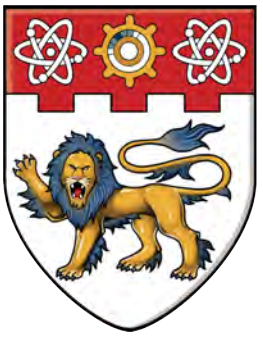


NANYANG TECHNOLOGICAL UNIVERSITY

COMPLEX TASK ALLOCATION FOR CROWDSOURCING IN SOCIAL NETWORK CONTEXT

JIUCHUAN JIANG

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

2019

Complex Task Allocation for Crowdsourcing in Social Network Context

Jiuchuan Jiang

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University

in partial fulfillment of the requirement for the degree of

Doctor of Philosophy

2019

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

11-01-2019
11-Jan-2019

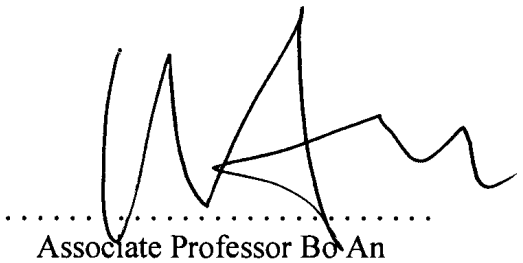
Jiuchuan Jiang
Jiuchuan Jiang

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

16/1/2019

.....
16-Jan-2019



.....
Associate Professor Bo An

Authorship Attribution Statement

This thesis contains material from 3 papers published in the following peer-reviewed journals where I was the first and/or corresponding author.

Chapter 3 is published as J. Jiang, B. An, Y. Jiang, D. Lin. Context-aware Reliable Crowdsourcing in Social Networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, accepted and published online, DOI: 10.1109/TSMC.2017.2777447.
The contributions of the co-authors are as follows:

- I proposed the idea, designed the study, wrote the drafts of the manuscript, and performed the experiments.
- Dr. B. An co-designed the study, and revised the manuscript.
- Dr. Y. Jiang helped building the model, and revised the manuscript.
- Dr. D. Lin helped improving the algorithms in the paper.

Chapter 4 has been accepted as J. Jiang, B. An, Y. Jiang, P. Shi, Z. Bu, J. Cao. Batch Allocation for Tasks with Overlapping Skill Requirements in Crowdsourcing. *IEEE Transactions on Parallel and Distributed Systems*, accepted.
The contributions of the co-authors are as follows:

- I proposed the idea, designed the study, wrote the drafts of the manuscript, and designed the experiments.
- Dr. B. An co-designed the study, and revised the manuscript.
- Dr. Y. Jiang helped building the model, and revised the manuscript.
- Mr. P. Shi helped designing the experiments and conducted the experiments.
- Dr. Z. Bu and J. Cao helped improving the algorithms in the paper.

Some parts of Chapter 2 is published as J. Jiang, B. An, Y. Jiang, D. Lin, Z. Bu, J. Cao, Z. Hao. Understanding Crowdsourcing Systems from a Multiagent Perspective and Approach. *ACM Transactions on Autonomous and Adaptive Systems*, 13(2): Article 8, 2018. DOI: 10.1145/3226028.
The contributions of the co-authors are as follows:

- I proposed the idea, designed the study, and wrote the drafts of the manuscript.
- Dr. B. An helped designing the study, improved the structure, and revised the manuscript drafts.
- Dr. Y. Jiang helped designing the study and revised the manuscript drafts.
- Dr. D. Lin, Z. Bu, J. Cao, and Z. Hao improved the structure of the draft, and gave suggestions on the scope of the study.

The paper of the preliminary result on modelling groups of social networks in Chapter 6 is published as J. Jiang, P. Shi, B. An, J. Yu, C. Wang. Measuring the Social Influences of Scientist Groups Based on Multiple Types of Collaboration Relations. *Information Processing & Management*, 53(1): pp.1-20, 2017. DOI:10.1016/j.ipm.2016.06.003.
The contributions of the co-authors are as follows:

- I proposed the idea, designed the study, and wrote the drafts of the manuscript.
- Mr. P. Shi helped designing the study, and made the experiments.
- Dr. B. An improved the structure and revised the manuscript drafts.
- Dr. J. Yu and C. Wang helped improving the structure and provided guidance in the interpretation of the data.

16-01-2019
16-Jan-2019

Jiuchuan Jiang
Jiuchuan Jiang

Acknowledgements

Reaching the end of my PhD study, I would like to acknowledge my appreciation to those who helped me during the past three years. In these three years study, I was inspired and helped by a lot of people.

First of all I want to express my deepest gratitude to my supervisor Associate Professor Bo An. His careful guidance, constant encouragement and helpful discussions have been the greatest support for me through the entire study. Without his expert guidance and insightful criticisms, the thesis would have not reached the present form. The knowledge and methodology I learnt from him are invaluable to my research, and even to my life. Except for guiding the research, Bo also attended a lot of activities in our daily life. It has too many times that he invited the group members for dinners at the canteens on campus or the the restaurants in the downtown. He even invited us to his home to celebrate with his family on Chinese New Year's Eve. Therefore, Bo has not only been a supervisor in research but also a really nice friend in life.

I would like to thank the Thesis Advisory Committee and Qualifying Examination panel members Prof. Xiaohui Bei, Prof. Zinovi Rabinovich, and Prof. Gao Cong. The meetings and discussions improved me a lot on my research, and build a better vision for me on my PhD study.

My sincere thanks are extended to all the members of our group. All of you gave me a very friendly and scientific environment, which helped me to overcome a lot of difficult times. You are forever friends: Zhen Wang, Jiarui Gan, Qingyu Guo, Haipeng Chen, Yanhai Xiong, Mengchen Zhao, Xinrun Wang, Zhenyi Wang, Ye Zhang, Youzhi Zhang, Wanyuan Wang, Xu He, Lei Feng, Aye Phyu, Shuxin Li, Hongxin Wei, Wei Qiu, Rundong Wang, Yanchen Deng, Jakub Cerny, and Jianfeng Li. I also thank the friends in Computational Intelligence Lab who accompanied my pleasant life these years.

Finally, my great thanks to my beloved family for their unflagging love and support throughout my life. My parents and brother encouraged me a lot when I faced difficulties, and they also share my happiness together. I hope they could enjoy with me this moment. Even my father is in heaven, I am sure he is smiling at me now. I am indebted to my wife Xiao, who gave me so much love and support. Her understanding and support, made me fully put my effort in study abroad these years. Thank my mother-in-law very much for babysitting my little daughter as well. This thesis is also dedicated to my daughter, Niuniu. Although she is just ten months old, she has brought me so much joy.

Contents

List of Figures

List of Tables

Abstract

Allocation of complex tasks has attracted significant attention in crowdsourcing area recently, which can be categorized into decomposition and monolithic allocations. Decomposition allocation means that each complex task will first be decomposed into a flow of simple subtasks and then the subtasks will be allocated to individual workers; monolithic allocation means that each complex task will be allocated as a whole, which includes individual-oriented and team formation-based approaches. However, those existing approaches have some problems for real crowdsourcing markets. On the other hand, workers are often connected through social networks, which can significantly facilitate crowdsourcing of complex tasks. Therefore, this thesis investigates crowdsourcing in social network context and presents models to address the typical problems in complex task allocation. The main contributions of this thesis are shown as follows.

First, traditional decomposition allocation for complex tasks has the following typical problems: 1) decomposing complex tasks into a set of subtasks requires the decomposition capability of the requesters; and 2) reliability may not be ensured when there are many malicious workers in the crowd. To this end, this thesis investigates the context-aware reliable crowdsourcing in social networks. In our approach, when a requester wishes to outsource a task, a worker candidate's self-situation and contextual-situation in the social network are considered. Complex tasks can be performed through autonomous coordination between the assigned worker and his contextual workers in the social network; thus, requesters can be exempt from decomposing complex tasks into subtasks. Moreover, the reliability of a worker is determined not only by the reputation of the worker himself but also by the reputations of the contextual workers, which can effectively address the unreliability of transient or malicious workers.

Second, traditional individual-oriented monolithic allocation for complex tasks often allocate tasks independently, which has the following typical problems: 1) the execution of one task seldom utilize the results of other tasks and the requester must pay in full for the task; and 2) many workers only undertake a very small number of tasks contemporaneously, thus the workers' skills and time may not be fully utilized. To this end, this thesis investigates the batch allocation for tasks with overlapping skill requirements. Then, two approaches are designed: layered batch allocation and core-based batch allocation. The former approach utilizes the hierarchy pattern to form all possible batches, which can achieve better performance but may require higher computational cost; the latter approach selects core tasks to form batches, which can achieve suboptimal performance with significantly reducing computational cost. If the assigned worker cannot complete a batch of tasks alone, he/she will cooperate with the contextual workers in the social network. Through the batch allocation, requesters' real payment can be discounted because the real execution cost of tasks can be reduced, and each worker's real earnings may increase because he/she can undertake more tasks contemporaneously.

Third, traditional team formation-based monolithic allocation for complex tasks has the following typical problems: 1) each team is created for only one task, which may be costly and cannot accommodate crowdsourcing markets with a large number of tasks; and 2) most existing studies form teams in a centralized manner, which may place a heavy burden on requesters. To this end, this thesis investigates the distributed team formation for a batch of tasks, in which similar tasks can be addressed in a batch to reduce computational costs and workers can self-organize through their social networks to form teams. In the presented team formation model, the requester only needs to select the first initiator worker and other team members are selected in a distributed manner, which avoids imposing all team formation computation loads on the requester. Then, two heuristic approaches are designed: one is to form a fixed team for all tasks in the batch, which has lower computational complexity; the other is to form a basic team that can be dynamically adjusted for each task in the batch, which performs better in reducing the total payments by requesters.

Forth, current workers are often naturally organized into groups through social networks. To address such common problem, this thesis investigates a new group-oriented

crowdsourcing paradigm in which the task allocation targets are naturally existing worker groups but not individual workers or artificially-formed teams as before. An assigned group often needs to coordinate with other groups in the social network contexts for performing a complex task since such natural group might not possess all of the required skills to complete the task. Therefore, a concept of contextual crowdsourcing value is presented to measure a group's capacity to complete a task by coordinating with its contextual groups, which determines the probability that the group is assigned the task; then the task allocation algorithms, including the allocations of groups and the workers actually participating in executing the task, are designed.

In summary, this thesis develops new models to cover the shortages of previous complex task allocation works and designs efficient algorithms to solve the corresponding problems by considering the social network contexts. Experimental results conducted on real-world datasets collected from some representative crowdsourcing platforms show that the presented approaches outperform existing benchmark approaches in previous studies.

Chapter 1

Introduction

Crowdsourcing is a task allocation paradigm in which a requester allocates tasks to a group of workers chosen from a population [1, 2]. Crowdsourcing is often suitable for tasks that are trivial for humans but difficult for computers [2, 3], which include simple tasks such as annotation [4] and classification tasks [5], and complex tasks such as software development [6] and product design [7]. Compared with traditional task allocation, the advantages of crowdsourcing include faster completion speed [8], lower costs [9], higher accuracy [10], and completion of tasks that computers cannot perform [11]. There are many crowdsourcing platforms have been developed, such as Amazon's Mechanical Turk [12], UPwork [13], and MicroWorkERs [14].

Despite the rapid development of crowdsourcing, it is usually challenging to allocate complex tasks in real crowdsourcing markets [15–18]. As more and more people are using crowdsourcing, the tasks released are often with complex characters, e.g., requiring different skills; and such complex tasks are too complex to be completed by a single worker. Existing studies on complex task allocation in crowdsourcing can be categorized into two categories: decomposition allocation [19, 20] and monolithic allocation [2, 21]. Decomposition allocation means that each complex task will first be decomposed into a flow of simple subtasks and then the decomposed subtasks will be allocated to individual workers; monolithic allocation means that each complex task will be allocated as a whole to an individual worker or a formed team of workers. However, we find that those approaches have certain problems, such as decomposing complex

tasks bring heavy burden to requesters and workers' skills as well as time may not be fully utilized, so they sometimes cannot adapt to the real crowdsourcing markets in which tasks and workers are numerous and dynamic.

Nowadays, with the significant development of social networks [22–24], the crowd of workers is often connected by social networks [25–29] and many related studies have harnessed the crowdsourcing power of social media [30, 31]. For example, social networks such as Facebook and Twitter can also be considered as crowd providers [32]. Specifically, the crowdsourcing power of social networks has been used for disaster relief [30]. Generally, the advantages of social networks for crowdsourcing include at least the following parts: 1) workers within a social network are more inclined to cooperate with each other to execute a complex task [23]; 2) it is easier to find professional and helpful workers through social networks because real-world social networks include many professional groups [33], and many users of social networks undertake outsourced tasks not only for monetary return but also due to interest or willingness to share with their friends [27]; and 3) social networks can broadcast outsourced tasks more rapidly, thereby hastening the completion of tasks [22, 30].

Therefore, by considering social network factors, this thesis is committed to solve the limitations of the related work on complex task allocation in crowdsourcing by proposing new approaches and models. First, in order to solve the problem that single worker cannot complete the complex task reliably, this thesis builds a context-aware reliable task allocation model and uses reputations to ensure reliability. Second, this thesis builds a new batch allocation model to solve the limitations of the retail-style task allocation where each task is executed independently and the requesters must pay in full, and workers' skills may not be fully utilized. Third, this thesis proposes distributed team formation mechanisms to solve the limitations of existing team formation studies in which each team is created for only one task and teams are formed in a centralized manner by requesters. Fourth, this thesis builds a group-oriented task allocation model to solve the limitations of the existing team formation work where workers are not sufficiently cooperative, reliable and effective. The four pieces of work are shown as follows.

1.1 Context-Aware Reliable Crowdsourcing

Generally, the micro-tasks-oriented crowdsourcing platforms have two typical limitations: complex tasks cannot be executed directly, and the workers may be transient and unreliable. Existing studies often addressed the two limitations as follows: complex tasks are decomposed into a set of micro-subtasks that can be solved in multiple phases [2], and reliability is achieved by assigning each task redundantly to multiple workers [34, 35]. However, they have the following problems:

1) In existing studies of crowdsourcing complex tasks in which tasks are decomposed to a set of interdependent micro-subtasks, the requesters must determine the optimal task decomposition. Thus, some requesters may abandon using crowdsourcing to accomplish a large number of complex tasks since they cannot bear such heavy burden by themselves. Moreover, the offline decomposition of tasks may not match the real-time situation of workers because the available workers may be dynamically changed at some crowdsourcing platforms [36].

2) In existing studies of achieving reliability by redundantly assigning each task to multiple workers, reliability may not be ensured when there are many malicious workers in the crowd [37–39]. Although some studies introduced the reputation mechanism to cope with malicious workers [40], reputation may be undeterminable in traditional crowdsourcing platforms due to the transient characteristics of workers.

Nowadays, with the significant development of social networks, the crowd of workers is often connected by social networks [25–29]. With the social network, crowds could undertake the task that is too hard to a single worker. Consider the following motivating scenario:

[Scenario 1.1: A crowdsourcing example utilizing contextual resources]*In the Lushan earthquake that occurred on April 20, 2013 in China, many donated supplies such as tents and foods were aggregated in nearby cities, but it was difficult to deliver these supplies to the rural villages. Therefore, the charitable organization called upon*

volunteers to help deliver supplies at some Chinese social media platforms such as Weibo and Renren. Usually, more than one person is required to deliver a block of tents; the required group includes one driver with a truck, a local guide, and several porters. Thus, if a volunteer wished to undertake a delivery task, he/she would find other partners in social networks and they would make the delivery together. Let there be a truck driver who offers, through a social media platform, to deliver and who is approved by the charitable organization, he/she will then seek other trusted partners, including a local guide and several porters, through his/her social network. After the truck driver finds the necessary partners, they can cooperate to complete the delivering task. Finally, the truck driver and his/her partners achieve certain reputations through completing the task, and they can apply for new delivery tasks by virtue of their reputations.

Motivated by this example, it is natural to think using social networks to form a team and then undertake the complex task. In fact, there are now some related studies that are often implemented by team formation in which a team of workers that can perform outsourced tasks is found through social networks [41–43]. However, the requesters must undertake heavy computing loads for selecting appropriate team members [41, 42]. Moreover, these related studies often assume that the team members are reliable, which may sometimes not conform to the actual situation [43].

To address the two critical problems of traditional crowdsourcing systems that are not solved by existing studies on crowdsourcing in social networks, this thesis presents an approach for context-aware reliable crowdsourcing in social networks. The context of a worker in the social network can be defined as the counterparts that interact with this worker through the social network [44]. In our approach, when a requester wishes to outsource a task, a worker candidate's self-situation and his/her contextual-situation in the social network are considered.

With our approach, the two problems in existing studies are addressed in social network environment as follows.

1) The requester only needs to assign a task directly to workers without decomposing the tasks; the assigned workers will then start to execute the complex tasks by autonomously coordinating with other workers in the social network context.

Therefore, the requester avoids heavy computing load when decomposing complex tasks, thereby enabling more requesters to accomplish a large number of complex tasks through crowdsourcing. Moreover, our approach is implemented by the autonomous coordination among workers, which offers an appropriate fit to real situation in which the set of available workers may be dynamically changed.

2) The reliability of a worker is determined not only by the reputation of the worker himself/herself but also by the reputations of his/her contextual workers. Even when a worker comes to the system for the first time, his/her initial reputation can be measured through the past experiences of his/her contextual workers. Therefore, reliability of the transient workers can be achieved.

Our context-aware crowdsourcing approach involves assigning a principal worker who will then recruit other assistant workers from the social network, which should satisfy the following general objectives: the communication cost and the reservation wages are minimized and the total reputations are maximized. This optimization problem is proved to be NP-hard in Chapter 3; thus, a heuristic approach that can be realized in reasonable time complexity is then presented. In the approach, a concept of crowdsourcing value is defined to measure the probability of a worker being assigned a task when the context of the worker in the social network is considered. In this way, workers with higher crowdsourcing values can be preferentially selected to perform the task. Experimental results show that the presented approach can achieve higher task allocation and execution efficiency than the previous benchmark approaches and can achieve higher reliability by adopting the contextual reputation mechanism.

1.2 Batch Allocation for Tasks with Overlapping Skill Requirements

In previous work on crowdsourcing, the task allocation is categorized into two types [45]: for simple tasks that are atomic computation operations, tasks are directly allocated to proper individual workers [46, 47]; for complex tasks involving many computational operations, tasks may either be decomposed into micro-subtasks allocated to individual workers [2, 48] or be directly allocated to a team of workers through team formation [21, 49].

In existing studies, the allocations of different tasks are independent from one another, regardless of whether the tasks are simple or complex. In other words, previous task allocation is similar to the retail business in markets, in which tasks are allocated individually and independently. Next, a real-world example of this approach is presented.

At the leading crowdsourcing website, www.upwork.com, there are two web development tasks that are posted at the same time: developing a B2C website and developing an O2O website. With existing task allocation methods, the two similar tasks will be dependently allocated to different workers. Obviously, such a solution may not be sufficient, since the two tasks have many similarities and may be mutually beneficial, e.g., many of the infrastructures and basic components of B2C and O2O websites are the same or similar. In fact, if the two tasks are allocated to the same worker, significant development cost may be saved because the development of two similar websites can be combined and experiences that are learned from developing one website can be applied when developing the other.

In general, there are some drawbacks with the retail-style task allocation in traditional crowdsourcing, which can be summarized as follows:

1) *Retail-style task allocation cannot scale to large-scale concurrent tasks, because each task needs to be allocated independently from scratch, which may involve repeatedly solving for the optimal matching between tasks and workers.* In fact,

it is common for some tasks on a crowdsourcing website to have similar sets of required skills and for each type of skill to be required by more than one task. For example, we find that each skill is required by at least 31.36 tasks on average at the Freelancer website and by at least 13.58 tasks on average at the Upwork website. Inspired by this observation, it may be possible to integrate the skill-overlapping tasks and allocate them in batches, which can save allocation cost and be scaled to large-scale concurrent tasks.

2) *Retail-style task allocation requires each requester to pay in full for his/her task because each task will be completed independently from scratch.* In fact, many tasks within the same category at a crowdsourcing website may be similar. For example, by analyzing 4950 tasks that were randomly collected from the category of web and mobile development at the Upwork website, we find that 769 tasks involve WordPress, 731 tasks involve PHP, and 683 tasks involve Website Development. It may be possible to integrate similar tasks and let them be performed by the same workers, which can allow the partial execution results of one task to be reused by another similar task. This approach can reduce the execution cost of tasks so that the system can discount the real payment that is required from requesters.

3) *Retail-style task allocation may not fully utilize workers' skills because many workers undertake no more than one task contemporaneously.* For example, we find that the period during which workers contemporaneously undertake no more than one task occupies 61.58% of the workers' total duration at the Upwork website and 80.16% at the Freelancer website. In fact, one task often uses only a fraction of a worker's resources. On the other hand, many crowdsourcing tasks have long deadlines; for example, by analyzing 7395 tasks that were randomly collected from the Upwork website, we find that the average completion time is 97.76 days. It is thus possible to allocate more than one task to the same worker. Moreover, when the system allocates a batch of tasks to a worker, the system can discount the real payment for each task so that the requesters pay less; when a worker can undertake more than one task contemporaneously, the worker can receive higher hourly payment compared with the retail-style task allocation.

To address the above problems, this thesis presents a novel batch allocation approach for tasks with overlapping skill requirements. Our approach is inspired by the concept of wholesaling which denotes that batched goods are allocated to individuals with the goal of lowering cost [50]. First, we integrate similar tasks into a batch according to their skill requirements; then, the integrated tasks will be allocated in batch to some workers. Next, we describe an example scenario to demonstrate our idea:

[Scenario 1.2: An example of batched tasks in crowdsourcing]*Currently, at www.upwork.com, there are three real tasks with long deadlines: 1) task 1, which requires skills 'Java, HTML, JavaScript, PHP' and has a budget of $50; 2) task 2, which requires skills 'Java, Web design, website development' and has a budget of $100; and 3) task 3, which requires skill 'Java' and has a budget of $50. The system can integrate the three similar tasks and determine that the overlapping skill is 'Java'; then, the system will allocate the three tasks in batch to a worker with excellent Java programming skills (such as the highest-rated Java developer at www.upwork.com). Moreover, the system may discount each requester's real payment because the three tasks are allocated in batch to the same worker and, thereby, the execution may be delayed and the execution cost may be reduced; on the other hand, the assigned worker can also earn more hourly wages by undertaking three similar tasks contemporaneously.*

If the workers assigned by the requesters can satisfy all skill requirements of the tasks, they will perform the tasks by themselves. However, if the workers cannot complete the tasks in the batch by themselves, they will coordinate with other workers in their social network to execute the tasks but not turn to the requesters, the reason is that the personal characteristics of workers are often invisible to the requesters [27] and the requesters may not have enough expertise to justify whether a worker has qualified professional skills to complete the tasks effectively; moreover, the requesters cannot bear the heavy burden of coordinating workers [51]. On the other hand, if the worker cannot complete the task by himself/herself and does not collaborate with others to complete the task, his/her reputation will be reduced.

Batch allocation has the following advantages: 1) batch allocation can save on task allocation time, which allow the system to scale to accommodate large-scale concurrent tasks; 2) the crowdsourcing systems discount the real payment for batched tasks so the

requesters pay less; and 3) the workers can receive higher hourly payment than with the previous retail-style task allocation.

The problem of achieving optimal batch allocation is NP-hard. To solve such an NP-hard problem, at first we present a layered heuristic approach, which utilizes the hierarchy pattern to form all possible batches and then performs batch allocation by considering the real situations of batches and available workers. This approach can achieve good performance but has high computational cost since all possible batches must be formed and observed. Then, we propose another core-based heuristic batch allocation approach, which selects core tasks to form suboptimal batches with lower computational cost.

1.3 Distributed Team Formation for a Batch of Tasks

In general, existing related studies on team formation in crowdsourcing typically have the following two characteristics. First, they are individual task-specific, i.e., a new team must be formed from scratch each time when a complex task is published at the crowdsourcing platform; thus, this method may be costly and cannot adapt to general crowdsourcing markets with a large number of tasks. Second, they are requester-focused, i.e., team formation is often implemented in a centralized manner by the requester. Although a few studies [52] presented a self-organized strategy where the hired workers can select the teammates themselves, the requesters must undertake a large amount of work around the hiring and training of the initial candidate workers. Such a requester-focused team formation approach may be extremely burdensome to the requesters, who are required to have information on all candidate workers to make good decision; moreover, the requesters sometimes may not have enough expertise to justify whether a bidding worker has qualified professional skills.

Our solution to address the above drawbacks in team formation is motivated by the following observations in popular crowdsourcing systems.

1) The skill requirements of many tasks in the same category at a crowdsourcing website are often similar. This phenomenon is common at many crowdsourcing websites, which is shown in Section 1.2. It would be useful to integrate these tasks with similar skill requirements into a batch to save on computational costs.

2) Workers are often connected through social networks and cooperate with each other. Some recent notable studies [26, 27] have shown that workers often communicate via phone, forums, chat, Facebook, or in person to share information about tasks and requesters and workers can self-report their social connections to other workers. Moreover, the workers within a social network are often cooperative [53, 54]. Therefore, it is possible that workers can self-organize team formation through the social networks. Moreover, this thesis attempts to explore a distributed approach different from the study in [52]: the requester needs not undertake hiring and training of the candidate workers but only needs to assign the worker for initializing the team formation for a batch of tasks.

Being inspired by the first observation, it is possible to form a versatile team for a batch of tasks with similar skill requirements, which can both reduce the team formation cost and make team members contemporaneously undertake more than one task to reduce real execution cost of tasks since the partial execution results of one task can be reused by another similar task undertaken by the same worker. Being inspired by the second observation, it is possible that workers can self-organize the team formation.

Therefore, this thesis investigates distributed team formation for a batch of tasks in crowdsourcing. In general, the cost of team formation-based crowdsourcing in existing benchmark studies [43, 55–57] includes computational cost of forming teams, payments by requesters, and cost of communication among team members; thus, we set weights to make trade-off among them to calculate the total cost. Minimizing the total cost of team formation-based crowdsourcing for a batch of tasks in a social network is proven to be an NP-hard in the thesis.

To solve such an NP-hard problem, at first we present an approach to form a fixed team for all tasks in the batch. This approach has lower computational complexity, but requesters need to pay more because each team member should be paid even such

member does not execute a task in the batch. Therefore, to reduce the total payments by requesters, we then present another approach to form a basic team that can be dynamically adjusted for each task in the batch. In both approaches, each team member's factual income can be improved although the total payments of requesters are reduced, because each team member undertakes more tasks contemporaneously.

In our approaches, first, a worker is selected by the requester to initialize the team formation process to select another qualified worker to join the team. The existing team members can select other qualified workers to join the team step by step until a suitable team for the batch of tasks is formed. Therefore, except for the initiator, the team members are selected in a distributed manner. The requester only needs to select the first initiator worker, which avoids imposing all team formation computation loads on the requester.

1.4 Group-Oriented Task Allocation for the Crowdsourcing

As stated in Section 1.3, team information has been used for crowdsourcing in which the requester seeks a team of workers who can cooperate to perform a complex task that requires various skills. The following scenario is a typical team formation case:

[Scenario 1.3: Team formation-based crowdsourcing] *At www.upwork.com, there is a real task 'C#.NET Developer to Teach Coding Course' that requires the skills of 'Asynchronous I/O', 'C#', 'CSS3', 'Genetic Algorithms', 'HTML5', and 'JavaScript'. Because the set of required skills of the task is large, it is difficult to find an appropriate worker who can satisfy all of the skill requirements. The requester must recruit a team of workers to perform such a complex task. Now, three workers might be recruited by the requester to form a team to perform the task: Alexander, who has the skills of 'JavaScript', 'CSS3', and 'HTML5', Jay, who has the skills of 'Asynchronous I/O' and 'C#', and Ivan, who has the skill of 'Genetic Algorithms'.*

Although the formed team can satisfy the skill requirements of the task, the above solution might have the following limitations. *1) Alexander, Jay and Ivan have no cooperation experience in the past, so they might not cooperate smoothly in executing the task, e.g., the code of Alexander might be not compatible with that of Jay. 2) The team of Alexander, Jay and Ivan is only hired by the requester of task 'C#.NET Developer to Teach Coding Course'. Now, there is another task 'C#.NET Developer to Teach Senior Level Coding Course' that requires additional skill of 'PHP', a new team needs to be formed from scratch if we use the previous team formation-based crowdsourcing approach; clearly, this method will waste great computational costs of team formation when there are many similar tasks.*

From the above case, we can generalize the typical limitations of existing team formation-based crowdsourcing studies, shown as follows.

1) A *team is formed artificially and might be not sufficiently reliable*. The reason is that the team members are called transiently and have no cooperation experience in the past, so they might not cooperate smoothly and effectively for executing the assigned task. Moreover, workers in a team can come from different social organizations and might even be non-cooperative in performing the tasks [58].

2) A *team is tailored only for a special task*; thus, it is not effective for other tasks. Each time a complex task is published, a new team should be formed from scratch; thus, such a mechanism will bring heavy task allocation costs in crowdsourcing markets in which tasks are numerous and dynamic [44, 59].

To address the above limitations of crowdsourcing based on artificial team formation, this thesis explores a new crowdsourcing paradigm that can utilize the natural organization form of people in social networks. In reality, people are often naturally organized into groups through social connections [23, 33, 60, 61]. For example, many natural groups exist in which people are grouped according to gender, affiliations, research interests, or other common factors [22, 62, 63]; mobile device users within an area are grouped as a mobile cloud to provide pervasive crowdsourcing services [29]. The natural group is different from the artificially-formed team because the former does

naturally exist in social networked crowds, but the latter is formed artificially and transiently for a special task. The phenomenon of people forming groups is common in real crowdsourcing systems. For example, at www.upwork.com, we find that the workers affiliated with any groups (i.e., agency freelancers) can constitute 47% of the total workers; at the Github website, there were 226449 groups registered from January 1, 2016 to June 30, 2016.

Therefore, this thesis presents a novel group-oriented crowdsourcing paradigm, in which the task allocation targets are naturally existing worker groups but not artificial teams or individual workers. With this novel crowdsourcing paradigm, the two typical limitations of team formation can be directly addressed. *1) Because the workers within the same group have common characteristics and often have rich cooperation experience, it is more probable that the workers can cooperate smoothly and effectively in performing a new task. 2) Groups are organized naturally and can be assigned varying tasks; thus, groups can better fit the crowdsourcing markets with varying tasks than can teams tailored for specific tasks.*

The following motivating example is a typical group-oriented crowdsourcing case:

[Scenario 1.4: Natural group-oriented crowdsourcing] *At https://github.com, the workers are grouped according to their participating open source projects. Now there is a real project 'Iaravel- A PHP Framework for Web Artisans' involves a group of workers with the professional skills of "Java, JavaScript, and PHP". Now, there is another task 'design a responsive website' that requires the set of skills 'Java, JavaScript, PHP, and HTML5'. Because most skills of the task can be satisfied by the group, the requester can directly allocate the task to the group. Thus, the group is assigned varying tasks, which can utilize the group members' cooperation experiences to perform the new task and can save much task allocation costs.*

However, because groups are organized naturally, an allocated group might occasionally not possess the complete skills of the task. Moreover, some recent studies have shown that workers are often connected through social networks [26, 27]. Thus, the workers of one group can coordinate with other workers of the contextual groups for assistance [64], in which the context of a group primarily means the counterpart groups

interacting with this group through the social network. For example, in Scenario 1.4, the group lacks HTML5 skill required by the new task; thus, the group needs to coordinate with another contextual group with HTML5 skill.

Therefore, this thesis presents a context-aware approach for group-oriented task allocation in crowdsourcing, in which a group candidate's self-situation and its contextual-situation in the social network are considered when the requester wishes to assign a task to such group. Generally, the main challenges in this context-aware approach include: 1) how to measure the probability of a group being allocated a task by considering its contextual groups; and 2) how an assigned group coordinates with other contextual groups for seeking assistance in performing the task.

To solve the main challenges stated above, at first we present a metric of contextual crowdsourcing value of a group to measure the group's capacity to perform a task by coordinating with its contextual groups, which determines the probability that the groups is assigned the task; we then present a method to model the coordination among groups for performing the task. Finally, we present the task allocation algorithms that include: 1) assigning the task to a principal group according to the candidate groups' contextual crowdsourcing values; 2) allocation of assistant groups if the principal group cannot complete the task along by itself; and 3) selection of workers actually participating in executing task from the principal and assistant groups.

1.5 Research Contributions

Our major contributions in this thesis are summarized as follows:

- We propose an approach for context-aware reliable crowdsourcing in which requesters avoid decomposing complex tasks into micro-subtasks, and the reliability of a worker is determined not only by the reputation of the worker himself/herself but also by the reputations of his/her contextual workers. Thus, the two critical problems of traditional crowdsourcing systems are solved by the presented approach. We first prove the research problem is NP-hard. Then, we define a concept of crowdsourcing value and propose a heuristic approach to

solve the research problem in reasonable time complexity. Theoretical analyses and experiments on a real-world dataset testify that the presented approach can achieve significantly higher task allocation and execution efficiency than the previous benchmark task allocation approaches, and the presented contextual reputation mechanism can achieve relatively higher reliability when there are many malicious workers in the crowd.

- We propose a batch allocation approach in which similar tasks can be integrated and allocated in a batch, which can make requesters pay less and workers receive higher payment than can the previous retail-style task allocation. We first prove the problem of achieving optimal batch allocation is NP-hard. Then, we present two approaches to solve the problem. The first one is called layered heuristic approach which utilizes the hierarchy pattern to form all possible batches and then performs batch allocation by considering the real situations of batches and available workers. The layered heuristic approach can achieve good performance but has high computational cost, so we propose another core-based heuristic batch allocation approach which selects core tasks to form suboptimal batches with lower computational cost. The theoretical analyses and experiments testify that the batch allocation achieves better performances comparing with the previous benchmark methods.

- We propose a distributed team formation approach for a batch of tasks in crowdsourcing. The proposed approach can solve the two typical drawbacks of existing team formation studies: each team is created for only one task, which may be costly and cannot accommodate crowdsourcing markets with a large number of tasks; and most existing studies form team in a centralized manner by the requesters, which may place a heavy burden on requesters. In the approach, we set weights to make trade-off among the different costs to calculate the total cost of forming a team, and we prove the problem of minimizing the total cost is NP-hard. Then, we propose two approaches to solve the problem. One is to form a fixed team for all tasks in the batch, and the other is to form a basic team that can be dynamically adjusted for each task in the batch. In comparison, the former approach has lower computational complexity but the latter approach performs better in reducing the

total payments by requesters. Experiments on a real-world dataset show that our approaches outperform the previous benchmark approaches.

- We propose a group-oriented task allocation approach to address the previous individual-oriented or team formation-based crowdsourcing studies which do not consider the situation that workers are often naturally organized into groups through social networks. With the proposed approach, workers would be more cooperative and reliable to execute the assigned task, and a group could be better adapted to the crowdsourcing markets in which tasks vary significantly. At first we present a metric of contextual crowdsourcing value of a group and use it to measure the probability of a group being allocated a task. Then, we present a method to model the coordination among groups for performing the task. The experiments conducted on a real-world dataset show that our group-oriented approach outperform the previous benchmark individual-oriented and team formation approaches in several metrics.

1.6 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 reviews the related works to provide context of this thesis. Chapter 3 explores a context-aware reliable task allocation model for crowdsourcing. Chapter 4 presents a batch allocation method to solve the drawbacks of traditional retail-style task allocation. Chapter 5 presents a distributed team formation approach for a batch of tasks to solve the limitations of previous team formation studies. Chapter 6 explores a new group-oriented crowdsourcing paradigm to solve the limitations of existing team formation studies. Chapter 7 summarizes the thesis and presents possible directions for future work.

Chapter 2

Related Work

In this chapter, we review existing research that is relevant to this thesis. First. we will introduce the two types of studies attempting to address the above problems for crowdsourcing complex tasks: decomposition allocation-based crowdsourcing and monolithic allocation-based crowdsourcing. In particular, we categorize the monolithic allocation-based crowdsourcing into individual-oriented crowdsourcing and team formation-based crowdsourcing. Then, we will introduce the current works on reliable crowdsourcing. Next, the works on crowdsourcing in social networks will be introduced. Finally, we will introduce the works on context-aware crowdsourcing and some other task allocation approaches.

2.1 Decomposition Allocation of Complex Tasks

Traditional crowdsourcing platforms, such as Amazon's Mechanical Turk, are often oriented to micro-task markets. A popular method of performing a complex task is to decompose the task into a flow of simple subtasks and then combine the partial results of the subtasks to obtain the final answer [2, 20]. This approach is mainly used in the following two situations: in crowdsourcing systems that are oriented to micro-task markets, such as Amazon's Mechanical Turk [65, 66], and when the workers are non-professional and can only complete simple or micro-tasks [67, 68]. In general, decomposing a complex task involves the following aspects: the task structure, which is

the subtask decomposition for the given complex task [69, 70]; dependencies, which are the constraints among the subtasks [71, 72]; and the workflow, which is the control flow among subtasks [73, 74].

An efficient task decomposition method is necessary for the crowdsourcing of complex tasks [19]. Tran-Thanh et al. [2] proposed the first crowdsourcing algorithm, BudgetFix, to solve complex tasks that involve various types of interdependent micro-tasks structured into complex workflows. Then, in their succeeding work [48], Tran-Thanh et al. further investigated the problem of multiple complex workflows, and they proposed an algorithm, Budgeteer, to determine the number of interdependent micro-tasks and the price to pay for each task within each workflow. Bernstein et al. [75] used Mechanical Turk to present a word processing interface that can be used for proofreading and editing documents, which decomposed a complex task into three stages: Find, in which workers identify patches of the requester's task that need more attention; Fix, in which other workers revise the patches that were identified in the previous stage; and Verify, in which newly allocated workers vote on the best answers from the Fix stage and perform quality control on the revisions. Moreover, Dai et al. [76] used Bayesian network learning and Partially-Observable Markov Decision Processes (POMDP) to make dynamic control for workflow optimization.

Overall, the studies described above may place heavy computing loads on requesters, rendering them inappropriate when the number of complex tasks is large [45].

2.2 Monolithic Allocation of Complex Tasks

Monolithic allocation means that each complex task will be allocated as a whole to an individual worker or a team of workers and does not require the requesters to decompose complex tasks.

2.2.1 Individual-Oriented Allocation

Most previous studies adopted an individual-oriented crowdsourcing approach; thus, each worker can complete the allocated task individually and independently. Generally, there are two typical types of tasks in crowdsourcing; one is *micro-task*, and the other is *complex task*.

The micro-tasks are atomic computation operations and can be completed in minutes by non-professional individual workers [77, 78]. In fact, many traditional crowdsourcing systems, such as Amazon's Mechanical Turk, are often designed for micro-task markets [79]. In the crowdsourcing of micro-tasks, each task is redundantly allocated to more than one individual worker to improve accuracy, and each worker executes the task independently; finally, the requester will select the correct result from the multiple answers from the redundant allocated individual workers [77].

On some crowdsourcing websites for complex tasks, such as www.upwork.com, complex tasks are often allocated to professional workers [78, 80] by considering the following three factors: matching degree between the task's required skills and the worker's skills [81–83], the reputation (or experience) of the worker [84–86], and the reservation wage of the worker [87–89]. Complex tasks are often allocated non-redundantly (e.g., in Section 6.3.1 Chapter 6, by randomly counting 6271 tasks in Upwork website, we find that 79.3% of tasks are allocated non-redundantly). Moreover, sometimes the requesters may interview the candidate workers using instant messaging software tools to determine whether the workers can complete the tasks [90, 91].

Many workers are connected by social networks [22, 92]; therefore, Bozzon et al. [41] presented an approach based on finding the most knowledgeable people in social networks to address the task. Moreover, if a complex task is allocated to a worker who cannot complete the task by himself, the assigned worker needs to forward the complex task to another worker with the necessary skills. Heidari and Kearns [93] performed a study on designing efficient forwarding structures from a worker to another worker.

In comparison, our study in this thesis not only allocates an entire complex task to a worker but also integrates some similar complex tasks with overlapping skill requirements, which can reduce requesters' real payment and make use of the time of

workers better than previous studies since the assigned workers in our approach can undertake more tasks contemporaneously. Besides, the approach in Chapter 6 directly allocates the complex task to a group, and the workers in the allocated group cooperate to complete the complex task.

2.2.2 Team Formation-Based Crowdsourcing

Team formation is a general concept in many areas [49, 94–96], which mainly cares about the resource/skill requirements of tasks. In many existing studies, the team formation of workers is centrally controlled by the requester, in which interested candidate workers advertise their skills and bid for participation on the team. Liu et al. [43] presented an efficient method that is implemented through profitable and truthful pricing mechanisms. Kargar et al. [57] presented a team formation method to satisfy the two objectives in social networks: finding a team of experts that covers all the required skills for the tasks and minimizing the communication cost between workers in the team. Fathian et al. [94] integrated the collaboration network and reliability of people and presented an optimization model for the formation of a reliable team.

A small number of studies on self-organized team formation exist in which some workers organize a team to bid for the task [97–99]. For example, Lykourentzou et al. [52] presented a self-organized team formation strategy where the hired workers can select the teammates themselves. Rokicki et al. [21] explored a strategy for team formation in which workers themselves decide on which team they want to participate. However, in these related studies on self-organized team formation, the requester still must undertake hiring and training of the candidate workers and a team is formed for only one task; in comparison, the requester in our study only needs to assign the worker for initializing the team formation for a batch of tasks.

Another related area is the open source software development, in which either a centralized mechanism or a self-organized mechanism is used. In the centralized mechanism, there is a manager who assigns individuals to a team; in the self-organized mechanism, developers may join a project at their own discretion while a project is created by an initiator. Nan and Kumar [100] found that the impact of centralized teams on the

project performance for varying software project structural interdependencies. Hahn et al. [101] investigated how individuals make decisions about which teams to join in the open source software development and explored how the collaborative network affects developers' choice of new teams to participate in.

In summary, previous studies on team formation are oriented to individual tasks, i.e., each team is tailored for a special task but is not efficient in performing other tasks. Although a few studies in engineering management [102] explored the multifunctional team that may be reused for more than one task, they often implemented the team formation in a centralized manner and did not focus on how to address a batch of tasks. In comparison, the work in Chapter 4 focuses on the tasks instead of the workers and provides approaches for batch formation of tasks; in Chapter 5 this thesis studies the distributed team formation for a batch of tasks, which aims to reduce the cost in team formation; the groups in Chapter 6 are organized naturally and can be assigned varying tasks. Note that Chamberlain [25] presented the concept of groupsourcing in which the task is allocated to a group of people of varying expertise connected through a social network. However, the task allocation mechanism to groups and the task coordination among groups were not systematically investigated.

2.3 Reliable Crowdsourcing

Workers may be transient and unreliable [37, 103–105]. Therefore, it is necessary to ensure reliability in crowdsourcing to make the workers really work on the task [38, 106–108].

A typical solution is redundantly assigning each task to more than one worker and combing the answers by some measures such as majority voting [34, 109]. For example, Karger et al. [34] presented an algorithm for deciding which tasks to assign to which workers and for inferring correct answers from the workers' answers. However, this type of approach may be infeasible when many malicious workers exist; moreover, redundant allocation of complex tasks may be too expensive.

Another typical solution for reliable crowdsourcing is using a trust and reputation mechanism. Ren et al. [28] integrated the social relationship and reputation management into mobile crowdsourcing and proposed a Social Aware Crowdsourcing with Reputation Management (SACRM) scheme, which can efficiently improve the crowdsourcing utility and the quality of sensing reports. Venanzi et al. [35] addressed the problem of fusing untrustworthy answers provided by a crowd of workers and incorporated the trust model into a fusion method. Zhang and Schaar [40] proposed protocols to incentivize workers to perform tasks well and reliably by using a novel game-theoretic model and integrating reputation mechanisms.

Byzantine Fault Tolerance (BFT) based approaches are promising solutions which are often used in distributed systems to ensure distributed consensus [110, 111]. In distributed systems, BFT is the ability to function as desired and correctly reach a sufficient consensus despite malicious nodes of the system failing or even propagating incorrect information to other peers. BFT has been introduced to ensure reliability in crowdsourcing [112]. In [112], BFT-based algorithms are used to design consensus protocol in crowdsourcing systems.

Unlike these studies, our study in Chapter 3 integrates the redundancy mechanism and the reputation mechanism. Moreover, the reliability of a worker in our study will be determined not only by the reputation of the worker himself but also by the reputations of the contextual workers. Even when a worker comes to the system for the first time, his initial reputation can be measured through the past experiences of his/her contextual workers. This approach can solve the problem that the reputation may be undeterminable due to the transient characteristics of the workers [40, 113]. BFT method could work well especially when it has deliberated malicious workers in the system, but it consumes much resources [114]. Compared to BFT, our current reputation approach is simple to be implemented in the complex task allocation approaches. In the future, we may design mechanism of ensuring reliability by using BFT approaches.

