# Batch Allocation for Tasks with Overlapping Skill Requirements in Crowdsourcing

Jiuchuan Jiang, Bo An, Yichuan Jiang, *Senior Member, IEEE*, Peng Shi, Zhan Bu, and Jie Cao

**Abstract—**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 problems: 1) each task is executed independently from scratch, thus the execution of one task seldom 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 paper 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 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.

**Index Terms—**Task allocation, crowdsourcing, batch formation, discounting, skill overlapping

## 1 Introduction

In previous work on crowdsourcing, the task allocation is categorized into two types [1]: for simple tasks that are atomic computation operations, tasks are directly allocated to proper individual workers [2]; for complex tasks involving many computational operations, tasks may either be decomposed into micro-subtasks allocated to individual workers [3], [4] or be directly allocated to a team of workers through team formation [5], [6].

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.*

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 general, the problems with the retail-style task allocation in traditional crowdsourcing can be summarized as follows:

1) *Retail-style task allocation requires each requester to pay in full for his/her task because each task will be allocated and completed independently from scratch.* In fact, it is common that many tasks within the same category at a crowdsourcing website may be similar on required skills. For example, we find that each skill is required by at least 31.36 tasks on average at the Freelancer

- *J. Jiang and B. An are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, Singapore. E-mail: jiangjiuchuan@163.com, boan@ntu.edu.sg.*
- *Y. Jiang and P. Shi are with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China and also with the Co-innovation Center of Shandong Universities for Future Intelligent Computing, Shandong Technology and Business University, Yantai 264005, China. E-mail: {yjiang, pengshi}@seu.edu.cn.*
- *Z. Bu and J. Cao are with the College of Information Engineering, Nanjing University of Finance and Economics, Nanjing 210003, China. E-mail: buzhan@nuaa.edu.cn, caojie690929@163.com.*

*Manuscript received 2 Sept. 2017; revised 15 Dec. 2018; accepted 15 Jan. 2019. Date of publication 21 Jan. 2019; date of current version 8 July 2019. (Corresponding author: Jiuchuan Jiang.) Recommended for acceptance by B. Ravindran. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TPDS.2019.2894146*

website and by at least 13.58 tasks on average at the Upwork website. 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. Therefore, it may be possible to integrate the skill-overlapping tasks and allocate them in batches, which can save allocation cost; then, the batched similar tasks will be performed by the same workers, which can allow the partial execution results of one task to be reused by another similar task of the same workers. This approach can reduce the execution cost of tasks so that the system can discount the real payment that is required from requesters.

2) *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 percent of the workers' total duration at the Upwork website and 80.16 percent 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 randomly counting 7395 tasks 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 undertakes more tasks contemporaneously, the worker can receive higher hourly payment compared with the retail-style task allocation.

To address the above problems, this paper 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 [7]. First, we integrate similar tasks into a batch according to their skill requirements; then, the integrated tasks will be allocated in batch to workers. Batch allocation has the following advantages: 1) batch allocation can save on task allocation time, which allows 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.

If the assigned workers can satisfy all skill requirements of the tasks in the batch, they will perform the tasks by themselves. Otherwise, they will coordinate with other workers in their social network to execute the tasks rather than turning to the requesters, the reason is that the personal characteristics of workers are often invisible to the requesters [23] and the requesters may not have enough expertise to justify whether a worker has qualified professional skills to complete the tasks effectively.

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.

Through theoretical analyses and experiments on a real-world dataset from the Upwork website, we find that our two approaches achieve much better performances in terms of total payment by requesters and average income of workers, while maintaining close successful task completion probabilities and consuming less task allocation time compared with the previous benchmark retail-style allocation approach.

The remainder of this paper is organized as follows. In Section 2, we compare our work with related work; in Section 3, we present the motivation and problem description; in Section 4, we present the approach for layered batch formation and allocation of tasks; in Section 5, we present the approach for core-based batch formation and allocation of tasks; in Section 6, we provide the experimental results; finally, we conclude our paper in Section 7.

## 2 Related Work

We now introduce the related work on retail-style allocation of complex tasks in crowdsourcing, which includes two types: decomposition and monolithic allocations.

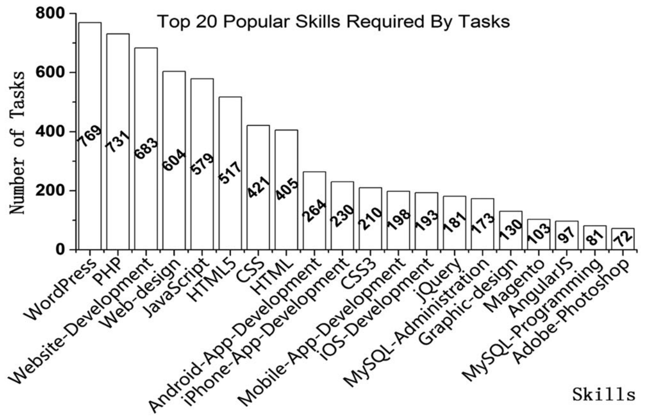
### 2.1 Decomposition Allocation of Complex Tasks

Decomposition allocation of a complex task means that the task will first be decomposed into a flow of simple sub-tasks and then the decomposed sub-tasks will be allocated to individual workers [3]. 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, and when the workers are non-professional and can only complete simple or micro-tasks.

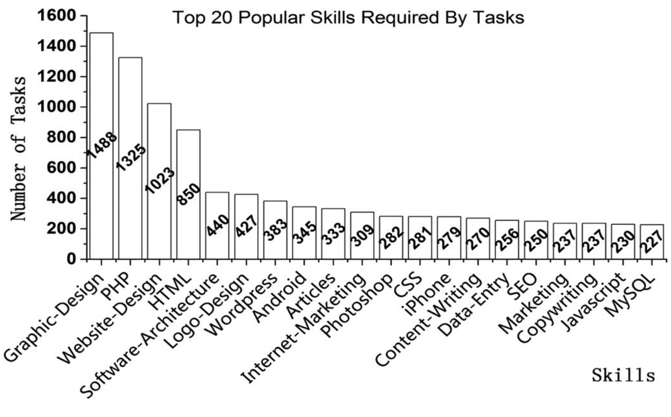
Tran-Thanh et al. [3] proposed a crowdsourcing algorithm to specify the number of micro-tasks to be allocated at each phase of a workflow and dynamically allocate its budget to each micro-task. Then, Tran-Thanh et al. [4] further investigated the problem of *multiple* complex workflows, and they proposed an algorithm to determine the number of inter-dependent micro-tasks and the price to pay for each task within each workflow.

Bernstein et al. [11] 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.

In comparison, this paper provides a contrary approach that integrates some complex tasks with overlapping skill requirements into batches instead of decomposing them, which can reduce the computational cost and real payment by requesters.



(a) Upwork website



(b) Freelancer website

Fig. 1. The numbers of tasks requiring some given skills.

## 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 Worker-Oriented Allocation

Currently, there are some crowdsourcing websites for complex tasks, such as www.upwork.com. These complex tasks are often allocated to professional workers [12], [13] by considering the following two factors: the matching degree between the task's required skills and the worker's skills, and the reputation (or experience) of the worker. Complex tasks are often allocated non-redundantly (e.g., by randomly counting 6271 tasks in Upwork website, we find that 79.3 percent of tasks are allocated non-redundantly). Moreover, on some crowdsourcing websites, such as Crowdworks, the requesters may interview the candidate workers using instant messaging software tools to determine whether the workers can complete the tasks.

Many workers are connected by social networks [24]; therefore, Bozzon et al. [14] presented an approach to find the most knowledgeable people in social networks to address the task. Moreover, if a worker cannot complete the allocated complex task by himself, he needs to forward the complex task to another worker. Heidari and Kearns [15] performed a study on designing efficient forwarding structures from a worker to another worker.

