Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 5570192, 11 pages
https://doi.org/10.1155/2021/5570192

Research Article

Fairness of Task Allocation in Crowdsourcing Workflows

Donglai Fu [1,2] **and Yanhua Liu** [3]

[1]*Software School, North University of China, Taiyuan 030051, Shanxi, China*
[2]*Shanxi Province Military-Civilian Integration Software Engineering Technology Research Center, Taiyuan 030051, Shanxi, China*
[3]*Affiliated Hospital, North University of China, Taiyuan 030051, Shanxi, China*

Correspondence should be addressed to Donglai Fu; hhluci@163.com

Received 20 January 2021; Revised 31 March 2021; Accepted 12 April 2021; Published 24 April 2021

Academic Editor: Zhiguo Yan

Fairness plays a vital role in crowd computing by attracting its workers. The power of crowd computing stems from a large number of workers potentially available to provide high quality of service and reduce costs. An important challenge in the crowdsourcing market today is the task allocation of crowdsourcing workflows. Requester-centric task allocation algorithms aim to maximize the completion quality of the entire workflow and minimize its total cost, which are discriminatory for workers. The crowdsourcing workflow needs to balance two objectives, namely, fairness and cost. In this study, we propose an alternative greedy approach with four heuristic strategies to address such an issue. In particular, the proposed approach aims to monitor the current status of workflow execution and use heuristic strategies to adjust the parameters of task allocation. We design a two-phase allocation model to accurately match the tasks with workers. The F-Aware allocates each task to the worker that maximizes the fairness and minimizes the cost. We conduct extensive experiments to quantitatively evaluate the proposed algorithms in terms of running time, fairness, and cost by using a customer objective function on the WorkflowSim, a well-known cloud simulation tool. Experimental results based on real-world workflows show that the F-Aware, which is 1% better than the best competitor algorithm, outperforms other optimal solutions in finding the tradeoff between fairness and cost.

1. Introduction

In China, crowdsourcing has been quickly making progress in various fields in the past years. Zhubajie (http://www.zbj.com) has established itself as a crowdsourcing leader with more than 22 million active workers. This company covers a range of online and offline services, including tutoring and logo and product design. Didi Chuxing (DiDi) is another representative example. DiDi is China's leading mobile transportation platform that provides a full range of application-based transportation services (including taxi; express, premier, and deluxe bus; designated driving; enterprise solutions; bike and e-bike sharing, automobile solutions, and food delivery) for numerous people. Tens of millions of drivers find flexible work opportunities on the DiDi platform. The platform provides more than 10 billion passenger trips a year. Crowdsourcing has become a fast, convenient, and cost-effective mode of research and production to obtain flexible and cheap resources. Various organizations flexibly outsource work, such as collaborative sensing [1, 2] and human-powered online security [3, 4], to a global pool of workers on a temporary basis. Therefore, addressing fairness in crowdsourcing systems is relevant as a modern issue in ethics.

In crowdsourcing systems, tasks are posted to a crowdsourcing platform, and these tasks are solved by a large group of workers registered at the platform. The Amazon Mechanical Turk (MTurk) [5] is a successful crowdsourcing system that enables individuals or companies to harness collective intelligence from a global workforce to accomplish various tasks, such as human intelligence tasks (HITs). Employers (known as requesters) recruit employees (known as workers) to execute HITs and reward the employees for their labor.

Fairness is important for requesters because workers are likely to engage with HITs, which are published by fair requesters. Fairness is also a key point for crowdsourcing platforms because a negative correlation exists between workers' satisfaction and turnover. Fairness has a positive effect on the job satisfaction of workers. Such effect of workers' expectations on platforms' fairness on the likelihood of their participation is more than that of the considerations of self-interest [6, 7]. For example, workers expect to be allocated a fair number of tasks which is proportional to their matching availabilities for the task. Fairness is an essential concept for sustaining a powerful crowd with substantial participation of workers. Therefore, fairness should be considered when allocating tasks to workers.

Solving complex tasks by using crowdsourcing platforms has emerged in recent years. A collaborative crowdsourcing model is an alternative approach to solve complex tasks on crowdsourcing platforms. In this model, a complex task is generally decomposed into a series of interrelated subtasks. These subtasks are organized in terms of workflow. However, in this scenario, task design and allocation remain challenging. We propose a general method that enables the requesters and workers to work collaboratively for improving the quality of task design. A relevant paper has passed the review of a journal. This study focuses on the fairness-aware task allocation in a collaborative crowdsourcing model.