2.4 Crowdsourcing in Social Networks

The existing related studies can be categorized into two classes. The first class of studies mainly consider how to harness the crowdsourcing power of social media [30, 31] and how to use social networks as crowdsourcing platforms [32, 115], and the second class of studies mainly consider how to find a group of workers in social networks to complete outsourced tasks [25, 41].

In the first class of studies, Gao et al. [30] addressed harnessing the crowdsourcing power of social media for disaster relief. Ren et al. [29] exploited the capabilities of mobile device users connected via wireless networks to form a mobile cloud to provide pervasive crowdsourcing services. Lim et al. [31] combined the power of crowdsourcing with that of social networks and provided a web-based tool for the automation of stakeholder analysis. Here, the social networks were considered as new crowdsourcing platforms, but the allocation and reliability of complex tasks were seldom investigated systematically.

In the second class of studies, Chamberlain [25] proposed a definition for group-sourcing that includes the idea that a group of people connected through a social network is used to complete a task. However, in this class of studies, the requesters must undertake heavy computing loads to coordinate the organization of workers.

2.5 Context-Aware Crowdsourcing and Other Task Allocation

There are some studies on context-aware crowdsourcing. Tamilin et al. [113] presented a prototype implementation of a context-aware mobile crowdsourcing system that makes it possible to conduct crowdsourcing campaigns with users carrying mobile devices. Rana et al. [116] developed a context-aware crowdsourcing method by using smart phones for noise mapping, which can provide a feasible platform to assess noise pollution. Hu et al. [117] presented a multidimensional social network architecture

for mobile crowdsensing, which enables context awareness in the mobile crowdsensing applications. Alt et al. combined the web-based crowdsourcing and user-generated content to integrate location as a context parameter for distributing tasks to workers. In summary, the existing studies of context-aware crowdsourcing mainly cared about the context of mobile devices, but they did not address the context of social networks among workers and the reliability in context-aware crowdsourcing.

Task allocation and self-organization in open environments have been investigated in previous studies [97, 118]. In [119], the authors explored the context-aware task allocation in social networks. The problems in [119] differ from those addressed in this thesis. This thesis aims to maximize the probability of task completion through autonomous cooperation of workers in social networks which is an important and realistic feature of crowdsourcing. However, the studies in [119] only focus on minimizing resource access time in social networks, a feature that is not crucial in crowdsourcing.

Chapter 3

Context-Aware Reliable Crowdsourcing in Social Networks

There are two typical problems in traditional crowdsourcing systems for handling complex tasks. First, decomposing complex tasks into a set of micro-subtasks requires the decomposition capability of the requesters; thus, some requesters may abandon using crowdsourcing to accomplish a large number of complex tasks since they cannot bear such heavy burden by themselves. Second, tasks are often assigned redundantly to multiple workers to achieve reliable results, but reliability may not be ensured when there are many malicious workers in the crowd. Currently, it is observed that workers are often connected through social networks, a feature that can significantly facilitate task allocation and task execution in crowdsourcing. Therefore, this chapter investigates crowdsourcing in social networks and presents a novel context-aware reliable crowdsourcing approach. In our presented approach, the two problems in traditional crowdsourcing are addressed as follows. 1) Complex tasks can be performed through autonomous coordination between the assigned worker and his/her contextual workers in the social network; thus, requesters can be exempt from a heavy computing load for decomposing complex tasks into subtasks and combing the partial results of subtasks, thereby enabling more requesters to accomplish a large number of complex tasks through crowdsourcing. 2) The reliability of a worker is determined not only

by the reputation of the worker himself/herself but also by the reputations of the contextual workers in the social network; thus, the unreliability of transient or malicious workers can be effectively addressed. Based on theoretical analyses and experiments on a real-world dataset, we find that the presented approach can achieve significantly higher task allocation and execution efficiency than the previous benchmark task allocation approaches; moreover, the presented contextual reputation mechanism can achieve relatively higher reliability when there are many malicious workers in the crowd.

The remainder of this chapter is organized as follows. In Section 3.1, we present the problem description. In Section 3.2, we present the context-aware task allocation model. In Section 3.3, we present the context-aware task execution model. In Section 3.4, we present the reward mechanism. In Section 3.5, we provide the experimental results. Finally, in Section 3.6, we conclude this chapter.

3.1 Motivation and Problem Description

To clearly illustrate the research problem, we will first introduce the original framework for reliable allocation of simple tasks in crowdsourcing systems in Section 3.1.1. Then, in Section 3.1.2, we introduce the motivation and state our research problem. Finally, we analyze the complexity of the problem in Section 3.1.3.

3.1.1 Original Framework for Reliable Allocation of Simple Tasks in Crowdsourcing Systems

A simple task can be completed by one worker independently. Given a budget b_t for a simple task t, the necessary skills to complete t are represented by S_t. Let there be a crowd of workers, W. Then, the simple task allocation is to redundantly assign the task to multiple workers under the budget constraint to improve the accuracy of the result [2, 120]. This can be defined as follows: the requester or system assigns t to a set of workers, W_t, $W_t \subseteq W$; $\forall w_i \in W_t$, w_i can satisfy the skills required for task t and

complete t independently; and the sum of the reservation wages of the workers in W_t does not exceed b_t.

Moreover, the reputation mechanism can be used to encourage workers to complete the assigned tasks reliably [40, 121]. The reputation of a worker is mainly determined by the worker's past experiences in completing tasks; if a worker has richer experience of successful completion of tasks, his/her reputation is higher, and vice versa.

Therefore, the objective of reliable task allocation in general crowdsourcing systems is to select a set of workers that maximizes the following values: 1) the degree of redundancy, which denotes that the system redundantly assigns task t to as many workers as possible (each of whom fully possesses the skills required for t) under the constraint of the budget; and 2) the reputations of the assigned workers. Given a simple task t; $\forall w_i \in W$, the set of skills of w_i is S_{w_i}, and the reservation wage of w_i is γ_{w_i}. The objective of reliably allocating t can then be formalized as selecting a set of workers W_t that can satisfy the following equations:

$$W_t = \arg\max_{W_t \subseteq W} (\alpha_1 \cdot (\overbrace{\frac{|W_t|}{\sum_{w_i \in W_t} \gamma_{w_i}}}^{\text{Redundancy}}) + \alpha_2 \cdot (\overbrace{\sum_{w_i \in W_t} R_{w_i}}^{\text{Reputations}})) \tag{3.1}$$

$$subject\,to\ S_{w_i} \supseteq S_t,\ \forall w_i \in W_t \tag{3.2}$$

$$\sum_{w_i \in W_t} \gamma_{w_i} \leq b_t \tag{3.3}$$

where α_1 and α_2 are used to determine the relative importance of the two factors.

3.1.2 Motivation and Problem Statement of Complex Task Crowdsourcing in Social Networks

We will now present some notable observations made in related studies to motivate our study on crowdsourcing in social networks. Gray et al. [27] observed four popular crowdsourcing platforms, MTurk, UHRS, LeadGenius, and Amara, and found that the crowd of workers is actually a rich collaborative network and that workers often communicate via phone, forums, chat, Facebook, or in person, to share information

about tasks and requesters. Yin and Gray [26] specifically observed the collaboration network of workers on the MTurk platform where they executed a task in which over 10,000 workers self-reported their communication links to other workers. They found that there is a substantial communication network within the crowd of workers that is related to the workers' usage on the online forum. Therefore, these observations motivate our study on crowdsourcing in social networks.

In current crowdsourcing markets, there are many complex tasks. A complex task involves many computational operations and may not be completed independently by a non-professional individual worker. Therefore, *the situation in which each assigned worker can fully cover the required skills of* t*, i.e.,* $(S_{w_i} \supseteq S_t,\ \forall w_i \in W_t)$ *in Equation* (3.2) *cannot be satisfied. Because the self-owned skills of the assigned worker may only partially cover the necessary skills required by a complex task, the situation should be revised to* $(S_{w_i} \cap S_t \neq \Phi,\ \forall w_i \in W_t)$.

The new issues of complex task crowdsourcing in social networks can be described as follows.

1) An assigned worker may not perform the complex task alone and should coordinate with other workers in the social network to request assistance for the lacking skills. Therefore, when the crowdsourcing system wishes to assign a task t to a worker w_i, it considers not only the individual skills of w_i but also the skills of the contextual workers of w_i, CS_{w_i}. (*Considering contextual skills in task allocation*)

2) Because an assigned worker's performance in executing a task is influenced by his/her coordination with the contextual workers, a worker's reliability is determined not only by his/her own reputation but also by the reputations of his/her contextual workers. Therefore, the reputation R_{w_i} in Equation (3.1) should consider this factor. (*Considering contextual reputations in task allocation*)

3) Because the assigned workers can coordinate autonomously with the contextual workers in the social network to execute complex tasks, the system or the requester does not need to decompose the complex task into micro-tasks. However,

an efficient coordination mechanism between the assigned worker and the contextual workers should be designed. (*Coordinating with contextual workers in task execution*)

4) To promote the cooperation of workers in executing tasks, a proper reward mechanism should be designed that encourages not only workers to accept the tasks by themselves but also them to contribute skills to assist others in executing tasks. In addition, the reward mechanism considers not only monetary reward but also reputation reward. (*Rewarding contextual workers after task execution*)

Moreover, communication costs among workers may significantly influence performance in social networks [56, 57, 122]. Therefore, we should consider the communication costs of each assigned worker, w_i, with his/her contextual workers. The communication cost Com_{w_i} is defined as.

$$Com_{w_i} = \psi \left(\sum_{w_j \in (W - \{w_i\})} d_{ij} \right) \tag{3.4}$$

where the communication distance d_{ij} is the length of the shortest path between worker w_i and w_j in the social network, and $\psi(X)$ is a monotonically increasing function.

We now extend the original framework in Section 3.1.1 by considering these new issues. Then, the objective of reliable task allocation of crowdsourcing in social networks is to select a set of workers that maximizes the following values: 1) the coverage degree of each assigned worker's contextual skills for the task's required skills; 2) the contextual reputation of each assigned worker; 3) the degree of redundancy; and 4) the inverse of the communication cost of each assigned worker with his/her contextual workers. Therefore, the objective can now be formalized as selecting a set of workers

W_t that can satisfy the following equations:

$$W_t = \arg\max_{W_t \subseteq W} \frac{\overbrace{\sum_{w_i \in W_t} |CS_{w_i} \cap S_t|}^{\text{Contextual skills}} + \overbrace{\sum_{w_i \in W_t} CR_{w_i}}^{\text{Contextual reputations}} + \overbrace{\frac{|W_t|}{\sum_{w_i \in W_t} \gamma_{w_i}}}^{\text{Redundancy}}}{\overbrace{\sum_{w_i \in W_t} Com_{w_i}}^{\text{Communication}}} \tag{3.5}$$

$$subject\,to: \quad S_{w_i} \cap S_t \neq \phi, \forall w_i \in W_t \tag{3.6}$$

$$\sum_{w_i \in W_t} \gamma_{w_i} \leq b_t \tag{3.7}$$

With the above objective function, the requester can assign the task to the workers to maximize the possibility that the task's required skills can be satisfied and the accuracy of execution results. Moreover, the communication cost for executing the task will be minimized, which is also focused in previous benchmark studies on task allocation in social networks [56].

Then, each assigned worker, $w_i(\forall w_i \in W_t)$, will coordinate with his/her contexts to execute t. Finally, the system will reward w_i and his/her contextual workers for executing t after t is completed.

Our approach includes the following three components: 1) *context-aware task allocation*, which aims to assign the task to workers who can maximize the combination of contextual skills, contextual reputations, and redundancy and minimize the communication costs, as shown in Section 3.2; 2) *context-aware task execution*, which describes how each assigned worker coordinates autonomously with his/her contextual workers in the social network to execute complex tasks, as shown in Section 3.3; and 3) *reward after task execution*, which describes how to distribute rewards among the assigned workers and their contextual workers, as shown in Section 3.4.

3.1.3 Complexity Analyses

As stated above, the core of our research problem mainly involves assigning a principal worker who will then recruit other assistant workers from the social network. Thus, the

set of workers participating in the task includes the assigned principal worker and the assistant workers, which should possess all of the skills required by the task and satisfy the following conditions: 1) the communication costs among participating workers are minimized; 2) the total reservation wages of the participating workers are minimized; and 3) the total reputations of the participating workers are maximized.

Theorem 3.1. *Let there be a crowd of workers W that is organized in a social network. Assigning a set of participating workers including a principal worker and some assistant workers from W to cover the skills required by a task t and satisfy the above three conditions is NP-hard.*

Proof. Our research problem includes three independent sub-problems that can be described to assign a set of participating workers within a social network to achieve the following three objectives: 1) minimizing the total communication costs; 2) minimizing the total reservation wages; and 3) maximizing the total reputations. The first sub-problem has already been proven in previous benchmark studies to be NP-hard [56, 57, 123]. Since our research problem involves this NP-hard sub-problem in combination with two other independent sub-problems, we have Theorem 3.1. □

Because the research problem is NP-hard, we will present a heuristic approach that can be realized in reasonable time complexity but not present an approach to compute the optimal solution. In our heuristic approach, we define a concept of crowdsourcing value that combines the factors in Equation (3.5) to measure the probability of a given worker's being selected to participate in a task. Moreover, the principal worker will find assistant workers within the social network using the breadth-first search method, which can effectively reduce the costs of communication between the principal worker and the assistant workers.

3.2 Context-Aware Task Allocation

This section investigates how to allocate a task to workers by considering their contextual situations within their social networks. We first design a metric of contextual

crowdsourcing value to measure the probability of a given worker's being assigned the task. We then present algorithms for allocating the task to workers to maximize the contextual crowdsourcing values.

3.2.1 Contextual Crowdsourcing Value

Each worker has three properties: *skills*, *reputation*, and *reservation wage*. Among these three factors, the skills determine the worker's capacity to complete a task, the reputation determines the worker's reliability in completing the task, and the reservation wage is the minimum wage a worker is willing to accept as compensation in exchange for completing a crowdsourcing task [87, 124].

The probability of a given worker being assigned a task can be defined as the crowdsourcing value of the worker. To optimize the first three factors in Equation (3.5), this value is defined to be determined by the following attributes: 1) the coverage degree of the worker's skills for the necessary skills required by the task; 2) the reputation of the worker; and 3) the occupancy rate of the worker's wage in the task's budget.

Definition 3.1 (Self-owned crowdsourcing value of a worker)**.** *Given a budget b_t for a task t, the set of necessary skills to complete t is S_t. The self-owned crowdsourcing value of a worker, w_i, for a task t is defined as:*

$$v_i(t) = \frac{\alpha_1 \left(|S_{w_i} \cap S_t|/|S_t|\right) + \alpha_2(R_{w_i})}{\alpha_3(\gamma_{w_i}/b_t)} \tag{3.8}$$

where S_{w_i}, R_{w_i}, and γ_{w_i} denote the skills, reputation, and reservation wage of w_i, respectively, and α_1, α_2, and α_3 are parameters that determine the relative importance of the three factors.

In the social network context, when a worker receives a request from another worker for assistance in executing a task, the worker can decide whether to accept the request according to the benefits associated with the task and the worker's own expectation threshold. Therefore, each worker w_i has a predefined threshold τ_{w_i}; the worker will accept a request to assist another worker only if the benefit associated with the ongoing

task exceeds the threshold. The threshold of a worker is related not only to the worker's reservation wage but also to other factors such as reputation and credit; this will be shown in Section 3.3.1.

To optimize the four factors in Equation (3.5), the contextual crowdsourcing value of w_i can be defined to be determined by the crowdsourcing value of w_i himself/herself and the following four attributes of the contextual workers: the coverage degree of the contextual workers' skills for current lacking skills of w_i for t, the reputations and thresholds of the contextual workers, and the communication distance between w_i and the contextual workers.

Definition 3.2 (Contextual crowdsourcing value of a worker)**.** *The contextual crowdsourcing value of a worker w_i for a task t is defined as*

$$Cv_i(t) = \beta_1 \cdot v_i(t) + \beta_2 \cdot \sum_{w_j \in (W - \{w_i\})} \frac{\alpha_1 \left(|(S_{w_j} - S_i) \cap S_t| / |S_t| \right) + \alpha_2 R_{w_j}}{\alpha_3 \tau_{w_j} d_{ij}} \tag{3.9}$$

where W denotes the crowd of workers in the social network. The communication distance can be defined as the length of the shortest path between the two workers in the network. β_1 and β_2 are used to determine the relative importance of context.

Lemma 3.1. *Let there be two workers, w_i and w_j. If $Cv_i(t) > Cv_j(t)$, it is more probable that task t will be assigned to w_i by a crowdsourcing system in social networks to achieve the task allocation objective in Equation (3.5).*

Proof. The three factors that influence the task allocation objective in Equation (3.5) are determined as follows: the coverage degree for the skills required by a task is determined by $|S_{w_i} \cap S_t|/|S_t|$ and $|(S_{w_j} - S_{w_i}) \cap S_t|/|S_t|$, the reputations of the workers executing the task are determined by R_{w_i} and R_{w_j}, and the redundancy of the assigned workers is determined by γ_{w_i}/b_t. Because $Cv_i(t) > Cv_j(t)$, the comprehensive value of the three factors for w_i is higher than that of w_j; accordingly, w_i can satisfy the objective in Equation (3.5) to achieve the target W_t better than w_j. Therefore, it is more probable that task t will be assigned to w_i. □

3.2.2 Task Allocation Mechanism

In our model, two types of workers participate in the tasks: one type is the *principal worker*, who is assigned by the system or the requester to be responsible for a task; the other type is the *assistant workers* who are autonomously sought by the principal worker in the context of the social network. The assignment of the principal worker is implemented by the task allocation model in this section, and the selection of the assistant workers is implemented by the task execution model described in Section 3.3.

In the allocation of task t, we will select the candidate worker, from the crowd, who has the highest contextual crowdsourcing value for t and whose reservation wage does not exceed t's budget; the task will then be assigned to this candidate worker. We can repeat the allocation process to redundantly assign the task to other workers until the budget of the task has been used up.

To avoid the situation in which some workers are heavily loaded for a task, we make the following assumption: each worker can perform only one role in a task, i.e., if a worker is assigned as the principal worker for a task, he/she cannot serve as an assistant worker (requested by another assigned principal worker) for the task. Therefore, when the system calculates a worker's contextual crowdsourcing value for task t, the workers who have already been assigned to t will be excluded from the context.

The task allocation can be described as Algorithm 1. The time complexity of Algorithm 1 is $O(|W|^2)$, where $|W|$ is the number of workers in the social networked crowd. Although the worker set W is quite large, in reality the algorithm could be terminated in a reasonable time by pruning some workers. For instance, if the task is related to software developing, the workers whose skills are about translation could be excluded from the candidate sets. Moreover, the complexity of the algorithm is $O(|W| \times |W|)$ where one $|W|$ is related to searching the contextual resources of a worker. In reality, it is not necessary to search all the worker set in the searching of contextual resources, and the maximal quantity of the workers needs to search could be predefined by the system, i.e., the number of a worker's contextual workers should not exceed this maximal number.

Theorem 3.2. *Let the set of assigned workers for task t using Algorithm 1 be W_t. It is, then, assumed that there is another worker set in the crowd, W'_t, and that the total*

Algorithm 1: Task allocation based on workers' contextual crowdsourcing values (t, W).

$b_1 = 0$;
$W_t = \{\}$;
$W_{temp} = W$;
while $b_1 == 0$ **do**
  forall the $w_i \in W_{temp1}$ **do**
    $Cv_i(t) =$
    $\beta_1 \cdot v_i(t) + \beta_2 \cdot \sum_{w_j \in (W_{temp1} - \{w_i\})} \left(\frac{\alpha_1\left(|(S_{w_j} - S_{w_i}) \cap S_t| / |S_t|\right) \cdot \alpha_2(R_{w_j})}{\alpha_3(\tau_{w_j})} / d_{ij} \right)$;
  $w_* = \arg\max_{w_i \in W_{temp1}} Cv_i(t)$;
  $b_2 = 0$;
  $W_{temp2} = W_{temp1}$;
  if $\gamma_{w_*} > b_t$ **then**
    $b_2 = 1$;
  while $W_{temp2} \neq \phi$ *and* $b_2 == 1$ **do**
    $W_{temp2} = W_{temp2} - \{w_*\}$;
    if $W_{temp2} \neq \phi$ **then**
      $w_* = \arg\max_{w_i \in W_{temp2}} Cv_i(t)$;
      if $\gamma_{w_*} \leq b_t$ **then**
        $b_2 = 0$;
  if $W_{temp2} \neq \phi$ **then**
    $W_t = W_t \cup \{w_*\}$;
    $b_t = b_t - \gamma_{w_*}$;
    $W_{temp1} = W_{temp1} - \{w_*\}$;
    else
      $b_1 = 1$;
  if $b_t \leq 0$ **then**
    $b_1 = 1$;
Output (W_t);

reservation wages of all workers in W_t' do not exceed b_t. Thus, we have

$$\forall W_t' \subseteq W, \sum_{w_i \in W_t'} \gamma_{w_i} \leq b_t \Rightarrow \left(\frac{\sum_{w_i \in W_t} CV_i(t)}{|W_t|} \geq \frac{\sum_{w_j \in W_t'} CV_j(t)}{|W_t'|} \right) \tag{3.10}$$

Proof. We can use reductio ad absurdum to prove Theorem 3.2. Assume there is a set of workers W_t', $W_t' \neq W_t$, that is assigned by the system to task t and that the total

reservation wages of all workers in W_t' do not exceed b_t. If the assumption

$$\left(\sum\nolimits_{w_i \in W_t} CV_i(t)\right) / |W_t| < \left(\sum\nolimits_{w_j \in W_t'} CV_j(t)\right) / |W_t'|$$

is true, then there exists at least one worker with a higher contextual crowdsourcing value and whose reservation wage does not exceed the remaining budget of t but who cannot be selected by Algorithm 1, and another worker with lower contextual crowdsourcing value will be assigned to t. However, from Steps 7 and 15, in each round for selecting the assigned worker, the worker with the highest contextual crowdsourcing value whose reservation wage does not exceed the remaining budget of t will be the first to be definitely assigned to t. Therefore, the above assumption cannot occur in reality when Algorithm 1 is used. □

Theorem 3.2 ensures that the set of assigned workers with the highest average contextual crowdsourcing values can be achieved by Algorithm 1. Then, according to Lemma 3.1, the objective of task allocation of crowdsourcing in social networks can be approached the most efficiently.

3.3 Context-Aware Task Execution

After the task is assigned to a worker through the task allocation approach described in Section 3.2, the assigned worker will start to execute the task. If the assigned worker cannot execute the task by himself/herself, he/she will execute the task with the coordination of contextual workers.

3.3.1 Preliminaries

The workers in a social network are often coadjutant [33, 63]. A worker w_i will have certain obligations to provide assistance for another worker, w_j, if w_j has provided assistances for w_i in the past. Therefore, w_j may accept the requests of w_i for assistance even if the monetary reward provided by w_i is less than w_j's reservation wage because

w_j expects to get the possible assistance from w_i in the future. To measure the obligation to provide assistance between two workers, we define the credit between them to be determined by their cooperation history.

Definition 3.3 (Credit between two workers)**.** *Let there be two workers, w_i and w_j. We use $n_{i \leftarrow j}$ to denote the number of w_j's real assistance for w_i's executing tasks. The credit of w_j paid by w_i is in proportion to $n_{i \leftarrow j}$:*

$$c_j(\leftarrow i) = f(n_{i \leftarrow j}) \tag{3.11}$$

where f is a monotonically increasing function. Obviously, the higher $c_j(\leftarrow i)$ is, the more compulsory that w_i should provide assistance for w_j's request even if w_j cannot provide enough monetary reward to w_i. The reason is that in the past w_j has provided lots of assistance for w_i so that now w_i is obligated to compensate w_j. This credit is different from reputation because the credit involves only two workers whereas the reputation of a worker is perceived by all workers.

If the assigned worker w_i lacks the necessary skills required by task t, i.e., $S_t - S_{w_i} \neq \phi$, w_i will seek the assistance of other contextual workers to provide the skills w_i lacks. When w_i requests assistance from another worker, assuming w_j, w_i will offer w_j with two items:

1) *The possible monetary reward for executing task t.* Let $\overline{S_t}$ be the set of skills required for t that are currently lacking and S_{w_j} be the set of skills possessed by worker w_j. Therefore, the possible skill contribution of w_j for t is $S_{w_j} \cap \overline{S_t}$. Let the reservation wage of w_i for task t be γ_{w_i}. The possible monetary reward paid by w_i to w_j is

$$m_{i \rightarrow j}(t) = \lambda \cdot \gamma_{w_i} \cdot |S_{w_j} \cap \overline{S_t}| / |S_t| \tag{3.12}$$

where $0 \leq \lambda \leq 1$, which denotes the percentage of reservation wage that w_i is willing to distribute to other assistant workers.

2) *The credit paid by w_i to w_j for executing task t, $c_{i \rightarrow j}(t)$.* If $c_{i \rightarrow j}(t)$ is high and w_j hopes to obtain assistance from w_i in the future, w_j may accept the current request from w_i even if w_j cannot receive a satisfactory monetary reward for this request. If

w_j accepts the request from w_i, then: $c_i(\leftarrow j) = c_i(\leftarrow j) - c_{i\to j}(t)$, $c_j(\leftarrow i) = c_j(\leftarrow i) + c_{i\to j}(t)$.

Then, w_j will decide whether to accept the request from w_i for assistance in executing task t according to the following four conditions: 1) the possible monetary reward for executing task t, $m_{i\to j}(t)$; 2) the credit paid by w_i to w_j for executing task t, $c_{i\to j}(t)$; 3) the total credits of w_i paid by w_j in the past, $c_i(\leftarrow j)$; and 4) the reputation of w_i, R_i.

In the real world, each person may cooperate initially with his/her neighbors, and he/she will then cooperate with other people according to the breadth-first search in the social network [125]. Therefore, we now have:

Definition 3.4 (Coordination tree in the social network for a task)**.** *Let w_i be the assigned worker for task t. If w_i requests assistance from other workers in the social network, then the interaction relations between w_i and other workers form a tree whose root is w_i and the depth of each worker in the tree is the shortest interaction distance between w_i and the worker in the social network. Obviously, the coordination tree can be constructed on the basis of the breadth-first traversal method for the social network without considering the link types, and the workers can be decomposed into varying levels such that the shortest path from w_i to each worker (assuming w_j) in Level L_x is with distance x, i.e., $d_{ij} = x$.*

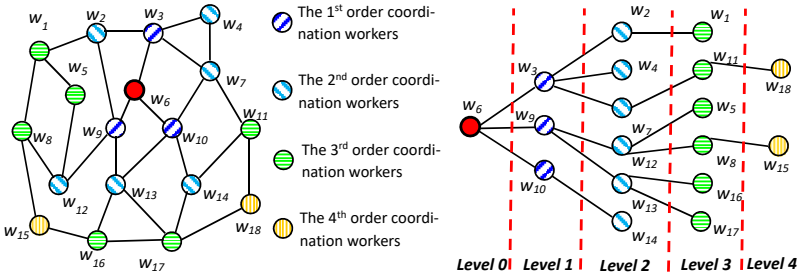


Figure 3.1: An example coordination tree in a social network.

Example. *In Figure 3.1, let w_6 be the assigned worker. We first compute the varying orders of coordination workers in the social network, and then the coordination tree is achieved.*

3.3.2 Task Execution Mechanism

Let the threshold of w_j be τ_{w_j}. Worker w_j will accept the request of w_i if the following condition can be satisfied:

$$(\eta_1 \cdot m_{i \to j}(t) + \eta_2 \cdot c_{i \to j}(t) + \eta_3 \cdot c_i(\leftarrow j) + \eta_4 \cdot R_{w_i}) \geq \tau_{w_j} \tag{3.13}$$

where η_1, η_2, η_3 and η_4 are four parameters that are used to determine the relative importance of the four factors.

To optimize the four factors in Equation (3.5), we can define the assistance value of a worker to be determined by the four attributes in Definition 3.2 and the credit between the assigned worker and the assistant worker.

Definition 3.5 (Assistance value of a worker)**.** *Let w_i be the assigned worker for task t. If $\overline{S_t}$ is the set of skills for t that are currently lacking, the assistance value of w_j perceived by w_i for executing t is defined as*

$$v_j(i-t) = \frac{\beta_1 \cdot \left(|S_{w_j} \cap \overline{S_t}|/|\overline{S_t}|\right) + \beta_2 \cdot (R_{w_j}) + \beta_3 \cdot c_i(\leftarrow j)}{\beta_4 \cdot \tau_{w_j} + \beta_5 \cdot d_{ij}} \tag{3.14}$$

where β_1, β_2, β_3, β_4 and β_5 are five parameters.

The task execution mechanism in a social network can be designed based on the coordination tree, shown as Algorithm 2 whose time complexity is $O(|W|^2)$, where $|W|$ is the number of workers. Then, $\forall w_i \in W_t$, w_i will execute Algorithm 2 to perform task t, and his/her contextual workers will decide whether to execute the task cooperatively according to 3.13. Finally, the results of the different assigned workers will be combined, and the final result will be achieved by majority voting.

3.4 Reward After Task Execution

After the task is executed by the assigned workers and their assistant workers, the requester or the system will reward them according to the execution results. The rewards include monetary reward and reputation reward.

Algorithm 2: Task execution in a social network (t, w_i).

$b = 0$;
$\overline{S_t} = S_t - S_{w_i}$;
$W_i(t) = \{w_i\}$;
Set the tags of all workers in the social network to 0;
Create Queue (Q);
Insert Queue (Q, w_i);
Set the tag of w_i to 1;
while *(!EmptyQueue* (Q)*) and (*$b == 0$*)* **do**
    w_{temp} =**Out** Queue(Q);
    forall the $w_x \in N(w_{temp})$ **do**
        // Neighbours of w_{temp}
        if $b == 0$ **then**
            $w_j = \underset{w_x \in N(w_{temp})}{\arg\max} \; v_x(i-t)$;
            if *the tag of* w_j *is 0* **then**
                Insert Queue(Q, w_j);
                Set the tag of w_j to 1;
                if $\eta_1 \cdot m_{i \to j}(t) + \eta_2 \cdot c_{i \to j}(t) + \eta_3 \cdot c_i(\leftarrow j) + \eta_4 \cdot R_{w_i} \geq \tau_{w_j}$ **then**
                    if $\overline{S_t} \cap S_{w_j} \neq \phi$ **then**
                        $c_i(\leftarrow j) = c_i(\leftarrow j) - c_{i \to j}(t)$;
                        $c_j(\leftarrow i) = c_j(\leftarrow i) + c_{i \to j}(t)$;
                        $\overline{S_t} = \overline{S_t} - S_{w_j}$;
                        $W_i(t) = W_i(t) \cup \{w_j\}$;
            if $\overline{S_t} == \phi$ **then**
                $b = 1$;
forall the $w_x \in W_i(t)$ **do**
    cooperating to execute task t;
Output (the executing result);

3.4.1 Monetary Reward

Let the final result of t after majority voting be $\omega(t)$ and the executing result of t by ω_i $(\forall \omega_i \in W_t)$ be $\omega_i(t)$. We can then set two tolerance values to evaluate the accuracy of $\omega_i(t)$, Δ_1, and Δ_2, $0 \leq \Delta_1 \leq \Delta_2$. If $|\omega_i(t) - \omega(t)| \leq \Delta_1$, we can say that the accuracy of w_i's executing task t is satisfactory, and w_i can now obtain the full monetary reward. If$\Delta_1 \leq |\omega_i(t) - \omega(t)| \leq \Delta_2$, we can say that the accuracy of w_i's executing task t is partially satisfactory, and w_i can now obtain part of the monetary reward; if $\Delta_2 \leq |\omega_i(t) - \omega(t)|$, we can say that the accuracy of w_i's executing task t is unsatisfactory,

and w_i cannot obtain a monetary reward. Moreover, finally there may be an unspent part of the budget of t, which can be used as a bonus to reward the assigned workers that can achieve satisfactory accuracy.

After w_i receives the monetary reward, M_i, he/she will distribute some part of M_i to the assistant workers. We can set a parameter λ, $0 \leq \lambda \leq 1$, that denotes the percentage of monetary reward that w_i is willing to distribute to the assistant workers, i.e., w_i will distribute λM_i to $\{w_j | w_j \in (W_i(t) - \{w_i\})\}$.

Let the reservation wage of w_i for task t be γ_{w_i}. γ_{w_i} is also thought of as the full monetary reward received by w_i from the system for successfully executing task t. We can now set the monetary reward mechanism as Algorithm 3, in which S_j^t denotes the real skills contributed by w_j for task t.

Algorithm 3: Monetary reward mechanism (t).

$W_t(ok) = \{\}$; `// Workers providing satisfactory accuracy`
forall the $w_i \in W_t$ **do**
  if $|\omega_i(t) - \omega(t)| \leq \Delta_1$ **then**
    $M_i = \gamma_{w_i}$;
    $b_t = b_t - M_i$;
    $W_t(ok) = W_t(ok) \cup \{w_i\}$;
  if $\Delta_1 \leq |\omega_i(t) - \omega(t)| \leq \Delta_2$ **then**
    $M_i = (\Delta_1 / |\omega_i(t) - \omega(t)|) \cdot \gamma_{w_i}$;
    $b_t = b_t - M_i$;
  if $\Delta_2 \leq |\omega_i(t) - \omega(t)|$ **then**
    $M_i = 0$;
forall the $w_i \in W_t(ok)$ **do**
  $M_i = M_i + b_t / |W_t(ok)|$;
forall the $w_i \in W_t$ **do**
  Reward $(1 - \lambda) M_i$ to w_i;
  forall the $w_j \in (W_i(t) - \{w_i\})$ **do**
    Reward $\lambda M_i \cdot (|S_j^t| / |S_t - S_{w_i}|)$;

3.4.2 Reputation Reward

We first set a value ζ_t for the reputation reward of task t; moreover, we also set another value μ that measures the relative obligation incurred by the assigned worker to the assistant workers for executing the task. Generally, we can set $0.5 \leq \mu \leq 1$; this means that the assigned worker assume a primary obligation to complete the task successfully and the assistant workers only undertake the secondary obligation. The reputation reward mechanism can then be designed as Algorithm 4.

Algorithm 4: Reputation reward mechanism (t).

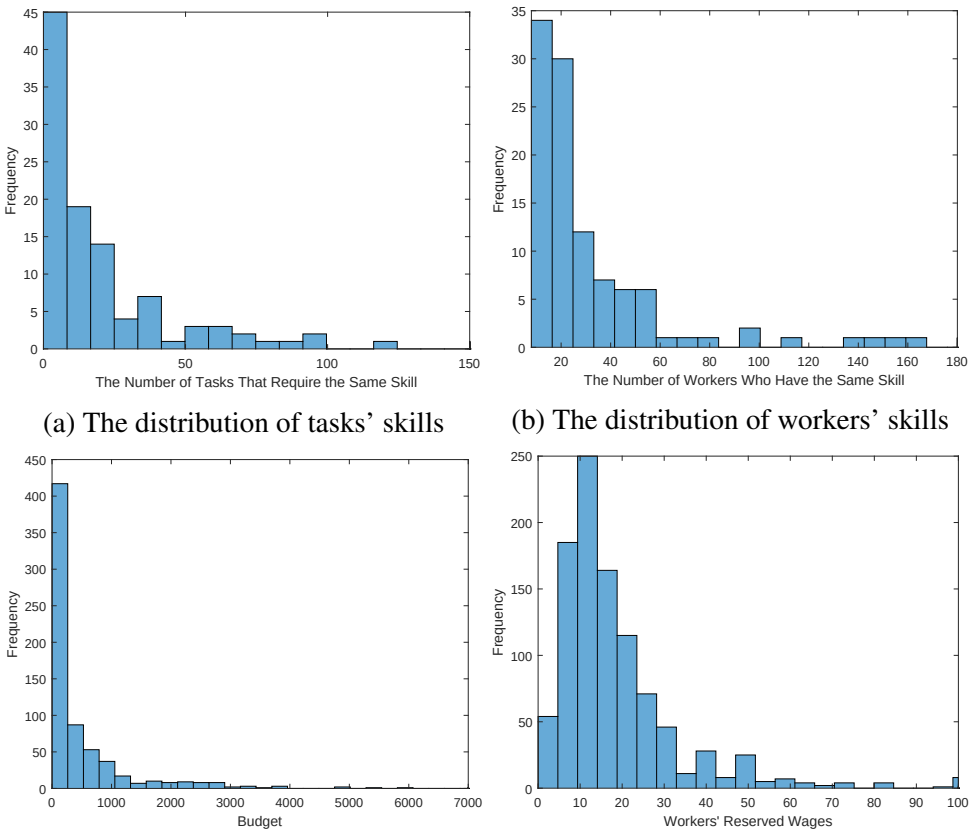
forall the $w_i \in W_t$ **do**
  if $|\omega_i(t) - \omega(t)| \leq \Delta_1$ **then**
    $R_{w_i} = R_{w_i} + \mu\zeta_t$;
    forall the $w_j \in (W_i(t) - \{w_i\})$ **do**
      $R_{w_j} = R_{w_j} + (1-\mu)\zeta_t$;
  if $\Delta_1 \leq |\omega_i(t) - \omega(t)| \leq \Delta_2$ **then**
    $R_{w_j} = R_{w_j} + \mu \cdot (\Delta_1/|\omega_i(t) - \omega(t)|) \cdot \zeta_t$;
    forall the $w_j \in (W_i(t) - \{w_i\})$ **do**
      $R_{w_j} = R_{w_j} + (1-\mu) \cdot (\Delta_1/|\omega_i(t) - \omega(t)|) \cdot \zeta_t$;
  if $\Delta_2 \leq |\omega_i(t) - \omega(t)|$ **then**
    $R_{w_j} = R_{w_j} - \mu\zeta_t$;
    forall the $w_j \in (W_i(t) - \{w_i\})$ **do**
      $R_{w_j} = R_{w_j} - (1-\mu)\zeta_t$;

3.5 Experiments

3.5.1 Experimental Settings

The experiments are conducted using a real-world dataset extracted from www.freelancer.com. The dataset includes information on the workers and information on the tasks. Specifically, the information for each worker includes the set of skills he/she has and the reserved wage he/she declares; and for each task, the information includes the required skills and the budget presented by the requester. At Freelancer,

each worker can list at most 5 different skills in his/her profile, and each requester can also input at most 5 different required skills in his/her profile.



(a) The distribution of tasks' skills

(b) The distribution of workers' skills

(c) The distribution of tasks' budgets

(d) The distribution of workers' reserved wages

Figure 3.2: The statistics of the collected data at Freelancer website.

The original collected data include 9642 tasks and 997 workers. The tasks require 644 different skills in total, but the workers only possess 107 different skills. Thus, many tasks cannot be completed by these workers. To allow every task a chance to be completed, we remove the tasks that require skills that are not possessed by any of the workers. The final dataset contains 697 tasks and 997 workers. Figure 3.2 shows the distribution of the tasks' required skills, the workers' skills, the tasks' budgets, and the workers' reservation wages; the y-axis denotes the occurrence frequencies of the varying numbers of the four factors, respectively, within the collected data. From Figure 3.2, we can see that both the tasks and the workers' skills follow the power law distribution; this indicates that some skills are very popular whereas others are not. If a task requires some unpopular skills, it may be difficult for the task to be matched with

perfectly suitable workers. The task budgets also follow the power law distribution, whereas the workers' reservation wages follow a normal distribution. Here, it is noted that the average budget is much higher than the average reserved wage. Thus it is possible that a task can be allocated to several workers.

In the experiments, we use Facebook social networks to model the workers' social networks, i.e., we combine the Facebook network with the Freelancer dataset to construct a crowdsourcing platform with a social network structure. We randomly extract 997 nodes from the downloaded Facebook network, and we find that these nodes are connected. The average degree of the network is 24.93. The height of the breadth-first traversal tree starting from any node is 4.6, so it is not difficult to find coordination workers for each assigned worker in the network.

In the experiments, for simplification, all factors are considered equally. Therefore, we set all parameters, including α_1, α_2, α_3, β_1, β_2, β_3, β_4, β_5, η_1, η_2, η_3, and η_4 to be normalized 1. If the parameters are not normalized, some factors would be overwhelmed by other factors since the value scopes of varying factors are very different. The distribution of workers' accuracies follows a normal distribution $N(0.7, 0.1)$; moreover, the accuracy values are set within the range [0, 1].

3.5.2 Benchmark Approaches

In the experiments, we compare our presented context-aware task allocation approach with two previous benchmark approaches: the *straightforward task allocation approach* and the *decomposition-based task allocation approach*.

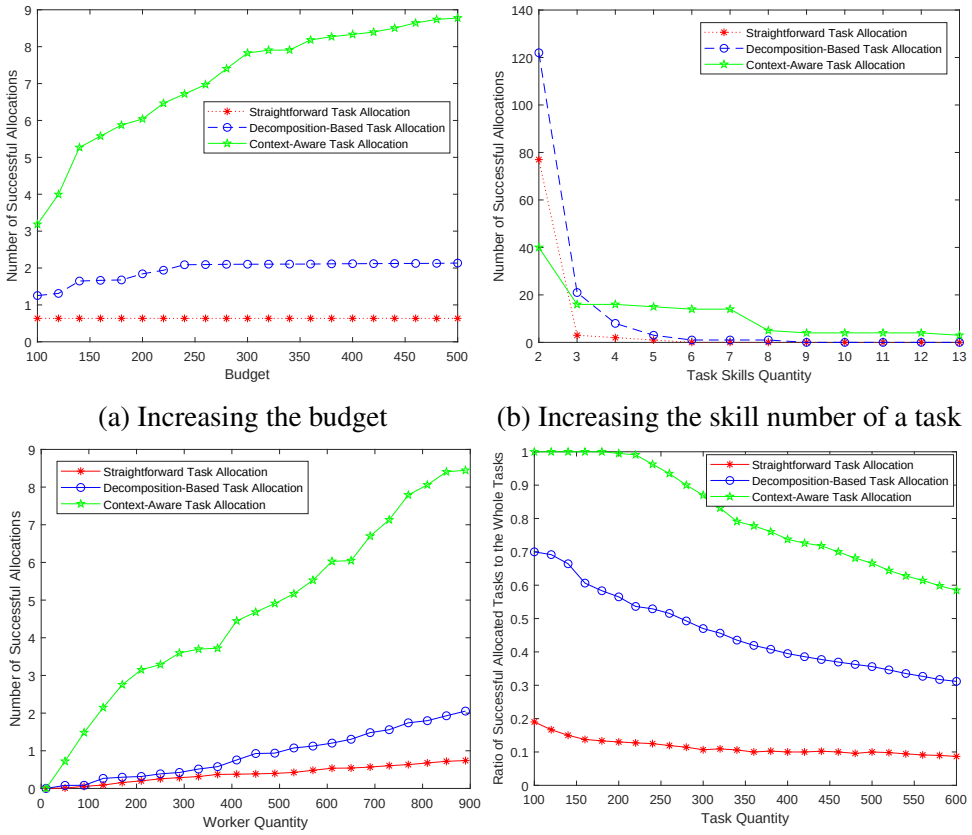
The *straightforward task allocation approach* is a traditional approach that has been used in previous crowdsourcing systems [79]. In this approach, the requester (or the crowdsourcing system) allocates the task to a worker who fully satisfies the skill requirements of the task. Therefore, each assigned worker can perform the task individually and independently. If the assigned worker's required wage does not exceed the budget, the task can be redundantly assigned to other workers until the budget is used up. If the requester (or the system) cannot find a worker who fully satisfies the skill requirements of the task, the task cannot be allocated successfully.

The *decomposition-based task allocation approach* [2] is a popular approach for performing complex tasks in which each complex task is decomposed into a flow of simple subtasks; the subtasks are then allocated to workers, each of whom can fully satisfy the skill requirements of the assigned subtask and perform the assigned subtask independently. If all of the subtasks are allocated successfully, the original complex task is deemed to have been allocated successfully. If the budget is not used up, the complex task will be redundantly allocated multiple times.

3.5.3 Tests on the Task Allocation Efficiency

In the task allocation objective defined in Equations (3.1) and (3.5), a task will be redundantly assigned to as many workers as possible under a given budget, which can improve the accuracy of the solution [2]. Thus, one important task allocation objective is to maximize the redundancy degree of task allocation. Therefore, we can use the number of successful allocations to measure the allocation efficiencies of the three approaches.

First, we test the number of successful redundant allocations when the budget is varied. In the test, we enter the tasks into the system one by one and then record the average allocation number for all tasks. Figure 3.3a shows that with the same budget, the average successful allocation number for all tasks in our approach is much higher than the ones obtained using the other two approaches; moreover, the average successful allocation number of our approach increases with the increase of the budget more drastically than do the other two benchmark approaches. The reason is: when the task's budget increases, our approach allows each assigned worker to find more appropriate assistant workers, thus increasing the number of successful allocations (here, a successful allocation means that an assigned worker and his/her assistant workers can satisfy all of the skill requirements of the task). In comparison, since the reserved wage of each worker is much lower than the task's budget, the possible workers who can fully satisfy the skill requirements of the task can be found no matter the budget is low or high; therefore, an increase in the budget has no obvious effect on the other two benchmark approaches.