In comparison, our study in this paper not only allocates an entire complex task to a worker but also integrates some similar complex tasks, which can reduce requesters' real payment and make use of the time of workers better than previous studies.

### 2.2.2 Team Formation-Based Allocation

Team formation is a new method for crowdsourcing complex tasks, where individuals with different skills form a team to complete tasks collaboratively [6], [16]. Liu et al. [17] presented an efficient method for team formation in crowdsourcing, which are implemented by involving economic incentives and guaranteeing profitability for requesters and workers. They designed some incentive mechanisms for forming a team that can complete the task and determine each individual worker's payment.

Currently, there are some self-organization approaches for the team formation. Rokicki et al. [5] explored a self-organization strategy for team formation in which the workers make decisions for forming a team; each worker initially forms a one-man team and becomes its administrator; workers can decide by themselves which team they want to join. Lykourentzou et al. [18] presented a self-organized team formation strategy in which the hired workers can select the teammates by themselves.

In summary, each team is tailored artificially for a special task, which cannot fit for a batch of tasks. In comparison, this paper focuses on integrating tasks instead of grouping workers.

## 3 Motivation and Problem Description

## 3.1 Motivation

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

- *Overlapping skill requirements of different tasks*. We consider the typical category of complex tasks at the Upwork website: web-mobile development. We select the 20 skills that are most often required by tasks in the category, which are shown on the $x$-axis in Fig. 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 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 Fig. 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*.
- *Similarities of required skills among tasks*. Let there be $n$ tasks. We use $S_{t_x}$ to denote the set of skills required by

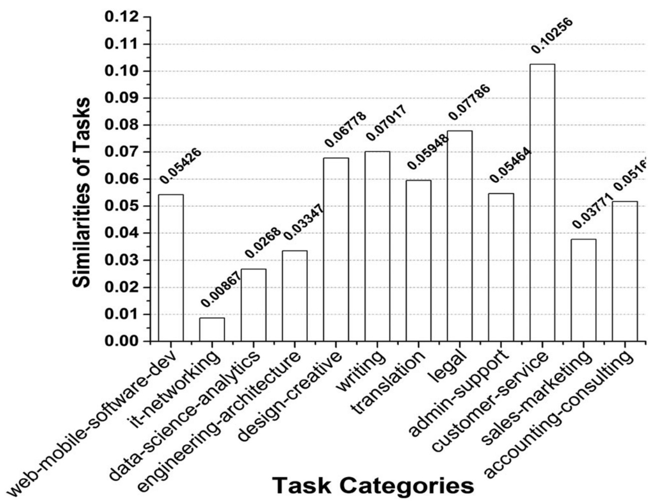


Fig. 2. The similarities of required skills of tasks in different categories at the Upwork website.

task $t_x$. Then, the similarities among the $n$ tasks can be measured as the average pairwise Jaccard similarity:

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

Fig. 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*.

- *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 Fig. 3. In Fig. 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 percent of the workers' total duration at the Upwork website and 80.16 percent at the Freelancer website. *Therefore, most workers undertake only a very small number of tasks contemporaneously*.
- *The deadlines of tasks*. Many crowdsourcing tasks have long deadlines; for example, after counting 7395 tasks at the Upwork website, we find the average duration is 97.76 days; and after counting 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 websites, where workers may have uncertain skill levels and be unreliable.

*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 [2]. In comparison, the novelty of this paper is that we explore the batching of *complex* tasks.

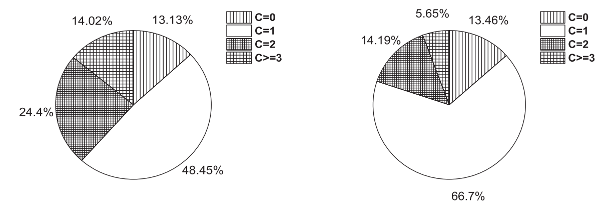


Fig. 3. The numbers of tasks undertaken by workers during a given period.

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 website 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 Fig. 6 in Section 6.5.2.

## 3.2 Problem Description

### 3.2.1 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 [19]. 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 $w_i$, $|B_{w_i}|$. Now, we can define the discounting function as follows:

$$Pay(w_i, t) = \psi(|B_{w_i}|) \cdot b_t, \tag{2}$$

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

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

$$Pay(w_i, B) = \sum_{x=1,\ldots,|B|} (\psi(x) \cdot b_{t_x}). \tag{3}$$

### 3.2.2 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).

The optimization objective is to form proper batches and to allocate the batches to the workers with the maximum crowdsourcing values (which measure workers' probabilities of performing the tasks successfully and will be defined afterwards) to minimize the real payments by requesters under the following two constraints: the discounted payment for each task is not less than the reservation wage of the assigned worker; and the completion time of each task cannot exceed the deadline of that task.

Let $T$ be a set of tasks. $\Pi$ 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 $V(w_i, B_x)$ 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 expressed later in Section 4.2. Our objective is to find a batching scheme that can minimize the total real payment by all requesters of tasks in $T$:

$$\Pi_* = \arg\min_{\Pi} \left( \sum_{B_x \in \prod} Pay(w_j, B_x) \right), \tag{4}$$

where

$$w_j = \arg\max_{w_i \in W} (V(w_j, B_x)) \tag{5}$$

$$subject\,to:$$

$$\forall B_x \in \Pi \wedge \forall t \in B_x \wedge time(w_j, t) \le d_t \tag{6}$$

$$\forall B_x \in \Pi \wedge \forall t \in B_x \wedge Pay(w_j, t) \ge \gamma_{w_j}, \tag{7}$$

where $time(w_j, t)$ denotes the completion time of task $t$ by worker $w_j$, $d_t$ denotes the deadline of $t$, and $\gamma_{w_j}$ denotes the reservation wage of $w_j$. Equations (5), (6), and (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 $w_j$.

### 3.2.3 Property Analyses

**Theorem 1.** *The batch allocation problem with the optimization objective in Equations (4), (5), (6), (7) is NP-hard.*

**Proof.** 3-dimensional matching(3DM) is a classical NP-hard problem [30]. 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 $|M| \ge k$.

In an instance of our problem, there is a set of tasks, $T = \{t_1, t_2, \ldots, t_n\}$. We can obtain a collection $C = \{C_1, C_2, \ldots, C_x\}$ of 3-element subsets of $T$ and a collection $A = \{A_1, A_2, \ldots, A_n\}$ 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 $|M| \ge k$, if and only if our problem has an exact cover $E$ for $T$ with the sum of weights $\sum_{i \in E} w_i \le J$, 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 $w_a \cdot |A'| + w_c \cdot |C'| = w_a \cdot (|T| - 3|C'|) + w_c \cdot |C'| = 2n/3 - |C'| \le J$. Thus, $|C'| \ge 2n/3 - J$. 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 $|M| = |C'| \ge k$, $k = 2n/3 - J$.
2) *if*. If there exists a 3-dimensional matching $M$ in the 3DM problem, $|M| \ge k$, 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 $2/3 \cdot (|T| - 3|C'|) + 1 \cdot |C'| = 2n/3 - |C'| \le J$.

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) is compatible to another objective of improving the real earnings of individual workers. 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.

