForge* divides complex task into smaller sub-tasks akin to map-reduce fashion [40]. *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 [43].

The common aspect 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 a complex task which may not be 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 [13].

One prior work investigates how to assign workers to the task of knowledge intensive crowdsourcing [61] and its computational challenges. However, this former work neither investigates 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 [57]. Crowd4u, an academic crowdsourcing platform, effectively integrate our previous work in their framework [26]. The effectiveness of collaborative teams drawn from the online labor marketplaces for solving innovative tasks is studied in [9]. Estimation of human factors (such as skill or expertise) in complex tasks is studied in [55,58]. These works justify our modeling for considering social interaction variables such as affinity and individual human factors. Recently, there has been work on important aspects of collaborative crowdsourcing such as rapidly identifying workers [68]. There are also a number of works on designing effective collaborative crowdsourcing approaches for specific tasks such as prototype design [64], fiction writing [37], creative tasks [36] and so on. [62] studies the problem of ensuring repeated and familiar crowd teams using sophisticated approaches. Note that each of these papers is orthogonal to our efforts and our algorithms can be used to improve the task assignment of any of these works.

Automated team formation Although tangentially related with crowdsourcing, automated team formation is widely studied in computer-assisted cooperative systems. [45] forms a team of experts in social networks with the focus on minimizing coordination cost among team members. Although their coordination cost is akin to our affinity, 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 [4,51], where coordination cost is posed as a constraint. Density-based coordination is introduced in [17], where multiple workers with similar skill are required in a team, such as ours. Formation of team with a leader (moderator) is studied in [30]. Minimizing both communication cost and budget while forming a team is first considered in [31,32]. The concept of pareto optimal groups related to the skyline research is studied in [31].

While several elements of our optimization model are actually adapted from these related work, there are many stark differences that preclude 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 of 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 [12], authors demonstrate empirically that the utility is decreased for larger teams which validate our approach to divide a group into multiple subgroups obeying *upper critical mass*. However, no optimization is proposed to solve the problem. Recently, there has also been interesting work in spatial crowdsourcing and team formation [18,19,67]. [67] uses a microtask scenario where each worker works independently and does not require collaborative team formation. Furthermore, they seek to adopt an algorithm based on maximum weighted bipartite matching. This graph structure cannot handle team centric human factors such as worker–worker affinity. In contrast, our work tries to ensure that task assignment objective function is cognizant of the worker–worker affinity through worker affinity graph to identify cliques that satisfy some minimal distance constraints. [18,19] consider collaborative teams but in a much narrower sense. They are interested in teams such that there exists at least one member who has the necessary skill to cover the desired requirement. In contrast, we use a SUM aggregation function that can handle a broader class of tasks. To see why, if we treat the skill of workers as a boolean variable, then our method naturally subsumes their method. Furthermore, they do not take into account the team dynamics.

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.

10 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, upper critical mass) that are significant for successful completion of collaborative tasks. We propose a set of optimization objectives, which maximize the collaboration while appropriately considering the complex interplay of human factors. We show that our overall problem is NP-complete, and then provide a two-staged solution to our problem. Furthermore, we show that the problem at each individual stage is also NP-Complete. This prompts us to design efficient approximation algorithm for both of the stages. Our extensive experiments on real data collected from Amazon Mechanical Turk show the superiority of our algorithms on their respective baseline counterparts.