(a) Increasing the budget

(b) Increasing the skill number of a task

(c) Increasing the worker quantity

(d) Increasing the task quantity

Figure 3.3: Number of successful allocations made using the three approaches under varying situations.

Second, we test the number of successful redundant allocations when the number of skills required by task increases, shown as Figure 3.3b. At first when a task requires few skills, our approach cannot achieve better performance because the budget is quickly used up with our approach. However, when the tasks require more and more skills, the other two benchmark approaches cannot find appropriate workers who can fully satisfy the skill requirements of the tasks. Thus, their successful allocation numbers deteriorate more drastically than does our approach. Moreover, when the number of skills needed for the task exceeds the system's specified number of skills (i.e., five skills), our approach can still allocate the task quite successfully since it can utilize the contextual workers' skills, whereas the decomposition-based task allocation can only allocate very few times, and the straightforward task allocation cannot find any workers who can fully satisfy the need for more than five skills.

Third, we test the number of successful redundant allocations when the number of

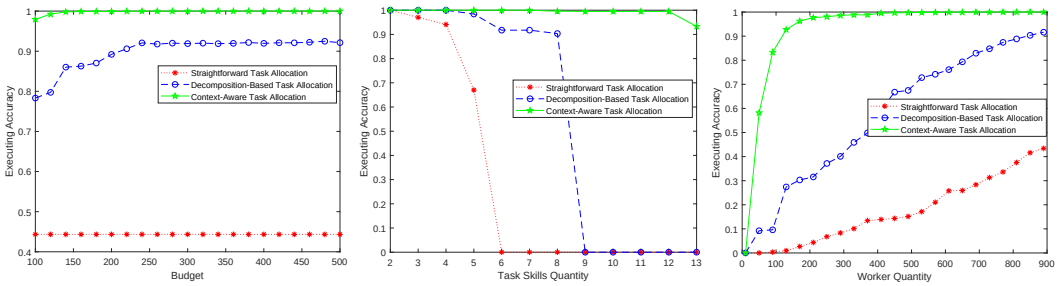
workers in the crowd increases, shown as Figure 3.3c. We can see that the allocation number of our approach increases more rapidly than that of the other two benchmark approaches. The reason is that our model is able to harness the power of more workers when a single worker cannot complete the task individually; in comparison, other two benchmark approaches can only utilize the power of the workers who can fully satisfy the skill requirements of the task.

Fourth, to consider the real-world situation in which requesters may outsource numerous tasks simultaneously, we test the throughput of the three approaches. In the experiment, we assume that each task is only allocated one time and a worker cannot be assigned to other tasks if that worker has already been assigned to a task. We then test the ratio of successful allocated tasks to all tasks, shown as Figure 3.3d. We can see that our approach performs better than the other two benchmark approaches. The reason is that our approach allows more workers to be considered in the task allocation even the workers themselves cannot fully meet the skill requirements of the tasks; in comparison, other two benchmark approaches only consider the workers who can fully satisfy the skill requirements of the tasks; thus, the qualified workers in those two approaches are fewer than that of in our approach.

3.5.4 Tests on the Task Execution Efficiency

We now compare the task executing accuracy in the three approaches. In the experiment, each worker w_i has a random executing accuracy a_{w_i}; the distribution of workers' accuracies follows a normal distribution $N(0.7, 0.1)$. Moreover, the accuracy values are set within the range [0, 1].

If a task is allocated redundantly, the task will be executed more than one time. The accuracy of one execution of a task t (e.g., the j-th execution), $a_j(t)$, is determined by the average accuracy of the workers participating in the execution. In the straightforward approach, there is only one worker in each execution; thus, the accuracy of each execution is the accuracy of the assigned worker, $a_j(t) = a_{w_i}$. In the decomposition-based task allocation, let the task be decomposed into m subtasks and let only one



(a) Increasing the budget (b) Increasing the skill number of a task (c) Increasing the worker quantity

Figure 3.4: Task execution accuracy of the three approaches under varying situations.

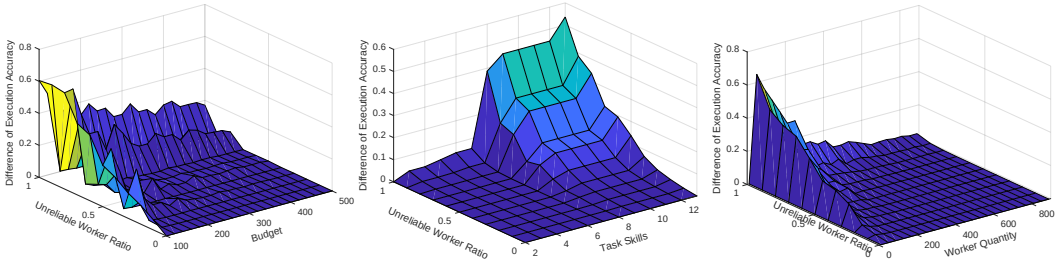
worker participate in one execution of each subtask; thus, there are m workers participating in one execution of the entire task. In our presented approach, let there be m workers participating in one execution of the task; thus, there are one assigned worker and $m-1$ assistant workers. Therefore, the accuracy of one execution of a task in the decomposition-based task allocation approach and our presented approach is:

$$a_j(t) = \frac{1}{m}\sum_{i=1}^{m} a_{w_i} \tag{3.15}$$

Now, since each task is allocated redundantly multiple times (n), the overall accuracy of the task is:

$$a(t) = 1 - \prod_{j=1}^{n}(1 - a_j(t)) \tag{3.16}$$

Intuitively, if a task is executed more times, the result will be more accurate. The experimental results are shown in Figure 3.4. Figure 3.4a shows that the executing accuracy with our approach increases quite rapidly with the increase of the task's budget, finally reaching 1. Figure 3.4b shows the execution accuracy when we increase the number of the task's skills; we can see that the execution accuracy of our approach remains very close to 1 when the number of required skills is less than 13, but the execution accuracy in other two approaches deteriorates drastically as the number of required skills increases. Figure 3.4c shows the execution accuracy when the worker quantity increases. Compared with the other two benchmark approaches, our presented approach achieves much higher execution accuracy even when the quantity of workers is small.



(a) Varying the budgets (b) Varying the task skills (c) Varying the worker quantity

Figure 3.5: Differences in task execution accuracies when a reputation mechanism for different ratios of unreliable workers is or is not adopted under varying situations.

Therefore, our presented approach achieves better performance in task execution efficiency than the other two benchmark approaches. The reason is similar to that described in Section 3.5.3; for brevity, a detailed description is omitted.

3.5.5 Tests on the Reputation Mechanism

We now test the effects of our reputation mechanism when there are unreliable or malicious workers. In particular, we compare the task execution accuracies when the reputation mechanism is adopted and when it is not. In the experiments, we randomly set some workers as unreliable. For simplicity, we assume that the results of task execution are binary values. We can then adopt a method similar to that described in [126] in which the unreliable workers also contribute in some degree to the accuracy of task and hence, even if all workers are unreliable, the task can still achieve some accuracy if the reputation mechanism is adopted. The unreliable workers' accuracies follow the normal distribution $N(0.2, 0.1)$. When the reputation mechanism is adopted, the system gradually identifies the unreliable workers and then utilizes the results generated by the unreliable workers. The results are shown in Figure 3.5, in which z-axis shows the differences of task execution when the reputation mechanism is or is not adopted in a variety of situations.

Figure 3.5a shows the results for varying task budgets (x-axis) and proportions of unreliable workers (y-axis). We can see that when there are more unreliable workers, the model adopting a reputation mechanism performs much better than the model that does

not; therefore, our presented reputation mechanism can effectively address the presence of unreliable workers. Moreover, we can also see that the reputation mechanism can provide more obvious benefits when the task's budget is insufficient.

Figure 3.5b shows the results for varying numbers of task skills (x-axix) and proportions of unreliable workers (y-axix). When a task requires more skills, i.e., the task is more complex, the reputation mechanism performs much better than the model that does not incorporate a reputation mechanism. Therefore, our presented reputation mechanism can effectively address the presence of the unreliable workers, especially when the task is complex.

Figure 3.5c shows the results for varying numbers of workers (x-axix) and proportions of unreliable workers (y-axix). The results show that the reputation mechanism can perform better especially when the workers are not sufficient.

3.6 Summary

This chapter aims to solve two typical problems noted in previous studies of crowdsourcing complex tasks: the requesters undertake a heavy burden when decomposing complex tasks into a set of micro-subtasks, and reliability may not be ensured when there are many malicious workers in the crowd. By considering a current general situation in which the workers are often connected by social networks, this chapter explores a context-aware reliable crowdsourcing approach that can solve the above two problems in social network environments.

This chapter implements the approach by defining a reasonable concept of crowdsourcing value that can be used to measure the probability of a worker's being assigned a task when the context of the worker in the social network is considered. The experiments on a real-world dataset show that the presented approach outperforms previous benchmark approaches with respect to task allocation and execution efficiencies; moreover, the presented approach can effectively address the situation in which there are many unreliable workers.

Chapter 4

Batch Allocation for Tasks with Overlapping Skill Requirements in Crowdsourcing

Existing studies on crowdsourcing often adopt the retail-style allocation approach, in which tasks are allocated individually and independently. However, such retail-style task allocation has the following drawbacks: 1) each task is executed independently from scratch, thus the execution of one task cannot utilize the results of other tasks and the requester must pay in full for the task; 2) many workers only undertake a very small number of tasks contemporaneously, thus the workers' skills and time may not be fully utilized. We observe that many complex tasks in real-world crowdsourcing platforms have similar skill requirements and long deadlines. Based on these real-world observations, this chapter presents a novel batch allocation approach for tasks with overlapping skill requirements. Requesters' real payment can be discounted because the real execution cost of tasks can be reduced due to batch allocation and execution, and each worker's real earnings may increase because he/she can undertake more tasks contemporaneously. This batch allocation optimization problem is proved to be NP-hard. Then, two types of heuristic approaches are designed: layered batch allocation and core-based batch allocation. The former approach mainly utilizes the hierarchy pattern to form all

possible batches, which can achieve better performance but may require higher computational cost since all possible batches are formed and observed; the latter approach selects core tasks to form batches, which can achieve suboptimal performance with lower complexity and significantly reduce computational cost. With the theoretical analyses and experiments on a real-world Upwork dataset in which the proposed approaches are compared with the previous benchmark retail-style allocation approach, we find that our two batch allocation approaches have better performances in terms of total payment by requesters and average income of workers, as well as maintaining close successful task completion probability and consuming less task allocation time.

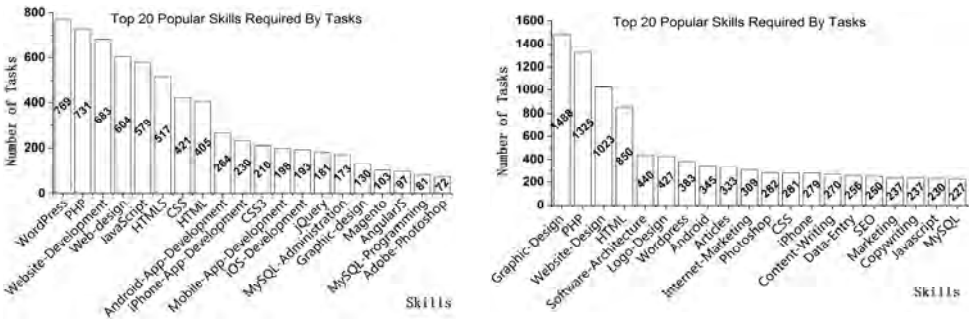
The remainder of this chapter is organized as follows. In Section 4.1, we present the motivation and problem description; in Section 4.2, we present the approach for layered batch formation and allocation of tasks; in Section 4.3, we present the approach for core-based batch formation and allocation of tasks; in Section 4.4, we provide the experimental results; finally, we conclude this chapter in Section 4.5

4.1 Motivation and Problem Description

4.1.1 Motivation

We have analyzed some data from two leading complex-task-oriented crowdsourcing websites, www.upwork.com and www.freelancer.com, which include 1353 randomly selected workers and 4950 randomly selected tasks from the Upwork website, and 578 workers and 6968 tasks from the Freelancer website. We summarize the following main characteristics:

1) **Overlapping skill requirements of different tasks.** We consider the typical category of complex tasks at the Upwork website: web-mobile development. We find the 20 skills that are most often required by tasks in the category, which are shown on the x-axis in Figure 4.1a; the number of tasks requiring each type of skill is represented by the y-axis. We find that each of 20 most popular skills is



(a) Upwork website (b) Freelancer website

Figure 4.1: The numbers of tasks requiring some given skills.

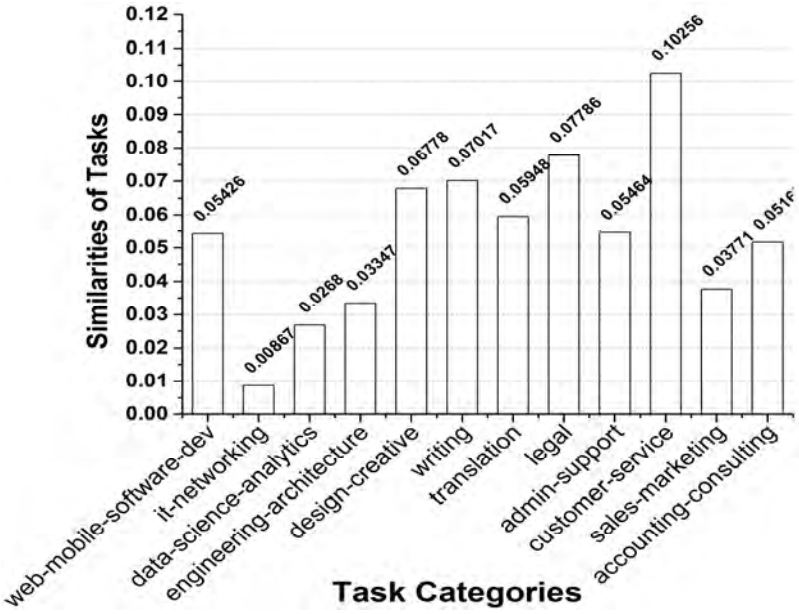
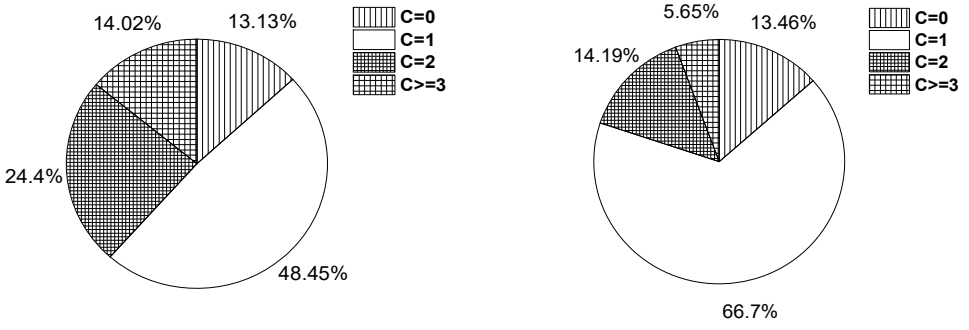


Figure 4.2: The similarities of required skills of tasks in different categories at the Upwork website.



(a) Upwork website (b) Freelancer website

Figure 4.3: The numbers of tasks undertaken by workers during a given period.

required by 332 tasks on average, while the total number of tasks is 4950 and each task may require more than one skill.

Because the Freelancer website does not have categories of tasks, we consider all categories of tasks. The statistical results are shown as Figure 4.1b. We can see that each of 20 most popular skills at the whole website is required by 473 tasks on average, while the total number of tasks is 6968 and each task may require more than one skill. *Therefore, many real-world crowdsourcing tasks may have overlapping skill requirements.*

2) **Similarities of required skills among tasks.** Let there be n tasks. We use S_{t_x} to denote the set of skills required by task t_x. Then, the similarities among the n tasks can be measured as the average pairwise Jaccard similarity (mean similarity between each pair of n tasks):

$$Sim = \frac{1}{n \cdot (n-1)} \sum_{x \neq y} \frac{|S_{t_x} \cap S_{t_y}|}{|S_{t_x} \cup S_{t_y}|} \tag{4.1}$$

Figure 4.2 shows the similarities of different categories of tasks at the Upwork website, where some tasks within each category are similar. In particular, the tasks of customer-service, legal, and writing have higher similarities. Although the tasks at the Freelancer website are not categorized clearly, we also find that the software development tasks at the Freelancer website have a similarity with an average value of 0.033. *Therefore, the skill requirements of some complex tasks within the same category may often be similar.*

3) **The number of tasks undertaken by workers contemporaneously.** We count the number of tasks undertaken by each worker during a given period; then, we calculate the percentage of the time in which the worker has undertaken a given number of tasks, shown as Figure 4.3. In Figure 4.3, C denotes the number of tasks undertaken contemporaneously, and the percentage indicates the time occupancy. We find that the period in which workers undertake no more than one task occupies 61.58% of the workers' total duration at the Upwork website and 80.16% at the Freelancer website. *Therefore, most workers undertake only a very small number of tasks contemporaneously.*

4) **The deadlines of tasks.** Many crowdsourcing tasks have long deadlines; for example, after analyzing 7395 tasks at the Upwork website, we find the average duration is 97.76 days; and after analyzing 11614 tasks at the Freelancer website, we find that the average duration is 6.53 days. *The crowdsourcing tasks often have long deadlines* because urgent tasks may not be applicable to crowdsourcing web-sites, where workers may have uncertain skill levels and be unreliable.

In summary, from the observations, it can be concluded that there are four popular characteristics of current complex-task-oriented websites: *the skills required by tasks are often overlapping; many complex tasks within the same category have similarities; most workers undertake only a very small number of tasks contemporaneously; and the deadlines of many tasks are long.* Therefore, these observations motivate our study on batch allocation of tasks. In fact, it is well known that grouping simple tasks in larger batches is attractive to workers, since working on large batches of tasks avoids overhead of selecting tasks to work on, as well as reading instructions and learning how to perform the task [46]. In comparison, the novelty of this thesis is that we explore the batching of complex tasks.

With the batching of complex tasks, there are two obvious advantages. *One is that the total execution costs of tasks in a batch by the same workers can be reduced by comparing with the case in which each task is executed independently by different workers.* The reason is that the partial execution results of one task can be reused (or with certain modifications) in another similar task by the same workers. For example, if a worker undertakes two similar tasks contemporaneously: developing a B2C website and developing an O2O website, some of the infrastructures and basic components in developing the B2C web-site can be reused with certain modification in the developing of the O2O website; because the repetitive works can be saved, the total execution costs of the two tasks can be reduced by comparing to the case in which the two websites are developed from scratch. *Another advantage is that the allocation time can be saved* because the tasks in the batch can be allocated in whole by comparing the retail-style task allocation approach in which each task is allocated non-redundantly and independently to a professional worker, which is verified by the experimental results in Figure 4.6 in Section 4.4.5.2.

4.1.2 Problem Description

4.1.2.1 Discounting of Payment for Batch Allocation

Discounting is a general market mechanism in which the payment can be discounted in exchange for cheaper or less satisfactory service [127]. It is well known that a key challenge within crowdsourcing is to minimize the real payments by requesters [2, 128]. Discounting can be introduced to reduce the real payment by requesters since a worker who accepts a task batch might produce delayed service as the worker needs to handle the multiple tasks in the batch. Therefore, requesters pay less by discounting their real payment due to the delayed service that results from batch allocation. On the other hand, as stated above, batch allocation can reduce the real execution cost and improve the real earnings of the worker, so the worker will be willing to accept the discounted payment.

Formally, the crowdsourcing system can discount a requester's real payment to an assigned worker w_i for completing task t according to the number of batched tasks that are queueing for w_i, $B_{w_i} = \{t^1_{w_i}, t^2_{w_i}, ..., t^n_{w_i}\}$ in which $t^j_{w_i}$ is the task that executed by worker w_i. Now, we can define the discounting function as follows:

$$Pay_{w_i}(t) = \psi(|B_{w_i}|) \cdot b_t \tag{4.2}$$

where b_t is the original budget of task t and ψ is a discounting function, $0 \leq \psi \leq 1$, where the value of $\psi(X)$ decreases monotonically from 1 to 0 with the increase of X. In this thesis, ψ is defined as $exp(\sigma \cdot (-|B_{w_i}| + 1))$, where σ is a given discounting factor.

Given a free worker w_i, let there be a batch of tasks, $B = \{t_1, t_2, ..., t_{|B|}\}$, allocated to w_i; we assume that t_x is executed after $t_{x-1}(1 < x \leq |B|)$ and the original budget for task $t_x(1 \leq x \leq |B|)$ is b_{t_x}. Then, the real payment that w_i can get from batch B is

$$Pay_{w_i}(B) = \sum_{x=1,...,|B|} (\psi(x) \cdot b_{t_x}) \tag{4.3}$$

4.1.2.2 Optimization Objective

According to the above discounting function, the more tasks that are allocated in batch to the same worker, the more significantly the system can discount the requesters' real payment for these tasks; the worker can also earn more hourly wages. However, if too many tasks are allocated to the same worker, some issues may arise: first, the assigned worker and his/her collaborators may have difficulty satisfying some of the skill requirements of the tasks; second, the payment will be heavily discounted and the real payment may be too low to satisfy the worker's reservation wage; third, if too many tasks are waiting for the same worker, the real completion time of some tasks may exceed their deadlines (although the deadlines are long).

Reducing the real payments of requesters is associated with increasing the batch size, which might also lead to higher earnings of workers. Therefore, the optimization objective is to form proper batches and to allocate the batches to the workers with the maximum crowdsourcing values to minimize the real payments of requesters under the following constraints: the discounted payment for each task is not less than the worker's reservation wage; and the completion time cannot exceed the deadline of the task.

Let T be a set of tasks. Π is a possible batching scheme for T, i.e., a combination scheme of the tasks in T; $\Pi = \{B_x | \bigcup B_x = T\}$, where $B_x (B_x \subseteq T)$ denotes a batch that includes a subset of T. We use $CV_{w_i}(B_x)$ to denote the crowdsourcing value of w_i for B_x, i.e., the probability of w_i being assigned to the batch of tasks B_x, which can be designed for certain criteria (e.g., skill coverage, wage occupancy, completion time, and reputation) and expressed later in Section 4.2.2. Our objective is to find a batching scheme that can minimize the total real payment by all requesters of tasks in T:

$$\Pi_* = \arg\min_{\Pi} \sum_{B_x \in \prod} Pay_{w_j}(B_x) \tag{4.4}$$

$$where\ w_j = \arg\max_{w_i \in W} CV_{w_i}(B_x) \tag{4.5}$$

$$subject\ to:$$

$$time(w_j, t) \leq d_t, \forall B_x \in \Pi, t \in B_x \tag{4.6}$$

$$Pay(w_j, t) \geq \gamma_{w_j}, \forall B_x \in \Pi, t \in B_x \tag{4.7}$$

where $time_{w_j}(t)$ denotes the completion time of task t by worker w_j, d_t denotes the deadline of t, and r_{w_j} denotes the reservation wage of w_j. Equations (4.5)-(4.7) denote that batch B_x is allocated to a worker w_j who has the highest crowdsourcing value for B_x and satisfies the following two constraints: 1) the completion time of each task in B_x by w_j should not exceed the deadline of that task; and 2) the real payment by the requester of each task in B_x to w_j should be higher than the reservation wage of w_j.

4.1.2.3 Property Analyses

Theorem 4.1. *The batch allocation problem with the optimization objective in Equations* (4.4)-(4.7) *is NP-hard.*

Proof. 3-dimensional matching (3DM) is a classical NP-hard problem [129]. There are assumed 3 finite disjoint sets, X, Y, and Z. All elements are defined as $N = X \cup Y \cup Z$. Let L be a set of candidate triples, $L = \{(x, y, z)|x \in X, y \in Y, z \in Z\}$. A 3-dimensional matching, M, is a subset of L: for any two distinct triples, $(x_1, y_1, z_1) \in M$ and $(x_2, y_2, z_2) \in M$, we have $x_1 \neq x_2$, $y_1 \neq y_2$ and $z_1 \neq z_2$. Given a set N, a set of triples L and an integer k, the decision problem of 3DM is to decide whether there exists a 3-dimensional matching $M \subseteq L$ with $|M| \geq k$.

In an instance of our problem, there is a set of tasks, $T = \{t_1, t_2, ..., t_n\}$. We can obtain a collection $C = \{C_1, C_2, ..., C_x\}$ of 3-element subsets of T and a collection $A = \{A_1, A_2, ..., A_n\}$ of 1-element subsets of T. We set a positive payment weight $w_c = 1$ for each C_i and $w_a = 2/3$ for each A_i. The 3DM problem can be transformed to our problem as follows: each triple of L corresponds to $C_i \in C$, and each element of N corresponds to $t_i \in T$. Let J be a positive number. Then, we will show that the 3DM problem has a 3-dimensional matching M, with $|M| \geq k$, if and only if our problem has an exact cover E for T with the sum of weights $\sum_{i \in E} w_i \leq J$, where $J = 2n/3 - k$.

1) *only if.* If there exists an exact cover E for T, where $E = A' \cup C'$ and $A' \subseteq A \wedge C' \subseteq C$, the sum of the weights of E is $w_a \cdot |A'| + w_c \cdot |C'| = w_a \cdot (|T| - 3|C'|) + w_c \cdot |C'| = 2n/3 - |C'| \leq J$. Thus, $|C'| \geq 2n/3 - J$. Each triple of M corresponds to $C_i \in C'$, and the size n of T and N that can be seen as a constant for the

specific problem. This indicates that for the 3DM problem, there is a matching M with $|M| = |C'| \geq k, k = 2n/3 - J$.

2) *if*. If there exists a 3-dimensional matching M in the 3DM problem, $|M| \geq k$, it can be proved that our problem has an exact cover E for T with $J = 2n/3 - k$. M corresponds to C', and other elements of T are covered by the subset of A. Thus, the sum of the weights of E is $2/3 \cdot (|T| - 3|C'|) + 1 \cdot |C'| = 2n/3 - |C'| \leq J$.

The 3DM can be restricted to the instance of our problem in polynomial time, thus our problem is NP-hard. □

The objective of minimizing the total real payment by requesters in Equation (4.4) is compatible to another objective of improving the real earnings of individual workers. For example, to reduce the real payment by requesters, we can make the batch larger and the worker can earn more from a larger batch than from a smaller batch. The following lemma ensures that workers can earn more when they undertake larger batches:

Lemma 4.1. *Let there be two batches, B_1 and B_2. It is assumed that $B_1 \subseteq B_2$ and that all tasks in $B_2 - B_1$ are executed after the tasks in B_1. If the two batches are assigned to worker w_i, we have: $Pay_{w_i}(B_1) \leq Pay_{w_i}(B_2)$.*

Proof. Let ξ_x denote the start time of the execution of task t_x, we have

$$\forall t_x, \ \ t_x \in B_2 \Rightarrow t_x \in B_1 \vee t_x \in (B_2 - B_1),$$

and $$\forall \mathrm{t_x} \in (\mathrm{B_2} - B_1), \ \ t_y \in B_1 \Rightarrow \xi_x \geq \xi_y.$$

According to Equation (4.3)

$$\begin{aligned} Pay_{w_i}(B_2) &= \sum_{x=1,\ldots,|B_1|} \psi(x) \cdot b_{t_x} + \sum_{x=|B_1|+1,\ldots,|B_2|} \psi(x) \cdot b_{t_x} \\ &= Pay_{w_i}(B_1) + \sum_{x=|B_1|+1,\ldots,|B_2|} \psi(x) \cdot b_{t_x}. \end{aligned}$$

$Therefore,$ $$Pay_{w_i}(B_1) \leq Pay_{w_i}(B_2)$$

□

Moreover, the batching for improving the worker's utility is submodular, which has the diminishing returns property, as stated in the following lemma:

Lemma 4.2. *Let there be two batches, B_1 and B_2. It is assumed that $B_1 \subseteq B_2$. Now suppose there is another set of tasks, T'. If the assigned worker is w_i and the tasks in T' are executed after the tasks in B_1 and B_2, we have $Pay_{w_i}(B_1 \bigcup T') - Pay_{w_i}(B_1) \geq Pay_{w_i}(B_2 \bigcup T') - Pay_{w_i}(B_2)$.*

Proof.

$$\text{We have} \quad \mathrm{Pay_{w_i}(B_1 \cup T')} = \sum_{x=1,...,|B_1|} \psi(x) \cdot b_{t_x} + \sum_{x=|B_1|+1,...,|B_1|+|T'|} \psi(x) \cdot b_{t_x}$$

$$\text{and} \quad \mathrm{Pay_{w_i}(B_1)} = \sum_{x=1,...,|B_1|} \psi(x) \cdot b_{t_x},$$

$$\text{thus } \mathrm{Pay_{w_i}(B_1 \cup T')} - \mathrm{Pay_{w_i}(B_1)} = \sum_{x=|B_1|+1,...,|B_1|+|T'|} \psi(x) \cdot b_{t_x}.$$

$$\text{We have} \quad \mathrm{Pay_{w_i}(B_2 \cup T')} = \sum_{x=1,...,|B_2|} \psi(x) \cdot b_{t_x} + \sum_{x=|B_2|+1,...,|B_2|+|T'|} \psi(x) \cdot b_{t_x}$$

$$\text{and} \quad \mathrm{Pay_{w_i}(B_2)} = \sum_{x=1,...,|B_2|} \psi(x) \cdot b_{t_x}$$

$$\text{thus } \mathrm{Pay_{w_i}(B_2 \cup T')} - \mathrm{Pay_{w_i}(B_2)} = \sum_{x=|B_2|+1,...,|B_2|+|T'|} \psi(x) \cdot b_{t_x}.$$

Because $|B_1| \leq |B_2|$, $\forall i = 1, ..., |T'|$ we have $b_{t_{|B_1|+i}} = b_{t_{|B_2|+i}}$, and according to the discounting function, we have

$$\psi(|B_1| + i) b_{t_{|B_1|+i}} \geq \psi(|B_2| + i) b_{t_{|B_2|+i}}, \ \forall i = 1, ..., |T'|.$$

Then, we have Lemma 4.2. □

According to Lemma 4.1, we should try to enlarge the batch if the constraints in our allocation objective can be satisfied; however, from Lemma 4.2, we can see that the marginal benefit will diminish as the batch size increases.

To solve the batch allocation problem, which is NP-hard, this thesis presents two heuristic approaches according to the tasks' required skills: the layered approach and the core-based approach. The layered approach uses hierarchical batching and can produce different layers of batching schemes; this approach can achieve good performance in minimizing requesters' total real payment, but may incur high computational cost

since all possible batches must be formed and observed. In contrast, the core-based approach first selects the core tasks that have minimum skill differences with other tasks, and then the batches are formed based on the core tasks. Although this approach cannot achieve the optimal batching results, it can significantly reduce the computational cost.

4.2 Layered Batch Formation and Allocation of Tasks

4.2.1 Layered-Batch Formation

With layered batching, we use the "bottom-up" pattern to iteratively form the batches step by step. In the first layer, each task forms an initial batch; the batching results in one layer feed into the next-higher layer. Therefore, the higher the layer is, the larger the batches in the layer are.

Let there be a set of tasks, $T = \{t_1, t_2, ..., t_n\}$. Suppose budget b_{t_x} is provided by the requester for each task t_x, $1 \leq x \leq n$, and the set of necessary skills required by t_x is $S_{t_x} = \{s_{x_1}, s_{x_2}, ..., s_{x_m}\}$. In a batch of tasks, it is assumed that the execution sequence is determined by the deadline of the tasks, i.e., t_x is executed before t_y if $d_{t_x} < d_{t_y}$. A batch is a set of time-sorted tasks and is denoted by an ordered tuple, e.g., $[t_1, t_2]$ is a batch that includes t_1 and t_2, and t_1 is executed before t_2.

Definition 4.1 (Candidate worker set of a batch)**.** *Let W denote the set of all workers in the crowd and B denote a batch of tasks. There is a subset $W_c \subseteq W$ that is defined as the candidate worker set of B if and only if its elements satisfy the wage and time constraints of all tasks in B : $\forall w \in W_c$, w's reservation wage is not higher than the discounted payment of each task, and w's completion time of every task $t \in B$, $time(t)$, does not exceed the deadline of t, d_t; see Equation (4.8).*

$$\forall t \in B, \;\; w \in W_c \;\Leftrightarrow\; \gamma_w \leq Pay_w(t) \wedge time_w(t) \leq d_t \tag{4.8}$$

The layered batch formation process is described as Algorithm 5. In the bottom layer, each task forms a batch initially, i.e., $Layer_1 = \{[t_1], [t_2], ..., [t_n]\}$. If any worker

satisfies the time and wage constraints of one batch (i.e., the constraints described in Equations (4.6) and (4.7)), it can be added to the set of candidate workers for the batch. Then, to form the higher layer, we iteratively generate all possible batches by appending one task to each batch of the lower layer; we can omit the batch in the layer if no workers satisfy the batch's time and wage constraints. Apparently, at layer i, each batch has i tasks, which are sorted according to their deadlines.

In Algorithm 5, 'w can satisfy Batch' means that the constraints of w's reservation wage and all tasks' deadlines are satisfied. Although the complexity of Algorithm 5 is high, $O(2^n)$, it is feasible to implement in practice. This is because the size of a batch that can be undertaken by an individual worker is limited according to the discounting mechanism; meanwhile, the time and wage constraints can remove some batches without candidate workers.

Theorem 4.2. *Let there be two batches B_1 and B_2 at $Layer_i$. Now it is assumed that B_1 and B_2 can be merged into a new batch, B_3, at $Layer_{i+x}(x > 0)$, which means $B_1 \cap B_2 = \phi$ and $B_1 \cup B_2 = B_3$. We use $Pay_w(B)$ to denote the real payment that worker w can get from batch B. Then, we have $Pay_w(B_1) + Pay_w(B_2) \geq Pay_w(B_3)$.*

Proof. All tasks in B_1, B_2, and B_3, are sorted according to their deadlines. Apparently, $\forall t \in B_1, B_2$, the order of task t in B_1 or B_2 is always no more than the order of t in B_3. The discounting function monotonically decreases as the task order increases. Thus, we have $Pay_w(B_1) + Pay_w(B_2) \geq Pay_w(B_3)$. □

According to Theorem 4.2, the larger a batch is, the more significantly the real payment by requesters will be discounted. Therefore, Theorem 4.2 ensures that the layered batch formation can approach our optimization objective of minimizing the real payment by requesters.

Theorem 4.3. *Suppose that when we use Algorithm 5 to perform batch formation for T, there are two layers: $Layer_i$ and $Layer_{i+1}$. Suppose we are given two batches $B_{i,j}$ and $B_{i+1,k}$, which respectively belong to $Layer_i$ and $Layer_{i+1}$, and assume $B_{i+1,k}$ is generated based on $B_{i,j}$. $BWset_{i,j}$ and $BWset_{i+1,k}$ are respectively the candidate worker sets of $B_{i,j}$ and $B_{i+1,k}$. Then, we have (1) $BWset_{i+1,k} \subseteq BWset_{i,j}$; (2) for a*

Algorithm 5: Layered-batch formation for tasks.
$/* \ T = \{t_1, t_2, ..., t_n\}$ denotes a set of n tasks; W is the set of workers in the crowd. $*/$

```
1  i = 1;
2  c = 0;
3  Layer_i = {};
4  LWS_i = {};// Candidate worker set for batches of layer i
5  forall the t_j ∈ T do
6      B_{i,j} = [t_j];
7      BWSet_{i,j} = {}; // Candidate worker set of the batch
8      forall the w ∈ W do
9          if w can satisfy B_{i,j} then
10             BWSet_{i,j} = BWSet_{i,j} ∪ {w};
11     if BWSet_{i,j} is not Empty then
12         Layer_i = Layer_i ∪ {B_{i,j}};
13         LWS_i = LWS_i ∪ {BWSet_{i,j}};
14 while c == 0 do
15     i = i + 1;
16     Layer_i = {};
17     LWS_i = {};
18     k = 1;
19     forall the B_{i-1,j} ∈ Layer_{i-1} do
20         forall the t_x ∈ T − {B_{i-1,j}} do
21             B_{i,k} = B_{i-1,j} + [t_x]; // Add a task to the batch
22             BWSet_{i,k} = {};
23             forall the w ∈ LWS_{i-1,j} do
24                 if w can satisfy B_{i,k} then
25                     BWSet_{i,k} = BWSet_{i,k} ∪ {w};
26             if BWSet is not Empty then
27                 Layer_i = Layer_i ∪ {B_{i,k}};
28                 LWS_i = LWS_i ∪ {BWSet_{i,k}};
29                 k = k + 1;
30     if Layer_i is Empty then
31         c = 1;
```

worker w, $Pay_w(B_{i,j}) < Pay_w(B_{i+1,k})$; and (3) if $BWset_{i,j}$ is empty, the generation of $Layer_{i+1}$ needs not to involve appending one task from batch $B_{i,j}$.

Proof. (1) We assume that there is an arbitrary candidate worker w_c who can satisfy $B_{i+1,k}$. Then, worker w_c, can solve all tasks of batch $B_{i+1,k}$ under the time and wage

constraints. Moreover, $B_{i,j} \subseteq B_{i+1,k}$ is known and batch $B_{i+1,k}$'s first i tasks are the same as the tasks in batch $B_{i,j}$. Therefore, worker w_c can also satisfy all tasks in batch $B_{i,j}$ under the constraints. Thus, $BWset_{i+1,k} \subseteq BWset_{i,j}$.

(2) $B_{i,j} \subseteq B_{i+1,k}$ is known, so $B_{i+1,k}$'s first i tasks are the same as the tasks in $B_{i,j}$. According to the discounting function, we have $Pay_w(B_{i,j}) = Pay_w(B_{i+1,k}[1, ..., i])$. Therefore, we have $Pay_w(B_{i,j}) < Pay_w(B_{i+1,k}[1, .., i]) + Pay_w(B_{i+1,k}[i + 1]) = Pay_w(B_{i+1,k})$. For worker w, $B_{i+1,k}$ can result in higher earnings, compared with $B_{i,j}$.

(3) From (1), if $BWset_{i,j}$ is empty, $BWset_{i+1,k}$ is also empty and the desired conclusion follows. □

Theorem 4.3 ensures the correctness of Algorithm 5 and guarantees that another aspect of our objective, improving the real earnings of the worker, can be achieved.

4.2.2 Worker's Crowdsourcing Value for a Batch of Tasks

The probability for a worker to be assigned a batch of tasks is influenced by the following four factors:

1) *The coverage degree of the worker's skills for the skills required by the tasks in the batch.* Let B be a batch. The set of skills required by all tasks in batch B is $S_B = \bigcup_{t_x \in B} S_{t_x}$, where S_{t_x} denotes the set of skills required by task t_x. For all $s_a \in S_B$, we use the number of tasks in the batch that require s_a as the weight of s_a in the batch, n_a. Now, we use a binary value to indicate whether worker w_i has the skill $s_a : b_{ia} = 1$ if w_i has skill s_a; $b_{ia} = 0$ if w_i does not have skill s_a. Then, the skill coverage degree of w_i for batch B will consider each skill's weight in the batch:

$$CS_{w_i}(B) = \left(\sum_{s_a \in S_B} (b_{ia} \cdot n_a) \right) / \left(\sum_{s_a \in S_B} n_a \right) \tag{4.9}$$

2) *The occupancy rate of the worker's reservation wage on the task's real payment.* Suppose the real payment that the requester of task t_x will pay to w_i is $Pay_{w_i}(t_x)$, which is calculated according to Equation (4.2). Let the reservation wage of w_i

be γ_{w_i}. Then, the occupancy rate of w_i's reservation wage on batch B's payment is

$$Occ_{w_i}(B) = \left(\sum_{t_x \in B} (\gamma_{w_i} / Pay_{w_i}(t_x)) \right) / |\{t_x | \forall t_x \in B\}| \tag{4.10}$$

If the assigned worker w_i has a lower $Occ_{w_i}(B)$, w_i may have more potential to distribute more utility to other workers who assist w_i in executing tasks in B.

3) *The estimated completion time of the worker for tasks in the batch.* When a worker wants to bid for a batch of tasks, he/she will estimate the completion time for each task in the batch according to the current real situation and his/her experiences. Moreover, the estimated completion time of a worker for a task can also be estimated by the system [130]. Let f_{ix} be the estimated completion time of worker w_i for task t_x. The index can be calculated as follows:

$$Est_{w_i}(B) = \left(\sum_{t_x \in B} f_{ix} \right) / |\{t_x | \forall t_x \in B\}| \tag{4.11}$$

4) *The reputation of the worker.* The reputation of the worker is mainly determined by the worker's past experiences in completing tasks; if the worker always successfully completed assigned tasks, his/her reputation is higher, and vice versa. We use R_{w_i} to denote the reputation of worker w_i.

People are often connected by social networks [23, 51, 131]. Some recent studies [26, 27] have shown that workers are also often connected through social networks. If a worker cannot complete a batch of tasks by himself/herself, he/she needs to seek the help of other workers through his/her social networks. Therefore, we present the following definition of crowdsourcing value to measure the probability of a worker being assigned a batch of tasks, which considers four factors that involve both the worker and other workers in his/her social network contexts:

Definition 4.2 (Crowdsourcing value of a worker for a batch of tasks)**.** *The crowdsourcing value of a worker, w_i, for a batch of tasks B is*

$$CV_{w_i}(B) = \alpha_1 \frac{\beta_1 \cdot CS_{w_i}(B) + \beta_2 \cdot R_{w_i}}{\beta_3 \cdot Occ_{w_i}(B) + \beta_4 \cdot Est_{w_i}(B)} + \alpha_2 \frac{\sum_{w_j \in (W - \{w_i\})} \left(\frac{\beta_1 \cdot CS_{w_j}(B) + \beta_2 \cdot R_{w_j}}{\beta_3 \cdot Occ_{w_j}(B) + \beta_4 \cdot Est_{w_j}(B)} / d_{ij} \right)}{|W - \{w_i\}|} \tag{4.12}$$

where W denotes the crowd of workers in the social network context, which includes w_i, and d_{ij} denotes the distance between w_i and w_j in the social network. The distance can be defined as the length of the shortest path between the two workers in the social network. $\alpha_1 + \alpha_2 = 1$*;* $\beta_1 + \beta_2 + \beta_3 + \beta_4 = 1$.

Although the number of worker set W is large, the crowdsourcing value can be calculated in a reasonable time. The reason is similar to the explantation on Page 34.

4.2.3 Allocation Algorithm

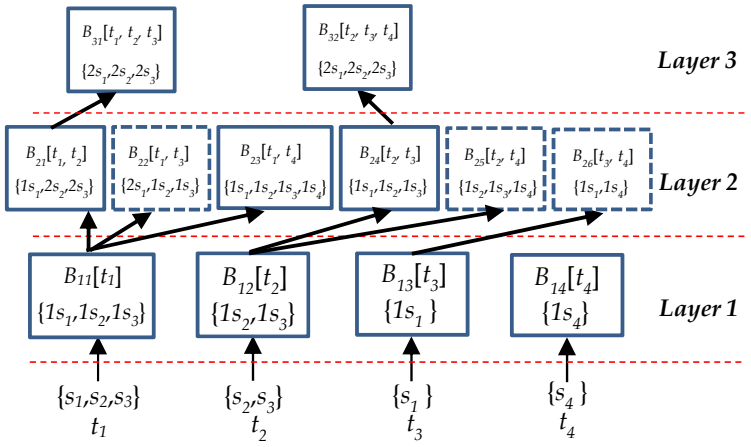
To achieve the optimization objective, we should first allocate the batches at the highest layer. If the system can find an appropriate worker for a batch and satisfy the constraints in Equations (4.4)-(4.7), the batch will be assigned to the worker. Therefore, assigned workers should be selected from the candidate worker set of the batch. If a batch at a layer can be allocated successfully, it is not necessary to consider the batches at the same or lower layers that have any common tasks with that batch, which can avoid the redundant allocation of common tasks between two batches. This process is repeated until all tasks have been allocated successfully or all workers have been considered.