**Lemma 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.** $\forall t_x \in B_2 \Rightarrow t_x \in B_1 \vee t_x \in (B_2 - B_1)$. Let $\xi_x$ denote the start time of the execution of task $t_x$. Then, $\forall t_x \in (B_2 - B_1) \wedge t_y \in B_1 \Rightarrow \xi_x \geq \xi_y$. According to Equation (3), $Pay(w_i, B_2) = \sum_{x=1,\ldots,|B_1|} (\psi(x) \cdot b_{t_x}) + \sum_{x=|B_1|+1,\ldots,|B_2|} (\psi(x) \cdot b_{t_x}) = Pay(w_i, B_1) + \sum_{x=|B_1|+1,\ldots,|B_2|} (\psi(x) \cdot b_{t_x})$. 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 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 $Pay(w_i, B_1 \cup T')$ - $Pay(w_i, B_1) \geq Pay(w_i, B_2 \cup T') - Pay(w_i, B_2)$.*

**Proof.** $Pay(w_i, B_1 \cup T') = \sum_{x=1,\ldots,|B_1|} (\psi(x) \cdot b_{t_x}) + \sum_{x=|B_1|+1,\ldots,|B_1|+|T'|} (\psi(x) \cdot b_{t_x})$ and $Pay(w_i, B_1) = \sum_{x=1,\ldots,|B_1|} (\psi(x) \cdot b_{t_x})$, thus $Pay(w_i, B_1 \cup T') - Pay(w_i, B_1) = \sum_{x=|B_1|+1,\ldots,|B_1|+|T'|} (\psi(x) \cdot b_{t_x})$. $Pay(w_i, B_2 \cup T') = \sum_{x=1,\ldots,|B_2|} (\psi(x) \cdot b_{t_x}) + \sum_{x=|B_2|+1,\ldots,|B_2|+|T'|} (\psi(x) \cdot b_{t_x})$ and $Pay(w_i, B_2) = \sum_{x=1,\ldots,|B_2|} (\psi(x) \cdot b_{t_x})$, thus $Pay(w_i, B_2 \cup T') - Pay(w_i, B_2) = \sum_{x=|B_2|+1,\ldots,|B_2|+|T'|} (\psi(x) \cdot b_{t_x})$. $|B_1| \leq |B_2|$, $\forall i = 1, \ldots, |T'| \Rightarrow b_{t_{|B_1|+i}} = b_{t_{|B_2|+i}}$, and according to the discounting function, we have $\forall i = 1, \ldots, |T'| \Rightarrow \psi(|B_1| + i) b_{t_{|B_1|+i}} \geq \psi(|B_2| + i) b_{t_{|B_2|+i}}$. Therefore, we can obtain Lemma 2. □

According to Lemma 1, we should try to enlarge the batch if the constraints in our allocation objective can be satisfied; however, from Lemma 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 paper 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 during batch formation and allocation.

## 4 Layered Batch Formation and Allocation of Tasks

### 4.1 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, $T = \{t_1, t_2, \ldots, t_n\}$. 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 $S_{t_x} = \{s_{x1}, s_{x2}, \ldots, s_{xm}\}$. 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 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 $t \in B, time(w,t)$, does not exceed the deadline of $t$, $d_t$:*

$$w \in W_c \Leftrightarrow \forall t \in B \wedge \gamma_w \leq Pay(w,t) \wedge time(w,t) \leq d_t. \tag{8}$$

**Algorithm 1** Layered-Batch Formation for Tasks

/* $T = \{t_1, t_2, \ldots, t_n\}$ denotes a set of *n* tasks; *W* is the set of workers in the crowd */

```
1)  $i = 1; c = 0; Layer_i = \{\};$
2)  $LWS_i = \{\};$ //Candidate worker set for batches of layer i
3)  $\forall t_j \in T$: // Form the first layer
4)      $Batch_{i,j} = [t_j];$
5)      $BWSet_{i,j} = \{\};$ // Candidate worker set of the batch
6)      $\forall w \in W$:
7)          If $w$ can satisfy $Batch_{i,j}$:
8)              $BWSet_{i,j} = BWSet_{i,j} \cup \{w\};$
9)      If $BWSet_{i,j}$ is not Empty:
10)         $Layer_i = Layer_i \cup \{Batch_{i,j}\};$
11)         $LWS_i = LWS_i \cup \{BWSet_{i,j}\};$
12) While ($c == 0$) do:
13)     $i = i + 1; Layer_i = \{\}; LWS_i = \{\}; k = 1;$
14)     $\forall Batch_{i-1,j} \in Layer_{i-1}$:
15)         $\forall t_x \in T - \{Batch_{i-1,j}\}$:
16)             $Batch_{i,k} = Batch_{i-1,j} + [t_x];$//Add a task to the batch
17)             $BWSet_{i,k} = \{\};$
18)             $\forall w \in LWS_{i-1,j}$:
19)                 If $w$ can satisfy $Batch_{i,k}$:
20)                     $BWSet_{i,k} = BWSet_{i,k} \cup \{w\};$
21)             If $BWSet$ is not Empty:
22)                 $Layer_i = Layer_i \cup \{Batch_{i,k}\};$
23)                 $LWS_i = LWS_i \cup \{BWSet_{i,k}\};$
24)                 $k = k + 1;$
25)     If $Layer_i$ is Empty: $c = 1;$
26) End.
```

The layered batch formation process is described as Algorithm 1. In the bottom layer, each task forms a batch initially, i.e., $Layer_1 = \{[t_1], [t_2], \ldots, [t_n]\}$. If any worker satisfies the time and wage constraints of one batch (i.e., the constraints in Equations (6) and (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 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 1, '$w$ can satisfy Batch' means that the constraints of $w$'s reservation wage and all tasks' deadlines are satisfied. Although the complexity of Algorithm 1 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 with no candidate workers.

**Theorem 2.** *Let there be two batches $B_1$ and $B_2$ at Layer$_i$. Now it is assumed that $B_1$ and $B_2$ are 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. We have Pay(w,$B_1$) + Pay(w,$B_2$) $\geq$ Pay(w, $B_3$).*

**Proof sketch.** 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 Theorem 2. □

According to Theorem 2, the larger a batch is, the more significantly the real payment by requesters will be discounted. Therefore, Theorem 2 ensures that the layered batch formation can approach our optimization objective of minimizing the real payment by requesters.

**Theorem 3.** *Suppose that when we use Algorithm 1 to perform batch formation for T, there are two layers: Layer$_i$, 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 $B_{i,j}$.$W_{i,j}$ and $W_{i+1,k}$ are respectively the candidate worker sets of $B_{i,j}$ and $B_{i+1,k}$. Then (1) $W_{i+1,k} \subseteq W_{i,j}$; (2) for a worker w, Pay(w, $B_{i,j}$) < Pay(w, $B_{i+1,k}$); (3) if $W_{i,j}$ is empty, the generation of Layer$_{i+1}$ needs not to involve appending one task from batch $B_{i,j}$.*

**Proof sketch.**

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, $W_{i+1,k} \subseteq W_{i,j}$.
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 $Pay(w, B_{i,j}) = Pay(w, B_{i+1,k}[1, \ldots, \mathrm{i}])$. Thus, 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 $W_{i,j}$ is empty, $W_{i+1,k}$ is also empty and the desired conclusion follows. □

Theorem 3 ensures the correctness of Algorithm 1 and guarantees that another aspect of our objective, improving the real earnings of the worker, can be achieved.