In 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. Ahmad, S., Battle, A., Malkani, Z., Kamvar, S.: The Jabberwocky programming environment for structured social computing. In: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, pp. 53–64. ACM (2011)
2. Amer-Yahia, S., Basu Roy, S.: From complex object exploration to complex crowdsourcing. In: Proceedings of the 24th International Conference on World Wide Web, pp. 1531–1532. ACM (2015)
3. Anagnostopoulos, A., Becchetti, L., Castillo, C., Gionis, A., Leonardi, S.: Power in unity: forming teams in large-scale community systems. In: CIKM '10 (2010)
4. Anagnostopoulos, A., Becchetti, L., Castillo, C., Gionis, A., Leonardi, S.: Online team formation in social networks. In: WWW '12 (2012)
5. Andres, H.P.: Team cognition using collaborative technology: a behavioral analysis. J. Manage. Psychol. **28**, 38–54 (2013)
6. Bernstein, M.S., Little, G., Miller, R.C., Hartmann, B., Ackerman, M.S., Karger, D.R., Crowell, D., Panovich, K.: Soylent: a word processor with a crowd inside. In: Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology, pp. 313–322. ACM (2010)
7. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific, Belmont (1999)
8. Bisschop, J.: AIMMS - optimization modeling. Lulu.com (2006). **ISBN:1411698991**
9. Boudreau, K., Gaule, P., Lakhani, K.R., Riedl, C., Woolley, A.W.: From crowds to collaborators: initiating effort and catalyzing interactions among online creative workers (2014)
10. Chai, K., Potdar, V., Dillon, T.: Content quality assessment related frameworks for social media. In: ICCSA(2009)
11. Chen, D.L., Dolan, W.B.: Building a persistent workforce on mechanical turk for multilingual data collection. In: HCOMP (2011)
12. Chhabra, M., Das, S., Szymanski, B.: Team formation in social networks. In: Computer and Information Sciences III, pp. 291–299. Springer (2013)
13. Daniela Retelny, A.T. M.B., Robaszkiewicz, S.: Expert crowdsourcing with flash teams. In: CrowdConf 2013 Poster
14. Difallah, D.E., Catasta, M., Demartini, G., Ipeirotis, P.G., Cudré-Mauroux, P.: The dynamics of micro-task crowdsourcing: the case of Amazon MTurk. In: Proceedings of the 24th International Conference on World Wide Web, pp. 238–247. ACM (2015)
15. Downs, J.S., Holbrook, M.B., Sheng, S., Cranor, L.F.: Are your participants gaming the system? Screening mechanical turk workers. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '10, pp. 2399–2402. ACM, New York, NY (2010). https://doi.org/10.1145/1753326.1753688
16. Duh, K., Hirao, T., Kimura, A., Ishiguro, K., Iwata, T., Yeung, C.-M.A.: Creating stories: Social curation of twitter messages. In: ICWSM (2012)
17. Gajewar, A., Sarma, A.D.: Multi-skill collaborative teams based on densest subgraphs. In: Proceedings of the 2012 SIAM International Conference on Data Mining, pp. 165–176 (2012)
18. Gao, D., Tong, Y., She, J., Song, T., Chen, L., Xu, K.: Top-k team recommendation in spatial crowdsourcing. In: International Conference on Web-Age Information Management, pp. 191–204. Springer (2016)