Let the number of layers obtained by Algorithm 5 be λ and the set of available workers be W. To avoid the crowding of too many batches of tasks on certain workers, it is assumed that each worker can only undertake one batch in an allocation. The overall payment for batch B_{ij} is denoted as $Pay_w(B_{ij})$. The task allocation algorithm is shown as Algorithm 6. According to the discounting function, the larger the batches are, the more payment it can reduce. Since Algorithm 5 ensures that the sizes of batches at a higher layer are larger than those at a lower layer and the sizes of all batches within

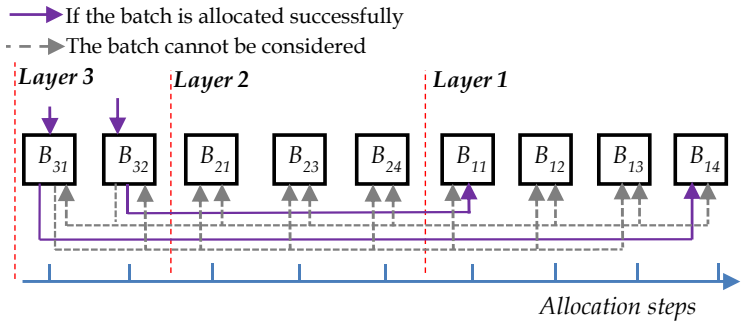
Algorithm 6: Batch layer-oriented task allocation.
$/* W$ is the set of workers in the crowd; $Layer_i$ and LWS_i are obtained by Algorithm 5. $*/$

```
1  i = λ;
2  c_1 = 0;
3  while i > 0 and c_1 == 0 do
4      c_2 = 0;
5      while c_2 == 0 do
6          min_budget = +∞;
7          W_c = {};
8          forall the B_ij ∈ Layer_i do
               // Find the minimum-payment batch
9              if min_budget > Pay_w(B_ij) then
10                 Batch = B_ij;
11                 BWSet = LWS_ij;
12             W_temp = W ∩ BWSet;
13             if W_temp is not Empty then
                   // Allocating the batch
14                 w_* = arg max_{w∈W_temp} CV_w(Batch);
15                 assign Batch to w_*;
16                 W = W − {w_*};
17                 for k=1 to i do
18                     forall the B_kx ∈ Layer_k do
19                         if B_kx ∩ Batch! = φ then
20                             Layer_k = Layer_k − {B_kx};
21             else
22                 Layer_i = Layer_i − {Batch};
23                 LWS_i = LWS_i − {BWSet};
24             if Layer_i is Empty then
25                 c_2 = 1;
26     i = i − 1;
27     if i > 0 then
28         if Layer_i or W is Empty then
29             c_1 = 1;
```

one layer are the same, we greedily allocate batches from upper layers to lower layers. The complexity of Algorithm 6 is $O(nm)$, where n is the number of tasks and m is the number of workers.



(a) Batch formation



(b) Batch allocation process

Figure 4.4: An example of layered batch formation and task allocation.

Example. Figure 4.4 shows an example of layered batch formation and task allocation, where there are four tasks $\{t_1, t_2, t_3, t_4\}$. Let the deadlines of the four tasks be $d_1 \leq d_2 \leq d_3 \leq d_4$. First, we can carry out batch formation according to Algorithm 5, which is shown as Figure 4.4a. The dotted framework represents the batch with no candidate workers who can satisfy both the time and reservation wage constraints of the batch. If batches B_{22}, B_{25} and B_{26} have no candidate workers, they and their descendant batches need not be considered. We assume there are no candidate workers who can satisfy the batch of four tasks, so the batch formation process ends at layer 3. Then, we can perform batch allocation according to Algorithm 6, which is shown as Figure 4.4b. The allocation process starts from batch B_{31} or B_{32}, depending on which

batch is associated with lower payment, because B_{31} and B_{32} are both at the top layer and their sizes are the same. The dotted line represents the batch that need not be considered later and the solid line denotes a possible allocation sequence.

4.3 Core-Based Batch Formation and Allocation of Tasks

The complexity of the above layered batch formation algorithm is $O(2^n)$, which may incur heavy computational cost when n (the number of tasks) is large. Moreover, the layered batch formation process may produce many useless batches. Therefore, we present a new suboptimal approach that can significantly reduce the computational cost.

Let there be two tasks, t_x and t_y. The sets of necessary skills required by t_x and t_y are $S_{t_x} = \{s_{x_1}, s_{x_2}, ..., s_{x_m}\}$ and $S_{t_y} = \{s_{y_1}, s_{y_2}, ..., s_{y_m}\}$, respectively. Then, the distance between tasks t_x and t_y is

$$\delta_{xy} = 1 - \frac{|S_{t_x} \cap S_{t_y}|}{|S_{t_x} \cup S_{t_y}|} \tag{4.13}$$

The main steps of the core-based batch formation and allocation approach are as follows:

1) Given a set of tasks T, we define the core task in T to be the one that has the minimum sum of distances to other tasks in T:

$$t_c = \arg\min_{t_x \in T} \sum_{t_y \in (T - \{t_x\})} \delta_{xy} \tag{4.14}$$

2) At first, t_c forms the initial batch; other tasks will be considered for integration into the batch with the core of t_c. In each round, the task of the batch with the minimum distance to t_c, t_x, is selected to be added into the batch. Then, the system checks whether there are candidate workers who can satisfy the two constraints in Equations (4.4)-(4.7). If candidate workers cannot be found, another task with

the second minimum distance is considered. This batch formation process for t_c is repeated until δ_{ci} is 1, i.e., no task has overlapping skills with t_c.

3) We can use $batch(t_c)$ to denote the batch that is formed with core task t_c. After the batch is formed, a worker w_* is assigned the batch.

$$w_* = \arg\max_{w_i \in W_c} CV_{w_i}(batch(t_c)) \tag{4.15}$$

4) $T = T - batch(t_c)$, we will repeat the above processes 1)→3) for the remaining tasks T. The stopping criterion of the whole core-based batch formation and allocation is that no new core tasks or workers can be found.

With the core-based approach, the order of the execution of tasks in a batch is determined by their distances to t_c, i.e., the smaller the skill distance of the task to t_c, the earlier the task can be executed. This idea is practical because in the execution of one task, the existing execution results of similar finished tasks can be utilized. The core-based batch formation and allocation algorithm is shown as Algorithm 7, where τ_{w,t_c} denotes the completion time of task t_c by worker w and d_{t_x} denotes the deadline of task t_x.

The worst-case complexity of Algorithm 7 is $O(nm)$, where n is the number of tasks and m is the number of workers. Therefore, the core-based approach can significantly reduce the computational cost compared to the layered approach, whose complexity is $O(2^n)$+$O(nm)$.

Theorem 4.4. *In the process of batch formation with core task t_c, we assume there are two batches, B_i and B_j, where $B_i \subseteq B_j$. W_i and W_j are the sets of candidate workers (who can satisfy the wage and time constraints of the batch) of B_i and B_j, respectively. Then, (1) $W_j \subseteq W_i$ and (2) $Pay_w(B_i) \leq Pay_w(B_j)$.*

Proof. (1) We can use reductio ad absurdum to prove this theorem. Assume there is a candidate worker $w' \in W_j$ and $w' \notin W_i$. Then, w' can solve all tasks of batch B_j under the time and wage constraints, but not those of batch B_i. However, $B_i \subseteq B_j$ implies that B_j's first i tasks are the same as those of B_i. Thus, worker w' can also solve the tasks

Algorithm 7: Core-Based Batch Formation and Allocation.
$/*T = \{t_1, t_2, ..., t_n\}$ denotes a set of n tasks; W is the set of workers in the crowd. $*/$

$c_1 = 0$;
while $c_1 == 0$ **do**
  $t_c = \arg\min_{t_x \in T} \sum_{t_y \in (T - \{t_x\})} \delta_{xy}$; `// Selecting the core task` t_c
  $W_{temp} = W$;
  $c_2 = 1$;
  $W_t = \{\}$;
  $Batch = \{\}$;
  forall the $w \in W_{temp}$ **do**
    if $\tau_{w,t_c} \leq d_{t_c}$ *and* $\gamma_w \leq Pay_w(t_c)$ **then**
      $time_w = \tau_{w,t_c}$;
      $W_t = W_t \cup \{w\}$;
  if $W_t \neq \phi$ **then**
    $Batch = \{t_c\}$;
    $c_2 = 0$;
    $W_{temp} = W_t$;
    $T_{temp} = T$;
  while $c_2 == 0$ **do**
    $t_b = \arg\min_{t_y \in T_{temp} \wedge \delta_{cy} \neq 0} \delta_{cy}$;
    if t_b *can be found* **then**
      $W_t = \{\}$;
      forall the $w \in W_{temp}$ **do**
        if $(\tau_{w,t_b} + time_w \leq d_{t_b}) \wedge (\gamma_w \leq Pay_w(t_b))$ **then**
          $W_t = W_t \cup \{w\}$;
          $time_w = time_w + \tau_{w,t_b}$;
      if $W_t \neq \{\}$ **then**
        $W_{temp} = W_t$;
        $Batch = Batch \cup \{t_b\}$;
      else
        $T_{temp} = T_{temp} - \{t_b\}$;
    else
      $c_2 = 1$;
  if $batch \neq \phi$ **then**
    $w_* = \arg\max_{w_i \in W_{temp}} CV_{w_i}(Batch)$;
    assign Batch to w_*;
    $W = W - \{w_*\}$;
    $T = T - batch$;
  if $T == \phi$ *or* $W == \phi$ **then**
    $c_1 = 1$;

in B_i. This is a contradiction; thus, $W_j \subseteq W_i$. (2) $B_i \subseteq B_j$ is known, so batch B_j's first i tasks are the same as those of batch B_i. According to the discounting function, $Pay_w(B_i) \leq Pay_w(B_j)$. □

Theorem 4.4 ensures the correctness of our core-cased batch formation and allocation algorithm and that our algorithm tends to select larger batches, which implies that it can approach the optimization objective in Equation (4.4).

4.4 Experiments

We now conduct experiments for our approaches on a real dataset and realistic settings for comparison with the previous benchmark approach: retail-style task allocation. Moreover, we compare the results of our approaches to those of the exhaustive batch allocation algorithm, which exhaustively enumerates all possible batch assignments.

4.4.1 Metric for Task Execution

In general, the successful execution of a task is affected by the following four factors: 1) the probability that the task's required skills are satisfied; 2) whether the task is finished before the deadline; 3) whether the assigned worker is satisfied with the payment; and 4) the assigned worker's reputation for reliable execution of the task. Let there be a worker w_i assigned to task t. The probability of successful execution of t by w_i, $p_{i,t}$, can be defined as follows:

$$p_{i,t} = s_{i,t} \cdot (R_{w_i}/R_{\max}) \cdot (time_{w_i}(t) \leq d_t) \cdot (Pay_{w_i}(t) \geq \gamma_{w_i}) \tag{4.16}$$

where

$$s_{i,t} = \prod_{k=1}^{m} e^{-D_k \cdot \tau} \tag{4.17}$$

If the completion time of t, $time_{w_i}(t)$, is later than d_t or the assigned worker's reservation wage γ_{w_i} is not satisfied by the real payment $Pay_{w_i}(t)$, $p_{i,t}$ is 0. $s_{i,t}$ indicates the possibility of w_i obtaining the required skills through himself/herself and his/her

social network. The probability of w_i getting s_k from a worker in w_i's social network is inversely proportional to the distance between w_i and the worker; we use D_k to denote the minimum distance of w_i to a worker with s_k in the social network; if the assigned worker has s_k, $D_k = 0$. m is the total number of skills required by the task. τ represents the decay factor. R_i is the reputation of w_i and R_{max} is the upper bound of the reputation.

4.4.2 Benchmark Approach

On many current leading complex task-oriented crowdsourcing websites, such as Upwork.com, Freelancer.com, and zbj.com, each task is often allocated non-redundantly and independently to a professional worker without decomposition, which is *retail-style task allocation approach*. Generally, the following three factors are considered: 1) the skill coverage degree of the candidate worker for the task; 2) the reputation of the candidate worker; and 3) the reservation wage of the candidate worker.

Our two proposed batch allocation approaches consider the assigned worker's associations with other workers in his/her social networks; for fair comparison, we extend the traditional retail-style allocation approach by considering the social network contexts of candidate workers. Suppose there is a crowd of workers W. For each $w_i \in W$, the set of skills of w_i is S_{w_i}, the reservation wage of w_i is γ_{w_i} and the reputation of w_i is R_{w_i}. Let t be a task and let the set of necessary skills to complete t be S_t. d_{ij} denotes the distance between worker w_i and another worker w_j in the social network. We define the crowdsourcing value, $CV_{w_i}(t)$, as the probability of w_i being assigned t, which is shown as follows:

$$CV_{w_i}(t) = \alpha_1 \cdot \frac{\beta_1 \cdot |S_t \cap S_{w_i}| + \beta_2 \cdot R_{w_i}}{\beta_3 \cdot \gamma_{w_i}} + \alpha_2 \cdot \frac{\sum_{w_j \in (W - \{w_i\})} \left(\frac{\beta_1 \cdot |(S_t - S_{w_i}) \cap S_{w_j}| + \beta_2 \cdot R_{w_j}}{\beta_3 \cdot \gamma_{w_j} \cdot d_{ij}} \right)}{|W - \{w_i\}|} \tag{4.18}$$

where α_1 and α_2 denote the relative contributions of w_i and w_i's social contextual workers to $CV_{w_i}(t)$, and β_1, β_2, and β_3 denote the relative importance of the three factors. $CV_{w_i}(t)$ is different from $CV_{w_i}(B)$ in Equation (4.12), because $CV_{w_i}(t)$ is the crowdsourcing value for an individual task but $CV_{w_i}(B)$ is the one for a batch of tasks. The

tasks are allocated one by one in increasing order of deadline. Each task is greedily assigned to the worker with the maximum crowdsourcing value that satisfies the time and wage constraints.

4.4.3 Dataset and Settings

Now many representative studies on crowdsourcing of complex tasks are validated by simulation experiments [44, 78, 80, 132], since the online experiments on complex tasks may produce very high unaffordable payments. The simulation experimental results with real data and realistic settings are generally accepted in this area. In this thesis, we also make simulation experiments by using real data set collected from Upwork.com and making realistic settings by referring real-world crowdsourcing processes from Upwork.com.

We collect the data of workers and tasks from Upwork.com. Worker data are crawled from search result pages, which include the skills, historical tasks, reputations of 4409 workers. The data that are extracted for each worker contain more than one task completion record. The average income for historical tasks of a worker is assumed to be the worker's expected wage. Tasks are selected from the "web-mobile-software-development" category at Upwork.com, which include 4096 tasks' budgets, required skills, and publishing time. To ensure the generality of our experimental results, we delete the data for extreme cases as follows: the workers whose wages are less than \$200 or more than \$700 are excluded; the tasks whose budgets are greater than \$1400 or less than \$400 are excluded; and tasks that require rare skills (for which number of workers is no more than 2) are excluded. Finally, there are 864 workers and 354 tasks in the dataset that is used for our experiments. By considering the general deadlines of tasks at the website, the deadline of each task d_t is a random value in [67, 127]; thus, the mean value is close to 97.7. The estimated task completion time of a worker is a random number in the range $[d_t/2, d_t]$. These workers are interconnected by three typical social network structures [133]: random networks, scale-free networks, and small-world networks.

We now describe how these networks are constructed.

- *Random Network* [133]. The random network is generated by randomly adding connections between workers, which results in the network average degree being equal to 6.

- *Scale-Free Network* [134]. The scale-free network starts with $m_0 = 12$ workers which are all connected. At each step, we add a new worker and connect this new worker to three workers already existing in the network with a probability. The probability that a new worker v connects an existing worker u is proportional to the degree of u. Finally, the final network average degree is about 6.

- *Small-World Network* [135]. This network starts from a regular ring lattice in which each worker connects with 6 nearest neighbors, and this worker has a probability of $p = 0.2$ of rewiring each connection to another worker. Finally, the final network average degree is 6.

TABLE 4.1: The properties of the three social network structures.

Properties / Network	Diameter	Average shortest path length	Clustering coefficient	Average degree
$Random$	7.02	4.00	0.01	6
$Scale-free$	5.91	3.29	0.12	6.01
$Small\ world$	8.51	4.98	0.63	6

In summary, the properties of the three types of social network structures among workers in experiments are shown in Table 4.1.

4.4.4 Performance Indices

According to the optimization objective, we define below four indices to evaluate the performances of our approaches, *layered batch allocation (Layer)* and *core-based batch allocation (Core)*, in comparison with the previous benchmark approach, *retail-style allocation (Retail)*:

- *Task Completion Proportion*: According to Equation (4.16), we assume the probability of completing task x is p_x and there are a total of m tasks. The completion proportion of all tasks is defined as $\sum_{x=1}^{m} p_x/m$. Higher completion proportion indicates that more tasks can be completed successfully.

- *Total Payments by Requesters*: We define this index as the sum of all requesters' real payments to evaluate our optimization objective of minimizing the requesters' total real payment.
- *Average Income of Workers*: We define this index as the average of all assigned workers' real earnings within a given duration to evaluate another aspect of our optimization objective: improving the real hourly wages of workers.
- *Task Allocation Time (Running Time of Algorithm)*: This index is used to measure the computational efficiency of the task allocation approaches. The running time of our approaches includes the batch formation and allocation processes.

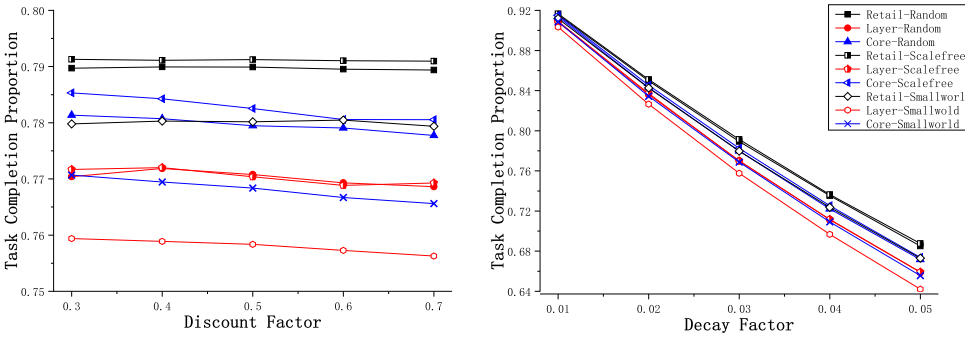
4.4.5 Experimental Results

There are four main factors that may influence the results: 1) discounting factor σ for the discounting function in Equation (4.2) (which is set ranging from 0.3 to 0.7); 2) decay factor τ in Equation (4.17), which influences the decay rate of the probability to get skills as a function of social distance; 3) structures of social networks, which affect the cooperation among workers; and 4) number of tasks, which may influence the performance. The experiments test the impacts of these four factors on the four performance indices.

4.4.5.1 Tests on Task Completion Proportion

Now we test the performances on *Task Completion Proportion* of the three approaches.

Figure 4.5a shows the results on *Task Completion Proportion* of the three approaches under different social network structures. The decay factor is set to a fixed value, $\tau = 0.03$. In the same network structure, the three allocation approaches have similar *Task Completion Proportion* performances, and core-based batch allocation is slightly better than layered batch allocation; the reason is that layered batch allocation greedily selects batches with lower budgets in the process of batch allocation. For the three types of social networks, *Task Completion Proportion* performance under Random Networks



(a) Different social network structures (b) The effects of the decay factor

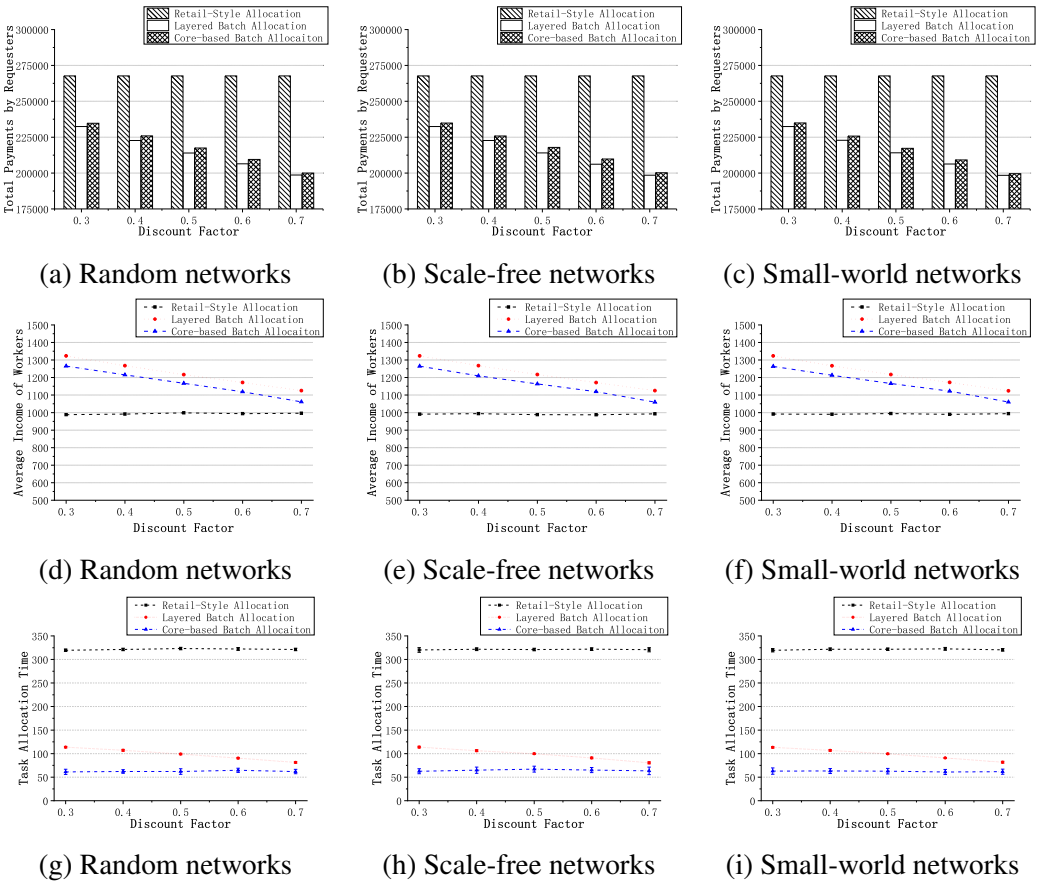
Figure 4.5: Tests on the performances of Task Completion Proportion

is better than that under Small-World Networks; this is because *Task Completion Proportion* is related to the social distance from a worker with the required skill according to Equation (4.17), and the average distance between two workers in a random network is slightly smaller than that in a small-world network. Moreover, in the three networks, the scale-free network's average distance is the smallest, which causes it to have the highest *Task Completion Proportion* performance.

Figure 4.5b shows the effects of the decay factor. With the growth of the decay factor (from $\tau = 0.01$ to $\tau = 0.05$, while $\sigma = 0.3$), the *Task Completion Proportion* performances of all approaches decrease drastically. The *Task Completion Proportion* performance under random networks is slightly higher than that under small-world networks. Moreover, the scale-free network's smallest average distance causes it to have the highest *Task Completion Proportion* performance.

4.4.5.2 Tests on Total Payment by Requesters, Average Income of Workers, and Task Allocation Time

This series of experiments tests the performances on *Total Payments by Requesters*, *Average Income of Workers*, and *Task Allocation Time* of the three approaches under different discounting factors and social networks. Because the decay factor in Equation (4.17) only influences the stage of task execution and these three performance indices are determined in the stage of task allocation, it is not necessary to test the effects of the decay factor on these three performance indices. Therefore, the decay factor can be set to 0.03.



(a) Random networks (b) Scale-free networks (c) Small-world networks

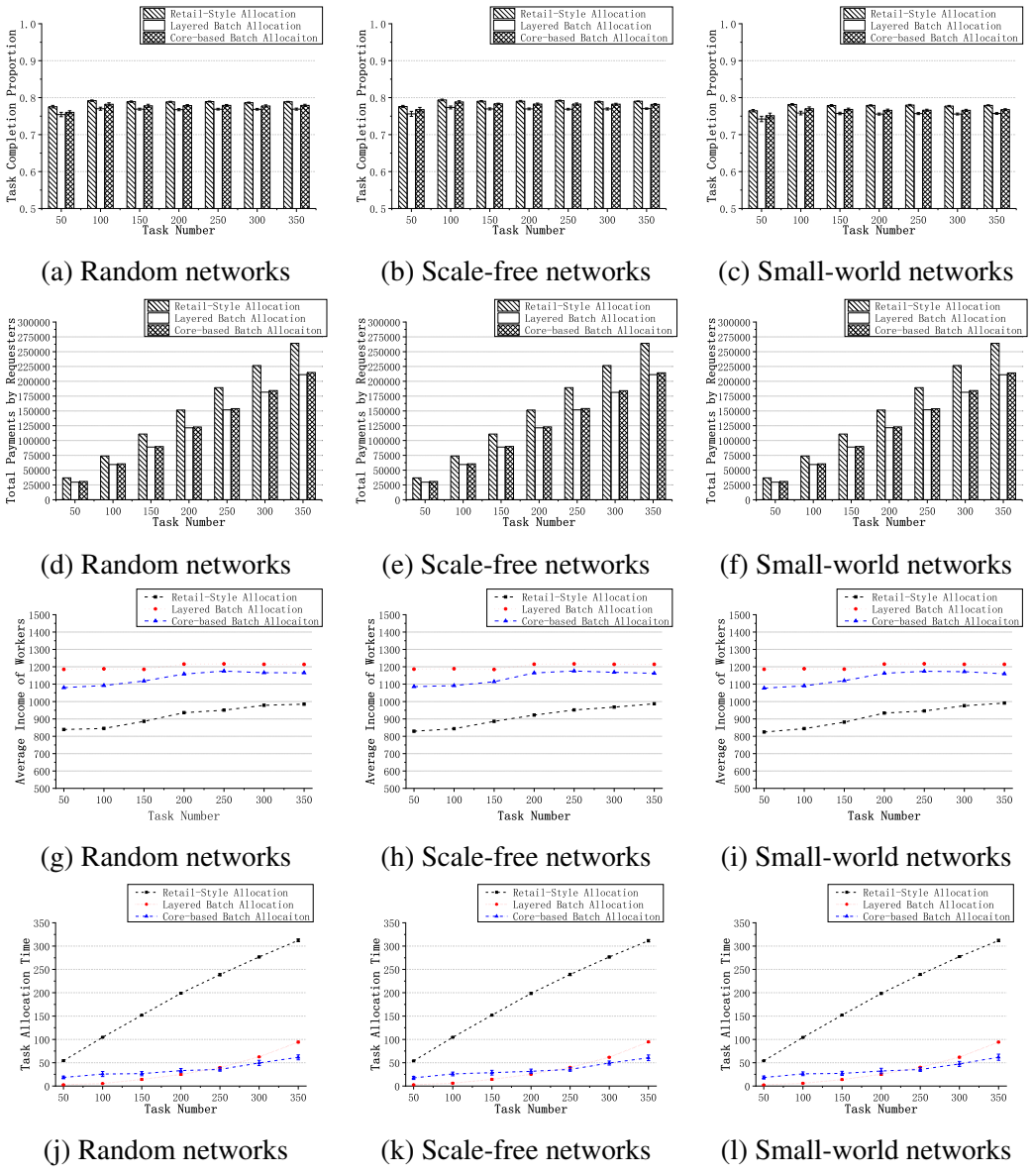
(d) Random networks (e) Scale-free networks (f) Small-world networks

(g) Random networks (h) Scale-free networks (i) Small-world networks

Figure 4.6: The scalability to the number of tasks in terms of the four performance indices.

Figure 4.6 shows the effects of the discounting factor on the three performance indices. We can see that our two approaches achieve lower *Total Payments by Requesters* and higher *Average Income of Workers* while using less allocation time than the retail-style allocation approach. Comparing with core-based batch allocation, layered batch allocation performs better on *Total Payments by Requesters* and *Average Income of Workers*, but it needs to consume more allocation time. In our approaches, the values of *Total Payments by Requesters* and *Average Income of Workers* decrease with the increase of the discounting factor. With the increase of the discounting factor, the runtime of our layered batch allocation declines slightly; the reason is that a higher discounting factor means a lower payment for the task, which may reduce the size of the candidate worker set in layered batch allocation, which speeds up the algorithm. The effects of the network structures are not significant; this is because these three performance indices are related to task allocation, whereas social network structures have more influence on task execution.

4.4.5.3 Tests on the Scalability to the Number of Tasks



(a) Random networks (b) Scale-free networks (c) Small-world networks

(d) Random networks (e) Scale-free networks (f) Small-world networks

(g) Random networks (h) Scale-free networks (i) Small-world networks

(j) Random networks (k) Scale-free networks (l) Small-world networks

Figure 4.7: The scalability to the number of tasks in terms of the five performance indices

This section examines the performances of three approaches under different numbers of tasks; the results are shown as Figure 4.7. Without loss of generality, we set $\tau = 0.03$ and $\sigma = 0.5$. We can see that the number of tasks has no significant influence on *Task Completion Proportion*. On *Total Payments by Requesters*, the advantage of our approaches becomes more obvious with the increase of the number of tasks. The performances on *Average Income of Workers* of the three approaches increase with the increase of the number of tasks. Comparing with the retail-style task allocation, our

two batch allocation approaches both have better scalability performances in terms of runtime. In addition, when number of tasks is small, the number of possible batches is limited, and the greedy batch allocation of the layered approach may spend less time than the core-based batch allocation; with the increase of the number of tasks, the core-based batch allocation may spend less time due to its lower complexity.

4.4.5.4 Comparison with the Exhaustive Batch Allocation Algorithm

We design an exhaustive algorithm based on recursion to enumerate all possible batch assignments. The recurrence formula is as follows:

$$ExactBatchAlloc(T, W) = \min_{B \in C(T)-\{\phi\}} (ExactBatchAlloc(T-B, W-\{w_B^*\}) + Pay_w(B))$$

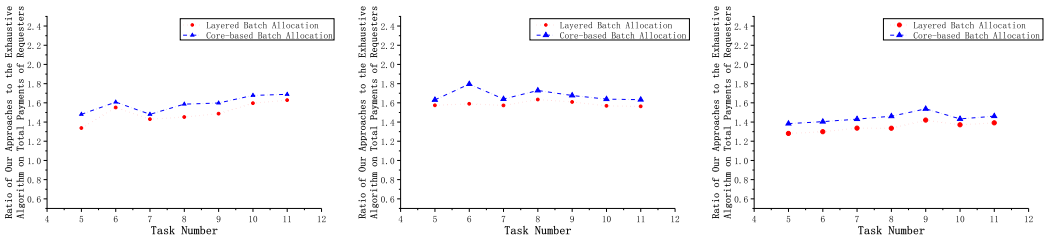
where $ExactBatchAlloc(T, W)$ represents the minimum total payment of all possible batch allocations. Let T denote the set of all tasks and $C(T)$ be the power set of T. We assume $B \in C(T) - \{\phi\}$ is a batch of the task set T. All tasks in B should be executed in increasing order of deadline. w_B^* represents the worker with the highest crowdsourcing value for batch B. $Pay_w(B)$ is the payment associated with batch B. The exhaustive algorithm is shown as Algorithm 8.

The exhaustive algorithm has very high computational complexity, $O(n!)$, so it can only be used for small number of tasks; however, it can be used as a benchmark for comparison. Because our main optimization objective is to minimize the total real payment by requesters, here we only compare the performance index of *Total Payments by Requesters*.

First, we randomly select a certain number of tasks from the set of all tasks and only retain workers who have skills related to the tasks. Then, we compare the performances of our two approaches with that of the exhaustive algorithm. We set $\tau = 0.03$ and $\sigma = 0.5$. We calculate the ratios of our approaches' results to those of the exhaustive algorithm, shown as Figure 4.8. The *Total Payments by Requesters* of our approaches are approximately 1.6 times that of exhaustive algorithm for a small number of tasks.

Algorithm 8: Exhaustive Batch Allocation Algorithm.
$/* T = \{t_1, t_2, ..., t_n\}$ denotes the set of tasks; W denotes the set of workers. $*/$

```
1  Total_budget = 0;
2  Min_budget = ∞;
3  Function ExactBatchAllocation(T, |T|, W)
4  if |T| == 0 or |W| == 0 then
       // Batch allocations are over
5      if Total_budget < Min_budget then
6          Min_budget = Total_budget;
7          return
8  C(T) = {S|S ⊆ T}; // C(T) is the power set of T
9  forall the B ∈ C(T) − {φ} do
10     BWSet = {};
11     forall the w ∈ W do
12         if w can solve Batch B then
13             BWSet = BWSet ∪ {w};
14     if BWSet ≠ {} then
15         w* = arg max_{w∈BWSet} CV_w(B);
16         Total_budget = Total_budget + Pay_w(B);
17         W_t = W − {w*};
18         T_t = T − B;
19         ExactBatchAllocation(T_t, |T_t|, W_t);
20         Total_budget = Total_budget − Pay_w(B);
21 End of the Function
```



(a) Random networks (b) Scale-free networks (c) Small-world networks

Figure 4.8: Comparison among our two approaches and the exhaustive algorithm.

In summary, according to a comparison with the retail-style task allocation approach, our two approaches both achieve better performances in terms of *Total Payments by Requesters* and *Average Income of Workers*, while maintaining higher *Task Completion Proportion* and consuming less task allocation time. Between our two approaches, the core-based batch allocation approach incurs lower time cost and produces suboptimal results. Moreover, although our approaches' *Total Payments by Requesters*

are 1.6 times that of the exhaustive algorithm, our approaches can scale to large number of tasks, whereas the exhaustive algorithm can only be applied when the number of tasks is very small.

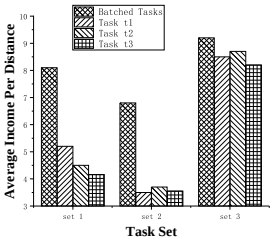
Certainly, because our experiments are conducted in a simulation environment with real data and realistic settings, there are some limitations with the current results.

- The strategies and skills of workers are assumed to be fixed. In fact, real workers may sometimes switch their strategies and their skills may change with their learning and execution histories. Therefore, the dynamics of strategies and skills of workers in the batch crowdsourcing should be explored in the future research.
- The crowdsourcing system is assumed to be dependable. In fact, the system may be undependable in real world; for example, some workers may join and depart the system dynamically, some workers may abandon the assigned tasks temporally, and the operations of the system may be undependable. Therefore, the real-world undependable systems may produce some unpredictable results. Therefore, in the future we will investigate the batching crowdsourcing in undependable environments.

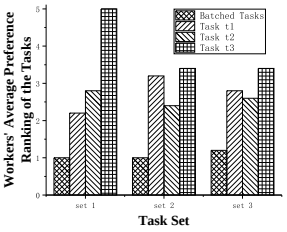
4.4.6 Real Online Experiments

Besides the above simulation experiments with real dataset and realistic settings, now we conduct a series of real online experiments to validate the effectiveness of our batch allocation approach.

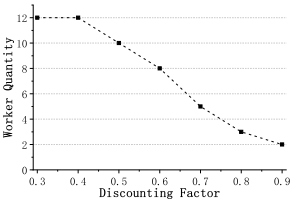
In this series of real online experiments, we do not use the popular crowdsourcing platforms such as Upwork and Freelancer, but we use a very popular Chinese social media WeChat (https://en.wikipedia.org/wiki/WeChat). The reason is that at current crowdsourcing platforms, workers can often find only one task which is posted by a requester, and they will not be aware of the other related tasks posted by the same requester. Therefore, the batch experiments cannot be conducted at those crowdsourcing platforms. In WeChat, there is a feature named *people nearby* which can be used to recruit workers. In detail, we post advertisements in the feature *moments* and make the



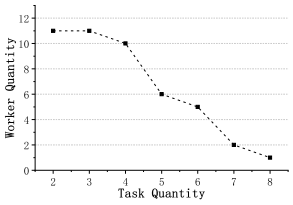
(a) Cost-performance index of workers



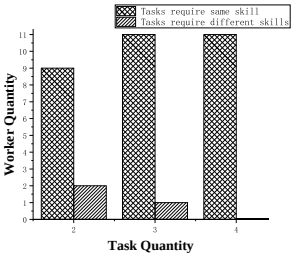
(b) The efficiency of allocation



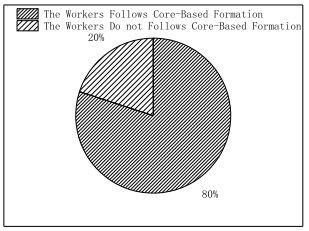
(c) The effect of discounting factor



(d) The effect of task quantity



(e) The performance of layered batch allocation



(f) The performance of core-based batch allocation

Figure 4.9: Tests on the performances in real online experiments.

advertisements public to the people nearby. Then, the nearby WeChat users who are interested with our advertisements will contact us through WeChat and then may undertake the tasks. Because of the limited budget and non-professional workers, we only post simple tasks in this experiment. The main tasks include three task sets, and each task set is to ask workers to use their phones to take three pictures in three different designated locations. These three locations are in a same region, so they are not far away. The first task set is asking workers to take pictures in Xuanwu Lake Park (a famous park in Nanjing, China) which has a big lake inside and there are several collected islands in the lake. The task set includes four different tasks: task t_1 is taking a picture in island a, task t_2 is taking a picture in island b, task t_3 is taking a picture in island c, and task t_4 is the batch of the former tasks which take three pictures in island a, b, and c respectively. The rewards of tasks t_1, t_2, and t_3 are same, 5 CNY for each, and the reward of the batched task is 11.4 CNY (the discounting factor is set as 0.3). The task sets 2 and 3 are similar to task set 1, and the main difference is just that the regions are two different universities and the corresponding tasks t_1, t_2, and t_3 are taking pictures of some buildings on campuses. The testing results of real online experiments are shown in Figure 4.9.

4.4.6.1 Tests on the Overall Effectiveness of Batch Allocation

Figure 4.9a shows the result of cost-performance index of workers. The cost-performance index is defined as the ratio between a worker's income (CNY) and the journey (kilometers) he/she spends. The higher the value is, the higher rewards the worker can earn for the same effort. We can see that in each task set, workers have the highest cost-performance on batched tasks. Figure 4.9b compares the allocation efficiency between batched task and single tasks. In the advertisement, we let workers know that they can take only one task in a task set, and ask workers to give their selections orders on each task, e.g., t_2, t_3, t_1, t_4. Then, the preference ranking of t_1, t_2, t_3 and t_4 are 3, 1, 2, and 4 respectively, which means that the worker's most favorite task in the task set is t_2 and least favorite task is t_4. From the figure, we can see that almost all workers prefer to undertake the batched task. The reason is that comparing with the effort spent on the other tasks in the same task set, a worker just spends a little more effort/time on the batched task but earns much higher reward. Therefore, batched tasks can be allocated more easily than the corresponding retail-style tasks. Meanwhile, requesters can save money if the batch allocation is used. In these experiments, the requester can save 3.6 CNY for each task set when he/she batches the tasks together and allocates them to the same worker.

4.4.6.2 Tests on the Effectiveness of Parameters

Figure 4.9c shows the effect of discounting factor. In this test, there are three tasks in a batch. We increase the discounting factor gradually which means that the worker's income will be decreased accordingly. We can see that as the discounting factor increases, less workers are willing to undertake the task. When the discounting factor is 0.9, the worker can only earn 0.8 CNY for the third task in the batch. Such a discount would be good for the requester, but it is bad for the worker because the origin salary of a task is 5 CNY. In the next test, we fix the discounting factor as 0.5 and increase the number of tasks one by one in the batch. The results are shown in Figure 4.9d. We can see that as the tasks' quantity increases, it will have less workers who are willing to undertake the tasks because the discounting function decreases the reward of the latter tasks a lot.

Certainly, in these tests, although it can only get very little reward from the latter tasks, some workers are still willing to undertake the task set. The reason is that the worker has earned enough reward from the former tasks, and the task is quite simple such that the worker does not need to pay too much cost for an extra task.

4.4.6.3 Tests on the Effectiveness of the Two Batch Allocation Approaches

First we test the effectiveness of the layered batch allocation, shown in Fig 4.9e. In the test, we consider the skill is related to the distance between the worker's and the task's locations. If a worker is close to the task's location, then the worker has the skill to finish that task. Hence, we can say that the tasks in the same region required the same skills and the tasks in the different region required different skills. We use two different ways of combining tasks to construct batches. The first way is combing the tasks that require same skills, i.e., the tasks' related locations are in the same region. The second way is combing the tasks that require different skills, i.e., the tasks' related locations are in different regions. Here if a worker is not far away from the tasks' corresponding locations, the worker is considered as the candidate worker who has the required skills of the tasks. In fact, the worker is willing to undertake the nearby tasks but not to the far away tasks. We can see that if we combine the tasks that require the same skills, there would be more workers who are willing to undertake the batched task, but if we combine the tasks that require different skills, there would be very few workers who are willing to undertake the batched task. Therefore, we can allocate the tasks successfully when we use the right combination way in the layered batch method.

Next, we test the effectiveness of the core-based batch allocation. In task set 1, the three inlands a, b, and c are connected serially. Usually, if a worker want to go to island c he/she will pass a and b in advance, and if the worker want to go to b he/she will pass a in advance. Then, task t_1 which is to take picture on island a requires the most basic skills, t_2 (taking picture on island b) requires more skills than t_1, and t_3 (taking picture on island c) requires more skills than t_2. Tasks t_1, t_2, and t_3 construct the batched task. In the test, there are 5 workers who finish the batch tasks, and among them 4 workers finish the batched task with the sequences of t_1, t_2, and t_3. We can see that the

majority workers use the same way of core-based batch formation, so the core-based batch approach is reasonable. The result is shown as Fig 4.9f. Therefore, in reality many workers will follow core-based formation, which can show the effectiveness of the core-based batch allocation.

4.5 Summary

Because each task needs to be allocated independently from the scratch, the retail-style task allocation cannot scale to large numbers of concurrent tasks due to the large computational cost of allocation. Moreover, many workers' skills and time may not be fully utilized since they often undertake a very small number of tasks contemporaneously.

To address these drawbacks, we present a batch allocation method with the objective of reducing requesters' total real payment as well as improving each worker' earnings. First we prove that the optimal batch allocation is NP-hard; then, we present two approaches: layered batch allocation, which can achieve better performance but may incur higher computational cost, and core-based batch allocation, which may achieve suboptimal performance but can significantly reduce the computational cost.

This chapter performs theoretical analyses and extensive experiments on real data to show the effectiveness and advantages of the proposed approaches. First, the optimization objective of our approaches of reducing the requesters' total real payment and improving workers' real earnings is validated by comparison with the benchmark retail-style task allocation. Then, it is demonstrated that our approaches can achieve higher successful task completion probability and consume less task allocation time by comparison with the retail-style task allocation. Moreover, we conduct experiments on the exhaustive batch allocation algorithm for very small numbers of tasks, and it is found that the total payment by requesters with our two approaches are approximately 1.6 times that of the exhaustive algorithm (the exhaustive algorithm can only be applied when the number of tasks is very small, whereas our approaches can scale to the big number of tasks). We also conduct real online experiments which testify the advantages of the proposed batch task allocation.

Chapter 5

Distributed Team Formation for a Batch of Tasks in Crowdsourcing

Team formation has been extensively studied in crowdsourcing, in which a set of workers are hired to form a team to complete a complex task collaboratively. However, existing studies have two typical drawbacks: 1) each team is created for only one task, which may be costly and cannot accommodate crowdsourcing markets with a large number of tasks; and 2) most existing studies form teams in a centralized manner by the requesters, which may place a heavy burden on requesters. In fact, we observe that many complex tasks at real-world crowdsourcing platforms have similar skill requirements and workers are often connected through social networks. Therefore, this chapter explores distributed team formation for a batch of tasks to address the drawbacks in existing studies, in which similar tasks can be addressed in a batch to reduce computational costs and workers can self-organize through their social networks to form teams. To solve such an NP-hard problem, this chapter presents two approaches: one is to form a fixed team for all tasks in the batch; the other is to form a basic team that can be dynamically adjusted for each task in the batch. In comparison, the former approach has lower computational complexity but the latter approach performs better in reducing the total payments by requesters. With the experiments on a real-world dataset comparing with previous benchmark approaches, it is shown that the presented approaches have better performance in saving the costs of forming teams, payments by requesters,

and communication among team members; moreover, the presented approaches have higher success rate of tasks and much better scalability.

The rest of this chapter is organized as follows. In Section 5.1, we present the motivation and problem description; in Section 5.2, we present the distributed formation approach of a fixed team for a batch of tasks; in Section 5.3, we present the distributed formation approach of a dynamic team for a batch of tasks; in Section 5.4, we provide experimental results; in Section 5.5, we conclude the chapter.

5.1 Motivation and Problem Description

5.1.1 Motivation

Section 4.1.1 of Chapter 4 has shown that: 1) it is common that many complex tasks within the same category have strong similarities, so *it may be useful to combine tasks that are highly similar as a batch and allocate them to the same workers, which may save computational costs*; 2) and most workers undertake only a small number of tasks contemporaneously in the real world, so *it is possible to allocate more than one task to a worker, which can utilize the worker's skills as much as possible*. Section 3.1.2 of Chapter 3 has shown that it is common that workers often collaborate through social networks, so *workers can self-organize teams through social networks*.