## 4.2 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 $S_B = \cup_{\forall t_x \in B} S_{t_x}$, 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:

$$CS(w_i, B) = \frac{\sum_{s_a \in S_B} (b_{ia} \cdot n_a)}{\sum_{s_a \in S_B} n_a}. \tag{9}$$

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 (2). Let the reservation wage of $w_i$ be $\gamma_{w_i}$. Then, the occupancy rate of $w_i$'s reservation wage on batch *B*'s payment is

$$Occ(w_i, B) = \left( \sum_{t_x in B} \left( \gamma_{w_i} / Pay(w_i, t_x) \right) \right) / |\{t_x | t_x \in B\}|. \tag{10}$$

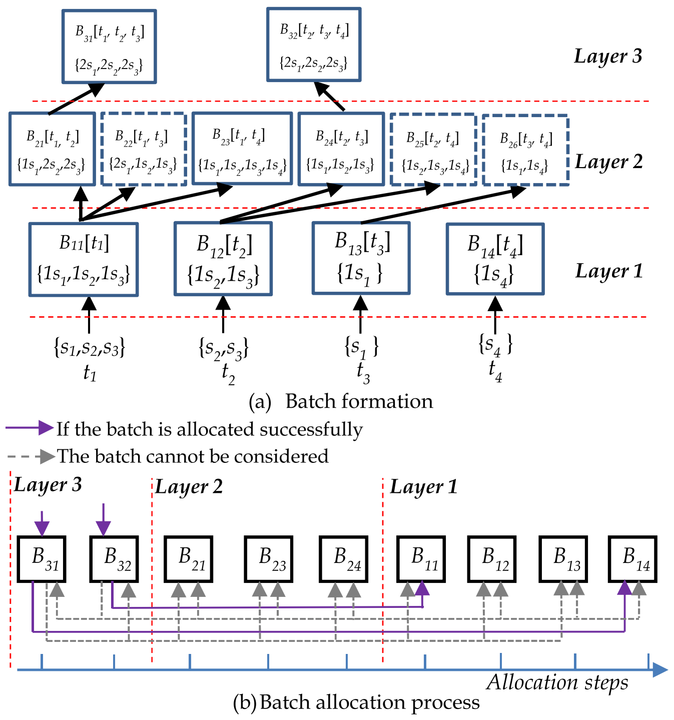
If the assigned worker $w_i$ has a lower $Occ(w_i, B)$, $w_i$ may have more potential to distribute more utility to other workers who assist $w_i$ in executing tasks in *B*.

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 [28]. Let $f(w_i, t_x)$ be the estimated completion time of worker $w_i$ for task $t_x$. The index can be calculated as follows:

$$Est(w_i, B) = \left( \sum_{t_x in B} f(w_i, t_x) \right) / |\{t_x | t_x \in B\}|. \tag{11}$$

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 or 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 [25], [26], [27]. Some recent studies [22], [23] 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



B31[t1, t2, t3] {2s1,2s2,2s3}
B32[t2, t3, t4] {2s1,2s2,2s3}
Layer 3
B21[t1, t2] {1s1,2s2,2s3}
B22[t1, t3] {2s1,1s2,1s3}
B23[t1, t4] {1s1,1s2,1s3,1s4}
B24[t2, t3] {1s1,1s2,1s3}
B25[t2, t4] {1s2,1s3,1s4}
B26[t3, t4] {1s1,1s4}
Layer 2
B11[t1] {1s1,1s2,1s3}
B12[t2] {1s2,1s3}
B13[t3] {1s1}
B14[t4] {1s4}
Layer 1
{s1,s2,s3} t1
{s2,s3} t2
{s1} t3
{s4} t4
(a) Batch formation
If the batch is allocated successfully
The batch cannot be considered
Layer 3
Layer 2
Layer 1
B31 B32 B21 B23 B24 B11 B12 B13 B14
Allocation steps
(b) Batch allocation process

Fig. 4. An example of layered batch formation and task allocation.

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 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*

$$V(w_i, B) = \alpha_1 \left( \frac{\beta_1 \cdot CS(w_i, B) + \beta_2 \cdot R_{w_i}}{\beta_3 \cdot Occ(w_i, B) + \beta_4 \cdot Est(w_i, B)} \right) + \alpha_2 \frac{\sum_{w_j \in (W - \{w_i\})} \left( \frac{\beta_1 \cdot CS(w_j, B) + \beta_2 \cdot R_{w_j}}{\beta_3 \cdot Occ(w_j, B) + \beta_4 \cdot Est(w_j, B)} / d_{ij} \right)}{|W - \{w_i\}|}. \tag{12}$$

*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.* $\alpha_1 + \alpha_2 = 1; \beta_1 + \beta_2 + \beta_3 + \beta_4 = 1$.

Definition 2 shows that the probability of a worker to be assigned a task is determined not only by the worker himself/herself but also by his/her contextual workers in the social network. If a worker has higher values for the four factors, he/she has a higher probability to be assigned the task. However, a worker may also have a higher probability to be assigned the task even he/she has lower values but his/her contextual workers have higher values for the four factors.

## 4.3 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), (5), (6), (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 1 be $\lambda$ 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 2. Since Algorithm 1 ensures that the sizes of batches at a higher layer are larger than those at a lower layer and the sizes of batches within one layer are the same, we greedily allocate batches from upper layers to lower layers. The complexity of Algorithm 2 is $O(nm)$, where $n$ is the number of tasks and $m$ is the number of workers.

**Algorithm 2** Batch Layer-Oriented Task Allocation

```
1)  $i = \lambda; c_1 = 0;$
2)  While $((i > 0)$ and $(c_1 == 0))$ do:
3)    $c_2 = 0;$
4)    While $(c_2 == 0)$ do:
5)        $min\_budget = +\infty; W_c = \{\};$
6)        $\forall B_{ij} \in Layer_i$: // Find minimum-payment batch
7)          If $min\_budget > Pay(w, B_{ij})$:
8)            $min\_budget = Pay(w, B_{ij});$
9)            $Batch = B_{ij}; BWSet = LWS_{i,j};$
10)       $W_{temp} = W \cap BWSet;$
11)       If $W_{temp}$ is not Empty: // Allocating Batch
12)         $w_* = \arg\max_{w \in W_{temp}} (V(w, Batch));$
13)         assign $Batch$ to $w_*$; $W = W - \{w_*\};$
14)         For $k = 1$ to $i$:
15)           $\forall B_{kx} \in Layer_k$:
16)             If $B_{kx} \cap Batch \,!= \phi$:
17)               $Layer_k = Layer_k - \{B_{kx}\};$
18)       Else:
19)         $Layer_i = Layer_i - \{Batch\};$
20)         $LWS_i = LWS_i - \{BWSet\};$
21)       If $Layer_i$ is Empty: $c_2 = 1$
22)   $i = i - 1;$
23)   If $i > 0$:
24)       If $Layer_i$ or $W$ is Empty: $c_1 = 1;$
25) End.
```

**Example 1.** Fig. 4 shows an example, where there are four tasks {t$_1$, t$_2$, t$_3$, t$_4$}. Let the deadlines of the four tasks be $d_1 \le d_2 \le d_3 \le d_4$. First, we can carry out batch formation according to Algorithm 1, which is shown as (a). 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 2, which is shown as (b). 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.

## 5 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 $S_{t_x} = \{s_{x1}, s_{x2}, \ldots, s_{xm}\}$ and $S_{t_y} = \{s_{y1}, s_{y2}, \ldots, s_{yn}\}$, respectively. Then, the distance between tasks $t_x$ and $t_y$ is

$$\delta_{xy} = 1 - \frac{|S_{t_x} \cap S_{t_y}|}{|S_{t_x} \cup S_{t_y}|}. \tag{13}$$

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 $T$:

$$t_c = \arg\min_{t_x \in T} \left( \sum_{t_y \in (T - \{t_x\})} \delta_{xy} \right). \tag{14}$$

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), (5), (6) (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 $\delta_{ci}$ 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:

$$w_* = \arg\max_{w_i \in W_c} (V(w_i, batch(t_c))). \tag{15}$$

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 easily utilized. The core-based approach is shown as Algorithm 3.