19. Gao, D., Tong, Y., She, J., Song, T., Chen, L., Xu, K.: Top-k team recommendation and its variants in spatial crowdsourcing. Data Sci. Eng. **2**(2), 136–150 (2017)
20. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York (1979)
21. Grotschel, M., Lovász, L.: Combinatorial optimization. Handb. Comb. **2**, 1541–1597 (1995)
22. Guo, S., Parameswaran, A.G., Garcia-Molina, H.: So who won? Dynamic max discovery with the crowd. In: SIGMOD Conference, pp. 385–396 (2012)
23. Guttmann-Beck, N., Hassin, R.: Approximation algorithms for minimum k-cut. Algorithmica **27**, 198–207 (2000)
24. Hertel, G.: Synergetic effects in working teams. J. Manage. Psychol. **26**, 176–184 (2011)
25. Hffmeier, J., Hertel, G.: When the whole is more than the sum of its parts: group motivation gains in the wild. J. Exp. Soc. Psychol. **47**(2), 455–459 (2011)
26. Ikeda, K., Morishima, A., Rahman, H., Roy, S.B., Thirumuruganathan, S., Amer-Yahia, S., Das, G.: Collaborative crowdsourcing with Crowd4U. Proc. VLDB Endow. **9**(13), 1497–1500 (2016)
27. Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decis. Support Syst. **43**(2), 618–644 (2007)
28. Kaplan, H., Lotosh, I., Milo, T., Novgorodov, S.: Answering planning queries with the crowd. PVLDB **6**(9), 697–708 (2013)
29. Krumke, S., et. al.: Compact location problems. Comput. Sci. (1996)
30. Kargar, M., An, A.: Discovering top-k teams of experts with/without a leader in social networks. In: CIKM '11 (2011)
31. Kargar, M., An, A., Zihayat, M.: Efficient bi-objective team formation in social networks. In: Flach, P., Bie, T., Cristianini, N. (eds.) Machine Learning and Knowledge Discovery in Databases, Lecture Notes in Computer Science, vol. 7524, pp. 483–498. Springer, Berlin (2012)
32. Kargar, M., Zihayat, M., An, A.: Finding affordable and collaborative teams from a network of experts In: Proceedings of the 2013 SIAM International Conference on Data Mining, pp. 587–595 (2013)
33. Karpinski, M.: Approximability of the minimum bisection problem: an algorithmic challenge. In: Mathematical Foundations of Computer Science (2002)
34. Katz, D., Kahn, R.L.: The social psychology of organizations, vol. 2. Wiley, New York (1978)
35. Kenna, R., Berche, B.: Managing research quality: critical mass and optimal academic research group size. IMA J. Manage. Math. **23**, 195–207 (2012)
36. Kim, J., Agrawala, M., Bernstein, M.S.: Mosaic: designing online creative communities for sharing works-in-progress. In: Design Thinking Research, pp. 105–129. Springer (2018)
37. Kim, J., Sterman, S., Cohen, A.A.B., Bernstein, M.S.: Mechanical novel: crowdsourcing complex work through reflection and revision. In: Design Thinking Research, pp. 79–104. Springer (2018)
38. Kittur, A., Kraut, R.E.: Harnessing the wisdom of crowds in Wikipedia: quality through coordination. In: Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work, CSCW '08, pp. 37–46. ACM, New York, NY, USA (2008)
39. Kittur, A., Nickerson, J.V., Bernstein, M., Gerber, E., Shaw, A., Zimmerman, J., Lease, M., Horton, J.: The future of crowd work. In: CSCW '13 (2013)
40. Kittur, A., Smus, B., Khamkar, S., Kraut, R.E.: Crowdforge: Crowdsourcing complex work. In: UIST (2011)
41. Kittur, A., Suh, B., Pendleton, B.A., Chi, E.H.: He says, she says: conflict and coordination in Wikipedia. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 453–462. ACM (2007)
42. Klakow, D., Peters, J.: Testing the correlation of word error rate and perplexity. Speech Commun. **38**(1), 19–28 (2002)
43. Kulkarni, A., Can, M., Hartmann, B.: Collaboratively crowdsourcing workflows with turkomatic. In: Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work, pp. 1003–1012. ACM (2012)
44. Lappas, T., Liu, K., Terzi. E.: Finding a team of experts in social networks. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 467–476 (2009)
45. Lappas, T., Liu, K., Terzi, E.: Finding a team of experts in social networks. In: SIGKDD, pp. 467–476 (2009)
46. Lasecki, W.S., Murray, K.I., White, S., Miller, R.C., Bigham, J.P.: Real-time crowd control of existing interfaces. In: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, UIST '11, pp. 23–32. ACM, New York, NY, USA (2011)
47. Lawler, E.L., Wood, D.E.: Branch-and-bound methods: a survey. Oper. Res. **14**(4), 699–719 (1966)
48. Li, G., Wang, J., Zheng, Y., Franklin, M.J.: Crowdsourced data management: a survey. IEEE Trans. Knowl. Data Eng. **28**(9), 2296–2319 (2016)
49. Li, G., Zheng, Y., Fan, J., Wang, J., Cheng, R.: Crowdsourced data management: overview and challenges. In: Proceedings of the 2017 ACM International Conference on Management of Data, pp. 1711–1716. ACM (2017)
50. Little, G., Chilton, L.B., Goldman, M., Miller, R.C.: Turkit: human computation algorithms on mechanical turk. In: Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology, pp. 57–66. ACM (2010)
51. Majumder, A., Datta, S., Naidu, K.: Capacitated team formation problem on social networks. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, pp. 1005–1013. ACM, New York, NY, USA (2012)
52. Marcus, A., Parameswaran, A., et al.: Crowdsourced data management: industry and academic perspectives. Found. Trends Databases **6**(1–2), 1–161 (2015)
53. Marwell, G., Oliver, P.E., Prahl, R.: Social networks and collective action: a theory of the critical mass. Am. J. Sociol. **94**, 502–534 (1988)
54. Myers, I.B., McCaulley, M.H.: Myers–Briggs Type Indicator: MBTI. Consulting Psychologists Press, Palo Alto (1988)
55. Ojha, P., Talukdar, P.: Quality estimation of workers in collaborative crowdsourcing using group testing. In: Fourth AAAI Conference on Human Computation and Crowdsourcing (2016)
56. Ravi, S.S., et. al.: Facility dispersion problems: heuristics and special cases. In: WADS (1991)
57. Rahman, H., Roy, S.B., Thirumuruganathan, S., Amer-Yahia, S., Das, G.: Task assignment optimization in collaborative crowdsourcing. In: 2015 IEEE International Conference on Data Mining (ICDM), pp. 949–954. IEEE (2015)
58. Rahman, H., Thirumuruganathan, S., Roy, S.B., Amer-Yahia, S., Das, G.: Worker skill estimation in team-based tasks. Proc. VLDB Endow. **8**(11), 1142–1153 (2015)
59. Rosenkrantz, D.J., Tayi, G.K., Ravi, S.S.: Facility dispersion problems under capacity and cost constraints. J. Comb. Optim. **4**, 7–33 (2000)
60. Roy, S.B., Lykourentzou, I., Thirumuruganathan, S., Amer-Yahia, S., Das, G.: Crowds, not drones: modeling human factors in interactive crowdsourcing. In: DBCrowd (2013)
61. Roy, S.B., Lykourentzou, I., Thirumuruganathan, S., Amer-Yahia, S., Das, G.: Optimization in knowledge-intensive crowdsourcing. CoRR, arXiv:1401.1302 (2014)