5.1.2 Problem Description

5.1.2.1 Discounting Mechanism for a Batch of Tasks

A batch of tasks denotes a collection of more than one task, such as tasks that have similar skill requirements, tasks attributed to the same category at a crowdsourcing website, or tasks published by the same requester. Let there be a batch of tasks, $B = \{t_1, t_2, ..., t_{|B|}\}$, where $|B|$ denotes the number of tasks in the batch.

Let there be two tasks t_x and t_y, $t_x, t_y \in B$. The sets of necessary skills required by t_x and t_y are $S_{t_x} = \{s_{x1}, s_{x2}, ..., s_{xm}\}$ and $S_{t_y} = \{s_{y1}, s_{y2}, ..., s_{yn}\}$, respectively, where

m and n denote the numbers of skills required by t_x and t_y. The skill distance between tasks t_x and t_y is:

$$\delta_{x,y} = 1 - \frac{|S_{t_x} \cap S_{t_y}|}{|S_{t_x} \cup S_{t_y}|} \tag{5.1}$$

Then, the *diversity of all tasks in* B is defined as:

$$\theta(B) = \frac{\sum_{x=1}^{|B|} \sum_{y=1}^{|B|} \delta_{x,y}}{2 \cdot |B|} \tag{5.2}$$

As we have explained the advantage of discounting in Section 4.1.2.1 of Chapter 4, we introduce discounting mechanism here. Let a batch of tasks, B, be allocated to a worker at once and is queueing for execution by that worker, w_i. The real payment to w_i for performing the tasks in B can be discounted according to the task number and diversity of B. Given an original utility $op_{w_i}(t_x)$ that the requester of task t_x should pay to w_i, we can define the *wholesale discounting function* as follows:

$$Pay_{w_i}(B) = \sum_{t_x \in B} \psi\left(\frac{|B|}{\theta(B)+1}\right) \cdot op_{w_i}(t_x)) \tag{5.3}$$

where ψ is the discounting function, $0 \leq \psi(X) \leq 1$, and the value of $\psi(X)$ decreases monotonically from 1 to 0 with the increase of X. We use '$\theta(B)+1$' to avoid a situation where the denominator is 0 due to $\theta(B)$ being 0.

If tasks in B are allocated to w_i one by one, discounting can be done progressively for each task in B. We can define the *progressive discounting function* as follows:

$$Pay_{w_i}(B) = \sum_{x=1,\cdots,|B|} \psi\left(\frac{x}{\delta_{x-1,x}+1}\right) \cdot op_{w_i}(t_x) \tag{5.4}$$

where '$\delta_{x-1,x}+1$' is used to avoid a situation where the denominator is 0 due to $\delta_{x-1,x}$ being 0; $\delta_{0,1} = 1$.

5.1.2.2 Cost of Team Formation-Based Crowdsourcing for a Batch of Tasks

In general, the cost of team formation-based crowdsourcing in existing benchmark studies [43, 55–57] includes three parts: costs of forming teams, payments by requesters, and communication among team members.

1) *The cost of forming teams.* In general, the cost of forming a team can be abstracted as a monotonically increasing function on the number of recruitment and elimination events of team members [55]. We can use $f(X)$ to denote a monotonically increasing function whose argument is X.

   Let there be a batch of tasks B. If we form a fixed team for all tasks in B, $W_B = \{w_1, ..., w_n\}$, where n denotes the number of workers in the team, the cost of forming a team for B is $C_{Form}(B) = f(|W_B|)$.

   If we form a dynamic team for B, the members will be dynamically adjusted for each task in the batch. Let there be two tasks: $t_x, t_y \in B$. The teams actually performing t_x and t_y are W_{t_x} and W_{t_y} , respectively; then, the number of elimination and recruitment events of team members from W_{t_x} to W_{t_y} is $|W_{t_x} - W_{t_y}| + |W_{t_y} - W_{t_x}|$. If the execution sequence of tasks in B is from t_1 to $t_{|B|}$, then the cost of team formation for B is:

$$C_{Form}(B) = f\left(|W_{t_1}| + \sum_{x=1,...,|B|-1} (|W_{t_x} - W_{t_{x+1}}| + |W_{t_{x+1}} - W_{t_x}|)\right) \quad (5.5)$$

2) *The cost of requesters paying team members for performing tasks.* If a fixed team is formed for a batch of tasks, some team members may not make real contributions for some tasks in the batch. Let $T_i(B)$ be the set of tasks in B that are actually performed by worker w_i, $T_i(B) \subseteq B$. $\forall t_x \in B$, the budget of t_x is b_{t_x}. The set of skills required by t_x is S_{t_x}; let $\overline{S_{t_x}}$ be the set of skills for t_x that are currently lacking; the set of skills of w_i that actually contribute to performing t_x is $\overline{S_{t_x}} \cap S_{w_i}$, where S_{w_i} denotes the set of skills possessed by w_i. Given a budget b_{t_x} provided by the requester of task t_x, b_{t_x} will be distributed to the assigned workers according to their real skill contribution; therefore, the original utility

$op_{w_i}(t_x)$ that the requester of task t_x should pay to w_i is determined by b_{t_x} and $|\overline{S_{t_x}} \cap S_{w_i}|/|S_{t_x}|$.

If we form a fixed team for all tasks in B, W_B, $\forall w_i \in W_B$, the requester of t_x will pay the following real utilities to w_i by considering w_i's real contribution to t_x and the discounting function:

$$\forall w_i \in W_B : \quad Pay_{w_i}(t_x) = \begin{cases} \psi\left(\frac{|T_i(B)|}{\theta(T_i(B))+1}\right) \cdot b_{t_x} \cdot \frac{|\overline{S_{t_x}} \cap S_{w_i}|}{|S_{t_x}|}, & if\ t_x \in T_i(B) \\ 0, & else \end{cases} \tag{5.6}$$

Thus, the total payment of requesters for performing the tasks in B by forming a fixed team is:

$$C_{Pay}(B) = \sum_{t_x \in B} \sum_{w_i \in W_B} Pay_{w_i}(t_x) \tag{5.7}$$

If we form a dynamic team for B, i.e., $\forall t_x \in B$, we form W_{t_x} for task t_x. $\forall w_i \in W_{t_x}$, the requester of t_x will pay the following real utilities to w_i:

$$\forall w_i \in W_{t_x} : \quad Pay_{w_i}(t_x) = \psi\left(\frac{|T_i(B)|}{\theta(T_i(B))+1}\right) \cdot b_{t_x} \cdot \frac{|\overline{S_{t_x}} \cap S_{w_i}|}{|S_{t_x}|} \tag{5.8}$$

Thus, the total cost of payments by requesters for performing the tasks in B by forming a dynamic team is:

$$C_{Pay}(B) = \sum_{t_x \in B} \sum_{w_i \in W_{t_x}} Pay_{w_i}(t_x) \tag{5.9}$$

3) *The communication cost among team members for performing tasks,* $C_{Com}(B)$, which is mainly decided by the size of the team and the distance between each other in the social networks.

Therefore, the total cost of team formation-based crowdsourcing for a batch of tasks B is:

$$C(B) = \alpha_1 \cdot C_{Form}(B) + \alpha_2 \cdot C_{Pay}(B) + \alpha_3 \cdot C_{Com}(B)) \tag{5.10}$$

where α_1, α_3, and α_3 are three weights to measure the relative importance of the three types of costs.

5.1.2.3 Optimization Objective and Heuristics

Generally, there are three typical measures to ensure the quality of crowdsourcing results: effective task design, redundant allocation with majority voting, and reputation-based worker selection [136]. Among the three measures, effective task design is implemented by the requester, which is not the focus of this thesis; the redundant allocation is mainly used for simple tasks, because the redundant allocation of complex tasks may produce extensive costs. Therefore, we introduce the reputation mechanism to encourage workers to complete the assigned tasks with high quality; the reputation of a worker is mainly determined by the worker's past experiences in completing tasks: if a worker has richer experience of successful completion of tasks, his/her reputation is higher, and vice versa [51].

When a worker responds to the request for joining a team, he/she will estimate the completion time for each task in the batch according to the current real situation and his/her experiences; moreover, the estimated completion time of worker for a task can also be estimated by the system. Therefore, it is assumed that the estimated completion time of each task can be achieved [130].

In this thesis, we abstract the team formation problem of a batch of tasks to minimize the total cost as well as maximize the reputations of team members under four constraints: the skill requirements of all tasks can be satisfied, the payment is more than the reservation wages of team members, all workers can only communicate with their neighbors in the social network, and the estimated completion time of each task cannot exceed the deadline of that task. Let W denote the crowd of workers, R_{w_i} denote the reputation of worker w_i, and γ_{w_i} denote the reservation wage of worker w_i. The optimization objective while forming a fixed team for all tasks in B is to form a team of workers as follows:

$$W_{B*} = \arg\min_{W_B \subseteq W} \frac{C(B)}{\sum\limits_{w_i \in W_B} R_{w_i}} \tag{5.11}$$

subject to

$$\bigcup_{w_i \in W_B} S_{w_i} \supseteq S_{t_x},\ \forall t_x \in B \tag{5.12}$$

$$(t_x \in T_i(B)) \wedge (Pay_{w_i}(t_x) \geq \gamma_{w_i}),\ \forall w_i \in W_B,\ \forall t_x \in B \tag{5.13}$$

$$\forall w_{i \in W_B} :\ w_i \text{ can only communicate with } w_i\text{'s neighbors in the social network} \tag{5.14}$$

$$time_{W_B}(t_x) \leq d_{t_x},\ \forall t_x \in B \tag{5.15}$$

where $time_{W_B}(t_x)$ denotes the completion time of task t by W_B, and d_{t_x} denotes the deadline of t_x.

Let W_{t_x} denote the team of workers performing task t_x. Let Π be a set of teams for all tasks in B, $\Pi = \{W_{t_x} | \forall t_x \in B\}$. For a worker in any team, w_i, the set of tasks assigned to w_i, is assumed to B_i. The optimization objective while we form a dynamic team for each task in B is to form the set of teams for B as follows:

$$\prod = \underset{\forall \prod \wedge (\forall W_{t_x} \in \prod \wedge W_{t_x} \subseteq W)}{\arg\min} \frac{C(B)}{\sum_{t_x \in B} \sum_{w_i \in W_{t_x}} R_{w_i}} \tag{5.16}$$

subject to

$$(W_{t_x} \in \prod) \wedge (\bigcup_{w_i \in W_{t_x}} S_{w_i} \supseteq S_{t_x}),\ \forall t_x \in B \tag{5.17}$$

$$(W_{t_x} \in \prod) \wedge (Pay_{w_i}(t_x) \geq \gamma_{w_i}),\ \forall t_x \in B,\ w_i \in W_{t_x} \tag{5.18}$$

$$\forall W_{t_x} \in \prod,\ w_i \in W_{t_x} :\ w_i \text{ can only communicate with } w_i^{'}\text{s neighbors in the social network} \tag{5.19}$$

$$time_{W_{t_x}}(t_x) \leq d_{t_x},\ \forall t_x \in B \tag{5.20}$$

where $time_{W_{t_x}}(t_x)$ denotes the completion time of task t by W_{t_x}, and d_{t_x} denotes the deadline of t_x.

Theorem 5.1. *The team formation problem for a batch of tasks with the optimization objective in Equations* (5.11)-(5.15) *or* (5.16)-(5.20) *is NP-hard.*

Proof. Our research problem includes three independent sub-problems that can be described to find a team of workers within a social network to satisfy the skill requirements of tasks and minimize the following three factors: the cost of forming teams, the cost

of requesters paying team members for performing tasks, and the communication cost among team members for performing tasks. The problem of finding a team of workers in a social network to satisfy the skill requirements of a task and minimize the communication cost among workers has already been proven in previous benchmark studies to be NP-hard [56, 57, 123]. Since our research problem involves this well-known NP-hard problem in combination with two other independent problems for optimizing two other factors and satisfying the skill requirements of a batch of tasks, we have Theorem 5.1. □

To reduce the cost of forming teams, we will attempt to reduce the number of team members and reduce the variation among dynamic teams for a batch of tasks; therefore, it is preferable to select the workers who cover more skills of the task batch. To reduce the payments by requesters, it is preferable to select workers whose reservation wages are low and workers who have already undertaken more tasks in the batch. To optimize the cost of communication between team members, it is preferable to select workers who are closer in the social network to form a team; therefore, existing team members will find other new teammates from the near to the distant in the social network. Moreover, it is preferable to select workers with higher reputations to ensure the quality of crowdsourcing results.

For the above considerations, this thesis presents the concept of the crowdsourcing value of a worker to measure the probability of that worker to be hired into a team, which is mainly determined by the following factors: 1) the coverage degree of the worker's skills for the skill requirements of the tasks; 2) the communication distance of the worker with other workers in the social network; 3) the reservation wage of the worker; and 4) the reputation of the worker.

Based on the concept of crowdsourcing value, this thesis presents two heuristic approaches. The first is a distributed formation of a fixed team for all tasks in the batch; the second one is a distributed formation of a dynamic team, i.e., a basic team will be formed first and will then be adjusted by recruiting or eliminating team members for each task in the batch.

5.1.2.4 Self-Organization Process of Team Formation Through Social Networks

In our approach, if a worker assigned by the requester can satisfy all skill requirements, he/she will perform the task independently. If the worker cannot complete the task by himself/herself, he/she has to collaborate with others to complete the task; or else, the worker's reputation will be reduced if he/she cannot complete the task and is not willing to collaborate with others. Now we describe how workers can self-organize a team through social networks.

First, a worker with the maximum probability for approaching the optimization objective whose reservation wage can be satisfied is assigned by the requester as the initiator forming the team. Then, the initiator will send requests to all his/her neighbors through the social network. If a neighbor can respond to the request within a predefined time, the initiator will observe the situation of this neighbor; then, the initiator recruit a worker from all his/her neighbors in the social network to join the team for approaching the optimization objective.

Then, the existing team members will send requests to all their neighbors through the social network, and they will observe the situations of the neighbors who can respond to the requests within the predefined time. They will recruit one worker from the observed neighbors to join the team for approaching the optimization objective.

The members of the new team will repeat the above recruiting process until the skills of all tasks in the batch can be satisfied or all workers in the social network are observed. Therefore, the team formation is implemented by the self-organization of the workers, and the formation process is like a breadth-first diffusion process from the initiator in the social network.

5.2 Distributed Formation of a Fixed Team for a Batch of Tasks

In this section, we will present an approach that can make distributed formation of a team that will remain fixed within the execution of all tasks in a batch. Because all

tasks in the batch will be allocated to a fixed team in whole, the wholesale discounting mechanism in Equation (5.3) is applied here.

5.2.1 Crowdsourcing Value for a Batch of Tasks

To approach the optimization objective, we define two types of crowdsourcing values of a worker for a batch of tasks: *global crowdsourcing value*, which measures the probability of a worker being assigned by the requester as the initiator to form the team for a batch of tasks; and *local crowdsourcing value*, which measures the probability of a worker being recruited by the existing team members to join the team for a batch of tasks.

5.2.1.1 Global Crowdsourcing Value for a Batch of Tasks

The global crowdsourcing value of a worker is generally determined by the following factors:

1) *The coverage degree of the worker's skills for the necessary skills required by the batch.* This factor is the same as the one in Section 4.2.2 of Chapter 4.

2) *The locality of the worker regarding other workers' skill coverage degrees for the batch.* We will attempt to select more central workers with respect to other workers' skills, i.e., the closer a worker is to other workers that can cover more skills for the batch, the more advantageous the worker's locality will be for being assigned to the batch. Let $CS_{w_i}(B)$ denote the coverage degree of worker w_i's skills for batch B. Let d_{ij} denote the distance between workers w_i and w_j in the social network. It is assumed that the set of all workers in the social network is W; the **locality** of w_i for B is defined as follows:

$$Loc_{w_i}(B) = \sum_{j=1\cdots|W|,\, j\neq i} \frac{d_{ij}}{CS_{w_j}(B)} \tag{5.21}$$

3) *The occupancy rate of the worker's reservation wage on the task's real payment.* Let the reservation wage of w_i be γ_{w_i}. Given an original budget b_{t_x} provided by

the requester of task t_x, the occupancy rate of w_i's reservation wage on batch B's payments is:

$$Occ_{w_i}(B) = \frac{\sum_{t_x \in B} \frac{\gamma_{w_i}}{\psi\left(\frac{|B|}{\theta(B)+1}\right) \cdot b_{t_x}}}{|B|} \tag{5.22}$$

If the assigned worker w_i has a lower $Occ_{w_i}(B)$, w_i may have more potential to distribute more utilities to other team members for executing tasks in B; therefore, it is more probable that w_i will form a successful team for B.

4) *The reputation of worker* w_i, R_{w_i}.

Finally, we have the following definition:

Definition 5.1 (Global crowdsourcing value of a worker for a batch of tasks)**.** *The global crowdsourcing value of a worker,* w_i*, for a batch of tasks* B *is:*

$$GV_{w_i}(B) = \frac{\alpha_1 \cdot CS_{w_i}(B) + \alpha_2 \cdot R_{w_i}}{\alpha_3 \cdot Loc_{w_i}(B) + \alpha_4 \cdot Occ_{w_i}(B)} \tag{5.23}$$

where $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1$.

Lemma 5.1. *Let there be two workers,* w_i *and* w_j*. If* $GV_{w_i}(B) > GV_{w_j}(B)$*,* w_i *can comprehensively approach the optimization objective better than* w_j *can.*

Proof. The optimization objective is to minimize the three types of costs and the reverse of reputations. Let $W_B(w_i)$ be the team formed by w_i for B, and let $W_B(w_j)$ be the team formed by w_j for B. Now $GV_{w_i}(B) > GV_{w_j}(B)$; we can say that the four factors of w_i in Equation (5.23) are higher than those of w_j if the three factors are endowed with equal weight. $GS_{w_i}(B) > GS_{w_j}(B)$ denotes that w_i's skills can cover the skills of B more than w_j's, and thus $|W_B(w_i)| < |W_B(w_j)|$; therefore, the three types of costs of w_i are less than that of w_j since the cost is influenced by the number of team members. $LOC_{w_i}(B) > LOC_{w_j}(B)$ denotes that w_i can recruit other team members with less numbers and communication distance than w_j; thus, the three types of cost of w_i are also less than that of w_j. $Occ_{w_i}(B) > Occ_{w_j}(B)$ denotes that the cost of paying w_i is less than to w_j. Thus, the third type of cost of w_i is less than that of w_j. All three types of cost of w_i are less than that of w_j and the reputation of w_i is higher than that of w_j; thus, we have Lemma 5.1. □

5.2.1.2 Local Crowdsourcing Value for a Batch of Tasks

The local crowdsourcing value of a worker is generally determined by the following four factors:

1) *The coverage degree of the worker's skills for the currently missing skills of the batch.* Let there be a batch, B. Now, let there be an existing team $W_B \subseteq W$. $\forall w_i \in W_B$, the set of skills possessed by w_i is S_{w_i}. Then, the skill coverage degree of w_i with respect to the existing members in W_B for batch B can be defined as follows:

$$\begin{aligned} \#(worker's\ \ effective\ \ skills) &= \sum_{s_a \in (S_B - \bigcup\limits_{w_j \in W_B} S_{w_j})} b_{ia} \cdot n_a \\ \#(missing\ \ skills) &= \sum_{s_a \in (S_B - \bigcup\limits_{w_j \in W_B} S_{w_j})} n_a \\ CS_{w_i}^{lacking}(B) &= \frac{\#(worker's\ \ effective\ \ skills)}{\#(missing\ \ skills)} \end{aligned} \tag{5.24}$$

where the symbols S_B, n_a, b_{ia} are the same as those in Equation (4.9) of Chapter 4.

2) *The distance between the worker and the existing team members.* Let $CS_{w_j}(B)$ denote the coverage degree of worker w_j's skills for batch B. The distance between the worker and W_B will consider the existing team members' skill coverage degrees, shown as:

$$D_{w_i}^{team}(B) = \frac{1}{|W_B|} \sum_{w_j \in W_B} \frac{d_{ij}}{CS_{w_j}(B)} \tag{5.25}$$

3) *The reservation wage of the worker,* γ_i, denotes the minimum wage requirement of worker w_i for one task.

4) *The reputation of worker* w_i, R_{w_i}.

Definition 5.2 (Local crowdsourcing value of a worker for a batch of tasks)**.** *Let there be an existing non-final team* W_B *for batch* B, $W_B \subseteq W$. *The local crowdsourcing*

value of w_i perceived by W_B for B is defined as:

$$LV_{w_i}(B) = \frac{\alpha_1 \cdot CS_{w_i}^{lacking}(B) + \alpha_2 \cdot R_{w_i}}{\alpha_3 \cdot D_{w_i}^{team}(B) + \alpha_4 \cdot \gamma_{w_i}} \tag{5.26}$$

where $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1$.

Lemma 5.2. *Let there be two workers, w_i and w_j. It is assumed that the set of existing team members formed for task batch B is W_B. If $LV_{w_i}(B) > LV_{w_j}(B)$, $W_B \cup \{w_i\}$ can approach the optimization objective better than $W_B \cup \{w_j\}$ can.*

Proof. The proof is similar to that for Lemma 5.1; here, we skip it to save space. □

5.2.2 Distributed Formation Algorithm

First, a worker with the maximum global crowdsourcing value whose reservation wage can be satisfied is assigned by the requester as the initiator forming the team. Then, the self-organization process of team formation described in Section 5.1.2.4 is used through observing the local crowdsourcing values of other workers in the social network.

Some workers may not respond to the request for joining a team in a reasonable time, and their delay would hinder the forming of teams. In order to deal with such cases, we set a maximum response time δ that denotes the worker should respond within such time, or else a new worker will be selected. On the other hand, to satisfy the deadline requirement of a task, the team's estimated completion time of the task cannot exceed its deadline, which can be simplified to that completion time of all workers in the team to perform the task cannot exceed the deadline.

However, if all neighbors of existing team members in one round cannot satisfy the requirements, the formation process may end before all skills of the batch can be satisfied, but some unobserved workers still exist in the social network. Therefore, we can revise the formation process by assuming that all observed workers (not only the team members) can help find their neighbors; this method is feasible because workers are often cooperative within the same social network [137].

Let B denote a batch of tasks and W denote the set of workers in the social network. We use S_B to denote the set of skills required by all tasks in B and W_B to denote the team formed for B. The distributed formation of a fixed team for a batch of tasks is shown as Algorithm 9. $Res(w)$ denotes the response time of worker w.

Algorithm 9 is $O(|S_B| \cdot |W|)$. Liking 3.3.1, we suppose workers are coadjutant, so in line 33, neighbours will have certain obligations to join the team. For simplification, here we do not define the detailed strategy of joining a team by a worker. The strategy of joining the team could be similar with that of in 3.3.2 where threshold mechanism is adopted. In Algorithm 9, the workers in the team are sought by an initiator from the near to the distant in the social network, which is similar to the breadth-first search method in the graph [138, 139]. Therefore, we have the following theorem.

Theorem 5.2. *Let the team formed for a batch of tasks B using Algorithm 9 be W_B; the initiator worker of W_B is w_i. It is then assumed that there is another team, W'_B, which is formed by any of the following two methods: another random initiator worker w_j and the workers in $W_B - \{w_i\}$ or w_i and other workers. It is assumed that the skills possessed by W'_B can fully cover the skill and deadline requirements of B and the reservation wage of each worker in W'_B can be satisfied. We use $C_{form}(B) - W_B$, $C_{pay}(B) - W_B$, and $C_{com}(B) - W_B$ to denote the three related types of costs in our optimization objective when we form team W_B to perform B and use R_W_B to denote the reputations of workers in W_B. Then, we have:*

$$\left(\forall W'_B \wedge W'_B \subseteq W \wedge \bigcup_{w_i \in W'_B} S_{w_i} \supseteq S_B \wedge (\forall w_i \in W'_B \wedge \gamma_{w_i} \leq \frac{Pay_{w_i}(B)}{|B|})\right)$$
$$\Rightarrow \left(\alpha_1 \cdot C_{Form}(B)\_W_B + \alpha_2 \cdot C_{Pay}(B)\_W_B + \alpha_3 \cdot C_{Com}(B)\_W_B + \alpha_4 / R\_W_B\right)$$
$$\leq (\alpha_1 \cdot C_{Form}(B)\_W'_B + \alpha_2 \cdot C_{Pay}(B)\_W'_B + \alpha_3 \cdot C_{Com}(B)\_W'_B + \alpha_4 / R\_W'_B)$$

Proof. If W'_B is formed by another random initiator worker w_j and the workers in $W_B - \{w_i\}$. According to Algorithm 9, we have $GV_{w_i}(B) > GV_{w_j}(B)$; then, Lemma 5.1 ensures that w_i can comprehensively approach the optimization objective better than w_j. Because $W_B - \{w_i\} = W_B - \{w_j\}$, we obtain the theorem. If W'_B is formed by w_i and other workers in the social network, let the order of w_k in W_B be the same as

Algorithm 9: Distributed formation of a fixed team for a batch of tasks.

$Lacking\_S_B = S_B$; $b_1 = 0$; $W_B = \{\}$; $W_{temp1} = W$; $W_{temp2} = \{\}$;
while $b_1 == 0$ **do**
    // Assign the initiator worker
    $w_* = \underset{w_i \in W_{temp1}}{\arg\max}\ GV_{w_i}(B)$;
    if $Res(w_*) \leq \delta$ *and* $(\forall t_x \in B) \wedge (time_{w_*}(t_x) \leq d_{t_x})$ *and* $(Pay_{w_*}(B)/|B|) \geq \gamma_{w_*}$ *and* $S_{w_*} \cap Lacking\_S_B \neq \phi$ **then**
        $Lacking\_S_B = Lacking\_S_B - S_{w_*}$;
        $b_1 = 1$;
    $W_{temp1} = W_{temp1} - \{w_*\}$;
    if $W_{temp1} == \phi$ **then**
        $b_1 = 1$;
$b_1 = 0$;
$W_B = \{w_*\}$;
$W_{temp2} = W_{temp2} \cup \{w_*\}$; // Observed workers
$W_{candidate} = \{\}$; // Workers to be observed
if $Lacking\_S_B == \phi$ **then**
    $b_1 = 1$;
while $b_1 == 0$ **do**
    forall the $w_i \in W_{temp2}$ **do**
        forall the $w_j \in Neigh(w_i)$ **do**
            // Neighbors of w_i
            if $w_j \notin W_{temp2}$ **then**
                $W_{candidate} = W_{candidate} \cup \{w_j\}$;
    $W_{temp2} = W_{temp2} \cup W_{candidate}$;
    $b_2 = 0$;
    $b_3 = 0$;
    while $b_2 == 0$ **do**
        // Search a qualified neighbor
        $w_* = \underset{w_i \in W_{candidate}}{\arg\max}\ LV_{w_i}(B)$;
        if $Res(w_*) \leq \delta$ *and* $(\forall t_x \in B) \wedge (time_{w_*}(t_x) \leq d_{t_x})$ *and* $(Pay_{w_*}(B)/|B|) \geq \gamma_{w_*}$ *and* $S_{w_*} \cap Lacking\_S_B \neq \phi$ **then**
            $b_2 = 1$;
            $b_3 = 1$;
        $W_{candidate} = W_{candidate} - \{w_*\}$;
        if $W_{candidate} == \phi$ **then**
            $b_2 = 1$;
    if $b_3 == 1$ **then**
        // Recruit the qualified neighbor
        $W_B = W_B \cup \{w_*\}$; $Lacking\_S_B = Lacking\_S_B - S_{w_*}$;
    if *(*$Lacking\_S_B == \phi$*) or (*$W_{temp2} == W$ *and* $W_{candidate} == \phi$*)* **then**
        $b_1 = 1$;
Output (W_B);

that of w'_k in W'_B; we then have $LV_{w_k}(B) > LV'_{w_k}(B)$ according to Algorithm 9. Now, Lemma 5.2 ensures that $\{w_i\} \cup \{w_k\}$ can approach the optimization objective better than $\{w_i\} \cup \{w'_k\}$. Thus, finally, W_B can approach the optimization objective better than W'_B. Therefore, we obtain the theorem. □

Therefore, Theorem 5.2 ensures that the optimization objective in Equation (5.11) can be approached by Algorithm 9.

5.3 Distributed Formation of a Dynamic Team for a Batch of Tasks

Since the skill requirements of tasks in a batch may sometimes vary significantly, a fixed team for all tasks may be disadvantageous. We present an approach for forming a dynamic team. First, we will find the basic set of skills of the batch, and we then form a basic team to satisfy the basic set of skills of the batch. Afterwards, the basic team will be adjusted by recruiting or eliminating members for each task in the batch.

5.3.1 Crowdsourcing Value for One Task

5.3.1.1 Global Crowdsourcing Value for One Task

The global crowdsourcing value of a worker for one task denotes the probability of that worker being assigned by the requester to be an initiator forming a team for that task, which is determined by the following factors: 1) the satisfaction degree of the worker's skills for the necessary skills required by the task; 2) the locality of the worker with respect to other workers' skill coverage degrees for the task; 3) the occupancy rate of the worker's wage on the task's budget; and 4) the reputation of the worker.

Definition 5.3 (Global crowdsourcing value of a worker for one task)**.** *The global crowdsourcing value of a worker, w_i, for task t_x is:*

$$GV_{w_i}(t_x) = \frac{\alpha_1 \cdot (|S_{w_i} \cap S_{t_x}|/|S_{t_x}|) + \alpha_2 \cdot R_{w_i}}{\alpha_3 \cdot Loc_{w_i}(t_x) + \alpha_4 \cdot (\gamma_{w_i}/b_{t_x})} \tag{5.27}$$

where $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1$*, and* $Loc_{w_i}(tx)$ *is calculated as follows:*

$$Loc_{w_i}(t_x) = \sum_{j=1}^{|W|} \frac{d_{ij}}{|S_{w_j} \cap (S_{t_x} - S_{w_j})|/|S_{t_x}|} \tag{5.28}$$

where d_{ij} *denotes the distance between workers* w_i *and* w_j *in the social network, and* W *denotes the crowd of workers in the social network.*

5.3.1.2 Local Crowdsourcing Value for One Task

The local crowdsourcing value of a worker denotes the probability of a worker to be recruited by the existing members within a team for a task, which is determined by: 1) the coverage degree of the worker's skills for the current lacking skills by the task; 2) the distance between the worker and the existing members in the team regarding the team members' skills; 3) the occupancy rate of the worker's wage on the task's budget; and 4) the reputation of the worker.

Definition 5.4 (Local crowdsourcing value of a worker for one task)**.** *Let there be an existing worker team* W_{t_x} *for task* t_x*;* $W_{t_x} \subseteq W$*. The local crowdsourcing value of* w_i *perceived by* W_{t_x} *for* t_x *is defined as:*

$$LV_{w_i}(t_x) = \frac{\alpha_1 \cdot (|(S_{t_x} - \bigcup_{w_j \in W_{t_x}} S_{w_j}) \cap S_{w_i}|/|S_{t_x}|) + \alpha_2 \cdot R_{w_i}}{\alpha_3 \cdot D_{w_i}^{team}(t_x) + \alpha_4 \cdot (\gamma_{w_i}/b_{t_x})} \tag{5.29}$$

where $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1$*, and*

$$D_{w_i}^{team}(t_x) = \frac{1}{|W_{t_x}|} \sum_{w_j \in W_{t_x}} \frac{d_{ij}}{|S_{t_x} \cap S_{w_j}|/|S_{t_x}|} \tag{5.30}$$

5.3.2 Distributed Formation of the Basic Team

5.3.2.1 Basic Set of Skills for a Batch of Tasks

We can use three methods to decide the basic set of skills for a batch of tasks.

1) Adopting the skills of the first task if the tasks in a batch need to be executed in sequence. Let $B = \{t_1, t_2, ..., t_n\}$. $\forall t_x \wedge (1 \leq x \leq n-1)$, it is assumed that t_x is executed before t_{x+1}; then, the basic set of skills of B is $Basic\_S_B = S_{t_1}$.

2) Adopting the skills of the core task if the tasks in a batch do not need to be executed in sequence. The core task is defined as the task with the minimum sum of distances with other tasks in B:

$$t_c = \arg\min_{t_x \in B} \sum_{t_y \in (B - \{t_x\})} \delta_{x,y} \tag{5.31}$$

Now the basic set of skills of B is $Basic\_S_B = S_{t_c}$.

3) Adopting the intersection of the skills of all tasks in B, i.e., $Basic\_S_B = \bigcap_{t_x \in B} S_{t_x}$. This method can avoid the costs of eliminating unutilized workers.

5.3.2.2 Forming the Basic Team

We construct a virtual basic task, t_v, which only requires the skills of $Basic\_S_B$. Similar to Algorithm 9, we will form a team for t_v. This team formation process can use Algorithm 9 by making certain revisions. The set of skills is the basic set of skills but not the set of all skills required by the batch of tasks. Revising the calculations of crowdsourcing values according to Definition 5.3 and 5.4 is required. Moreover, the calculation of real payment can be revised by only considering the individual virtual basic task but not the batch of tasks. The process can be shown as Algorithm 10. Finally, the original basic team for t_v can be achieved, denoted as W_{t_v}.

Moreover, because the basic team is crucial to the following teams for other tasks in the batch, we now refine the above team formation method. In Algorithm 9, the existing team members will remain fixed during the whole team formation process; such method may only form the local optimal team, but some better workers may be not recruited by the team. To address this problem, after the original team is formed by using the method similar to Algorithm 9, we can add a refinement process. Now, a worker with the maximum local crowdsourcing value is selected from the neighbors of all existing team members; the team can attempt to replace each team member by the

Algorithm 10: Distributed formation of an original basic team for a virtual task. /* $Basic\_S_B$ is the basic set of skills of batch B; t_v is the virtual basic task that requires $Basic\_S_B$; W is the set of workers in the social network */

$b_1 = 0$; $W_{t_v} = \{\}$; $W_{temp0} = W$; $W_{temp1} = \{\}$;
while $b_1 == 0$ **do**
    // Assign the initiator worker
    $w_* = \arg \max_{w_i \in W_{temp0}} GV_{w_i}(t_v)$;
    if $Res(w_*) \leq \delta$ *and* $time_{w_*}(t_v) \leq d_{t_v}$ *and* $Pay_{w_*}(t_v) \geq \gamma_{w_*}$ *and* $S_{w_*} \cap Basic\_S_B \neq \phi$ **then**
        $Basic\_S_B = Basic\_S_B - S_{w_*}$; $b_1 = 1$;
    $W_{temp0} = W_{temp0} - \{w_*\}$;
    if $W_{temp0} == \phi$ **then**
        $b_1 = 1$;
$b_1 = 0$; $W_B = \{w_*\}$;
$W_{temp1} = W_{temp1} \cup \{w_*\}$; // Observed workers
$W_{candidate} = \{\}$; // Workers to be observed
if $Basic\_S_B == \phi$ **then**
    $b_1 = 1$;
while $b_1 == 0$ **do**
    // Distributed formation of team
    forall the $w_i \in W_{temp1}$ **do**
        forall the $w_j \in Neigh(w_i)$ **do**
            // Neighbors of w_i
            if $w_j \notin W_{temp1}$ **then**
                $W_{candidate} = W_{candidate} \cup \{w_j\}$;
    if $W_{canditate} == \phi$ **then**
        // All workers have been considered
        $b_1 = 1$;
    $W_{temp1} = W_{temp1} \cup W_{candidate}$; $b_2 = 0$; $b_3 = 0$;
    while $b_2 == 0$ **do**
        // Search a qualified neighbor
        $w_* = \arg\max_{w_i \in W_{candidate}} LV_{w_i}(t_v)$;
        if $Res(w_*) \leq \delta$ *and* $time_{w_*}(t_v) \leq d_{t_v}$ *and* $Pay_{w_*}(t_v) \geq \gamma_{w_*}$ *and* $S_{w_*} \cap Basic\_S_B \neq \phi$ **then**
            $b_2 = 1$; $b_3 = 1$;
        $W_{candidate} = W_{candidate} - \{w_*\}$;
        if $W_{candidate} == \phi$ **then**
            $b_2 = 1$;
    if $b_3 == 1$ **then**
        // Recruit the qualified neighbor
        $W_{tv} = W_{tv} \cup \{w_*\}$; $Basic\_S_B = Basic\_S_B - S_{w_*}$;
    if *(*$Basic\_S_B == \phi$*)* **then**
        $b_1 = 1$;
Output (W_{t_v});

selected worker. If the performance of the new team is better than that of the previous team, the new team can replace the previous team. This process will repeat until no workers can be found to replace the existing team members.

To consider the optimization objective, we can use the following index to measure the performance of a team:

$$Per(W_{t_v}) = \alpha_1 \cdot |W_{t_v}| + \alpha_2 \cdot \sum_{w_i \in W_{t_v}} \gamma_{w_i} + \alpha_3 \cdot \frac{1}{2} \sum_{w_i, w_j \in W_{t_v}} d_{ij} + \alpha_4 \cdot \sum_{w_i \in W_{t_v}} \frac{1}{R_{w_i}} \quad (5.32)$$

Algorithm 11: Refinement of a team (W_{t_v}).
$/* W_{t_v}$ is the original basic team for virtual task t_v; W is the set of workers in the social network $*/$

$W_{temp1} = W_{t_v}$;
$b_1 = 0$;
while $b_1 == 0$ **do**
    // Refine the team
    $W_{neighbor} = \{\}$; // Neighbors of observed workers
    forall the $w_i \in W_{temp1}$ **do**
        forall the $w_j \in Neigh(w_i)$ **do**
            // Neighbors of w_i
            if $w_j \notin W_{temp1}$ **then**
                $W_{neighbor} = W_{neighbor} \cup \{w_j\}$;
    if $W_{neighbor} == \phi$ **then**
        $b_1 = 1$;// All workers have been considered
    $W_{temp1} = W_{temp1} \cup W_{neighbor}$;
    while $W_{neighbor} \neq \phi$ **do**
        $w_* = \underset{w_i \in W_{neigbhor}}{\arg\max} \; LV_{w_i}(t_v)$;
        $W_{temp2} = W_{t_v}$;
        forall the $w_i \in W_{t_v}$ **do**
            $W_{temp3} = (W_{t_v} - \{w_i\}) \cup \{w_*\}$;
            if $Res(w_*)$ *and* $time_{w_*}(t_v) \leq d_{t_v}$ *and* $S_{t_v} \subseteq \bigcup_{w_i \in W_{temp3}} S_{w_i}$ *and* $Per(W_{temp3}) < Per(W_{temp2})$ **then**
                $W_{temp2} = W_{temp3}$;
        $W_{t_v} = W_{temp2}$;
        $W_{neighbor} = W_{neighbor} - \{w_*\}$;
Output (W_{t_v})

The refinement process is shown as Algorithm 11. Algorithm 11's time complexity is $O(|W|^2)$, which is more complex than Algorithm 9 whose time complexity is $O(|S_B| \cdot |W|)$, because in reality $|S_B|$ is much less than $|W|$. This can explain why we do not use the refinement in Algorithm 9 because we want to save time complexity.

Theorem 5.2 ensures that the optimization objective in Equation (5.11) can be approached by Algorithm 9. Now the basic team is formed by Algorithm 9 and the refinement process in Algorithm 11. Because the refinement process can achieve better results than Algorithm 9, our approach can ensure that the resulted basic team can achieve good performance.

5.3.3 Dynamic Adjustment of the Team

5.3.3.1 Execution Sequence of Tasks

There are two methods for determining the execution sequence of tasks.

In the first method, the execution sequence of the tasks is predefined, i.e., given a batch of tasks, $B = t_1, t_2, ..., t_n$, $\forall t_x \wedge (1 \leq x \leq n-1)$, t_x is executed before t_{x+1}. If we use an array, $T[x]$, to denote the execution sequence, we have $T[x] = t_x$.

Algorithm 12: Determination of the execution sequence of a batch of tasks.

```
1  x = 0;
2  B_temp = B;
3  Min_dis = max_value;
4  while B_temp ≠ φ do
5      forall the t_y ∈ B_temp do
6          if δ_yc < Min_dis then
7              t_temp = t_y;
8              Min_dis = δ_y,c;
9      x++;
10     T[x] = t_temp;
11     B_temp = B_temp − {t_temp};
12     Min_dis = max_value;
13 Output (T[x]);
```

In the second method, the execution order of every task in the batch is determined by its skill distance to the virtual basic task. This idea is practical because the method can

make it easier for the execution of one task to utilize the existing execution results of other similar finished tasks. Let the virtual basic task of a batch of task B be t_v. The algorithm for determining the execution sequence of tasks in B is shown as Algorithm 12, which is $O(|B|^2)$.

5.3.3.2 Dynamic Adjustment

The basic team will be adjusted for each task in the batch by eliminating some existing members and recruiting new members. While an existing team is assigned a new task in the batch, the unutilized team member who cannot provide any skills required by the new task will be eliminated. The remaining team members will recruit new members to satisfy the skill requirement of the new task by observing the local crowdsourcing values of other workers from the near to the distant in the social network.

Let the virtual basic task of batch B be t_v. If the basic team for B is W_{t_v}, the dynamic adjustment of the team for each task in B is shown as Algorithm 13, which is $O(|B| \cdot |W|^2)$. W_{t_x} denotes the final team for task t_x.

Theorem 5.3. *Let the basic team for a batch of tasks B be W_{t_v}. For any task in B, $t_x(t_x \in B)$, the final team after adjustment using Algorithm 13 is W_{t_x}. It is then assumed that there is another worker team W'_{t_x}, which is formed by eliminating the members in W_{t_v} that cannot provide any skills for t_x and randomly recruiting workers whose reservation wages can be satisfied. The skills of all members in W'_{t_x} can fully satisfy the skill requirements of t_x. Thus, we have*

$$\begin{aligned}&\left(\alpha_1 \cdot C_{Form}(B)\_W_{tx} + \alpha_2 \cdot C_{Pay}(B)\_W_{tx} + \alpha_3 \cdot C_{Com}(B)\_W_{tx} + \alpha_4/R\_W_{tx}\right)\\ &\leq (\alpha_1 \cdot C_{Form}(B)\_W'_{tx} + \alpha_2 \cdot C_{Pay}(B)\_W'_{tx} + \alpha_3 \cdot C_{Com}(B)\_W'_{tx} + \alpha_4/R\_W'_{tx})\end{aligned}$$

Proof. We can use reductio ad absurdum to prove Theorem 5.3. Let the set of remaining team members of W_{t_v} by eliminating the members who cannot provide any skills for t_x be $Net\_W_{t_x}$. Assume there is a set of workers W'_{t_x}, $W'_{t_x} \neq W_{t_x}$, formed by $Net\_W_{t_x}$ randomly recruiting some workers whose reservation wages can be satisfied,

Algorithm 13: Dynamic team adjustment for each task in a batch.

for $(x = 1; x \leq |B|; x++)$ **do**
  // Team adjusting for each task
  $t_x = T[x]$;
  $W_{t_x} = W_{t_v}$;
  forall the $w_i \in W_{t_x}$ **do**
    // Eliminate the useless members
    if $S_{w_i} \cap S_{t_x} == \phi$ **then**
      $W_{t_x} = W_{t_x} - \{w_i\}$;
  $b_1 = 0$;
  $S_{t_x\_lacking} = S_{t_x} - \bigcup_{w_i \in W_{t_x}} S_{w_i}$;
  if $S_{t_x\_lacking} == \phi$ **then**
    $b_1 = 1$;
  while $b_1 == 0$ **do**
    // Recruit new team members
    $W_{neighbor} = \{\}$; // Neighbors of the team members
    forall the $w_i \in W_{t_x}$ **do**
      forall the $w_j \in Neigh(w_i)$ **do**
        // Neighbors of w_i
        if $w_j \notin W_{t_x}$ **then**
          $W_{neighbor} = W_{neighbor} \cup \{w_j\}$;
    if $W_{neighbor} == \phi$ **then**
      $b_1 = 1$;
    $b_2 = 0$;
    $b_3 = 0$;
    while $b_2 == 0$ **do**
      $w_* = \underset{w_i \in W_{neigbhor}}{\arg\max} \; LV_{w_i}(t_x)$;
      if $Res(w_*) \leq \delta$ *and* $time_{w_*}(t_x) \leq d_{t_x}$ *and* $Pay_{w_*}(t_x) \geq \gamma_{w_*}$ *and* $(S_{w_*} \cap S_{t_x\_lacking}) \neq \phi$ **then**
        $b_2 = 1$;
        $b_3 = 1$;
      $W_{neighbor} = W_{neighbor} - \{w_*\}$;
      if $W_{neighbor} == \phi$ **then**
        $b_2 = 1$;
    if $b_3 == 1$ **then**
      $W_{t_x} = W_{t_x} \cup \{w_*\}$;
      $S_{t_x\_lacking} = S_{t_x\_lacking} - S_{w_*}$;
    if $S_{t_x\_lacking} == \phi$ **then**
      $b_1 = 1$;
  Output (W_{t_x})

and $\sum_{w_i \in W'_{tx}} s_{w_i} \supseteq S_{t_x}$. If the assumption

$$\begin{aligned}&\left(\alpha_1 \cdot C_{Form}(B)\_W_{tx} + \alpha_2 \cdot C_{Pay}(B)\_W_{tx} + \alpha_3 \cdot C_{Com}(B)\_W_{tx} + \alpha_4/R\_W_{tx}\right)\\ &> (\alpha_1 \cdot C_{Form}(B)\_W'_{tx} + \alpha_2 \cdot C_{Pay}(B)\_W'_{tx} + \alpha_3 \cdot C_{Com}(B)\_W'_{tx} + \alpha_4/R\_W'_{tx})\end{aligned}$$

is true, then there exists at least one worker with a higher local crowdsourcing value perceived by $Net\_W_{t_x}$ for t_x and whose reservation wage can be satisfied (and can provide any lacking skills for t_x) but who cannot be selected autonomously by $Net\_W_{t_x}$ using Algorithm 13, and another worker with a lower local crowdsourcing value perceived by $Net\_W_{t_x}$ is selected. However, from Step 22 in Algorithm 13, in each round for recruiting new member, the worker with the highest local crowdsourcing value whose reservation wage can be satisfied will be the first to be definitely selected by $Net\_W_{t_x}$ if the worker can provide any currently lacking skills for t_x. Therefore, the above assumption cannot occur in reality when Algorithm 13 is used. □

Therefore, Theorem 5.3 ensures that the optimization objective in Equation (5.16) can be approached by Algorithm 13.