The worst-case complexity of Algorithm 3 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 $O(2^n) + \mathrm{O}(nm)$.

**Algorithm 3** Core-Based Batch Formation and Allocation. /* $T = \{t_1, t_2, \ldots, t_n\}$ denotes a set of $n$ tasks; $W$ is the set of workers in the crowd */

```
1)  $c_1 = 0$;
2)  While ($c_1 == 0$) do:
3)    $t_c = \arg\min_{t_x \in T}(\sum_{t_y \in (T-\{t_x\})} \delta_{xy})$; /*Select core task $t_c$*/
4)    $W_{temp} = W; c_2 = 1; W_t = \{\}; Batch = \{\}$;
5)    $\forall w \in W_{temp}$:
6)      If ($time(w,t_c) \le d_{t_c}$) and ($\gamma_w \le Pay(w,t_c)$):
7)        $\{time_w = time(w,t_c); W_t = W_t \cup \{w\};\}$
8)    If $W_t \ne \phi$:
9)      $\{Batch = \{t_c\}; c_2 = 0; W_{temp} = W_t; T_{temp} = T;\}$
10)   While ($c_2 == 0$) do:
11)     $t_b = \arg\min_{t_y \in T_{temp} \wedge \delta_{cy} \ne 0}(\delta_{cy})$;
12)     If $t_b$ can be found:
13)       $W_t = \{\}$;
14)       $\forall w \in W_{temp}$:
15)         If ($time(w,t_b) + time_w \le d_{t_b}$)
              $\wedge(\gamma_w \le Pay(w,t_b))$:
16)           $W_t = W_t \cup \{w\}$;
17)           $time_w = time_w + time(w,t_b)$;
18)       If $W_t \ne \phi$:
19)         $\{W_{temp} = W_t; Batch = Batch \cup \{t_b\};\}$
20)       else: $T_{temp} = T_{temp} - \{t_b\}$;
21)     Else: $c_2 = 1$;
22)   If batch $\ne \phi$:
23)     $w_* = \arg\max_{\forall w_i \in W_{temp}}(V(w_i, Batch))$;
24)     assign Batch to $w_*$; $W = W - \{w_*\}$;$T = T - batch$;
25)   If $T == \phi$ or $W == \phi$: $c_1 = 1$;
26) End.
```

**Theorem 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) $Pay(w, B_i) \le Pay(w, B_j)$.*

**Proof sketch.** (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 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, $Pay(w, B_i) \le Pay(w, B_j)$. □

Theorem 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).

## 6 Experiments

We now conduct experiments for our approaches on a real dataset and realistic settings by comparing 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.

### 6.1 Metric for Task Execution

In general, the successful execution of a task is affected by the following four factors: 1) the possibility 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:

$$p_{i,t} = s_{i,t} \cdot (R_{w_i}/R_{\max}) \cdot (time(w_i,t) \le d_t) \cdot (Pay(w_i,t) \ge \gamma_{w_i}), \tag{16}$$

where

$$s_{i,t} = \prod_{k=1}^{m} e^{-D_k \cdot \tau}. \tag{17}$$

If the completion time of $t$, $time(w_i, t)$, is later than $d_t$ or the assigned worker's reservation wage $\gamma_{w_i}$ 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. $\tau$ represents the decay factor. $R_{w_i}$ is the reputation of $w_i$ and $R_{max}$ is the upper bound of the reputation.

### 6.2 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 batch allocation approaches consider the assigned worker's social network contexts; 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 $W$. 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, $V(w_i,t)$, as the probability of $w_i$ being assigned $t$, which is shown as follows:

$$V(w_i,t) = \alpha_1 \cdot \frac{\beta_1 \cdot |S_t \cap S_{w_i}| + \beta_2 \cdot R_{w_i}}{\beta_3 \cdot \gamma_{w_i}} + \alpha_2 \cdot \frac{\sum_{w_j \in (W-\{w_i\})} \left( \frac{\beta_1 \cdot |(S_t - S_{w_i}) \cap S_{w_j}| + \beta_2 \cdot R_{w_j}}{\beta_3 \cdot \gamma_{w_j} \cdot d_{ij}} \right)}{|W - \{w_i\}|}. \tag{18}$$

where $\alpha_1$ and $\alpha_2$ denote the relative contributions of $w_i$ and $w_i$'s contextual workers to $V(w_i,t)$, and $\beta_1$, $\beta_2$, and $\beta_3$ denote the relative importance of the three factors. $V(w_i,t)$ is different from $V(w_i,B)$ in Equation (12), because $V(w_i,t)$ is the crowdsourcing value for an individual task but $V(w_i,B)$ 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.

### 6.3 Dataset and Settings

Now many representative studies on crowdsourcing of complex tasks are validated by simulation experiments [12], [13], [20], 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. Therefore, we make simulation experiments by using real data set 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 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 $[d_t/2, d_t]$.

In fact, the three types of structures (random network, scale-free network, and small-world network) are very typical in real social networks [9], [10], [29], thus we use them in the experiments to conduct a simulated social network environment. Then, we will test the universality of our approaches in varying social network structures.

We now describe how these networks are constructed.

- *Random Network*. By referring to [29], the random network is generated by randomly adding connections between workers, which results in the network average degree being equal to 6.

TABLE 1
The Properties of the Three Social Network Structures

| Network\ Properties | 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 |

- *Scale-Free Network*. By referring to [9], 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*. By referring to [10], 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.

In summary, the properties of the three types of social network structures among workers in experiments are shown in Table 1.

## 6.4 Performance Indices

According to the optimization objective of this paper, 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 (16), we assume the probability of successfully completing task $x$ is $p_x$ and there are a total of $m$ tasks. The completion proportion of all tasks is defined as $\sum_{x=1}^{m} p_x/m$.

◆ *Total Payment 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.

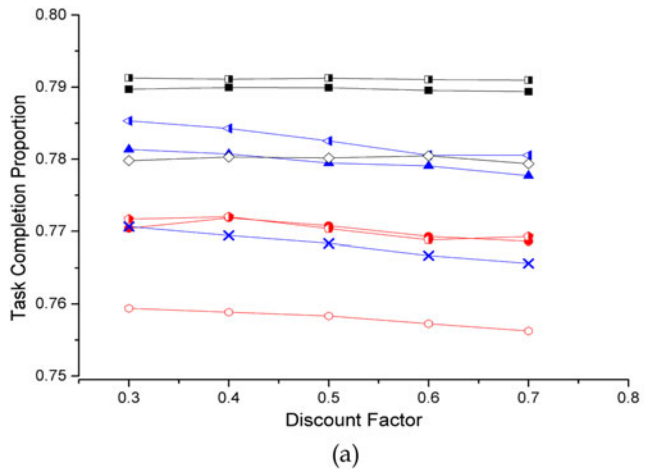
## 6.5 Experimental Results

Our experiments are implemented in Python 2.7 and tested on an Intel(R) Core(TM) CPU i7-4770 3.4 GHz and 16 G memory. There are four main factors that may influence the results: 1) discounting factor $\sigma$ for the discounting function in Equation (2) (which is set ranging from $\sigma = 0.3$ to $\sigma = 0.7$ in the experiments); 2) decay factor $\tau$ in Equation (17), which influences the decay rate of the probability to get skills as a function of social distance; 3) structures of social networks, which can be used to test the universality of our approaches in varying networks; and 4) number of tasks, which may influence the performance. The experiments test the impacts of these four factors on the performance indices.