62. Salehi, N., McCabe, A., Valentine, M., Bernstein, M.: Huddler: convening stable and familiar crowd teams despite unpredictable availability. ArXiv preprint arXiv:1610.08216 (2016)
63. Schrijver, A.: Theory of Linear and Integer Programming. Wiley, New York (1986)
64. Snehalkumar, G., Chhibber, N., Sehgal, V., Ballav, A., Mullings, C., Nasser, A., Richmond-Fuller, A., Gilbee, A., Gamage, D., Whiting, M., et al.: Prototype tasks: improving crowdsourcing results through rapid, iterative task design. ArXiv preprint arXiv:1707.05645 (2017)
65. Solomon, J., Wash, R.: Critical mass of what? Exploring community growth in wikiprojects. In: ICWSM (2014)
66. Surowiecki, J.: The wisdom of crowds: why the many are smarter than the few and how collective wisdom shapes business. Economies, Societies and Nations (2004)
67. Tong, Y., She, J., Ding, B., Wang, L., Chen, L.: Online mobile micro-task allocation in spatial crowdsourcing. In: 2016 IEEE 32nd International Conference on Data Engineering (ICDE), pp. 49–60. IEEE (2016)
68. Valentine, M.A., Retelny, D., To, A., Rahmati, N., Doshi, T., Bernstein, M.S.: Flash organizations: crowdsourcing complex work by structuring crowds as organizations. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, pp. 3523–3537. ACM (2017)
69. van Dongen, S., Enright, A.J.: Metric distances derived from cosine similarity and Pearson and Spearman correlations. CoRR arXiv:1208.3145 (2012)
70. Yan, R., Gao, M., Pavlick, E., Callison-Burch, C.: Are two heads better than one? Crowdsourced translation via a two-step collaboration of non-professional translators and editors In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), vol 1, pp. 1134–1144 (2014)