5.4 Experiments

We now conduct experiments for our presented two distributed team formation approaches on a real dataset by comparing with benchmark approaches.

5.4.1 Benchmark Approaches

Our two approaches, distributed formation approach of a fixed team for a batch of tasks (*Our algorithm-fixed*) and distributed formation approach of a dynamic team for a batch of tasks (*Our algorithm-dynamic*) are compared to the following three benchmark approaches:

- Individual task-specific team formation (*Individual algorithm*). For each task, a new team is formed from scratch. For each team, all members are selected by the requester centrally according to their crowdsourcing values.

- *Centralized greedy algorithm*. In this algorithm, a fixed team for all tasks in a batch is formed centrally using the greedy method. At each step, the system will select a new team member to achieve the locally optimal result for satisfying the optimization objective. Finally, a team can be achieved to satisfy the requirements of all tasks in the batch.

- *Distributed greedy algorithm*. In this algorithm, dynamic teams are formed for a batch of tasks, i.e., each task has a different team. In the formation of each team, first, the system randomly selects a worker as the initiator for forming the team. The existing team members will select new members greedily from their neighbors, i.e., the neighbor who can achieve the locally optimal result is recruited to join the team. The members of the new team will repeat this recruiting process to select other workers to enlarge the team until the skills of all tasks in the batch can be satisfied or all workers in the crowd are observed.

5.4.2 Dataset and Data Processing

We collect the data of workers and tasks from www.upwork.com. We extract worker data each of which contains more than one task completion record. The average income of historical tasks of a worker is regarded as the worker's reservation wage. Tasks are selected from the "web-mobile-software-development" category at Upwork.com. To ensure the generality of our experimental result, some extreme data will be deleted as follows: the workers whose wages are less than \$200 or more than \$700 are excluded; the tasks whose budgets are greater than \$1400 or less than \$400 are excluded; and rare skills (if the number of the workers possessing a skill is no more than 2, then such skill is rare) are excluded. Finally, there are 864 workers and 354 tasks used for our experiments. By considering the general deadline of tasks at the website, the deadline of each task in experiments, d_t, is a random value in [67,127]. The workers

in the experiments are interconnected by three typical social network structures: small-world networks, scale-free networks, and random networkers. In the experiments, the estimated completion time of a task in the batch is the maximum completion time of all workers in the team for completing the task.

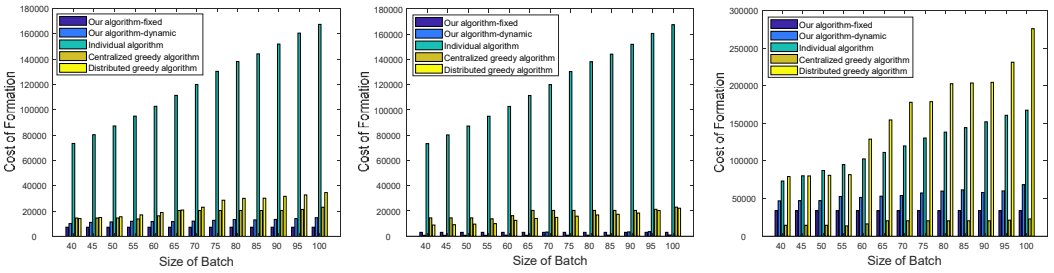
5.4.3 Performance Indices

According to the optimization objective, we define the following indices to evaluate the performance each approach.

- Cost of Team Formation: According to Equation (5.5), we assume that the cost of team formation is generated by the process of finding team members in the social network.
- Cost of Communication: After the team is formed, the requester will assign the batch of tasks to the team, and team members will communicate with each other to cooperate. This performance index is determined by the size of the team and distance between team members.
- Total Payment by Requester: According to Equations (5.6) and (5.7), this is a discount of payment to workers, so we define this index to evaluate our optimization objective-minimizing requesters' real payment.
- Success Rate of Tasks: Because some workers in the team may fabricate their skills, some tasks may not be completed successfully. This index is to measure the effectiveness of the reputation mechanism in our approaches.

5.4.4 Experimental Results

We repeat each experiment 20 times, and the final result is obtained from the average of the 20 times of results.



(a) Small-world networks (b) Scale-free networks (c) Random networks

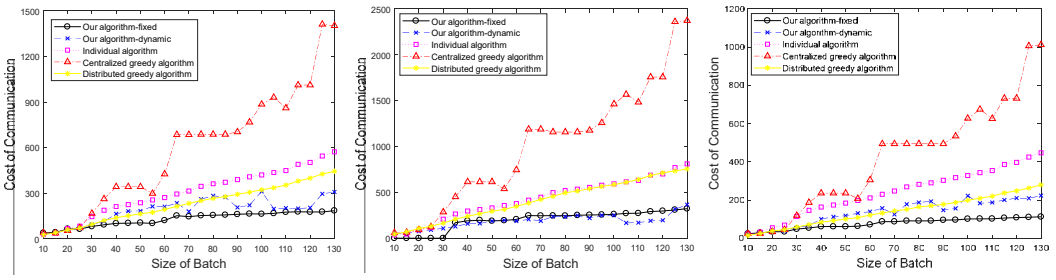
Figure 5.1: Tests on the cost of team formation.

5.4.4.1 Tests on the Cost of Team Formation

Now we test the performances on the cost of team formation, shown in Figure 5.1. We can see that both of our approaches can result in lower costs of team formation. Moreover, while the size of batch becomes larger, the costs of team formation of our two approaches have no obvious changes. In comparison, the cost of team formation of *Our algorithm-fixed* is less than that of *Our algorithm-dynamic* in small networks and random networks; therefore, it denotes that the dynamic adjustment of team will produce higher costs in these two types of networks.

We find that the *Individual algorithm* almost produces the highest cost for team formation because each task needs to form a new team. The cost of team formation of the *Centralized greedy algorithm* is less than that of the *Distributed greedy algorithm* because the former algorithm only needs to form a fixed team for all tasks in the batch, but the latter algorithm needs to form dynamic teams for tasks in the batch. Especially in Figure 5.1c, the performance of the cost of team formation of the *Batch-distributed-greedy algorithm* is higher than that of the *Individual algorithm* even when the size of the batch becomes larger. The potential reason is that the random network structure may make the former algorithm difficult to find new team members from the neighbors of existing team members.

In summary, our two approaches perform better than previous benchmark approaches in terms of the cost of team formation; moreover, our two approaches have good scalability for the sizes of the batches.



(a) Small-world networks (b) Scale-free networks (c) Random networks

Figure 5.2: Tests on the cost of communication.

5.4.4.2 Tests on the Cost of Communication

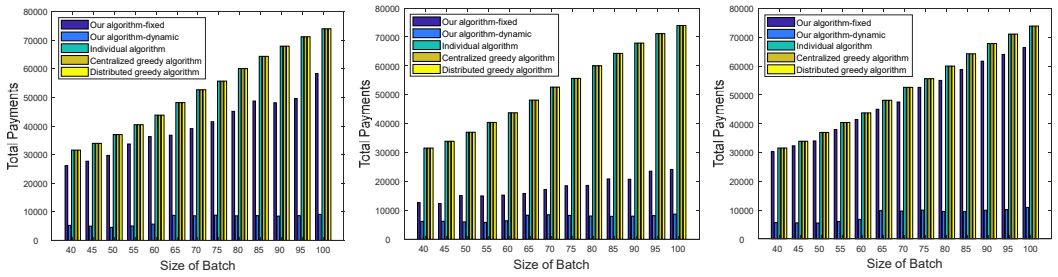
We now test the performances on the cost of communication, shown in Figure 5.2. We can see that our two presented approaches result in lower communication cost. In the small-world and random networks, *Our algorithm-fixed* performs better than *Our algorithm-dynamic*; but in the scale-free networks, the two presented approaches perform similarly. Therefore, it shows that the scale-free networks can provide shorter communication lengths for both fixed team and dynamic teams. Moreover, the cost of communication performance of our two approaches has no obvious fluctuations with increasing batch size.

Among the three benchmark approaches, the *Centralized greedy algorithm* performs the worst. The reason is that the team members are selected from the whole network by considering the optimization objective but overlooking the communication distances among team members.

In summary, our two approaches perform better than previous benchmark approaches in terms of the cost of communication; moreover, our two approaches have good scalability for the sizes of the batches.

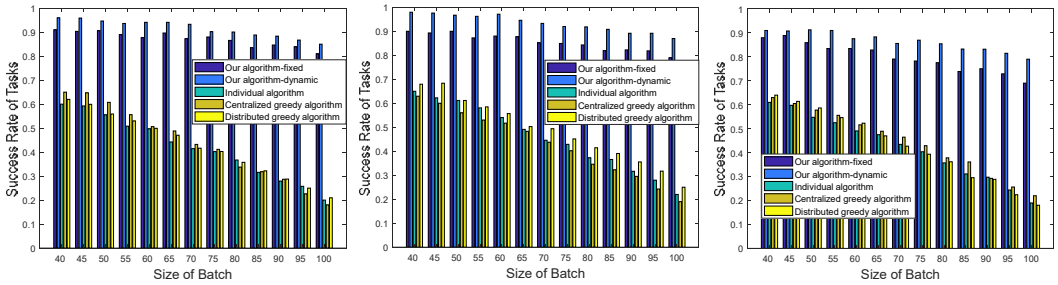
5.4.4.3 Tests on the Total Payment by Requesters

We now test the performance on the total payments by requesters, shown in Figure 5.3. We can see that our two presented approaches achieve lower total payments by requesters. The reason is that our approaches adopt the discounting mechanism, but the



(a) Small-world networks (b) Scale-free networks (c) Random networks

Figure 5.3: Tests on the total payments by requesters.



(a) Small-world networks (b) Scale-free networks (c) Random networks

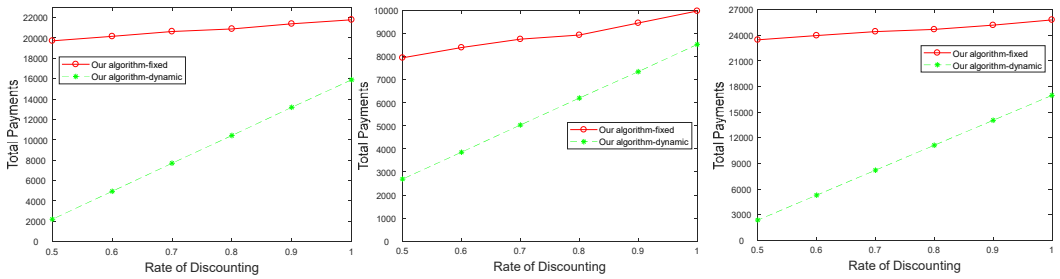
Figure 5.4: Tests on the success rate of tasks.

previous approaches did not. Moreover, we can see that *Our algorithm-fixed* may produce higher total payments by requesters than *Our algorithm-dynamic*; the reason is that each requester will pay all team members, even the members who do not factually execute the task in the former approach, but the requester will only pay the team members that factually execute the task in the latter approach.

Moreover, with the increase of size of batch, the total payments by the requesters of our two approaches have no obvious change, but those of the previous three approaches can increase with the increasing size of the batch. Therefore, our presented two approaches have good scalability in the performance of total payments.

5.4.4.4 Tests on the Success Rate of Tasks

We set some workers to fabricate their skills with certain probabilities. If an assigned worker can execute the task successfully, his/her reputation will become higher, and vice versa. Now we make a series of experiments on the success rate of tasks, in which



(a) Small-world networks (b) Scale-free networks (c) Random networks

Figure 5.5: Tests on the sensitivity to discounting factor.

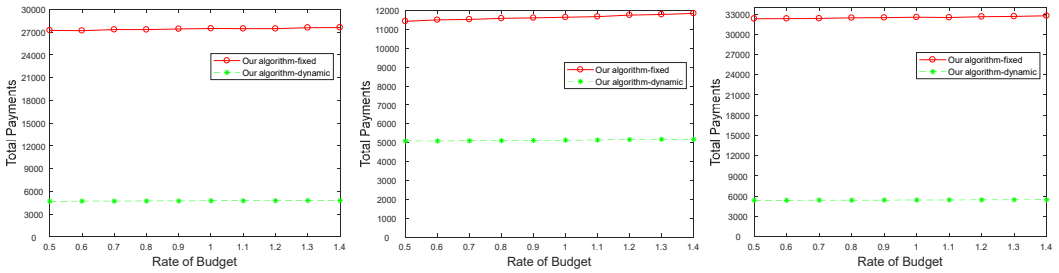
the subsequent experiments can utilize the reputations of workers achieved by the before experiments. The results are shown in Figure 5.4.

From the experiments results, we can see that our two presented approaches have higher success rate of tasks in all experiments under the three typical network structures; moreover, with the progress of experiments, the advantages of our approaches become more obvious. Therefore, it denotes that the reputation mechanism in our approaches can effectively improve the success rate of tasks. Moreover, we can see that *Our algorithm-dynamic* may produce a little higher success rate of tasks than *Our algorithm-fixed*; the reason is that the refinement algorithm can form a better team with higher reputations.

5.4.4.5 Tests on the Sensitivity to Discounting Factor

In Equation (5.3), there is a discounting function ψ, $0 \leq \psi \leq 1$, where the value of $\psi(X)$ decreases monotonically from 1 to 0. In the experiments, we define the function as $\psi(X) = \sigma \times e^{-x}$, and the above experiments are conducted assuming $\sigma = 1$. To test our approaches' sensitivity to the discount factor, we will set the factor from $\sigma = 1$ to $\sigma = 0.5$ to measure our approaches' performance of total payments in three types of network structures. The experimental results are shown in Figure 5.5.

We can see that *Our algorithm-dynamic* is more sensitive to the discounting factor by comparing it with *Our algorithm-fixed* for total payment. The potential reason is that in *Our algorithm-fixed*, the batch of tasks is considered a whole once the team is formed, and the workers will be paid a considerable part of the budget due to their



(a) Small-world networks (b) Scale-free networks (c) Random networks

Figure 5.6: Tests on the sensitivity to budget.

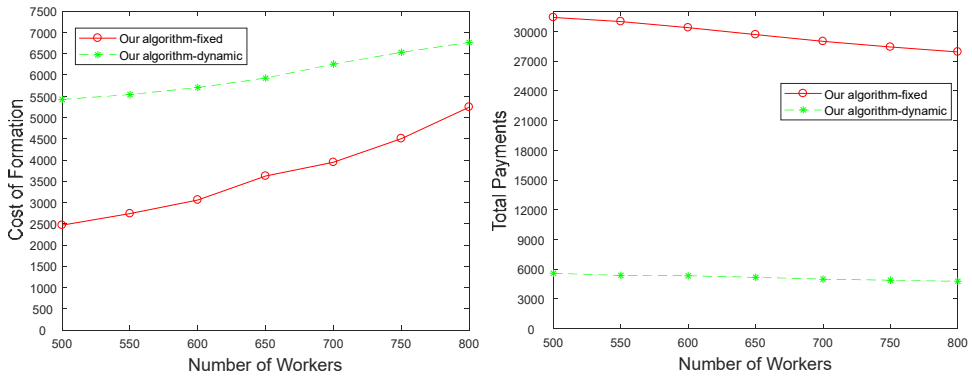
skills and location in the network, so the discounting factor has no obvious effect on the total payment. In *Our algorithm-dynamic*, the team will be adjusted to adapt to different tasks and the members of the team will be paid differently for different tasks individually; thus, it is more sensitive to the discounting factor of the budget.

5.4.4.6 Tests on the Sensitivity to Budget

Now, we test our approaches' sensitivity to the budget. The original budgets are obtained from the Upwork website. Then, we set the budget differently by multiplying the original budget by a parameter τ, which ranges from 0.5 to 1.4. Finally, the experimental results are shown in Figure 5.6. We can see that our approaches are not sensitive to the budgets of tasks in all social networks. The potential reason is that our discounting mechanism can result in the workers' real payments being less than the budget.

5.4.4.7 Tests on the Sensitivity to Number of Workers

In the above experiments, all 864 workers are considered in the team formation. We now adjust the number of candidate workers from 500 to 800 to test the performance of our approaches on the sensitivity to the number of workers. When the number of workers is small, the performance of our approaches may fluctuate erratically in random networks and scale-free networks, so here, we only present the experimental results in the small-world networks. Figure 5.7 shows the results on the cost of formation and total payments in small-world networks. We can see that while the number of workers increases, the cost of formation will also increase because more candidate workers



(a) Cost of formation

(b) Total payments

Figure 5.7: Test on the sensitivity to number of workers in small-world networks.

should be considered, but the cost of total payments will decrease slightly because more candidate workers can achieve better results in which the assigned workers' reservation wages are lower.

5.5 Summary

Previous team formation studies may be costly and cannot apply to crowdsourcing markets where the number of tasks is large because each team is tailored only for one task; moreover, the centralized manner used in these studies may place a heavy burden on requesters. Therefore, this thesis presents a distributed team formation approach for a batch of tasks in which tasks with similar skill requirements can be addressed in a batch to save computational cost and where workers can self-organize the team formation though their social networks.

We formalize the optimization objective of the problem and show that it is NP-hard; then, we present two types of heuristic approaches. The experiments on a real-world dataset show that our two presented approaches have better performance in terms of the three typical types of cost in team formation and the success ratio of tasks by comparing them with previous benchmark approaches.

Chapter 6

Group-Oriented Task Allocation for Crowdsourcing in Social Networks

Team formation in previous studies on crowdsourcing of complex tasks has two typical limitations: 1) team members are called transiently and might have no cooperation experience, thus they might not be sufficiently cooperative and reliable to execute the assigned task; and 2) a team is artificially tailored for only a special task and thus cannot adapt to general crowdsourcing markets in which tasks vary significantly. To address these two limitations while considering a notable observation that workers are often naturally organized into groups through social networks, this chapter explores a new crowdsourcing paradigm in which the task allocation targets are naturally existing worker groups but not artificially-formed teams or individual workers. An assigned group often needs to coordinate with other groups in the social network contexts for performing a complex task since such natural group might not possess all of the required skills to complete the task. Therefore, a metric of contextual crowdsourcing value is presented to measure a group's capacity to complete a task by coordinating with its contextual groups, which determines the probability that the group is assigned the task; then the task allocation algorithms, including the allocations of groups and the workers actually participating in executing the task, are designed. The experiments on a real-world dataset show that the two typical limitations in previous studies can be

addressed well by our presented approach with good synergy performance, consistency performance, conflict performance, adaptability, and effectiveness of assigned workers.

The rest of this chapter is organized as follows. In Section 6.1, we present the motivation and problem description; in Section 6.2, we model the groups in crowdsourcing; in Section 6.3, we present the context-aware task allocation model; in Section 6.4, we provide the experimental results; finally, we discuss and conclude the chapter in Section 6.5.

6.1 Motivation, Problem Description, and Complexity Analyses

To illustrate the research problem clearly, we first introduce the research motivation by analyzing some real-world crowdsourcing data, and state our investigated problem of group-oriented crowdsourcing in Section 6.1.1. Then, we analyze the complexity of the research problem in Section 6.1.2.

6.1.1 Motivation and Problem Description

We analyzed data from the leading crowdsourcing website, www.upwork.com, in which there are two types of freelancers, independent freelancers and agency freelancers. An independent freelancer is an individual worker, whereas an agency freelancer means a group that includes several managers and developers. After randomly collecting 9018 workers, we find that there are 4018 workers affiliated with at least one group, i.e., 45% workers are organized into groups. Moreover, we find that there were 226449 groups registered at https://github.com from January 1, 2016 to June 30, 2016. These data denotes that groups are quite common at many crowdsourcing websites. Therefore, these observations motivate our study on group-oriented crowdsourcing and offer us concrete cases.

As stated above, team formation is tailored artificially for a special task, and teammates might be non-cooperative. In comparison, the people within a group can cooperate with one another easily because they have some common characteristics. Therefore, we think that a group-oriented approach needs to be explored in the crowdsourcing systems, which can utilize the cooperation of workers within the groups.

However, because groups are organized naturally rather than tailored for any tasks, some problems need to be addressed when we allocate a task to a natural group. First, *how can the appropriate group that can mostly match the skill requirements of the task be found*. Second, *how can the appropriate group that requires fewer payments be found*, to save more budgets for the requester and retain more funds to seek the assistance of other groups. Third, *how can the appropriate group with lower communication distance and costs be found*, because previous benchmark studies have shown that the communication distance and costs between allocated workers will significantly influence the performance in completing the outsourced task [57]. The communication distance between two workers in a social network can determine the communication cost between them, such as the communication time (because two workers who have closer social connection may communicate more easily, and vice versus), which can influence the execution time of the outsourced task; moreover, the communication distance between two workers in a social network can influence the reliability of the cooperation between them (because two workers who have closer social connection may have more cooperation experiences), which influences the reliability of the task's result [56, 123].

Therefore, group-oriented crowdsourcing should find a group to optimize the following three factors: 1) the degree to which the skills of the workers in the group satisfy the necessary skills required by the task; 2) the occupancy rate of the wages of the workers in the group on the task's budget; and 3) the communication costs among the workers in the group to execute the task. A group might often not have the complete set of skills to implement the allocated task. Therefore, the group needs to coordinate with other contextual groups in the social network to obtain the lacking skills. Thus, the contextual groups' situations should be considered.

Let a crowd of workers be organized into n groups, $G = \{G_i | 1 \leq i \leq n\}$. Given a complex task t with a budget b_t, the set of skills required by t is S_t. Let the set of workers

in G_i actually participating in executing the outsourced task be $W_{G_i}(t)$, $W_{G_i}(t) \subseteq G_i$. We use γ_{ix} to denote the reservation wage of w_{ix}, where w_{ix} is a worker in G_i; $C_{ix,iy}$ denotes the communication cost between workers w_{ix} and w_{iy}, and d_{G_i,G_j} denotes the communication distance between groups G_i and G_j in the social network. Let α and $1-\alpha$ denote the relative importance between a group and its contexts, and, β_1, β_2 and β_3 denote the relative importance of the three factors. The context-aware task allocation objective in group-oriented crowdsourcing for t can now be formalized as selecting a group, G_*, which can satisfy the following equation:

$$gs = \frac{|(\bigcup_{w_{ix}} S_{ix}) \cap S_t|}{|S_t|}$$

$$gw = \sum_{w_{ix} \in W_{G_i}(t)} \gamma_{ix}$$

$$gcc = \sum_{w_{ix}, w_{iy} \in W_{G_i}(t)} C_{ix,iy}$$

$$gcr = \frac{|(\bigcup_{w_{jy} \in G_j} S_{jy}) \cap (S_t - \bigcup_{w_{ix} \in G_i} S_{ix})|}{|S_t - \bigcup_{w_{ix} \in G_i} S_{ix}|}$$

$$G_* = \arg\max_{G_i \in G} \left(\alpha \cdot \frac{\beta_1 \cdot gs}{(\beta_2 \cdot gw) \cdot (\beta_3 \cdot gcc)} + (1-\alpha) \sum_{G_j \in (G - \{G_i\})} gcr \cdot \frac{1}{d_{G_i,G_j}} \right) \quad (6.1)$$

$$subject\ to \sum_{w_{ix} \in W_{G_i}(t)} \gamma_{ix} \leq b_t \quad (6.2)$$

In the above formulas, gs, gw, and gcc denote skills coverage, wage, and communication costs factors within a group, and gcr denotes group's contextual resources. With the above objective, when a task is outsourced to a group, the group's self-situation, the contextual groups' situations, and the distance between the group and the contextual groups should all be considered. Therefore, the probability of a group to be assigned a task is determined not only by the group itself but also by its contextual groups. If a group has higher values for the three factors, it has higher probability to be assigned the task. However, a group may also have higher probability to be assigned the task even it has lower values for the three factors but its contextual groups have higher values for the three factors.

Therefore, the problems of *group-oriented crowdsourcing when contexts are aware* can be described as follows.

1) The situations of the contextual groups can influence the performance of a group in performing the task. Thus, the crowdsourcing value of G_i, i.e., the probability that G_i obtains the task, is influenced by G_i' contextual groups in the social network as well as G_i itself. Moreover, the communication costs between G_i and G_i' contextual groups in the social network will also influence the performance. Therefore, how to measure the crowdsourcing value of G_i by considering G_i' contexts is a problem to be investigated. (*Modeling the groups*)

2) In fact, a group often needs to coordinate with its contextual groups in the social network to execute complex tasks. Then, how can an efficient coordination mechanism between groups to ensure that the task can be completed with higher efficiency be designed? (*Coordination among groups*)

3) The execution of a complex task needs the skills of more than one group. Then, how can an efficient method to select assistant groups in the contexts as well as an efficient allocation mechanism to allocate the task to a principal group be designed? (*Allocation of groups*)

6.1.2 Complexity Analyses

Theorem 6.1. *The context-aware task allocation problem in group-oriented crowdsourcing satisfying the objective in Equation* (6.1) *is NP-hard.*

Proof. We consider a simple and special version of this problem; if the simple version is NP-hard, then our problem is also NP-hard. The simple version is that there is only one group in the social network, whether we can find a set of workers from the group that satisfies

$$\max \left(\overbrace{\sum_{w_{ix} \in W_{G_i}(t)} \gamma_{ix}}^{\text{Wages}} \cdot \overbrace{\sum_{w_{ix}, w_{iy} \in W_{G_i}(t)} C_{ix,iy}}^{\text{Communication costs}} \right)^{-1}.$$

We prove the decision version of the problem is NP-hard. The decision problem asks whether a set of workers exists from the group that satisfies $1/(\sum_{w_{ix} \in W_{G_i}(t)} \gamma_{ix} \cdot \sum_{w_{ix}, w_{iy} \in W_{G_i}(t)} C_{ix,iy}) = q$, where q is a constant number.

We prove the theorem by a reduction from the set cover problem which is well known to be NP-hard [140]. An instance of the set cover problem consists of a universe U of n elements, a collection of subsets of U, $S = \{S_1, ..., S_m\}$, and a cost function $b : S \rightarrow Q^+, \{b_1, ..., b_m\}$. Given a constant t, the decision problem asks whether we can find a sub collection of S that covers all elements of U, and the total cost of them is t.

We transform the instance of the set cover problem to an instance of our problem as follows. Every worker corresponds to a subset x, worker w_{ix}'s skill set is $S_{ix} = S_x$, his wage is $\gamma_{ix} = b_x$, the communication cost between worker w_{ix} and worker w_{iy} is $C_{ix,iy} = b_x + b_y$, $q = 1/((k-1)t^2)$, and k is the number of the selected subsets.

We shall show that the total cost is t if and only if the group satisfies $1/(\sum_{w_{ix} \in W_{G_i}(t)} \gamma_{ix} \cdot \sum_{w_{ix}, w_{iy} \in W_{G_i}(t)} C_{ix,iy}) = q$. Suppose that the total cost is t and the number of subsets selected is k. Thus, the number of selected workers is k, their communication cost $(k-1)t$, and $1/(\sum_{w_{ix} \in W_{G_i}(t)} \gamma_{ix} \cdot \sum_{w_{ix}, w_{iy} \in W_{G_i}(t)} C_{ix,iy}) = \frac{1}{(k-1)t^2}$.

Suppose the group we find satisfies $1/(\sum_{w_{ix} \in W_{G_i}(t)} \gamma_{ix} \cdot \sum_{w_{ix}, w_{iy} \in W_{G_i}(t)} C_{ix,iy}) = q$, and the number of selected workers is k. Then, $q = \frac{1}{(k-1)(\sum_{w_{ix} \in W_{G_i}(t)} \gamma_{ix})^2}$, and the total cost of selected subsets is t. The decision version of the simple version of our problem is now proved NP-hard, so we can obtain Theorem 6.1. □

To solve the NP-hard problem, we present a heuristic approach that can be realized within a limited time complexity. In the approach, we define a function of crowdsourcing value that combines the factors in Equation (6.1) to measure the probability of a group being selected to participate in a task. Then, the group with higher crowdsourcing value can be preferentially selected to participate in performing the task.

6.2 Modeling the Groups in Crowdsourcing

Groups are very common organization forms in society [33, 141]. In general, there are two typical types of groups, one is the groups with leaders, and the other is the groups without leaders [63, 142]. The former indicates that the persons in one group are all coordinated by the leader; thus, the communication costs are largely determined by the communication distances between the leader and all common persons in the group. The latter form indicates that the persons in the group can coordinate with one another; thus, the communication costs are largely determined by the total communication distances among the persons in the group.

In crowdsourcing environments, in which a crowd of workers is connected by social networks, the workers can cooperate to complete a complex task. Generally, intragroup cooperation is much easier than intergroup cooperation [143]. Therefore, without loss of generality, we make the following two assumptions: 1) a worker will definitely accept a cooperation request within the same group; 2) a group will accept a cooperation request outside the group only if the group's threshold can be satisfied, e.g., the group's desire for momentary award or other factors can be satisfied.

6.2.1 Groups with Leaders

In a group with a leader, all workers will communicate with the leader, and only the leader communicates with the requester. Let there be a group of workers in the social network, $G_i = \{w_{ix}\}$, whose leader is w_{il}, $w_{il} \in G_i$. In the group, all tasks should be executed under the control of w_{il}. Without loss of generality, when the leader wants to select a worker to participate in the execution of the task, the following three aspects of the worker will be considered: 1) the degree to which the worker's skills satisfy the current lacking skills required by the task; 2) the occupancy rate of the worker's wage on the task's budget; and 3) the communication distance between the leader and the worker. The probability of a worker being selected by the leader within the group to participate in the execution of task can be defined as the crowdsourcing value of the worker within the group.

Definition 6.1 (Crowdsourcing value of a worker within the group with a leader)**.** *Given a budget b_t for an outsourced task t, the necessary skills to complete t is S_t; let $\overline{S_t}$ be the set of skills for t that are currently lacking. The crowdsourcing value of w_{ix} within group G_i (whose leader is w_{il}) for executing task t is as follows:*

$$v_x^{G_{i,l}}(t) = \frac{|S_{ix} \cap \overline{S_t}|/|\overline{S_t}|}{(\gamma_{ix}/b_t) \cdot d_{ix,il}} \tag{6.3}$$

where S_{ix} and γ_{ix} denote the skills and reservation wage of w_{ix}; $d_{ix,il}$ is the communication distance between w_{ix} and w_{il} in the social network.

We assume that a crowdsourced task, t, is now allocated to a group with a leader, and the leader will select the workers within the group that actually participate in executing the task according to the worker's crowdsourcing value defined in Definition 6.1, shown as Algorithm 14.

Algorithm 14: Selection of workers within a group (with a leader) for participating in executing task (t, G_i).

```
b = 0;
S_t = S_t − S_il;
W_{G_i(t)} = {w_il};
Temp_G_i = G_i − {w_il};
while S_t ≠ φ and b == 0 do
    forall the w_ix ∈ Temp_G_i do
        calculate the crowdsourcing value of w_ix for satisfying the current S_t
        according to Definition 6.1;
    w_i* = arg max_{w_ix ∈ Temp_G_i} v_x^{Gi,l}(t);
    Temp_G_i = Temp_G_i − {w_i*};
    if S_i* ∩ S_t ≠ φ then
        S_t = S_t − S_i*;
        W_{G_i(t)} = W_{G_i(t)} ∪ {w_i*};
    if Temp_G_i == φ then
        b = 1;
Output (W_{G_i(t)});
```

Lemma 6.1. *Let there be two workers in group G_i, w_{ix} and w_{iy}. If $v_x^{G_{i,l}}(t) > v_y^{G_{i,l}}(t)$, it is more probable that worker w_{ix} rather than w_{iy} will be selected by group leader w_{il} to participate in executing task t. Moreover, the union of w_{il} and w_{ix} will satisfy the task*

allocation objective in crowdsourcing for t more significantly than the union of w_{il} and w_{iy}.

Proof. 1) Now $v_x^{G_{i,l}}(t) > v_y^{G_{i,l}}(t)$, assuming that w_{ix} is not selected by w_{il} but w_{iy} is selected by w_{il} to execute task t, which denotes that the worker with the higher crowdsourcing value is not selected but the one with the lower crowdsourcing value is selected. In Algorithm 14, the selection of the real participating worker is implemented by selecting one with the maximum crowdsourcing value in the crowd of candidates within the group. Therefore, such an assumption cannot occur in reality when Algorithm 14 is used.

2) The objective of task allocation in Equation (6.1) includes three parts: skills, wages, and communication costs. Now, we can find that the first and second factors in Definition 6.1 fully correspond to the first and second parts in Equation (6.1). In fact, the communications costs of executing t by G_i are determined as follows: the communication cost between w_{il} and the requester, and the one between w_{il} and the selected worker within G_i. Now, the former is fixed; thus, the actual communication costs are influenced by the latter. Accordingly, the third factor in Definition 6.1 factually decides the third part in Equation (6.1). Therefore, If $v_x^l(t) > v_y^l(t)$, according to Definition 6.1, the comprehensive value of the three factors $(|S_{ix} \cap \overline{S_t}|/|\overline{S_t}|, \gamma_{ix}/b_t, d_{ix,il})$ of w_{ix} is higher than that of w_{iy}. Because the situation of w_{il} is fixed, we can conclude that the union of w_{il} and w_{ix} will satisfy the task allocation objective in crowdsourcing for t more significantly than the union of w_{il} and w_{iy}. □

From Algorithm 14, the final set of workers within group G_i that will actually execute t is $W_{G_i}(t)$. Because each worker in $W_{G_i}(t)$ will communicate with the leader w_{il} for executing task t, we should calculate the total communication distances between all workers in $W_{G_i}(t)$ and w_{il}:

$$D(W_{G_i}(t)) = \sum_{w_{ix} \in W_{G_i}(t)} d_{ix,il} \tag{6.4}$$

To satisfy the task allocation objective, when the requester (or the crowdsourcing system) decides whether to allocate a task to group G_i, the requester will consider the

following three factors of G_i: 1) the satisfaction degree of all workers' skills in the group for the skills required by the task; 2) the occupancy rate of the wages of the workers in $W_{G_i}(t)$ on the task's budget; and 3) the total communication distances between the leader and all workers in $W_{G_i}(t)$. Therefore, the probability of a group being allocated the task can be defined as the crowdsourcing value of the group.

Definition 6.2 (Crowdsourcing value of a group with a leader)**.** *Given a budget b_t for a task t, the necessary skills to complete t are S_t, and the crowdsourcing value of group G_i for task t is as follows:*

$$v_{G_i}(t) = \frac{\beta_1 \left(|(\bigcup_{w_{ix} \in W_{Gi(t)}} S_{ix}) \cap S_t| / |S_t| \right)}{\beta_2 (\sum_{w_{ix} \in W_{Gi}(t)} \gamma_{ix}/b_t) + \beta_3 (\sum_{w_{ix} \in W_{Gi}(t)} d_{ix,il})} \tag{6.5}$$

The three weights, β_1, β_2, and β_3, are applied to reflect the relative importance among the three factors. Definition 6.2 shows that the crowdsourcing value can fully correspond to the task allocation objective. Therefore, the group with a leader that is allocated according to its $v_{G_i}(t)$ can effectively approach the task allocation objective.

6.2.2 Groups Without Leaders

In a group without leaders, all workers can coordinate with each other autonomously. In the social network, the locality of each worker can be measured by the following definition:

Definition 6.3 (Centrality of a worker within a group)**.** *Let there be a group, G_i; the centrality of a worker, w_{ix}, in G_i, is determined by the reciprocal of the ratio of the worker's total communication distances with others to the average total communication distances of all workers with others in the group:*

$$c_x{}^{G_i} = \left(\frac{\sum_{w_{iy} \in G_i} d_{ix,iy}}{(\sum_{w_{iy} \in G_i} \sum_{w_{iz} \in G_i} d_{iy,iz})/|G_i|} \right)^{-1} \tag{6.6}$$

where $d_{ix,iy}$ is the communication distance between workers w_{ix} and w_{iy}, and $|G_i|$ is the number of workers in G_i.

The more central a worker is in the group, the fewer communication costs are needed by such a worker to coordinate with other workers within the group.

In the group without leaders, each worker has a probability of being allocated the outsourced complex task, which can be defined as the initial crowdsourcing value of a worker in the group and is influenced by the following three factors: 1) the satisfaction degree of the worker's skills for the skills required by the task; 2) the occupancy rate of the worker's wage on the task's budget; and 3) the centrality of the worker in the group.

Definition 6.4 (The initial crowdsourcing value of a worker in a group without leaders)**.** *Given a budget b_t for a task t, the necessary skills to complete t are S_t, and the initial crowdsourcing value of w_{ix} in group G_i for task t is as follows:*

$$v_x{}^{G_i}(t) = \frac{(|S_{ix} \cap S_t|/|S_t|) \cdot c_x^{G_i}}{\gamma_{ix}/b_t} \tag{6.7}$$

Based on the previous work [64], we present a mechanism for selecting participating workers to execute the outsourced task. After a worker is allocated a complex task, he will then seek another worker within the group to obtain the highest probability that can help him complete the task efficiently. Then, these two workers become the already allocated sub-group. The workers in the already allocated sub-group will then autonomously seek the next assistant worker to obtain the highest probability that can help complete the task efficiently. This iterative process will repeat until all skills required by the task can be satisfied or all workers in the group are observed. To measure the probability of a worker that can help other workers completing the outsourced task, we have the following definition:

Definition 6.5 (Assistant crowdsourcing value of a worker perceived by other workers within the group)**.** *Let there be a sub-group of workers in group G_i that is allocated for task t, $W_{G_i}(t)$, $W_{G_i}(t) \subset G_i$. Let there be a worker, w_{ix}, $w_{ix} \in G_i \wedge w_{ix} \notin W_{G_i}$. The*

crowdsourcing value of w_{ix} for assisting $W_{G_i}(t)$ to perform task t is as follows:

$$av_x^{W_{G_i}(t)} = \frac{|S_{ix} \cap \overline{S_t}|/|\overline{S_t}|}{(\gamma_{ix}/b_t) \cdot d_{ix,W_{G_i}}} \tag{6.8}$$

where $d_{ix,W_{G_i}}$ is the communication distance between w_{ix} and $W_{G_i}(t)$, and $\overline{S_t}$ is the set of skills for t that are currently lacking by $W_{G_i}(t)$, which can be calculated as follows:

$$\begin{cases} d_{ix,W_{G_i}(t)} = \dfrac{\sum\limits_{w_{iy} \in W_{G_i}(t)} d_{ix,iy}}{|W_{G_i}(t)|} \\ \overline{S_t} = S_t - \bigcup\limits_{w_{iy} \in W_{G_i}(t)} S_{iy} \end{cases} \tag{6.9}$$

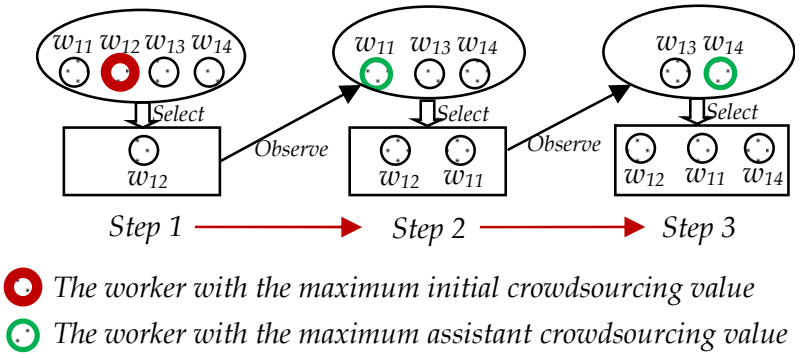


Figure 6.1: Iterative selection of participating workers in a group without leaders.

Figure 6.1 is an example illustrating this process. First, w_{12} is selected because it has the maximum initial crowdsourcing value; then, w_{12} seeks another worker with the maximum assistant crowdsourcing value for $\{w_{12}\}$, which is w_{11}; then w_{11} and w_{12} will seek another worker with the maximum assistant crowdsourcing value for $\{w_{12}, w_{11}\}$, which is w_{14}. The process can now be finished because the skill requirements of the task can be fully satisfied by $\{w_{12}, w_{11}, w_{14}\}$.

Finally, the process of selecting the workers actually participating in the execution of a task allocated to the group is shown as Algorithm 15.

Definition 6.6 (Crowdsourcing value of the group without leaders)**.** *Given a budget B_t for a task t, the necessary skills to complete t are S_t, and the crowdsourcing value of*

Algorithm 15: Selection of workers within a group (without leaders) to actually participate in executing task (t, G_i).

forall the $w_{ix} \in G_i$ **do**
  calculate the initial crowdsourcing values of w_{ix} for executing task t;
$w_{i*} = \arg\max_{w_{ix} \in G_i} {v_x}^{G_i}(t)$;
$b = 0$;
$\overline{S_t} = \overline{S_t} - S_{i*}$;
$W_{G_i(t)} = \{w_{i*}\}$;
$Temp\_G_i = G_i - W_{G_i(t)}$;
while $\overline{S_t} \neq \phi$ *and* $b == 0$ **do**
  forall the $w_{ix} \in Temp\_G_i$ **do**
    calculate the assistant crowdsourcing value of w_{ix} for assisting $W_{G_i}(t)$ to satisfy the **current** $\overline{S_t}$ according to Definition 6.5;
  $w_{i*} = \arg\max_{w_{ix} \in Temp\_G_i} {av_x}^{W_{G_i}(t)}$;
  $Temp\_G_i = Temp\_G_i - \{w_{i*}\}$;
  if $S_{i*} \cap \overline{S_t} \neq \phi$ **then**
    $\overline{S_t} = \overline{S_t} - S_{i*}$;
    $W_{G_i(t)} = W_{G_i(t)} \cup \{w_{i*}\}$;
  if $Temp\_G_i == \phi$ **then**
    $b = 1$;
Output $(W_{G_i(t)})$;

group G_i (G_i has no leaders) for task t is as follows:

$$v_{G_i}(t) = \frac{\beta_1 \left(|(\bigcup_{w_{ix} \in W_{G_i}(t)} S_{ix}) \cap S_t| / |S_t| \right)}{\beta_2(\sum_{w_{ix} \in W_{G_i}(t)} \gamma_{ix}/b_t) + \beta_3(\frac{1}{2} \sum_{w_{ix}, w_{iy} \in W_{G_i}(t)} d_{ix,iy})} \tag{6.10}$$

Lemma 6.2. *Let the set of workers participating in the outsourced task using Algorithm 15 in group G_i be $W_{G_i}(t)$ and the first selected worker with the maximum initial crowdsourcing value be w_{i*}. We use $P(X)$ to denote the probability that the task allocation objective defined in Equation (6.1) can be achieved. Another set of workers $W'(t)$ in group G_i is then assumed with the same first selected worker w_{i*}. We thus*

have the following formula:

$$\left(W'(t) \subseteq G_i \wedge w_{i*} \in W'(t) \wedge ((\bigcup_{w_{ix} \in W'(t)} S_{ix}) \cap S_t \neq \phi) \right) \Rightarrow P(W_{G_i}(t)) \geq P(W'(t))$$

Proof. The proof is similar to the second part of the proof for Lemma 6.1; here, we skip it for saving spaces. □

According to Lemma 6.2, the group without leaders that is allocated according to its $v_{G_i}(t)$ can effectively approach the task allocation objective.