### 6.5.1 Tests on Task Completion Proportion

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

Fig. 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, $\tau = 0.03$. Obviously, the discounting factor has no influence on the retail-style allocation approach. 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 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



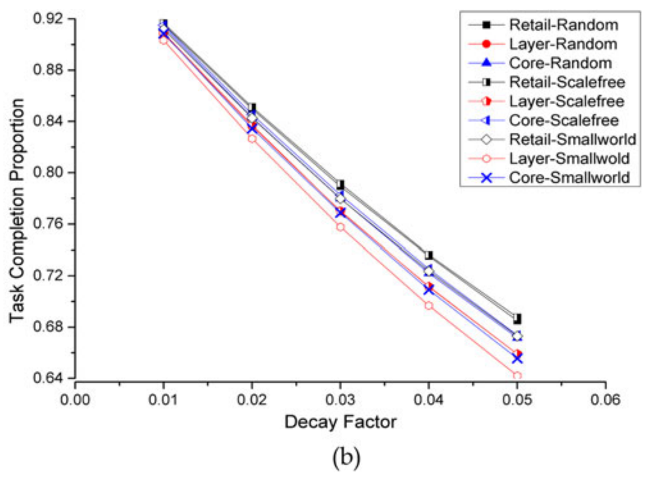


Fig. 5. Tests on the performances of task completion proportion.

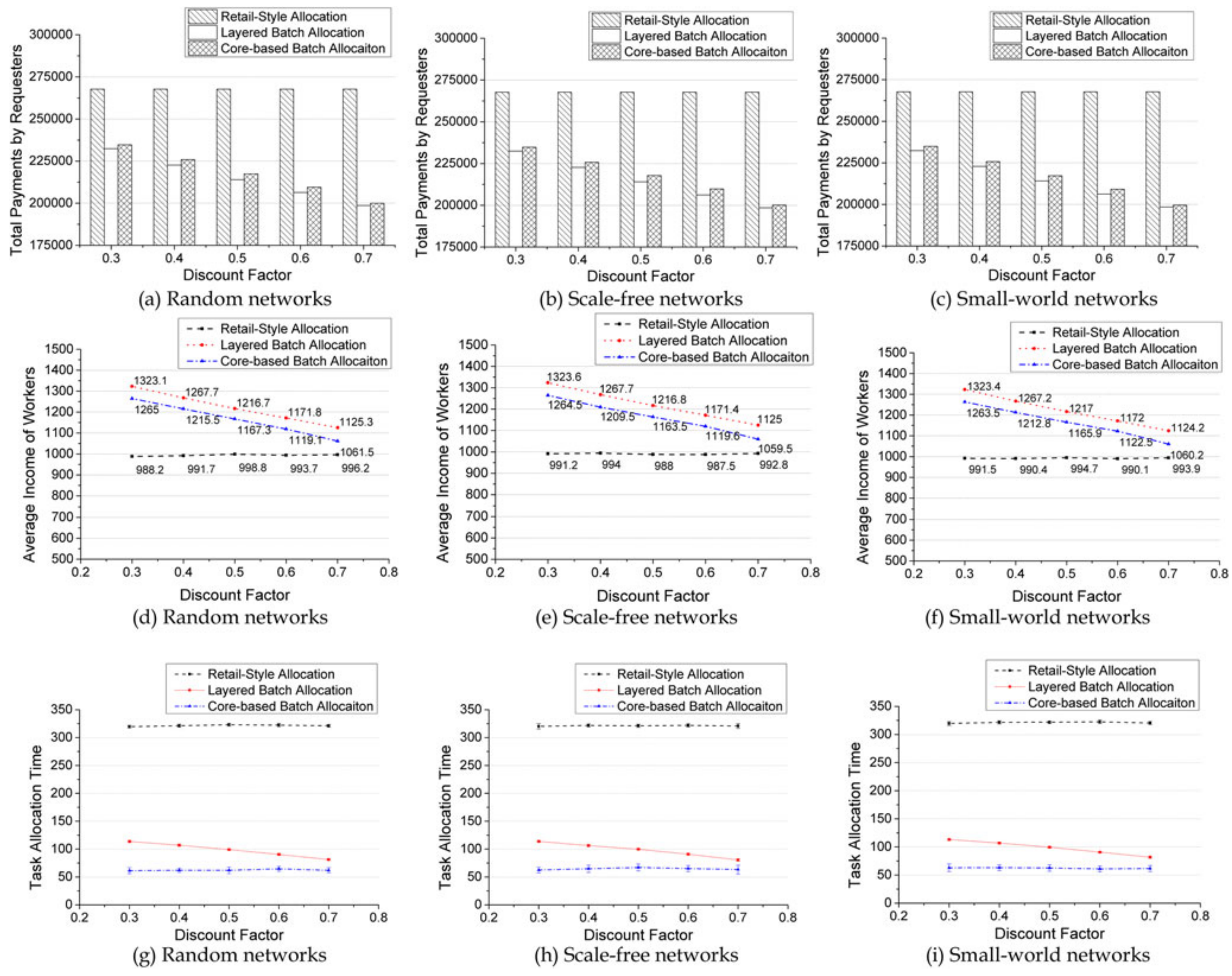


Fig. 6. Tests on the performances in terms of total payment by requesters, average income of workers, and task allocation time.

required skill according to Equation (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.

Fig. 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.

### 6.5.2 Tests on Total Payment By Requesters, Average Income of Workers, and Task Allocation Time

This series of experiments tests the performances on Total Payment by Requesters, Average Income of Workers, and Task Allocation Time of Algorithm of the three approaches under different discounting factors and social networks. Because the decay factor in Equation (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.

Fig. 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 Incomes 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 Payment by Requesters and Average Income of Workers, but it needs to consume more allocation time. In our approaches, the values of Total Payment 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.