6.2.3 Coordination and Contexts of Groups

6.2.3.1 Coordination Between Groups

Generally, a worker will definitely accept a cooperation request within the same group but will accept a cooperation request outside the group only if certain preconditions can be satisfied. To incentivize noncooperative groups to help one another, we are inspired by the coadjutant behaviors in society [144] and assume that the groups in the social network are coadjutant. A group G_i will have certain obligations to provide assistance for another group, G_j, if G_j has provided assistance to G_i in the past. Therefore, G_j can accept the requests of G_i for assistance even when G_i's offered monetary reward is less than G_j's reservation wage because G_j also expects to obtain the possible assistance from G_i in the future. To advance cooperation between groups, we present a term of credit between groups, shown as the following definition:

Definition 6.7 (Credit between groups)**.** *Let there be two groups, G_i and G_j, and $n_{G_j \leftarrow G_i}$ denotes the historical number of G_i's providing real assistance for G_j's executing tasks. The credit of G_i paid by G_j is in proportion to $n_{G_j \leftarrow G_i}$:*

$$c_{G_i}(\leftarrow G_j) = f(n_{G_j \leftarrow G_i}) \tag{6.11}$$

where f is a monotonically increasing function. Clearly, when $c_{G_i}(\leftarrow G_j)$ is higher, it is more compulsory that G_j should provide assistance for G_i's request even when G_i cannot provide sufficient monetary reward to G_j, because in the past, G_i has frequently assisted G_j; thus, G_j is now obligatory to compensate G_i.

When G_i requests assistance from another group G_j to execute the assigned task, G_i will promise two factors to G_j:

1) *The possible monetary reward for executing task t.* Such a factor will be estimated according to the possible contribution of G_j to satisfy the skill requirements of task t. Let S_t be the set of necessary skills required by task t, $\overline{S_t}$ be the set of skills for t that are currently lacking, and S_{jx} be the set of skills owned by worker w_{jx}. Let the amount of budget of t that is distributed to G_i be $b_t(G_i)$; then, the possible monetary reward paid by G_i to G_j is as follows:

$$m_{G_i \to G_j}(t) = \lambda \cdot (b_t(G_i) - \sum_{w_{ix} \in W_{G_i}(t)} \gamma_{ix}) \cdot \frac{|\bigcup_{w_{jx} \in G_j} S_{jx} \cap \overline{S_t}|}{|S_t|} \tag{6.12}$$

where $0 \leq \lambda \leq 1$, which denotes the percentage of monetary benefit that G_i is willing to distribute to other assistant groups.

2) *The credit paid by G_i to G_j for executing task t, $c_{G_i \to G_j}(t)$.* If $c_{G_i \to G_j}(t)$ is high and G_j hopes to request assistance from G_i in the future, G_j can accept the current request from G_i even when G_j cannot receive a satisfactory monetary reward for this request. If G_j accepts the request from G_i, then: $c_{G_i}(\leftarrow G_j) = c_{G_i}(\leftarrow G_j) - c_{G_i \to G_j}(t)$, $c_{G_j}(\leftarrow G_i) = c_{G_j}(\leftarrow G_i) + c_{G_i \to G_j}(t)$. Therefore, in real systems, we can have $|c_{G_i}(\leftarrow G_j)| = |c_{G_j}(\leftarrow G_i)|$.

Then, after receiving the request from G_i on assisting to execute task t, G_j will decide whether to accept the request according to the criteria shown in Section 6.3.2.

6.2.3.2 Contextual Crowdsourcing Values of Groups

In the social networked crowd, the communication costs will significantly influence the performance for completing the outsourced task [57].

Definition 6.8 (Communication distance between two groups)**.** *Let there be two groups G_i and G_j such that the communication distance between them, d_{G_i,G_j}, is defined as follows: 1) if G_i and G_j both have leaders w_{il} and w_{jl}, then $d_{G_i,G_j} = d_{il,jl}$; 2) if G_i has the leader w_{il} but G_j has no leaders, then $d_{G_i,G_j} = \min_{w_{jy} \in G_j} d_{il,jy}$; and 3) if G_i and G_j both have no leaders, then $d_{G_i,G_j} = \min_{w_{ix} \in G_i \wedge w_{jy} \in G_j} d_{ix,jy}$.*

The context of a group in the social network primarily means other groups that coordinate with this group through the social network. Clearly, every contextual group within the social network will contribute differently to the crowdsourcing value of a group, given G_i, which is determined by the skills owned by the workers in the contextual group, the communication distance between the contextual group and G_i, and the threshold of the contextual group.

Definition 6.9 (Contextual crowdsourcing value of a group)**.** *The contextual crowdsourcing value of group G_i for task t is defined as follows:*

$$Cv_{G_i}(t) = \alpha \cdot v_{G_i}(t) + \frac{(1-\alpha)}{|G|} \cdot \sum_{G_j \in (G - \{G_i\})} \left(\frac{|(\bigcup_{w_{jy} \in G_j} S_{jy}) \cap (S_t - \bigcup_{w_{ix} \in G_i} S_{ix})|}{|S_t - \bigcup_{w_{ix} \in G_i} S_{ix}|} \cdot \frac{\sum_{G_j \in (G - \{G_i\})} d_{G_i,G_j}}{d_{G_i,G_j} \cdot |G|} \right) \tag{6.13}$$

where G is the set of all groups in the social network; α is a parameter to measure the relative importance between a group itself and the contextual groups, $0 < \alpha < 1$.

From Definition 6.9, we have the following hypothesis: the group with closer interacting groups whose skills can further make up the skill shortcomings of the group for the outsourced task will be more effective for satisfying the task allocation objective than will other groups.

6.3 Context-Aware Task Allocation

When a complex task is outsourced, the requester (or the crowdsourcing system) will first allocate a principal group with the maximum contextual crowdsourcing value to take charge of the task. If the principal group lacks any necessary skills required by the task, other contextual groups should be allocated to assist the principal group to execute the task.

6.3.1 Allocation of Principal Group

Now the non-redundant allocation is very popular in real crowdsourcing of complex tasks, e.g., we find that 79.3% of tasks are allocated non-redundantly while we randomly count 6271 tasks at the Upwork website. Therefore, we assume that each complex task is allocated non-redundantly.

Algorithm 16: Allocation of the principal group for task t.
$/*$ Let the set of all candidate groups be G $*/$

$b = 0$;
$n = 1$;
$G_{temp} = G$;
while $b == 0$ *and* $n \leq |G|$ **do**
  $G_* = \arg\max_{G_i \in G_{temp}} Cv_{G_i}(t)$;
  $G_{temp} = G_{temp} - G_*$;
  if $\sum_{w_{ix} \in W_{G_i(t)}} \gamma_{ix} \leq b_t$ **then**
    $b = 1$;
  $n = n + 1$;
if $b == 1$ **then**
  Output (G_*);
else
  Output $(false)$;

As stated above, we will try to select the group with the maximum contextual crowdsourcing value (defined in Definition 6.9). However, each task has a budget constraint; therefore, the total reservation wages of the workers in the group that actually participate in executing task ($W_{G_i}(t)$) should not exceed the budget. Our task allocation criterion

is thus now to assign the task to a group that has the highest contextual crowdsourcing value in the set of groups in which each one's real participating workers' reservation wages do not exceed the task's budget, b_t. The process is shown as Algorithm 16.

The time complexity of Algorithm 16 is $O(|G|)$,where $|G|$ denotes the number of the candidate groups.

6.3.2 Allocation of Assistant Groups

Here we will present two approaches for the allocation of assistant groups: semi-supervised approach and fully-supervised approach.

First we apply the semi-supervised manner into the group-oriented allocation, i.e., only the allocation of the principal group is supervised by the requester (or the crowdsourcing system), and the allocation of assistant groups is conducted by the principal group autonomously. Moreover, we adopt another approach, fully-supervised manner, in which the principal group and assistant groups are all allocated by the requester (or the crowdsourcing system).

Each person has a threshold to decide his attitude to cooperate with others [145]. Therefore, here we also set each group, G_i, to have a predefined threshold τ_{G_i}; the group will accept a request for assistance from another group only if the ongoing task's benefit exceeds the threshold.

Generally, when a group, G_j, decides whether to accept a request from group G_i, it primarily considers the following factors: 1) the possible monetary reward for executing task t, $m_{G_i \to G_j}$; 2) the credit paid by G_i to G_j for executing task t, $c_{G_i \to G_j}$; 3) the cooperation history that G_i has assisted G_j, $c_{G_i}(\leftarrow G_j)$; and 4) the cooperation history that G_j has assisted G_i, $c_{G_j}(\leftarrow G_i)$. Let the threshold of G_j be τ_{G_i}. G_j will accept the request from G_i if the following condition can be satisfied:

$$\eta_1 \cdot m_{G_i \to G_j}(t) + \eta_2 \cdot c_{G_i \to G_j}(t) + \eta_3 \cdot c_{G_i}(\leftarrow G_j) - \eta_4 \cdot c_{G_j}(\leftarrow G_i) \geq \tau_{G_j} \quad (6.14)$$

where η_1, η_2, η_3 and η_4 are four parameters to determine the relative importance of the four factors.

6.3.2.1 Semi-Supervised Approach

The semi-supervised approach is often used in the situation in which the principal group has a leader who can search for other groups autonomously. Certainly, in this approach, the principal group also needs to consult the crowdsourcing system on the information of other groups, such as their skills, their thresholds, their distances, and their credits.

Algorithm 17: Semi-supervised approach for allocation of assistant groups (t, G_i).
$/*$ Let G_i be the principal group and the set of all candidate groups be G. $*/$

$\overline{S_t} = S_t - \bigcup_{w_{ix} \in G_i} S_{ix}$;
$Gass(t) = \{\}$;
$G = G - \{G_i\}$;
G_i enquires the information of other groups from the crowdsourcing system;
while $G \neq \phi$ *and* $\overline{S_t} \neq \phi$ **do**
  $G_* = \arg\min_{G_j \in G} d_{G_i,G_j}$;
  $G = G - \{G_*\}$;
  if $\bigcup_{w_{*x} \in G_*} S_{*x} \cap \overline{S_t} \neq \phi$ **then**
    if $\eta_1 \cdot m_{G_i \to G_*}(t) + \eta_2 \cdot c_{G_i \to G_*}(t) + \eta_3 \cdot c_{G_i}(\leftarrow G_*) - \eta_4 \cdot c_{G_*}(\leftarrow G_i) \geq \tau_{G_*}$ **then**
      $\overline{S_t} = \overline{S_t} - \bigcup_{w_{*x} \in G_*} S_{*x}$;
      $G_{ass}(t) = G_{ass}(t) \cup \{G_*\}$;
Output $(G_{ass}(t))$;

In real society, each person may cooperate with others initially who are neighbors. He then will cooperate with other people from near to far within the social network, which can reduce communication distances and make it more possible to cooperate with acquaintances [64]. Therefore, we now also make the principal group search for other groups from near to far within the social network until all required skills are satisfied or all groups within the social network are observed. The process is shown in Algorithm 17. The time complexity of Algorithm 17 is $O(|G|)$.

Theorem 6.2. *If all groups for performing task t can be obtained by using Algorithm 17, the total communication costs between the principal group and assistant groups can be minimized.*

Proof. Let G_i be the principal group; then the set of lacking skills of G_i to implement t is $\overline{S_t}$. If Algorithm 17 is used, the set of assistant groups is $G_{ass}(t)$, and the total communication costs between G_i and the groups in $G_{ass}(t)$ is C_*. However, if there is another set of groups, G', $G' \neq G_{ass}(t)$, that can provide all of the skills in $\overline{S_t}$, and the total communication costs between G_i and G' are C'; if $C' < C_*$, it denotes that there are any further groups that provide the required skills in $\overline{S_t}$, but the nearer groups with required skills do not provide the required skills in $\overline{S_t}$. Clearly, such a situation cannot occur in Algorithm 17. Therefore, we have Theorem 6.2. □

6.3.2.2 Fully-Supervised Approach

In the semi-supervised approach, the principal group can only observe other groups according to their distances, which might not achieve the optimal result of crowdsourcing values; moreover, the search process might be too complex for the principal group. Therefore, we now provide a fully-supervised approach, in which the assistant groups are all allocated by the requester (or the crowdsourcing system). When the requester (or the crowdsourcing system) wants to assign a group to act as an assistant group, he will measure the assistance value of such group according to the following definition.

Definition 6.10 (Assistance value of a group for another group)**.** *Let G_i be the principal group for task t. If $\overline{S_t}$ is the set of skills for t that are currently lacking, the assistance value of G_j for G_i on executing t is defined as follows:*

$$v_{G_j}(G_i - t) = \frac{\beta_1 \cdot \left(|\bigcup_{w_{jx} \in G_j} S_{jx} \cap \overline{S_t}| / |\overline{S_t}| \right) + \beta_2 \cdot c_{G_i}(\leftarrow G_j)}{\beta_3 \cdot d_{G_i,G_j} + \beta_4 \cdot \tau_{G_j}} \tag{6.15}$$

where β_1, β_2, β_3 and β_4 are four parameters to decide the relative importance of the four factors.

Finally, the fully-supervised approach for allocation of assistant groups is shown as Algorithm 18.

Theorem 6.3. *Let G_i be the principal group and the set of assistant groups for task t using Algorithm 18 be $Gass(t)$. It is, then, assumed that there is another set of groups*

Algorithm 18: Fully-supervised approach for allocation of assistant groups (t, G_i).
$/*$ Let G_i be the principal group and the set of all candidate groups be G. $*/$

$\overline{S_t} = S_t - \bigcup_{w_{ix} \in G_i} S_{ix}$;
$Gass(t) = \{\}$;
$G = G - \{G_i\}$;
while $G \neq \phi$ *and* $\overline{S_t} \neq \phi$ **do**
  $G_* = \arg\min_{G_j \in G} v_{G_j}(G_i - t)$;
  $G = G - \{G_*\}$;
  if $\bigcup_{w_{*x} \in G_*} S_{*x} \cap \overline{S_t} \neq \phi$ **then**
    if $\eta_1 \cdot m_{G_i \to G_*}(t) + \eta_2 \cdot c_{G_i \to G_*}(t) + \eta_3 \cdot c_{G_i}(\leftarrow G_*) - \eta_4 \cdot c_{G_*}(\leftarrow G_i) \geq \tau_{G_*}$ **then**
      $\overline{S_t} = \overline{S_t} - \bigcup_{w_{*x} \in G_*} S_{*x}$;
      $G_{ass}(t) = G_{ass}(t) \cup \{G_*\}$;
Output $(G_{ass}(t))$;

in the crowd, G', and the threshold of each group in G' can be satisfied if the group assists G_i for t. Thus, we have the following results:

$$\forall G' \subseteq G,\ G_j \in G',$$
$$\left(\eta_1 \cdot m_{G_i \to G_j}(t) + \eta_2 \cdot c_{G_i \to G_j}(t) + \eta_3 \cdot c_{G_i}(\leftarrow G_j) - \eta_4 \cdot c_{G_j}(\leftarrow G_i) \geq \tau_{G_j}\right)$$
$$\Rightarrow \left(\frac{\sum_{G_j \in G_{ass}(t)} v_{G_j}(G_i - t)}{|G_{ass}(t)|} \geq \frac{\sum_{G_j \in G'} v_{G_j}(G_i - t)}{|G'|}\right)$$

Proof. We can use reductio ad absurdum to prove Theorem 6.3. Assuming there is a set of groups G', $G' \neq G_{ass}(t) \wedge G' \subseteq G$, that are allocated for assisting G_i to execute the outsourced task t, and

$$\forall G_j \in G',\ (\eta_1 \cdot m_{G_i \to G_j}(t) + \eta_2 \cdot c_{G_i \to G_j}(t) + \eta_3 \cdot c_{G_i}(\leftarrow G_j) - \eta_4 \cdot c_{G_j}(\leftarrow G_i)) \geq \tau_{G_j}$$

If the assumption $(\sum_{G_j \in G_{ass}(t)} v_{G_j}(G_i - t))/|G_{ass}(t)| < \sum_{G_j \in G'} v_{G_j}(G_i - t)/|G'|$ is true, then there exists at least one group with higher assistance value perceived by G_i for t and whose threshold is satisfied by G_i but that cannot be selected by Algorithm 18, and another group with lower assistance value perceived by G_i for t will be allocated.

However, in each round for selecting the assistant group, the group with the highest assistance value for G_i for t and whose threshold is satisfied by G_i will definitely be the first to be assigned. Thus, the above assumption cannot occur in reality when Algorithm 18 is used. Therefore, we can have Theorem 6.3. □

Theorems 6.2 and 6.3 show that the advantage of the semi-supervised approach is that the communication costs between the principal group and the assistant groups can be optimized, which can significantly improve the performance for completing the outsourced task in social networks. In comparison, the advantage of the fully-supervised approach is that the maximum crowdsourcing values of the assigned groups can be achieved theoretically because all assistant groups are selected by the requester (or the crowdsourcing system) globally.

6.4 Experiments

6.4.1 Experimental Settings

The experiments are conducted on a real-world dataset extracted from a popular crowdsourcing website, GitHub (https:github.com), from which data on 3733 groups and 4661 workers (a worker can be affiliated with more than one group) are collected. For each worker, two types of personal information are collected: 1) the skills the worker owns, 2) following context (the people the worker follows) and follower context (the people who follow the worker). We initially extract the top 40 skills that are most frequently owned by workers; then, we select the workers who can provide at least one of these top 40 skills and select the groups that each contains at least two of these workers; finally, we obtain 602 groups and 1494 workers. The social network on these workers is constructed according to their following relationship, where the vertices denote the workers and the edge between two vertices denote that there is following or followed relationship between the two corresponding workers. In fact, the "following" or "being followed by" relationships are not cooperation relationships, which are only used to denote that two workers are connected in the social network.

Our two group-oriented task allocation approaches, semi-supervised approach (*Group_semi supervised*) and fully-supervised approach (*Group_fully supervised*), are compared with the following three benchmark approaches:

- Team formation algorithm based on Minimal Cost Contribution (*Team formation*) [57]. Several initial teams are formed; then the new members are selected to add to the initial teams by considering their communication costs with all of the current members of the team in addition to their personal costs. Finally, a best team is selected. The details are described in [57].

- Individual-oriented algorithm (*Individual*) [51]. The algorithm initially assigns the task to a principal worker who has the best contextual factor based on the skills the worker has, the communication distance with other workers, and the reservation wage of the worker. The principal worker then searches for other assistant workers based on the skills the worker has and the reservation wage of the worker from near to far within the social network until all of the skills required by the task are satisfied.

- Group-oriented algorithm based on skill local search (*Group_skill local search*). The algorithm initially assigns the task to a principal group that can best meet the skill requirement of the task. Then, the principal group will search from near to far within the social network for cooperation until all of the skills required by the task are satisfied.

In the experiments, all of the results are recorded by averaging over 100 instances. We run the experiment in 5 situations: the occupancy rate of groups with leaders is 0% (that is, all of the groups are without leaders), 30%, 50%, 80% and 100% (that is, all of the the groups have leaders). There are two series of parameters to determine the relative importance between different factors in the definitions of crowdsourcing values: we use α and $1-\alpha$ to denote the relative importance between a group and its contexts respectively; moreover, we use β_1, β_2 and β_3 to denote the relative importance among the three factors for determining crowdsourcing value. In the experiments in Sections 6.4.2, 6.4.3, and 6.4.4, these parameters are: $\alpha = 0.75$; $\beta_1 = 1$, $\beta_2 = 0.5$, and $\beta_3 = 0.5$.

6.4.2 Experiments on the Performance

We test the following six types of performances in experiments: 1) synergy performance among the assigned workers; 2) consistency performance of the assigned workers; 3) conflict performance of the assigned workers; 4) adaptability for varying tasks; 5) average pairwise communication costs among the assigned workers; and 6) reservation wages of the assigned workers. Metrics 1, 2 and 3 can test the effect of our approach on addressing the first limitation of team formation; metric 4 can test the effects of our approach on addressing the second limitation of team formation; and metrics 5 and 6 can measure the efficiency of our approach.

6.4.2.1 Synergy Performance Among Assigned Workers

In this section, we test the synergy performance between the workers selected by the group-oriented task allocation and that of selected by other benchmark approaches.

We use the weighted synergy graph to model the task-based relationship among workers, in which the vertices represent workers and the edges represent the task-based relationship between workers [146, 147]. If two workers are originally in a group or are assigned to cooperate for the first time to complete a task, the corresponding two vertices are connected by an edge with an initial weight of 100; then the weight will be reduced by one for each cooperation. The distance between two vertices is the weight between them. The shorter the distance is, the more cooperative they are. Suppose that there are two workers x and y, let $d(x, y)$ be the shortest distance between these two workers in the task-based relationship graph. Let $s(x, y)$ denote the pairwise synergy value of x and y:

$$s(x, y) = 5 - \ln(d(x, y)) \tag{6.16}$$

Let $S(A)$ denote the synergy value of the set of workers A, which is the average of the pairwise synergy values of all worker pairs in A:

$$S(A) = \frac{1}{\binom{A}{2}} \sum_{\{x,y\} \in A} s(x, y) \tag{6.17}$$

The larger the synergy value of a set is, the more harmonious the cooperation among workers in the set is [146].

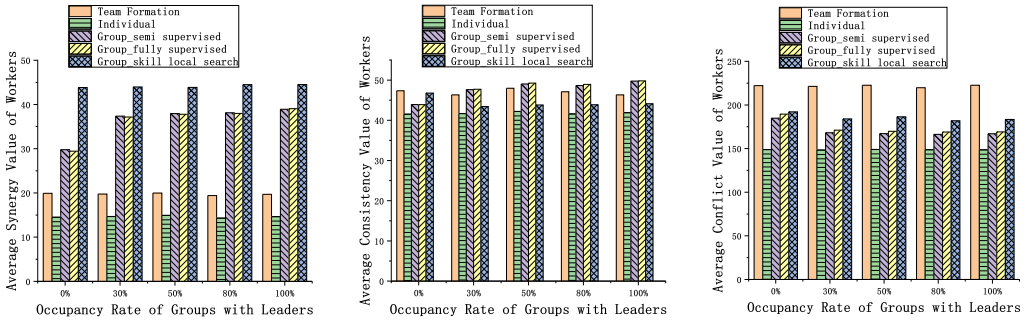
We now compare our proposed two group-oriented approaches with the other three benchmark approaches on the synergy performance of the assigned workers. The results on the average synergy values of 100 tasks in five situations (with varying occupancy rates of groups with leaders) are shown in Figure 6.2a. The results show that the three group-oriented approaches can achieve higher synergy values than can the other two approaches; thus, the group-oriented crowdsourcing paradigm can achieve better cooperation performance among workers. In addition, the workers selected by the *Group_skill local search* approach achieve the highest synergy value; the potential reason is that the approach only considers the skills of the workers but ignores the wages of the workers and the communication costs between them; therefore, the workers selected by the approach are within fewer groups and then can achieve higher synergy value.

6.4.2.2 Consistency Performance of the Assigned Workers

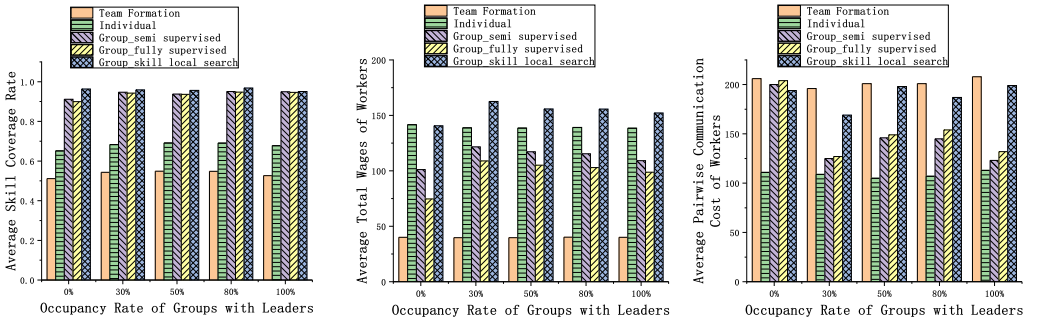
It is well known that decision-making is crucial in teamwork, thus whether members can reach agreements is an important aspect of judging the quality of the group [148]. In social networks, closer relationship helps groups achieve consensus more easily [135]. In graph theory, Clustering Coefficient is used to measure the degree of the nodes tend to cluster. Generally, the higher the Clustering Coefficient is, the closer the nodes are in the graph [149]. Therefore, we use the average Clustering Coefficient $C(A)$ to measure the consistency of the selected workers set A [135].

In graph $G(V, E)$, $v_i \in V$ indicates a worker in set A. Let N_i be the set of neighbors of node v_i in the social network contexts, and $e_{j,k} \in E$ indicates that there is a direct connection between nodes v_j and v_k. Thus, the consistency value of v_i can be defined as:

$$C_i = \frac{2 \cdot |\{e_{j,k} : v_j, v_k \in N_i, e_{j,k} \in E\}|}{|N_i| \cdot (|N_i| - 1)} \tag{6.18}$$



(a) Synergy performance (b) Consistency performance (c) Conflict performance



(d) Adaptability for varying tasks (e) Effectiveness-total reservation wages (f) Effectiveness-average pairwise communication cost

Figure 6.2: Experiments on the performance.

The average consistency value of set A is defined as:

$$C(A)=\frac{1}{|V|}\sum_{i=1}^{|V|}C_i \tag{6.19}$$

Intuitively, higher average Clustering Coefficient means that the group is closer and members can reach agreements more easily [150].

The results on the average consistency value of 100 tasks in the five situations are shown in Figure 6.2b. The results show that when the occupancy rate of groups with leaders is 0%, our two group-oriented approaches have relatively poor performance than the *Team formation* approach and *Group_skill local search* approach; while the rate increases to 30% and more, our two approaches achieve higher consistency value than other approaches. The reason is that the group with a leader will select workers who have social relationship with the leader more probably; therefore, the group members will be closer.

6.4.2.3 Conflict Performance of the Assigned Workers

The social relationship of the group members is one of the causes of conflict. The distance between two workers in a social network can reflect their familiarity, and two workers who are unfamiliar with each other may result in conflicts in teamwork [151]. The Harmonic Mean of Average Path Length (HMAPL) of the group in social networks could indicate the group's potential of conflict [149, 151].

In graph $G(V, E)$, $v_i \in V$ indicates a worker in set A. Let $d(x, y)$ be the shortest distance between workers x and y in the social network graph. The HMAPL of A is defined as:

$$APL(A) = \frac{|V| \cdot (|V| - 1)}{\sum\limits_{x \neq y} \frac{1}{d(x,y)}} \tag{6.20}$$

Intuitively, smaller HMAPL of a group means that the members are more familiar and the group does not have conflicts easily [149].

The results in Figure 6.2c show that our two group-oriented approaches have relatively better performance than *Team formation* approach and *Group_skill local search* approach, but not to *Individual* approach. The potential reason is that the principal worker searches for other assistant workers from the near to the distant within the social network in *Individual* approach; thus, the assigned workers are more familiar. In particularly, when the occupancy rate of groups with leaders increases to 30% and more, the gap between our two approaches and the *Individual* approach gradually narrows, as the group leader could select workers who have social relationship with the leader more probably.

6.4.2.4 Adaptability of the Assigned Workers

In this section, we test the adaptability of the workers selected by our group-oriented approaches for varying tasks.

We select 100 tasks randomly, and then we use the five approaches to select five sets of workers for the first task. Then, we use the remaining tasks to examine the adaptability of the five sets. First, for each of the remaining tasks, we apply the *Individual*

approach to the five worker sets to select five subsets of workers respectively. We then count the average skill coverage rate of the skills of the five selected worker subsets. The higher the average skill coverage of the workers selected by the approach is, the better the approach can adapt to varying tasks.

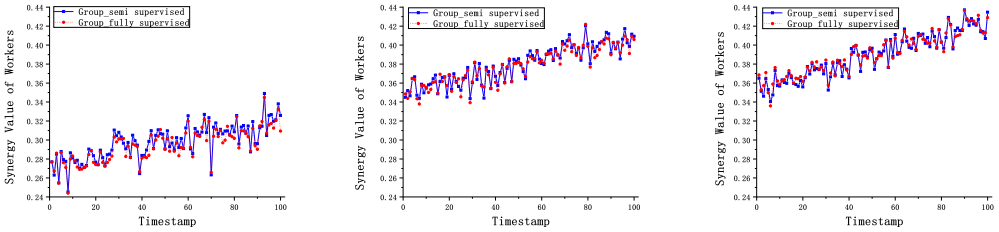
The results on the average skill coverage rate of the 100 tasks in five situations are shown in Figure 6.2d. The results show that the workers selected by the group-oriented approaches can obtain a higher skill coverage rate than can the workers selected by the other two approaches; thus, groups can be assigned varying tasks, and the group-oriented approaches are more adaptive than the *Team formation* approach and *Individual* approach in crowdsourcing markets.

6.4.2.5 Effectiveness of the Proposed Approaches

To evaluate the effectiveness of our approaches, we compare them with the benchmark approaches on the following performance metrics: the total reservation wages of all selected workers, and the average pairwise communication costs among all selected workers.

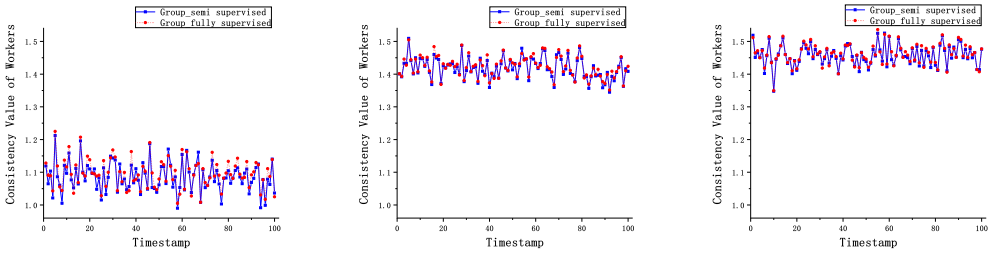
Figure 6.2e shows the average of the selected workers' total wages for 100 tasks. The results denote that both the average values of the total wages of the workers selected by the *Group_semi supervised* approach and *Group_fully supervised* approach are higher than that of the workers selected by *Team formation* approach, but lower than that of the workers selected by *Individual* approach and *Group_skill local search* approach. When the occupancy rate of groups with leaders is 0%, the average of total reservation wages of the workers selected by our two approaches is relatively lower than other situations in which the rate increases to 30% and more.

The results on the average pairwise communication cost of the assigned workers for 100 tasks in five situations are shown in Figure 6.2f. The results show that the performance of the average pairwise communication costs of workers selected by the *Group_semi supervised* approach and *Group_fully supervised* approach are superior to that of *Team formation* approach and are inferior to *Individual* approach. Moreover,



(a) (b) (c)

Figure 6.3: Dynamics experiments on the synergy of assigned workers.



(a) (b) (c)

Figure 6.4: Dynamics experiments on the consistency of assigned workers.

when the occupancy rate of groups with leaders is 0%, our two group-oriented approaches have relatively worse performance than *Group_skill local search* approach; while in other four situations in which the occupancy rate of groups with leaders is 30% and more, our two group-oriented approaches have an obvious advantage over *Group_skill local search* approach and narrow the gap with the *Individual* approach. The potential reason is that groups with leaders will select workers who have social relationship with leaders more probably; therefore, the average pairwise communication cost becomes lower.

6.4.3 Experiments on the Dynamics

6.4.3.1 Dynamics of the Synergy Performance of the Assigned Workers

From the definition on synergy performance in Section 6.4.2.1, the synergy values are dynamically changed with the cooperation number between workers. Now we set the number of tasks as the timestamp, which is set from 1 to 100; then we test the dynamics of synergy performance of the assigned workers of our two approaches, *Group_semi*

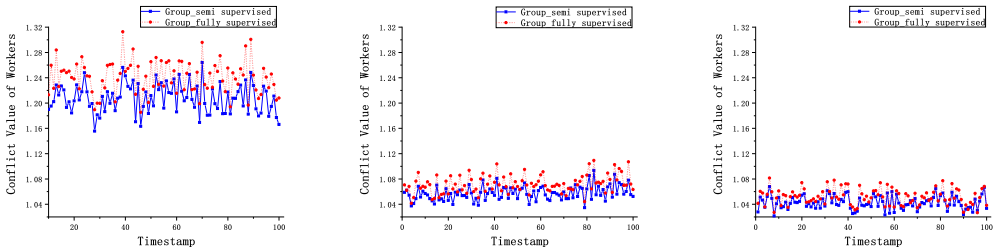


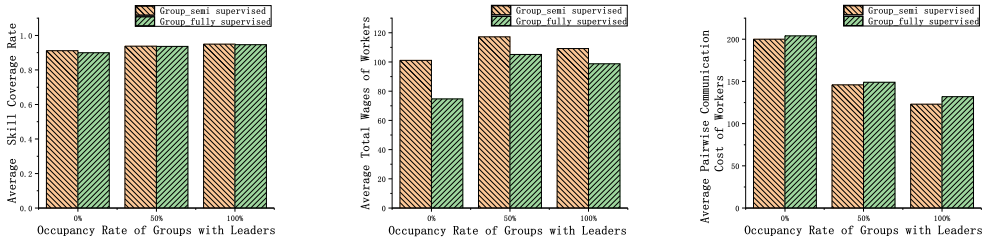
Figure 6.5: Dynamics experiments on the conflict of assigned workers. ((a): occupancy rate of groups with leaders is 0%; (b): occupancy rate of groups with leaders is 50%; (c): occupancy rate of groups with leaders is 100%)

supervised approach and *Group_semi supervised* approach. The results on three types of situations with different occupancy rates of groups with leaders are shown in Figure 6.3.

We can see that the synergy value of the workers selected by the *Group_semi supervised* approach shows a slight advantage compared to the *Group_fully supervised* approach. The potential reason is that when using the *Group_semi supervised* approach, the principal group searches other groups for help from the near to the distant within the social network; therefore, it is more possible that adjacent groups had cooperation in the past and that the workers selected by *Group_semi supervised* approach can achieve a higher synergy value.

6.4.3.2 Dynamics of the Consistency Performance of the Assigned Workers

According to Section 6.4.2.2, the consistency performance is related to the social relationship between workers. If two workers in the same group do not have social relationships originally, they will be connected after completing a task. Now we set the number of tasks as the timestamp, which is set from 1 to 100; then we test the dynamics of the consistency values of the assigned worker groups of our two approaches, shown in Figure 6.4. We can see that the dynamics of the consistency values of the workers selected by our approaches are very close.



(a) Adaptability for varying tasks

(b) Effectiveness-total reservation wages

(c) Effectiveness-average pairwise communication cost

Figure 6.6: Experiments on the comparison between our two approaches on three performance metrics.

6.4.3.3 Dynamics of the Conflict Performance of the Assigned Workers

Now we test the dynamics of the APL values of the selected workers, shown in Figure 6.5. We can see that the Harmonic Mean of Average Path Length of group of workers selected by the *Group_semi supervised* approach is significantly smaller than the *Group_fully supervised* approach, which indicates that the workers selected by the *Group_semi supervised* approach are less likely to have conflict. The potential reason is that when using *Group_semi supervised* approach, the principal group searches other groups for help from the near to the distant within the social network, thus the selected workers are more familiar.

6.4.4 Comparison between Our Semi-Supervised Approach and Fully-Supervised Approach on Other Performance Metrics

Now we compare the *Group_semi supervised* approach with the *Group_fully supervised* approach on other three performance metrics: adaptability performance, the selected workers' total wages, and the selected workers' pairwise communication costs.

The average skill coverage rate of 100 tasks is shown in Figure 6.6a, in which the average skill coverage rates of workers selected by *Group_semi supervised* approach and *Group_fully supervised* approach are close.

Figure 6.6b and c show the average of the total reservation wages and average of the pairwise communication costs of 100 tasks. As seen, the average of total reservation

wages of workers selected by the *Group_semi supervised* approach is higher than the one by the *Group_fully supervised* approach, whereas the average total communication costs of workers selected by the *Group_semi supervised* approach is lower than that selected by the *Group_fully supervised* approach. The reason is that the principal group, which uses the *Group_semi supervised* approach, searches other groups for help largely depending on distance between groups. However, when using the *Group_fully supervised* approach, the principal group searches for assistant groups by considering not only the distance between groups but also other information such as wages; therefore, the workers selected by the *Group_semi supervised* approach can result in lower communication costs, whereas the workers selected by the *Group_fully supervised* approach can result in lower reservation wages.

6.4.5 Experiments on the Influence of Parameters

Now we make a series of experiments on the uncertainty of our presented *Group_semi supervised* approach and *Group_fully supervised* approach resulted from the following three parameters: 1) relative importance factor between a group and its contextual groups; 2) number of skills of task; and 3) number of tasks. We run the experiment in three situations: the occupancy rate of groups with leaders is 0% (that is, all of the groups are without leaders), 50%, and 100% (that is, all of the groups have leaders). In the tables, ①, ②, ③, ④, and ⑤ denote *Team formation* approach, *Individual* approach, *Group_semi supervised* approach, *Group_fully supervised* approach, and *Group_skill local search* approach, respectively.

6.4.5.1 Tests on the Relative Importance Factor

In Equation (6.13), the parameter α is used to measure the relative importance between a group itself and the contextual groups, $0 < \alpha < 1$. Now we adjust the parameter from $\alpha = 0.25$ to $\alpha = 1.0$, with the interval of 0.25, to test the six types of performances of our two approaches in three situations. The experiment is conducted under the fixed wages of workers and tasks, that is, the number and types of skills required for the tasks are the same. The results are shown in Table 6.1.

TABLE 6.1: Experimental results on the influence of relative importance factor in Equation (6.13)

Occupancy Rate of Leader		0.0				0.5				1.0			
α		0.25	0.50	0.70	1.0	0.25	0.50	0.70	1.0	0.25	0.50	0.70	1.0
Average synergy value of workers	③	16.7	23.1	29.7	32.1	19.1	31.6	37.2	38.9	19.1	32.9	38.8	40.0
	④	14.6	22.3	29.4	31.9	16.0	30.8	37.0	38.8	16.6	32.5	38.9	40.1
Average consistency value of workers	③	38.3	41.3	44.2	45.7	36.9	44.6	48.4	49.9	37.1	46.5	50.2	51.1
	④	40.3	42.0	44.4	45.9	38.5	45.2	48.6	50.0	38.8	47.2	50.4	51.2
Average conflict value of workers	③	184.0	183.8	184.9	185.6	163.4	167.3	168.4	168.3	162.9	167.1	169.1	169.5
	④	203.5	193.1	190.2	189.7	184.7	175.0	171.6	170.9	184.8	174.5	171.5	171.1
Average skill coverage rate	③	0.85	0.88	0.89	0.89	0.92	0.93	0.92	0.93	0.93	0.95	0.95	0.95
	④	0.84	0.88	0.88	0.89	0.91	0.92	0.92	0.93	0.93	0.95	0.95	0.95
Average total wages of workers	③	81.9	61.6	50.9	47.6	107.5	73.2	61.7	58.3	106.2	67.5	55.2	52.6
	④	48.3	40.5	37.8	37.8	74.5	58.9	54.3	53.1	71.1	54.9	49.2	48.2
Average pairwise communication of workers	③	182	190	182	188	133	149	154	148	138	146	154	157
	④	199	197	190	191	174	162	158	151	175	161	160	161

From Table 6.1, as α increasing from 0.25 to 1.0, the average synergy values and average consistency value of workers selected by *Group_semi supervised* approach and *Group_fully supervised* approach are both increasing. This result indicates that when selecting a principal group for task, if the crowdsourcing value of the group itself has a greater weight, the workers selected in our two approaches will work more harmoniously and have closer relationship. It is worth noting that the trend of growth is getting slower with the increasing of α.

In terms of average conflict value, while α increases from 0.25 to 1.0, the value comes down slowly with the workers selected by *Group_fully supervised* approach but fluctuates slightly with the workers selected by *Group_semi supervised* approach in three situations. To achieve balance, it is recommended to set α to 0.75.

In terms of average skill coverage rates, our two approaches both have unobvious upward trend as α rises and approximately converge when α is 0.5. The average total wages of workers for our two approaches decrease slightly with the increasing of α. Finally, the two approaches both have slight fluctuation in terms of the average pairwise communication costs of the selected workers. From the results, we can see that it is appropriate to set α to 0.75.

6.4.5.2 Tests on the Skill Number of Tasks

Now we test the influence of skill number of tasks on the six types of performances of the five approaches. The scopes of skill number of tasks in the experiments are set as [5, 10], [10-15], [15-20], [20-25]. The parameter of the relative importance factor α is set to 0.75. The experimental results are shown in Table 6.2.

TABLE 6.2: Results of parameter test on skill number of tasks

Occupancy Rate of Leader		0.0				0.5				1.0			
Skill number of tasks		5∼10	10∼15	15∼20	20∼25	5∼10	10∼15	15∼20	20∼25	5∼10	10∼15	15∼20	20∼25
Average synergy value of workers	①	13.7	22.9	31.8	39.8	13.3	22.8	32.0	40.0	13.5	23.4	31.8	39.3
	②	12.8	16.3	19.7	19.6	13.2	16.5	18.6	19.6	13.6	16.8	18.5	19.7
	③	32.0	28.1	30.7	36.7	36.6	38.1	41.1	45.2	38.2	39.4	40.9	46.5
	④	31.8	27.6	29.5	34.6	36.4	37.8	40.7	44.6	38.3	39.4	40.6	46.3
	⑤	43.8	43.8	44.8	47.4	43.5	44.4	46.0	50.9	44.0	44.3	45.5	50.6
Average consistency value of workers	①	48.4	45.4	45.6	45.6	48.5	45.9	46.2	46.3	47.4	46.5	46.0	46.9
	②	43.4	41.3	39.8	37.6	43.4	41.5	39.3	37.9	43.3	41.5	39.6	37.1
	③	44.4	43.8	42.7	44.3	45.2	49.7	51.2	52.6	47.7	51.0	51.6	53.1
	④	44.5	43.3	41.9	42.4	45.4	50.0	51.5	52.9	47.9	51.1	52.0	53.7
	⑤	44.6	47.3	52.9	57.9	41.8	45.8	50.0	56.0	41.8	44.9	50.1	55.2
Average conflict value of workers	①	218.0	223.0	228.3	230.4	219.2	223.4	227.7	229.7	217.6	222.5	229.3	231.6
	②	145.6	150.0	152.0	156.6	146.8	149.9	153.6	156.3	145.9	149.6	154.3	155.9
	③	182.1	184.5	192.1	197.0	163.9	169.7	175.4	179.0	165.2	169.7	175.9	177.5
	④	184.8	189.1	200.4	207.8	166.8	173.1	178.8	182.3	167.4	172.1	179.8	181.1
	⑤	184.2	196.8	194.7	195.3	177.4	191.8	188.7	185.7	176.4	192.4	192.6	186.0
Average skill coverage rate	①	0.60	0.72	0.79	0.47	0.59	0.70	0.79	0.46	0.61	0.71	0.78	0.48
	②	0.74	0.81	0.88	0.63	0.73	0.81	0.87	0.63	0.76	0.81	0.87	0.61
	③	0.92	0.96	0.96	0.88	0.96	0.97	0.98	0.93	0.96	0.97	0.98	0.94
	④	0.90	0.94	0.95	0.86	0.96	0.96	0.98	0.93	0.96	0.97	0.98	0.94
	⑤	0.98	0.99	0.99	0.93	0.99	0.99	0.99	0.93	0.98	0.99	0.99	0.94
Average total wages of workers	①	15.0	23.4	30.8	37.8	15.0	23.8	30.6	37.5	15.0	23.8	30.6	37.5
	②	51.6	81.1	105.9	117.9	51.4	83.4	105.8	116.7	51.4	83.4	105.8	116.7
	③	34.5	61.4	86.2	106.3	41.6	64.9	87.9	99.5	41.6	64.9	87.9	99.5
	④	27.5	44.6	58.9	71.7	38.2	57.5	75.3	86.8	38.2	57.5	75.3	86.8
	⑤	60.0	78.5	86.1	85.4	64.4	86.8	91.1	97.7	64.4	86.8	91.1	97.7
Average pairwise communication of workers	①	201	201	200	200	206	199	200	200	200	200	200	200
	②	108	105	102	100	116	111	109	111	119	106	101	101
	③	196	169	166	180	146	143	127	122	157	114	150	154
	④	199	173	162	194	153	148	139	117	160	120	175	167
	⑤	203	195	181	197	189	177	180	174	201	209	193	173

- On the average synergy value of workers. When the occupancy rate of groups with leaders is 0%, three group-oriented task allocation approaches show slight fluctuation; the potential reason is that the selected workers are more disperse in the social network when the groups have no leaders. When the occupancy rate of groups with leaders are greater than 0%, this type of values will increase as

the skill number of tasks increases; therefore, the increasing of skill number of tasks can improve the average synergy value of assigned workers. Moreover, the *Team formation* approach and *Individual* approach can result in increased average synergy value of workers with the increasing of skill number of tasks under all types of occupancy rates of groups with leaders.

- On the average consistency value of workers. When the occupancy rate of groups with leaders is 0%, our two group-oriented task allocation approaches show slight fluctuation; as the occupancy rate of groups with leaders rises greater than 0%, average consistency value of workers selected by the two approaches increases with the number of skills required for the task. *Group_skill local search* approach can result in higher average consistency value of workers. In comparison, the *Individual* approach will result in decreased average consistency value of workers with the increasing of skill number of tasks under all types of occupancy rates of groups with leaders. Therefore, it shows that the group-oriented crowdsourcing approaches can have better performance on the average consistency value of workers when the skill number of tasks increases, especially when the groups have leaders.

- On the average conflict value of workers. Except that the *Group_skill local search* approach only presents slight fluctuation, other approaches all result in increased APL values when the skill number of tasks increases. Therefore, it can conclude that it may be more possible to generate conflicts when the skill number of tasks increases, because the increased skill number of tasks may result in more team members.

- On the average skill coverage rate. This type of values of all five approaches will increase first then decrease with the increasing of skill number of tasks. From the results, when the skill number of tasks is within [10, 20], it is more probable that the assigned workers can complete other tasks with similar skill numbers.

- On the average total wages of workers. This type of values of all five approaches will increase with the increasing of skill number of tasks; the reason is that higher skill number of tasks can result in higher number of team members.

- On the average pairwise communication costs of workers. This type of values of all five approaches will decrease first then increase with the increasing of skill number of tasks. From the results, the average pairwise communication costs of workers will be the minimum when the skill number of tasks is within [10-20].

6.4.5.3 Tests on the Number of Tasks

The purpose of this series of experiments is to evaluate the performance of the six performance metrics with different task quantities in three situations: the occupancy rate of groups with leaders is 0% (all of the groups are without leaders), 50% and 100% (all of the the groups have leaders) and to verify whether the *Group_semi supervised* approach and the *Group_fully supervised* approach are still maintaining advantages compared with other approaches. In this experiment, the number of tasks is increased from 20 to 100 with the interval of 20, the relative importance factor α is 0.75, and the number of skills required for the task is limited to 5-15.

It can be seen from Table 6.3 that under different task numbers, the values of the six performance metrics are basically the same as those presented in Section 6.4.2. The results show that the advantages of our approaches on the six performance metrics can well scale to the task sizes.

6.5 Summary

In previous studies on crowdsourcing, team formation was used to utilize the cooperation among workers to perform complex tasks. However, the team formation approach might not select sufficiently cooperative and reliable workers for executing task because team members are called transiently and might lack cooperation experience; moreover, a team cannot adapt to varying tasks because the team is artificially tailored only for a special task.

To solve the above drawbacks, this chapter turns to a novel group-oriented crowdsourcing paradigm, in which the tasks are allocated to the worker groups that are popular for the organization of workers at many crowdsourcing websites. Moreover, a group

TABLE 6.3: Results of parameter test on number of tasks

Occupancy Rate of Leader		0.0					0.5					1.0				
Number of tasks		20	40	60	80	100	20	40	60	80	100	20	40	60	80	100
Average synergy value of workers	①	1.2	4.3	8.7	14.1	19.9	1.3	4.4	8.8	14.1	20.0	1.2	4.2	8.6	13.9	19.7
	②	1.8	4.2	7.1	10.6	14.5	1.9	4.5	7.6	11.0	14.9	1.9	4.4	7.4	10.8	14.6
	③	5.5	11.3	17.3	23.5	29.8	7.1	14.4	22.0	29.9	37.9	7.2	14.6	22.5	30.6	38.9
	④	5.5	11.2	17.1	23.2	29.4	7.1	14.4	21.9	29.8	37.8	7.2	14.8	22.6	30.7	39.0
	⑤	8.5	17.1	25.8	34.8	43.8	8.2	16.8	25.5	34.6	43.9	8.4	17.0	25.8	35.0	44.5
Average consistency value of workers	①	9.4	18.9	28.4	37.8	47.3	9.6	19.1	28.7	38.4	48.0	9.3	18.5	27.7	37.0	46.3
	②	8.3	16.6	24.9	33.2	41.5	8.5	17.0	25.3	33.7	42.2	8.4	16.7	25.1	33.5	41.9
	③	8.8	17.6	26.3	35.3	43.9	9.9	19.7	29.5	39.3	49.0	9.9	19.9	29.8	39.8	49.7
	④	8.8	17.6	26.3	35.2	43.9	10.0	19.8	29.6	39.5	49.3	9.9	19.9	29.9	39.8	49.8
	⑤	9.3	18.6	28.0	37.4	46.8	8.8	17.6	26.4	35.1	43.8	8.8	17.5	26.4	35.2	44.1
Average conflict value of workers	①	44.5	88.9	133.3	177.6	222.1	44.5	88.9	133.3	177.6	222.1	44.6	89.0	133.5	178.0	222.5
	②	29.8	59.5	89.3	119.2	149.0	29.8	59.5	89.3	119.2	149.0	29.7	59.4	89.1	118.9	148.6
	③	37.0	74.1	111.2	148.1	184.8	37.0	74.1	111.2	148.1	184.8	33.4	66.8	100.2	133.6	167.0
	④	38.0	76.0	113.9	151.8	189.5	38.0	76.0	113.9	151.8	189.5	33.8	67.7	101.6	135.4	169.1
	⑤	38.4	76.8	115.4	153.7	192.1	38.4	76.8	115.4	153.7	192.1	36.3	73.0	109.9	146.8	183.4
Average skill coverage rate	①	0.54	0.51	0.50	0.51	0.51	0.57	0.58	0.56	0.55	0.55	0.54	0.52	0.53	0.52	0.53
	②	0.67	0.63	0.64	0.66	0.65	0.71	0.73	0.71	0.69	0.69	0.69	0.69	0.69	0.69	0.68
	③	0.94	0.89	0.91	0.91	0.91	0.90	0.93	0.93	0.94	0.94	0.93	0.93	0.94	0.95	0.95
	④	0.94	0.89	0.90	0.90	0.90	0.91	0.93	0.93	0.94	0.94	0.91	0.92	0.93	0.94	0.95
	⑤	0.96	0.95	0.96	0.96	0.96	0.98	0.96	0.95	0.95	0.96	0.96	0.95	0.96	0.95	0.95
Average total wages of workers	①	8.1	16.1	24.1	32.2	40.1	8.0	15.9	23.8	31.7	39.7	8.0	16.0	23.9	32.0	40.0
	②	28.7	57.4	85.4	113.9	141.6	27.7	55.0	82.3	110.4	138.6	27.9	55.7	83.1	110.8	138.4
	③	20.5	40.8	60.8	81.2	101.1	22.8	46.1	69.3	92.7	117.2	21.7	44.0	65.6	87.4	109.2
	④	14.9	30.0	44.9	60.0	74.7	20.8	41.8	62.6	83.6	105.1	19.6	39.5	59.2	79.0	98.7
	⑤	28.0	56.5	84.6	112.7	140.7	30.8	61.8	93.1	124.3	155.8	30.7	60.9	91.5	121.9	152.2
Average pairwise communication of workers	①	41	81	123	165	206	40	80	121	161	201	39	82	123	166	208
	②	21	43	66	87	111	22	42	63	85	105	23	45	67	90	113
	③	44	80	118	157	200	28	56	88	115	146	27	51	72	97	123
	④	43	80	123	160	204	29	58	89	117	149	28	56	77	104	132
	⑤	39	78	119	155	194	38	75	115	160	198	43	82	119	160	199

might often not have the complete skills to implement the allocated task; therefore, the group needs to coordinate with other contextual groups in the social network to obtain the needed skills. Therefore, this chapter investigates the context-aware task allocation problem for group-oriented crowdsourcing. We initially prove the problem is NP-hard; then, we present a heuristic approach that can be realized within a limited time complexity. In the heuristic approach, a function of crowdsourcing value is defined to measure the probability of a group being selected to participate in a task. This chapter theoretically proves that the heuristic approach can ensure that the optimization objective is approached.

Finally, we conduct extensive experiments on a real-world crowdsourcing dataset.

The experiments on a real-world dataset show that the two typical limitations in previous studies can be solved well by our presented approach with good synergy performance, consistency performance, conflict performance, adaptability, and effectiveness of assigned workers.

Chapter 7

Conclusion and Future Work

This thesis proposes a series of approaches in the area of complex task allocation in crowdsourcing. In this chapter, we will summarize the major contributions in this thesis and discuss several potential future research directions.

7.1 Conclusion

Since the term "crowdsourcing" was coined in 2005, crowdsourcing has experienced a lot of development and application these years and also has made significantly influence to our life. Along with its developing process, complex task allocation has become one of the critical issues in crowdsourcing. With considering the fact that workers in crowdsourcing are connected by social networks as well as motivated by analyses of the data from some leading complex-oriented crowdsourcing websites, this thesis investigates crowdsourcing in social network contexts and presents models to address the typical problems in complex task allocation.

First, in Chapter 3 we propose a context-aware reliable crowdsourcing approach aiming to solve two typical problems noted in previous studies of crowdsourcing complex tasks: the requesters undertake a heavy burden when decomposing complex tasks into a set of micro-subtasks, and reliability may not be ensured when there are many

malicious workers in the crowd. In our approach, we define the concept of crowdsourcing value by considering a worker's self-situation and contextual-situation in the social network, and we then use it to measure the priority of assigning a worker with the task. Besides, a worker's reliability is determined not only by the reputation of the worker himself/herself but also by the reputations of the his/her contextual workers, which can effectively address the unreliability of transient or malicious workers. The advantage of the presented approach is testified by the experiments on a real-world dataset.

Second, in Chapter 4, in order to address the limitations of the popular retail-style task allocation that cannot scale to large numbers of concurrent tasks due to the large computational cost of allocation and in which many workers' skills and time may not be fully utilized, we present a batch allocation method which can reduce requesters' total real payment as well as improving each worker' incomes. We first prove that the optimal batch allocation is an NP-hard problem. Then,two heuristic approaches, layered batch allocation and core-based batch allocation, are proposed to solve the problem. The layered approach can achieve good performance but has high computational cost, and the core-based achieves suboptimal performance but can reduce computational cost significantly. The theoretical analyses and experiments including real online testing show that the batch allocation achieves better performances comparing with the previous benchmark approaches.

Third, in Chapter 5, we propose a distributed team formation approach for a batch of tasks to solve the drawbacks of previous team formation studies that each team is tailored only for one task and the centralized manner used in these studies may place a heavy burden on requesters. In our approach, the team is formed for a batch of tasks with similar skill requirements so it saves computational cost, and workers self-organize the team formation though their social networks. We first formalize the optimization objective of the problem and prove that the problem is NP-hard. We then propose two heuristic approaches to solve it. One is to form a fixed team for all tasks in the batch, and the other is to form a basic team that can be dynamically adjusted for each task in the batch. In comparison, the first approach saves computational cost but the second one performs better in reducing the total payments by requesters. The experiments show

that our two presented approaches have better performance compared with previous benchmark approaches.

Fourth, in Chapter 6, we propose a group-oriented task allocation approach to address the limitations of previous individual-oriented or team formation-based crowdsourcing studies which do not consider the situation that workers are often naturally organized into groups through social networks. With our approach, workers within the same group often have rich cooperation experience so it is more probable that workers cooperate better in performing a new task, and groups are organized naturally so it can better fit the crowdsourcing with varying tasks. We first prove that the optimization problem is NP-hard; then, we present a heuristic approach that can be realized within a limited time complexity. The experiments on a real-world dataset show that the two typical limitations in previous studies can be solved well by our presented approach with good synergy performance, consistency performance, conflict performance, adaptability, and effectiveness of assigned workers.

7.2 Future work

In this thesis, we only made online experiments for batch allocation in Chapter 4 but conducted simulation experiments on software domain by using real dataset collected from crowdsourcing platforms for other models. The reason is that if we want to fully test other models by real online experiments on crowdsourcing platforms, it needs the support from the operators of crowdsourcing platforms. Therefore, in the future, we will build a testbed for testing the crowdsourcing models and algorithms. Comparing with the current crowdsourcing platforms, the testbed will consider more about the factor of social network structure, so it would harness the power of social network better. Then, with the testbed, we can further test our approaches presented in this thesis and also conduct real online experiments for other models more conveniently.

In the thesis, we use the worker's and his/her contextual workers' reputations to ensure the reliability. This approach has the advantage that even when a worker comes to the system for the first time, his/her initial reputation can be measured through the

past experiences of his/her contextual workers, so reliability of the transient workers can be achieved. Experiments are conducted and testify the effectiveness of the contextual reputation mechanism works when there are many malicious workers in the crowds on the macroscopic scale. However, if the malicious behaviour is deliberately designed, the current reputation mechanism may not work well. For instance, a malicious worker can pretend to be a normal one at the beginning and get high reputation record, and he/she may attack the task execution results using the way liking Byzantine attacks. Currently, our reputation approach cannot deal with such attacks. Therefore, in the future, we will explore how to design a mechanism to identify such malicious workers in the task allocation process, and design mechanism to ensure the consensus of the execution results.

In our models, the workers are supposed to be cooperative with each other, which is draw from the conclusion that workers connected by social network are often coadjutant [33, 63]. However, as many tasks are quite complex, we find that the budget of a task often be thousands dollars at the crowdsourcing platform. Therefore, in reality, some workers may not be coadjutant anymore because of tempting by the huge monetary reward. In other words, workers may play games with each other when they apply and perform tasks to enhance their utilities. Therefore, in the future, we will consider the factor of non-cooperation between workers for the complex task allocation.

This thesis assumes that the crowdsourcing environments are fixed during task allocation and execution. In reality, the environments may be dynamic because workers may depart or join the groups dynamically. Moreover, the social network structure between workers may be evolved in the allocation and execution process. Therefore, some of our current algorithms might not be efficient to such dynamics. The adaptive mechanism and self-organization mechanism will be introduced for addressing dynamic environments in the future.

Bibliography

[1] Jeff Howe. The rise of crowdsourcing. *Wired Magazine*, 14(6):1–4, 2006.

[2] Long Tran-Thanh, Trung Dong Huynh, Avi Rosenfeld, Sarvapali D Ramchurn, and Nicholas R Jennings. Budgetfix: budget limited crowdsourcing for interdependent task allocation with quality guarantees. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014)*, pages 477–484, 2014.

[3] Thomas Kohler and Marco Nickel. Crowdsourcing business models that last. *Journal of Business Strategy*, 38(2):25–32, 2017.

[4] Arkaitz Zubiaga, Maria Liakata, Rob Procter, Kalina Bontcheva, and Peter Tolmie. Crowdsourcing the annotation of rumourous conversations in social media. In *Proceedings of the 24th International Conference on World Wide Web (WWW 2015)*, pages 347–353, 2015.

[5] Qunwei Li, Aditya Vempaty, Lav R Varshney, and Pramod K Varshney. Multiobject classification via crowdsourcing with a reject option. *IEEE Transactions on Signal Processing*, 65(4):1068–1081, 2017.

[6] Xin Peng, Muhammad Ali Babar, and Christof Ebert. Collaborative software development platforms for crowdsourcing. *IEEE Software*, 31(2):30–36, 2014.

[7] Souad Djelassi and Isabelle Decoopman. Customers' participation in product development through crowdsourcing: issues and implications. *Industrial Marketing Management*, 42(5):683–692, 2013.

[8] Tobias Hoßfeld, Michael Seufert, Matthias Hirth, Thomas Zinner, Phuoc Tran-Gia, and Raimund Schatz. Quantification of YouTube QoE via crowdsourcing. In *Proceedings of 13th IEEE International Symposium on Multimedia (ISM 2011)*, pages 494–499, 2011.

[9] Omar Alonso and Stefano Mizzaro. Using crowdsourcing for TREC relevance assessment. *Information Processing & Management*, 48(6):1053–1066, 2012.

[10] Omar Alonso and Ricardo Baeza-Yates. Design and implementation of relevance assessments using crowdsourcing. In Paul Clough, Colum Foley, Cathal Gurrin, Gareth J. F. Jones, Wessel Kraaij, Hyowon Lee, and Vanessa Mudoch, editors, *Advances in Information Retrieval*, 2011.

[11] Chen Chen, Paweł W Woźniak, Andrzej Romanowski, Mohammad Obaid, Tomasz Jaworski, Jacek Kucharski, Krzysztof Grudzień, Shengdong Zhao, and Morten Fjeld. Using crowdsourcing for scientific analysis of industrial tomographic images. *ACM Transactions on Intelligent Systems and Technology*, 7(4): 52, 2016.

[12] Michael Buhrmester, Tracy Kwang, and Samuel D Gosling. Amazon's Mechanical Turk: A new source of inexpensive, yet high-quality, data? *Perspectives on Psychological Science*, 6(1):3–5, 2011.

[13] Ryo Suzuki, Niloufar Salehi, Michelle S Lam, Juan C Marroquin, and Michael S Bernstein. Atelier: Repurposing expert crowdsourcing tasks as micro-internships. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI 2016)*, pages 2645–2656, 2016.

[14] Matthias Hirth, Tobias Hoßfeld, and Phuoc Tran-Gia. Anatomy of a crowdsourcing platform-using the example of microworkers.com. In *Proceedings of Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS 2011)*, pages 322–329, 2011.

[15] Aniket Kittur, Boris Smus, Susheel Khamkar, and Robert E Kraut. Crowdforge: crowdsourcing complex work. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST 2011)*, pages 43–52, 2011.

[16] Shayan Doroudi, Ece Kamar, Emma Brunskill, and Eric Horvitz. Toward a learning science for complex crowdsourcing tasks. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI 2016)*, pages 2623–2634, 2016.

[17] Yudian Zheng, Jiannan Wang, Guoliang Li, Reynold Cheng, and Jianhua Feng. Qasca: A quality-aware task assignment system for crowdsourcing applications. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD 2015)*, pages 1031–1046. ACM, 2015.

[18] Ioannis Boutsis and Vana Kalogeraki. On task assignment for real-time reliable crowdsourcing. In *Proceedings of the 34th IEEE International Conference on Distributed Computing Systems (ICDCS 2014)*, pages 1–10, 2014.

[19] Huan Jiang and Shigeo Matsubara. Efficient task decomposition in crowdsourcing. In *Proceedings of 17th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA 2014)*, pages 65–73, 2014.

[20] Anand Kulkarni, Matthew Can, and Björn Hartmann. Collaboratively crowdsourcing workflows with turkomatic. In *Proceedings of the 15th ACM Conference on Computer Supported Cooperative Work (CSCW 2012)*, pages 1003–1012, 2012.

[21] Markus Rokicki, Sergej Zerr, and Stefan Siersdorfer. Groupsourcing: Team competition designs for crowdsourcing. In *Proceedings of the 24th International Conference on World Wide Web (WWW 2015)*, pages 906–915, 2015.

[22] Chao Gao and Jiming Liu. Network-based modeling for characterizing human collective behaviors during extreme events. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(1):171–183, 2017.

[23] Enrico Franchi, Agostino Poggi, and Michele Tomaiuolo. Social media for online collaboration in firms and organizations. *International Journal of Information System Modeling and Design*, 7(1):18–31, 2016.

[24] Julia Heidemann, Mathias Klier, and Florian Probst. Online social networks: A survey of a global phenomenon. *Computer Networks*, 56(18):3866–3878, 2012.

[25] Jon Chamberlain. Groupsourcing: Distributed problem solving using social networks. In *Proceedings of Second AAAI Conference on Human Computation and Crowdsourcing (HCOMP 2014)*, pages 22–29, 2014.

[26] Ming Yin, Mary L Gray, Siddharth Suri, and Jennifer Wortman Vaughan. The communication network within the crowd. In *Proceedings of the 25th International Conference on World Wide Web (WWW 2016)*, pages 1293–1303, 2016.

[27] Mary L Gray, Siddharth Suri, Syed Shoaib Ali, and Deepti Kulkarni. The crowd is a collaborative network. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW 2016)*, pages 134–147, 2016.

[28] Ju Ren, Yaoxue Zhang, Kuan Zhang, and Xuemin Sherman Shen. Sacrm: Social aware crowdsourcing with reputation management in mobile sensing. *Computer Communications*, 65:55–65, 2015.

[29] Ju Ren, Yaoxue Zhang, Kuan Zhang, and Xuemin Shen. Exploiting mobile crowdsourcing for pervasive cloud services: challenges and solutions. *IEEE Communications Magazine*, 53(3):98–105, 2015.

[30] Huiji Gao, Geoffrey Barbier, and Rebecca Goolsby. Harnessing the crowdsourcing power of social media for disaster relief. *IEEE Intelligent Systems*, 26(3): 10–14, 2011.

[31] Soo Ling Lim, Daniele Quercia, and Anthony Finkelstein. Stakesource: harnessing the power of crowdsourcing and social networks in stakeholder analysis. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE 2010)*, volume 2, pages 239–242, 2010.

[32] Tobias Hoßfeld, Matthias Hirth, Pavel Korshunov, Philippe Hanhart, Bruno Gardlo, Christian Keimel, and Christian Timmerer. Survey of web-based crowdsourcing frameworks for subjective quality assessment. In *Proceedings of the 16th IEEE International Workshop on Multimedia Signal Processing (MMSP 2014)*, pages 1–6, 2014.

[33] Can Wang, Longbing Cao, and Chi-Hung Chi. Formalization and verification of group behavior interactions. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(8):1109–1124, 2015.

[34] David R Karger, Sewoong Oh, and Devavrat Shah. Budget-optimal task allocation for reliable crowdsourcing systems. *Operations Research*, 62(1):1–24, 2014.

[35] Matteo Venanzi, Alex Rogers, and Nicholas R Jennings. Trust-based fusion of untrustworthy information in crowdsourcing applications. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent systems (AAMAS 2013)*, pages 829–836, 2013.

[36] Sepehr Assadi, Justin Hsu, and Shahin Jabbari. Online assignment of heterogeneous tasks in crowdsourcing markets. In *Proceedings of the Third AAAI Conference on Human Computation and Crowdsourcing (HCOMP 2015)*, pages 12–21, 2015.

[37] Jonghyuk Song, Sangho Lee, and Jong Kim. Crowdtarget: Target-based detection of crowdturfing in online social networks. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS 2015)*, pages 793–804, 2015.

[38] Luca De Alfaro, Vassilis Polychronopoulos, and Michael Shavlovsky. Reliable aggregation of boolean crowdsourced tasks. In *Proceedings of the Third AAAI Conference on Human Computation and Crowdsourcing (HCOMP 2015)*, pages 42–51, 2015.

[39] Tianyi Wang, Gang Wang, Xing Li, Haitao Zheng, and Ben Y Zhao. Characterizing and detecting malicious crowdsourcing. In *Proceedings of the 2013*

ACM SIGCOMM Conference on Data Communications (SIGCOMM 2013), volume 43, pages 537–538, 2013.

[40] Yu Zhang and Mihaela van der Schaar. Reputation-based incentive protocols in crowdsourcing applications. In *Proceedings of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM 2012)*, pages 2140–2148, 2012.

[41] Alessandro Bozzon, Marco Brambilla, Stefano Ceri, Matteo Silvestri, and Giuliano Vesci. Choosing the right crowd: expert finding in social networks. In *Proceedings of the 16th International Conference on Extending Database Technology (EDBT 2013)*, pages 637–648, 2013.

[42] Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, Antonino Nocera, and Domenico Ursino. Driving global team formation in social networks to obtain diversity. In Sven Casteleyn, Gustavo Rossi, and Marco Winckler, editors, *Web Engineering*, pages 410–419. Springer International Publishing, Cham, 2014.

[43] Qing Liu, Tie Luo, Ruiming Tang, and Stéphane Bressan. An efficient and truthful pricing mechanism for team formation in crowdsourcing markets. In *Proceedings of the 2015 IEEE International Conference on Communications (ICC 2015)*, pages 567–572, 2015.

[44] Shibo He, Dong-Hoon Shin, Junshan Zhang, and Jiming Chen. Toward optimal allocation of location dependent tasks in crowdsensing. In *Proceedings of the 33rd Annual IEEE International Conference on Computer Communications (INFOCOM 2014)*, pages 745–753, 2014.

[45] Nuno Luz, Nuno Silva, and Paulo Novais. A survey of task-oriented crowdsourcing. *Artificial Intelligence Review*, 44(2):187–213, 2015.

[46] Djellel Eddine Difallah, Michele Catasta, Gianluca Demartini, Panagiotis G Ipeirotis, and Philippe Cudré-Mauroux. The dynamics of micro-task crowdsourcing: The case of Amazon Mturk. In *Proceedings of the 24th International Conference on World Wide Web (WWW 2015)*, pages 238–247, 2015.

[47] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, pages 254–263, 2008.

[48] Long Tran-Thanh, Trung Dong Huynh, Avi Rosenfeld, Sarvapali D Ramchurn, and Nicholas R Jennings. Crowdsourcing complex workflows under budget constraints. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI 2015)*, pages 1298–1304, 2015.

[49] Wanyuan Wang, Jiuchuan Jiang, Bo An, Yichuan Jiang, and Bing Chen. Toward efficient team formation for crowdsourcing in noncooperative social networks. *IEEE Transactions on Cybernetics*, 47(12):4208–4222, 2017.

[50] Jan van Dalen, Johan Koerts, and Roy Thurik. The measurement of labour productivity in wholesaling. *International Journal of Research in Marketing*, 7: 21–34, Aug 1990.

[51] Jiuchuan Jiang, Bo An, Yichuan Jiang, and Donghui Lin. Context-aware reliable crowdsourcing in social networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, in press. doi: 10.1109/TSMC.2017.2777447.

[52] Ioanna Lykourentzou, Shannon Wang, Robert E Kraut, and Steven P Dow. Team dating: A self-organized team formation strategy for collaborative crowdsourcing. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI 2016)*, pages 1243–1249, 2016.

[53] Ming Li and Pan Li. Crowdsourcing in cyber-physical systems: Stochastic optimization with strong stability. *IEEE Transactions on Emerging Topics in Computing*, 1(2):218–231, 2013.

[54] Vaskar Raychoudhury, Shikhar Shrivastav, Sandeep Singh Sandha, and Jiannong Cao. Crowd-pan-360: Crowdsourcing based context-aware panoramic map generation for smartphone users. *IEEE Transactions on Parallel and Distributed Systems*, 26(8):2208–2219, 2015.

[55] Daniel P Forbes, Patricia S Borchert, Mary E Zellmer-Bruhn, and Harry J Sapienza. Entrepreneurial team formation: An exploration of new member addition. *Entrepreneurship Theory and Practice*, 30(2):225–248, 2006.

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

[57] Mehdi Kargar, Aijun An, and Morteza Zihayat. Efficient bi-objective team formation in social networks. In *Proceedings of the 2012 Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2012)*, pages 483–498. Springer, 2012.

[58] Kirill Osipov and Gita Sukthankar. Amalgacloud: Social network adaptation for human and computational agent team formation. *Human Journal*, 1(2):61–73, 2012.

[59] Chien-Ju Ho and Jennifer Wortman Vaughan. Online task assignment in crowdsourcing markets. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI 2012)*, pages 45–51, 2012.

[60] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006)*, pages 44–54, 2006.

[61] Mark EJ Newman and Juyong Park. Why social networks are different from other types of networks. *Physical Review E*, 68(3):036122, 2003.

[62] Massimo Bergami and Richard P Bagozzi. Self-categorization, affective commitment and group self-esteem as distinct aspects of social identity in the organization. *British Journal of Social Psychology*, 39(4):555–577, 2000.

[63] Yifeng Zeng, Xuefeng Chen, Yew-Soon Ong, Jing Tang, and Yanping Xiang. Structured memetic automation for online human-like social behavior learning. *IEEE Transactions on Evolutionary Computation*, 21(1):102–115, 2017.

[64] Yichuan Jiang, Yifeng Zhou, and Yunpeng Li. Reliable task allocation with load balancing in multiplex networks. *ACM Transactions on Autonomous and Adaptive Systems*, 10(1):3, 2015.

[65] Robert R Morris, Mira Dontcheva, and Elizabeth M Gerber. Priming for better performance in microtask crowdsourcing environments. *IEEE Internet Computing*, 16(5):13–19, 2012.

[66] Cristina Sarasua, Elena Simperl, and Natalya F Noy. Crowdmap: Crowdsourcing ontology alignment with microtasks. In *Proceedings of the 11th International Semantic Web Conference (ISWC 2012)*, pages 525–541, 2012.

[67] Habibur Rahman, Senjuti Basu Roy, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. Task assignment optimization in collaborative crowdsourcing. In *Proceedings of the 15th IEEE International Conference on Data Mining (ICDM 2015)*, pages 949–954, 2015.

[68] Rui Yan, Mingkun Gao, Ellie Pavlick, and Chris Callison-Burch. 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 (ACL 2014)*, volume 1, pages 1134–1144, 2014.

[69] Edmund H Durfee and Victor R Lesser. Chapter 10 - negotiating task decomposition and allocation using partial global planning. In Les Gasser and Michael N. Huhns, editors, *Distributed Artificial Intelligence*, pages 229 – 243. Morgan Kaufmann, San Francisco (CA), 1989.

[70] Yichuan Jiang, Jing Hu, and Donghui Lin. Decision making of networked multiagent systems for interaction structures. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 41(6):1107–1121, 2011.

[71] Alan Garvey, Marty Humphrey, and Victor R Lesser. Task interdependencies in design-to-time real-time scheduling. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI 1993)*, pages 580–585, 1993.

[72] Steven D Eppinger, Daniel E Whitney, Robert P Smith, and David A Gebala. A model-based method for organizing tasks in product development. *Research in Engineering Design*, 6(1):1–13, 1994.

[73] Robbie T Nakatsu, Elissa B Grossman, and Charalambos L Iacovou. A taxonomy of crowdsourcing based on task complexity. *Journal of Information Science*, 40 (6):823–834, 2014.

[74] Jiuchuan Jiang, Bo An, Yichuan Jiang, Donghui Lin, Zhan Bu, Jie Cao, and Zhifeng Hao. Understanding crowdsourcing systems from a multiagent perspective and approach. *ACM Transactions on Autonomous and Adaptive Systems*, 13 (2):8, 2018.

[75] Michael S Bernstein, Greg Little, Robert C Miller, Björn Hartmann, Mark S Ackerman, David R Karger, David Crowell, and Katrina Panovich. Soylent: a word processor with a crowd inside. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology (UIST 2010)*, pages 313–322, 2010.

[76] Peng Dai, Christopher H Lin, Daniel S Weld, et al. Pomdp-based control of workflows for crowdsourcing. *Artificial Intelligence*, 202:52–85, 2013.

[77] Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. A survey of crowdsourcing systems. In *Proceedings of the 3rd IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT 2011) and the 3rd IEEE Inernational Conference on Social Computing (SocialCom 2011)*, pages 766–773, 2011.

[78] Long Tran-Thanh, Sebastian Stein, Alex Rogers, and Nicholas R Jennings. Efficient crowdsourcing of unknown experts using bounded multi-armed bandits. *Artificial Intelligence*, 214:89–111, 2014.

[79] Aniket Kittur, Ed H Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2008)*, pages 453–456, 2008.

[80] Aris Anagnostopoulos, Luca Becchetti, Adriano Fazzone, Ida Mele, and Matteo Riondato. The importance of being expert: Efficient max-finding in crowdsourcing. In *Proceedings of the 36th ACM SIGMOD International Conference on Management of Data (SIGMOD 2015)*, pages 983–998, 2015.

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

[82] Roman Khazankin, Harald Psaier, Daniel Schall, and Schahram Dustdar. Qos-based task scheduling in crowdsourcing environments. In *International Conference on Service-Oriented Computing*, pages 297–311. Springer, 2011.

[83] Chien-Ju Ho, Shahin Jabbari, and Jennifer Wortman Vaughan. Adaptive task assignment for crowdsourced classification. In *International Conference on Machine Learning*, pages 534–542, 2013.

[84] Srikanth Jagabathula, Lakshminarayanan Subramanian, and Ashwin Venkataraman. Reputation-based worker filtering in crowdsourcing. In *Advances in Neural Information Processing Systems*, pages 2492–2500, 2014.

[85] Mohammad Allahbakhsh, Aleksandar Ignjatovic, Boualem Benatallah, Elisa Bertino, Norman Foo, et al. Reputation management in crowdsourcing systems. In *Proceedings of the 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2012)*, pages 664–671, 2012.

[86] Han Yu, Zhiqi Shen, Chunyan Miao, and Bo An. A reputation-aware decision-making approach for improving the efficiency of crowdsourcing systems. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, pages 1315–1316, 2013.

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

[88] Yaron Singer and Manas Mittal. Pricing mechanisms for crowdsourcing markets. In *Proceedings of the 22nd International Conference on World Wide Web (WWW 2013)*, pages 1157–1166, 2013.

[89] Yanru Zhang, Yunan Gu, Lanchao Liu, Miao Pan, Zaher Dawy, and Zhu Han. Incentive mechanism in crowdsourcing with moral hazard. In *Proceedings of the 2015 IEEE Wireless Communications and Networking Conference (WCNC 2015)*, pages 2085–2090. IEEE, 2015.

[90] Lilly C Irani and M Silberman. Turkopticon: Interrupting worker invisibility in Amazon Mechanical Turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2013)*, pages 611–620, 2013.

[91] Donna Vakharia and Matthew Lease. Beyond mechanical turk: An analysis of paid crowd work platforms. In *Proceedings of the iConference 2015*. Newport Beach, California, USA, 2015.

[92] Xinglin Zhang, Zheng Yang, Chenshu Wu, Wei Sun, Yunhao Liu, and Kai Xing. Robust trajectory estimation for crowdsourcing-based mobile applications. *IEEE Transactions on Parallel and Distributed Systems*, 25(7):1876–1885, 2014.

[93] Hoda Heidari and Michael Kearns. Depth-workload tradeoffs for workforce organization. In *Proceedings of the First AAAI Conference on Human Computation and Crowdsourcing (HCOMP 2013)*, pages 60–68, 2013.

[94] Mohammad Fathian, Mohamad Saei-Shahi, and Ahmad Makui. A new optimization model for reliable team formation problem considering experts' collaboration network. *IEEE Transactions on Engineering Management*, 64(4):586–593, 2017.

[95] Feng Li, Yongsheng Ding, Mengchu Zhou, Kuangrong Hao, and Lei Chen. An affection-based dynamic leader selection model for formation control in multi-robot systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(7):1217–1228, 2017.

[96] Muthukumaran Chandrasekaran, Prashant Doshi, Yifeng Zeng, and Yingke Chen. Can bounded and self-interested agents be teammates? application to planning in ad hoc teams. *Autonomous Agents and Multi-Agent Systems*, 31(4): 821–860, 2017.

[97] Dayong Ye, Minjie Zhang, and Athanasios V Vasilakos. A survey of self-organization mechanisms in multiagent systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(3):441–461, 2017.

[98] M Bernardine Dias, Thomas K Harris, Brett Browning, Edward Gil Jones, Brenna Argall, Manuela M Veloso, Anthony Stentz, and Alexander I Rudnicky. Dynamically formed human-robot teams performing coordinated tasks. In *Proceedings of the 2006 AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before*, pages 30–38, 2006.

[99] Aris Anagnostopoulos, Luca Becchetti, Carlos Castillo, Aristides Gionis, and Stefano Leonardi. Online team formation in social networks. In *Proceedings of the 21st International Conference on World Wide Web (WWW 2013)*, pages 839–848, 2012.

[100] Ning Nan and Sanjeev Kumar. Joint effect of team structure and software architecture in open source software development. *IEEE Transactions on Engineering Management*, 60(3):592–603, 2013.

[101] Jungpil Hahn, Jae Yun Moon, and Chen Zhang. Emergence of new project teams from open source software developer networks: impact of prior collaboration ties. *Information Systems Research*, 19(3):369–391, 2008.

[102] Shi-Jie Chen and Li Lin. Modeling team member characteristics for the formation of a multifunctional team in concurrent engineering. *IEEE Transactions on Engineering Management*, 51(2):111–124, 2004.

[103] David R Karger, Sewoong Oh, and Devavrat Shah. Iterative learning for reliable crowdsourcing systems. In *Proceedings of the 25th Conference on Neural Information Processing (NIPS 2011)*, pages 1953–1961, 2011.

[104] Thomas D LaToza, W Ben Towne, André Van Der Hoek, and James D Herbsleb. Crowd development. In *Proceedings of Sixth International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2013)*, pages 85–88, 2013.

[105] Christopher H Lin, Mausam Mausam, and Daniel S Weld. Dynamically switching between synergistic workflows for crowdsourcing. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI 2012)*, 2012.

[106] Bruno Gardlo, Sebastian Egger, Michael Seufert, and Raimund Schatz. Crowdsourcing 2.0: Enhancing execution speed and reliability of web-based qoe testing. In *Proceedings of the 2014 IEEE International Conference on Communications (ICC 2014),*, pages 1070–1075, 2014.

[107] Lav R Varshney, Aditya Vempaty, and Pramod K Varshney. Assuring privacy and reliability in crowdsourcing with coding. In *Proceedings of the 2014 Information Theory and Applications Workshop (ITA 2014)*, pages 1–6, 2014.

[108] Alec Burmania, Srinivas Parthasarathy, and Carlos Busso. Increasing the reliability of crowdsourcing evaluations using online quality assessment. *IEEE Transactions on Affective Computing*, 7(4):374–388, 2016.

[109] Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008)*, pages 614–622, 2008.

[110] Omar Bousbiba and Klaus Echtle. A fast byzantine fault-tolerant diagnostic agreement protocol for synchronous distributed systems. In *ARCS 2016; 29th International Conference on Architecture of Computing Systems*, pages 1–11. VDE, 2016.

[111] Miguel Correia, Nuno Ferreira Neves, and Paulo Verissimo. How to tolerate half less one byzantine nodes in practical distributed systems. In *Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems, 2004.*, pages 174–183. IEEE, 2004.

[112] Jun Zou, Bin Ye, Lie Qu, Yan Wang, Mehmet A Orgun, and Lei Li. A proof-of-trust consensus protocol for enhancing accountability in crowdsourcing services. *IEEE Transactions on Services Computing*, 2018.

[113] Andrei Tamilin, Iacopo Carreras, Emmanuel Ssebaggala, Alfonse Opira, and Nicola Conci. Context-aware mobile crowdsourcing. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp 2012)*, pages 717–720, 2012.

[114] Tobias Distler, Christian Cachin, and Rüdiger Kapitza. Resource-efficient byzantine fault tolerance. *IEEE transactions on computers*, 65(9):2807–2819, 2016.

[115] Dennis Linders. From e-government to we-government: Defining a typology for citizen coproduction in the age of social media. *Government Information Quarterly*, 29(4):446–454, 2012.

[116] Rajib Rana, Chun Tung Chou, Nirupama Bulusu, Salil Kanhere, and Wen Hu. Ear-phone: A context-aware noise mapping using smart phones. *Pervasive and Mobile Computing*, 17:1–22, 2015.

[117] Xiping Hu, Xitong Li, Edith Ngai, Victor Leung, and Philippe Kruchten. Multi-dimensional context-aware social network architecture for mobile crowdsensing. *IEEE Communications Magazine*, 52(6):78–87, 2014.

[118] Yan Kong, Minjie Zhang, and Dayong Ye. A belief propagation-based method for task allocation in open and dynamic cloud environments. *Knowledge-Based Systems*, 115:123–132, 2017.

[119] Yichuan Jiang, Yifeng Zhou, and Wanyuan Wang. Task allocation for undependable multiagent systems in social networks. *IEEE Transactions on Parallel and Distributed Systems*, 24(8):1671–1681, 2013.

[120] Omar F Zaidan and Chris Callison-Burch. Crowdsourcing translation: Professional quality from non-professionals. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*, volume 1, pages 1220–1229, 2011.

[121] Eyal Peer, Joachim Vosgerau, and Alessandro Acquisti. Reputation as a sufficient condition for data quality on Amazon Mechanical Turk. *Behavior Rsearch Methods*, 46(4):1023–1031, 2014.

[122] Andrew T Campbell, Shane B Eisenman, Nicholas D Lane, Emiliano Miluzzo, Ronald A Peterson, Hong Lu, Xiao Zheng, Mirco Musolesi, Kristóf Fodor, and Gahng-Seop Ahn. The rise of people-centric sensing. *IEEE Internet Computing*, 12(4), 2008.

[123] Mehdi Kargar and Aijun An. Discovering top-k teams of experts with/without a leader in social networks. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM 2011)*, pages 985–994, 2011.

[124] Satish Penmatsa and Anthony T Chronopoulos. Cost minimization in utility computing systems. *Concurrency and Computation: Practice and Experience*, 26(1): 287–307, 2014.

[125] Pedro C Pinto, Patrick Thiran, and Martin Vetterli. Locating the source of diffusion in large-scale networks. *Physical Review Letters*, 109(6):068702, 2012.

[126] Anirban Dasgupta and Arpita Ghosh. Crowdsourced judgement elicitation with endogenous proficiency. In *Proceedings of the 22nd International Conference on World Wide Web (WWW 2013)*, pages 319–330, 2013.

[127] Praveen K Kopalle, Carl F Mela, and Lawrence Marsh. The dynamic effect of discounting on sales: Empirical analysis and normative pricing implications. *Marketing Science*, 18(3):317–332, 1999.

[128] Tie Luo, Salil S Kanhere, Sajal K Das, and Hwee-Pink Tan. Incentive mechanism design for heterogeneous crowdsourcing using all-pay contests. *IEEE Transactions on Mobile Computing*, 15(9):2234–2246, 2016.

[129] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[130] Jing Wang, Siamak Faridani, and Panagiotis Ipeirotis. Estimating the completion time of crowdsourced tasks using survival analysis models. In *Proceedings of the 2011 WSDM Workshop on Crowdsourcing for Search and Data Mining (CSDM 2011)*, pages 31–34, Feb 2011.

[131] Hanhua Chen, Hai Jin, and Shaoliang Wu. Minimizing inter-server communications by exploiting self-similarity in online social networks. *IEEE Transactions on Parallel and Distributed Systems*, 24(4):1116–1130, 2016.

[132] Dejun Yang, Guoliang Xue, Xi Fang, and Jian Tang. Incentive mechanisms for crowdsensing: crowdsourcing with smartphones. *IEEE/ACM Transactions on Networking*, 24(3):1732–1744, 2016.

[133] Wanyuan Wang and Yichuan Jiang. Multiagent-based allocation of complex tasks in social networks. *IEEE Transactions on Emerging Topics in Computing*, 3(4):571–584, 2015.

[134] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[135] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[136] Mohammad Allahbakhsh, Boualem Benatallah, Aleksandar Ignjatovic, Hamid Reza Motahari-Nezhad, Elisa Bertino, and Schahram Dustdar. Quality control in crowdsourcing systems: issues and directions. *IEEE Internet Computing*, 17(2):76–81, 2013.

[137] Xu Chen, Brian Proulx, Xiaowen Gong, and Junshan Zhang. Exploiting social ties for cooperative d2d communications: A mobile social networking case. *IEEE/ACM Transactions on Networking*, 23(5):1471–1484, 2015.

[138] Yichuan Jiang and Jiuchuan Jiang. Contextual resource negotiation-based task allocation and load balancing in complex software systems. *IEEE Transactions on Parallel and Distributed Systems*, (5):641–653, 2008.

[139] Baruch Awerbuch and R Gallager. A new distributed algorithm to find breadth first search trees. *IEEE Transactions on Information Theory*, 33(3):315–322, 1987.

[140] Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

[141] Jiuchuan Jiang, Peng Shi, Bo An, Jianyong Yu, and Chongjun Wang. Measuring the social influences of scientist groups based on multiple types of collaboration relations. *Information Processing & Management*, 53(1):1–20, 2017.

[142] Hongseok Oh, Myung-Ho Chung, and Giuseppe Labianca. Group social capital and group effectiveness: the role of informal socializing ties. *Academy of Management Journal*, 47(6):860–875, 2004.

[143] Morton Goldman, Joseph W Stockbauer, and Timothy G McAuliffe. Intergroup and intragroup competition and cooperation. *Journal of Experimental Social Psychology*, 13(1):81–88, 1977.

[144] John N Constantino and Richard D Todd. Genetic structure of reciprocal social behavior. *American Journal of Psychiatry*, 157(12):2043–2045, 2000.

[145] Jing Wang, Feng Fu, Te Wu, and Long Wang. Emergence of social cooperation in threshold public goods games with collective risk. *Physical Review E*, 80(1): 016101, 2009.

[146] Somchaya Liemhetcharat and Manuela Veloso. Weighted synergy graphs for effective team formation with heterogeneous ad hoc agents. *Artificial Intelligence*, 208:41–65, 2014.

[147] Somchaya Liemhetcharat and Manuela Veloso. Modeling and learning synergy for team formation with heterogeneous agents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, pages 365–374, 2012.

[148] Norbert L. Kerr and R. Scott Tindale. Group performance and decision making. *Annual Review of Psychology*, 55(1):623–655, 2004.

[149] Mark EJ Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[150] Adriano O Sousa. Consensus formation on a triad scale-free network. *Physica A: Statistical Mechanics and its Applications*, 348:701–710, 2005.

[151] Jiming Liu, Hongjun Qiu, Ning Zhong, and Chao Gao. A dynamic trust network for autonomy-oriented partner finding. *Journal of Intelligent Information Systems*, 37(1):89–118, 2011.