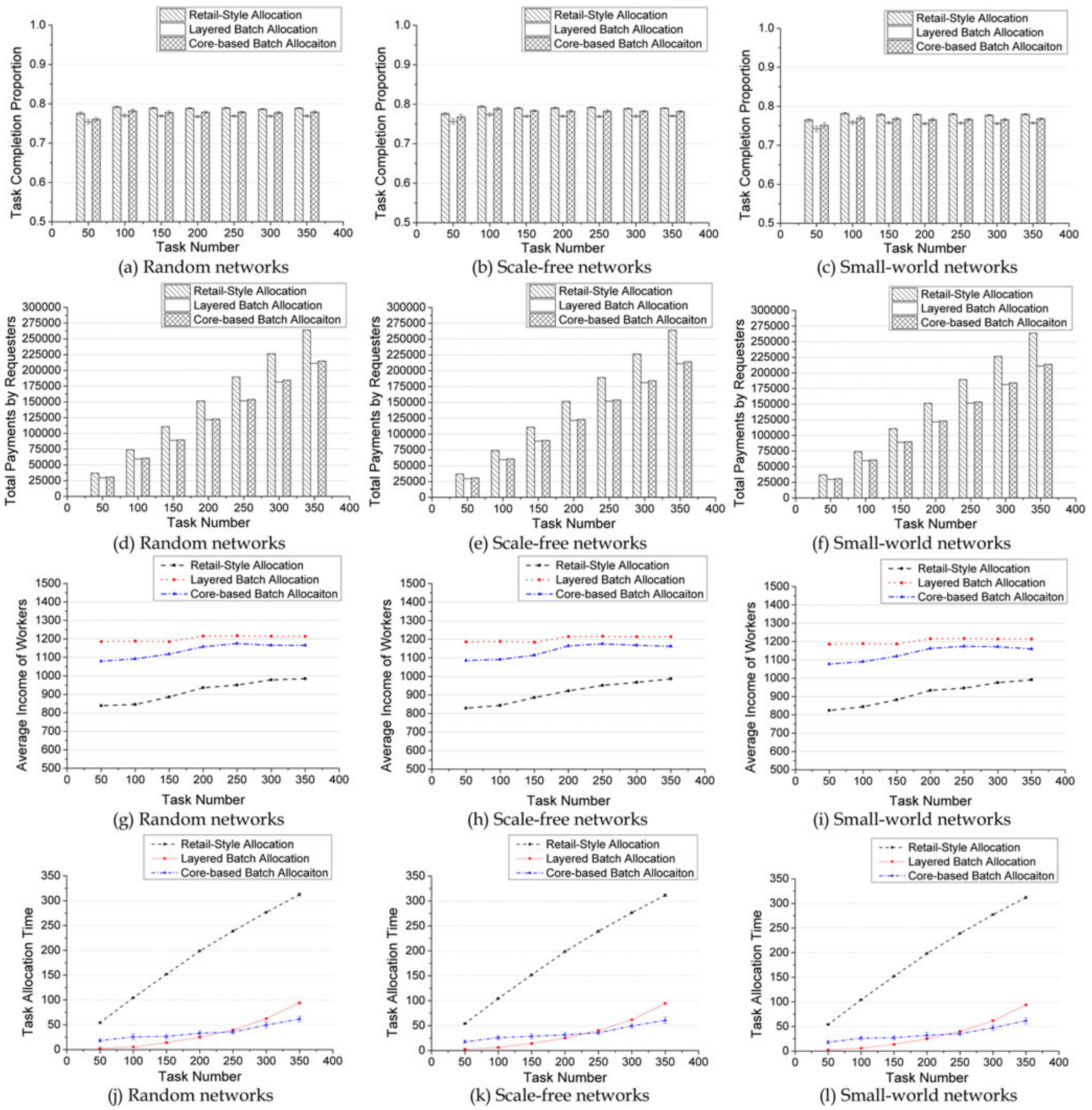


Fig. 7. The scalability to the number of tasks in terms of the four performance indices.

### 6.5.3 *Tests on the Scalability to the Number of Tasks*

After randomly counting the numbers of varying categories of tasks at Upwork newly published within one day, we find that they are all within several hundreds, e.g., 500 (web, mobile & software development), 70 (IT & Networking), 60 (Data Science & Analytics), 70 (Engineering & Architecture), 410 (Design & Creative), 250 (Writing), 50 (Translation). Our batching scheme makes batching for the tasks within the same category for one day (since the batching of tasks crossing multiple days may exceed the timeliness of tasks), thus we only made experiments based on hundreds of tasks.

Now we examine the performances of three approaches under different numbers of tasks; the results are shown as Fig. 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 Payment 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.

#### 6.5.4 *Comparison with the Exhaustive Batch Allocation Algorithm*

We design an exhaustive algorithm based on recursion to enumerate all possible batch assignments, as described in the Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2018.2894146. The exhaustive algorithm has very high computational complexity, O(n!), thus 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 Payment 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 in the Appendix available in the online supplemental material. The Total Payments by Requesters of our approaches are approximately 1.6 times that of exhaustive algorithm for a small number of tasks.

*In summary*, according to a comparison with the retail-style task allocation approach, our two approaches both achieve better performances in terms of Total Payment by Requesters and Average Income of Workers, while maintaining close Task Completion Proportion and consuming less task allocation time; moreover, our approaches are universal for varying social networks. Between our two approaches, the core-based approach incurs lower time cost and produces suboptimal results. Although our approaches' Total Payments by Requesters are 1.6 times that of the exhaustive algorithm, our approaches can scale to the 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, shown as follows.

- This paper assumes that the reusing is possible among the tasks by the same worker; moreover, this paper only considers the reusing costs without presenting a detailed reusing model. The limitation of such assumption is that it does not address the variety, dynamics, and complexity of reusing in real systems. In fact, reusing is very different in varying domains, for example, reusing in software domain is a very complex issue that is influenced by the software characteristics, the developers, and the application environments. Therefore, in the future we will explore the influence of varying reusing models on the effects of our batch allocation approach. Moreover, we will integrate the batch allocation approach with the software reusing technology to explore real applications in software domain.
- This paper assumes that the social networks are reliable during task allocation and execution, i.e., workers can reliably rely on their social network's skillset to complete the assigned tasks. However, due to the uncertainty and openness of social networks, the contextual workers may sometimes behave unreliably. Thus the reputation and trust mechanisms will be introduced for addressing unreliable workers in social networks.

## 7 Conclusion

The popular retail-style task allocation cannot scale to large numbers of concurrent tasks due to the large computational cost, because each task needs to be allocated and executed independently from scratch. 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, this paper presents a batch allocation method with the objective of reducing the requesters' total real payment as well as improving each worker' earnings. First this paper proves that the optimal batch allocation is NP-hard; then, this paper presents 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 paper performs theoretical analyses and extensive experiments on real data to prove 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.

In the future, we will mainly explore the adaption of the batch allocation mechanism to dynamic crowdsourcing environments in which workers may join or depart dynamically and the strategies and skills of workers are dynamic. Moreover, we will make large scale real online experiments through cooperating with popular crowdsourcing platforms.

## Acknowledgments

The work was supported in part by the National Natural Science Foundation of China (61472079, 61806053, 61807008), and the Natural Science Foundation of Jiangsu Province of China (BK20171363, BK20180356, BK20180369).

## References

[1] N. Luz, N. Silva, and P. Novais, "A survey of task-oriented crowdsourcing," *Artif. Intell. Rev.*, vol. 44, no. 2. pp. 187–213, 2015.
[2] D. E. Difallah, et al., "The dynamics of micro-task crowdsourcing: The case of amazon MTurk," in *Proc. 24th Int. Conf. World Wide Web*, May 18-22, 2015, pp. 238–247.
[3] L. Tran-Thanh, T. D. Huynh, A. Rosenfeld, S. Ramchurn, and N. R. Jennings, "BudgetFix: Budget limited crowdsourcing for interdependent task allocation with quality guarantees," in *Proc. 13th Int. Conf. Auton. Agents Multiagent Syst.*, May 5-9, 2014, pp. 477–484.

[4] L. Tran-Thanh, T. D. Huynh, A. Rosenfeld, S. D. Ramchurn, and N. R. Jennings, "Crowdsourcing complex workflows under budget constraints," in *Proc. 29th AAAI Conf. Artif. Intell.*, Jan 25–30, 2015, pp. 1298–1304.

[5] M. Rokicki, et al., "Groupsourcing: Team competition designs for crowdsourcing," in *Proc. 24th Int. Conf. World Wide Web*, May 18–22, 2015, pp. 906–915.

[6] W. Wang, J. Jiang, B. An, and Y. Jiang, "Towards efficient team formation for crowdsourcing in noncooperative social networks," *IEEE Trans. Cybern.*, 47, no. 2, pp. 4208–4222, Dec. 2017.

[7] J. van Dalen, J. Koerts, and A. R. Turik, "The measurement of labour productivity in wholesaling," *Int. J. Res. Marketing*, vol. 7, no. 1, pp. 21–34, 1990.

[8] D. Grosu and A. T. Chronopoulos, "Algorithm mechanism design for load balancing in distributed systems," *IEEE Trans. Syst. Man Cybern.-Part B: Cybern.*, vol. 34, no. 1, pp. 77–84, Feb. 2004.

[9] A. L. Barabási and R. Albert, "Emergence of scaling in random networks," *Sci.*, vol. 286, no. 5439, pp. 509–512, 1999.

[10] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[11] M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich, "Soylent: A word processor with a crowd inside," *Commun. ACM*, vol. 58, no. 8, pp. 313–322, 2015.

[12] A. Anagnostopoulos, L. Becchetti, A. Fazzone, I. Mele, and M. Riondato, "The importance of being expert: Efficient max-finding in crowdsourcing," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, May 31-Jun. 4, 2015, pp. 983–998.

[13] L. Tran-Thanh, S. Stein, A. Rogers, and N. R. Jennings, "Efficient crowdsourcing of unknown experts using multi-armed bandits," *Artif. Intell.*, vol. 214, pp. 89–111, 2014.

[14] A. Bozzon, M. Brambilla, S. Ceri, M. Silvestri, and G. Vesci, "Choosing the right crowd: Expert finding in social networks," in *Proc. 16th Int. Conf. Extending Database Technol.*, Mar. 18-22, 2013, pp. 637–648.

[15] H. Heidari and M. Kearns, "Depth-workload tradeoffs for workforce organization," in *Proc. 1st AAAI Conf. Hum. Comput. Crowdsourcing*, Nov. 6-9, 2013, pp. 60–68.

[16] M. Kargar, A. An, and M. Zihayat, "Efficient bi-objective team formation in social networks," in *Proc. Eur. Conf. Mach. Learn. Principles Practice Knowl. Discovery Databases*, Sep. 24-28, 2012, pp. 483–498.

[17] Q. Liu, T. Luo, R. Tang, and S. Bressan, "An efficient and truthful pricing mechanism for team formation in crowdsourcing markets," in *Proc. IEEE Int. Conf. Commun.*, Jun. 8-12, 2015, pp. 567–572.

[18] I. Lykourentzou, R. E. Kraut, S. Wang, and S. P. Dow, "Team dating: A self-organized team formation strategy for collaborative crowdsourcing," in *Proc. ACM CHI Conf. Hum. Factors Comput. Syst.*, May 07-12, 2016, pp. 1243–1249.

[19] P. K. Kopalle, C. F. Mela, and L. Marsh, "The dynamic effect of discounting on sales: Empirical analysis and normative pricing implications," *Marketing Sci.*, vol. 18, no. 3, pp. 317–332, 1999.

[20] D. Yang, G. Xue, X. Fang, and J. Tang, "Incentive mechanisms for crowdsensing: Crowdsourcing with smartphones," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1732–1744, Jun. 2016.

[21] L. Green and J. Myerson, "A discounting framework for choice with delayed and probabilistic rewards," *Psychological Bulletin*, vol. 130, no. 5, pp. 769–792, 2004.

[22] M. Yin and M. L. Gray, "The communication network within the crowd," in *Proc. 25th Int. World Wide Web Conf.*, Apr. 11-15, 2016, pp. 1293–1303.

[23] M. L. Gray and S. Suri, "The crowd is a collaborative network," in *Proc. 19th ACM Conf. Comput.-Supported Cooperative Work Soc. Comput.*, Feb. 27- Mar. 02, 2016, pp. 134–147.

[24] X. Zhang, Z. Yang, C. Wu, W. Sun, Y. Liu, and K. Xing, "Robust trajectory estimation for crowdsourcing-based mobile applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1876–1885, Jul. 2014.

[25] H. Chen, H. Jin, and S. Wu, "Minimizing inter-server communications by exploiting self-similarity in online social networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 1116–1130, 2016.

[26] E. Franchi, A. Poggi, and M. Tomaiuolo, "Social media for online collaboration in firms and organizations," *Int. J. Inf. Syst. Model. Des.*, vol. 7, no. 1, pp. 18–31, 2016.

[27] J. Jiang, B. An, Y. Jiang, and D. Lin. [Online]. "Context-aware reliable crowdsourcing in social networks," *IEEE Trans. Syst. Man Cybern.: Syst.*, https://ieeexplore.ieee.org/document/8226780, doi: 10.1109/TSMC.2017.2777447.

[28] J. Wang, S. Faridani, and P. G. Ipeirotis, "Estimating the completion time of crowdsourced tasks using survival analysis models," in *Proc. Workshop Crowdsourcing Search Data Mining*, Feb. 9, pp. 31–34, 2011.

[29] W. Wang and Y. Jiang, "Multiagent-based allocation of complex tasks in social networks," *IEEE Trans. Emerging Topics Comput.*, vol. 3, no. 4, pp. 571–584, Dec. 2015.

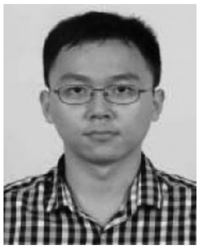
[30] R. M. Karp, "Reducibility among combinatorial problems," *Complexity of Computer Computations*. New York, NY, USA: Springer, 1972, pp. 85–103.

**Jiuchuan Jiang** received the ME degree in computer science from the Nanjing University of Aeronautics and Astronautics, in 2009. He is now with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His main research interests include crowdsourcing, multiagent systems, and social networks. He has published several scientific articles in refereed journals and conference proceedings, such as the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, the *ACM Transactions on Autonomous and Adaptive Systems*, and the International Conference on Autonomous Agents and Multiagent Systems (AAMAS).

**Bo An** received the PhD degree in computer science from the University of Massachusetts, Amherst. He is an associate professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His current research interests include artificial intelligence, multiagent systems, game theory, and optimization. His research results have been successfully applied to many domains including infrastructure security and e-commerce. He has published more than 90 referred papers at AAMAS, IJCAI, AAAI, ICAPS, KDD, JAAMAS, AIJ and ACM/IEEE Transactions. He was the recipient of the 2010 IFAAMAS Victor Lesser Distinguished Dissertation Award, an Operational Excellence Award from the Commander, First Coast Guard District of the United States, the 2012 INFORMS Daniel H. Wagner Prize for Excellence in Operations Research Practice, and 2018 Nanyang Research Award (Young Investigator). His publications won the Best Innovative Application Paper Award at AAMAS'12 and the Innovative Application Award at IAAI'16. He was invited to give Early Career Spotlight talk at IJCAI'17. He led the team HogRider which won the 2017 Microsoft Collaborative AI Challenge. He was named to IEEE Intelligent Systems' "AI's 10 to Watch" list for 2018. He was invited to be an Advisory Committee member of IJCAI'18. He is a member of the editorial board of JAIR and the Associate Editor of *Journal* of *Autonomous Agents and Multi-Agent Systems* and *IEEE Intelligent Systems*. He was elected to the board of directors of IFAAMAS.

**Yichuan Jiang** (SM'13) received the PhD degree in computer science from Fudan University, Shanghai, China, in 2005. He is currently a full professor with the School of Computer Science and Engineering, Southeast University, Nanjing, China. His main research interests include multiagent systems, crowdsourcing, and social networks. He has published more than 90 scientific articles in refereed journals and conference proceedings, such as the *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *IEEE Transactions on Cybernetics*, the *ACM Transactions on Autonomous and Adaptive Systems*, the *Journal of Autonomous Agents and Multi-Agent Systems*, IJCAI, AAMAS, and AAAI. He won the best paper award from PRIMA06 and best student paper awards twice from ICTAI13 and ICTAI14. He is a senior member of the IEEE.

**Peng Shi** received the ME degree in computer science from Southeast University, Nanjing, China, in 2018. His current research interests include crowdsourcing and multiagent systems. He has published several papers in refereed journals and conference proceedings, such as the knowledge-based Systems, and the International Conference on Autonomous Agents and Multiagent Systems (AAMAS).



**Zhan Bu** received the PhD degree in computer science from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2014. He is currently an associate professor with the College of Information Engineering, Nanjing University of Finance and Economics, Nanjing, China. His current research interests include social networks, and multiagent systems.

**Jie Cao** received the PhD degree from Southeast University, Nanjing, China, in 2002. He is currently a professor and the dean of the College of Information Engineering, Nanjing University of Finance and Economics, Nanjing, China. His current research interests include business intelligence, and social networks.