Traditionally, the optimization of task allocation problem is focused on cost instead of fairness. More challenges are faced when ones research the current problem of crowdsourcing workflows compared to that of simple HITs. The first problem is that the current crowdsourcing platforms, such as Zhubajie and MTurk, cannot directly execute a workflow because it describes a logical structure across tasks. Moreover, the maximal fairness of the whole workflow instead of one simple task needs to be considered. Thus, the research on addressing fairness-aware task allocation in scenarios is limited.

We propose an alternative fairness-aware task allocation approach to complete the workflows with minimal cost and maximal fairness before the deadline to bridge the gap. In particular, the proposed approach aims to monitor the current status of all tasks that are not allocated to workers during the workflow execution and dynamically update the parameter values of allocation algorithms.

This study investigates a practical and important problem, that is, dynamic fairness-aware task allocation, to maximize fairness and minimize cost in crowdsourcing workflows. The summary of the principal contributions of this work is provided as follows:

The fairness-aware task allocation problem in crowdsourcing workflows is formulated as a constraint optimization problem based on the customized fairness. A generic two-phase task allocation model inspired by [8] is designed to cover various crowdsourcing workflow scenarios based on the new fair criterion.

Four heuristic strategies are proposed to solve the current problem. A well-designed fairness-aware solution called the *F-Aware* is introduced to minimize the overall cost and target maximum fairness.

Performance is assessed using the workflow simulation tool named the WorkflowSim with different workflow benchmarks. Results show that the F-Aware outperforms other optimal solutions in finding the tradeoff between fairness and cost. The F-Aware performs 1% better than the best competitor algorithm, Max_Fairness.

The motivation of the research is to find the trade-off between two objectives, that is, minimal cost and maximal fairness, while allocating crowd tasks to workers. Furthermore, all tasks must be completed before the deadline. The ideas presented are beneficial to solve the problem of task allocation in the crowd context based on the workflow. The methods presented can promote the prosperity of the crowd platform. In addition, the results presented can be applied to other task allocation problems, in which the distributor wants to enforce the fairness among participants.

The remaining parts of this paper are organized as follows. The related work is explored in Section 2. The task allocation problem in the crowdsourcing workflow scenario is formulated in Section 3. The two-phase task allocation model is explained in detail in Section 4. The F-Aware algorithm is explained in detail in Section 5. The experimental evaluation is presented in Section 5. The conclusion is provided in Section 6.

2. Related Work

The combination of humans and computers to accomplish tasks that neither can do alone has attracted considerable attention from academic and industrial circles [9]. This idea dates back to the 1960s with the publication of "Man–Computer Symbiosis" by Likelier [10]. Tim Berners-Lee has proposed the concept of a social machine in 2009 and regarded the cooperation between machines and humans as the next direction of web application development [11]. The term "crowdsourcing" was coined by Jeff Howe in 2006 [12]. The MTurk is a pioneering crowdsourcing system and has been successfully used to solve multiple simple tasks. Solving complex tasks by leveraging the crowdsourcing systems remains challenging [13]. An alternative approach is to bring the workflow into the solution of crowdsourcing complex tasks. In particular, a complex task is decomposed into a series of interdependent subtasks which is relatively easy to solve. Workflows are used to express the logical structures across subtasks.

Promoting fair treatment is of utmost importance for effective and capable crowdsourcing systems. If workers believe that the environment that they are working within is fair, we can expect an improvement in the quality of workers' answers, which is beneficial for requesters, and an increase in retaining and recruiting additional workers, which is beneficial for crowdsourcing platforms [14]. However, existing efforts give much attention to other objectives instead of fairness especially in crowdsourcing workflows.

For the design of crowdsourcing workflows, Kittur et al. have proposed a solution to decompose a complex task on the basis of the MapReduce mechanisms [15]. In their proposed method, task designers must specify the execution sequence of subtasks. Little et al. have explored an iterative workflow paradigm for solving complex tasks, including image description, copyediting, handwriting recognition, and sorting [16, 17], and improved the quality of results by using an iterative algorithm in which the number of iterations is determined by the budget. However, requesters are required to divide each task by using a hard code before the task is posted on a third-party crowdsourcing system, such as MTurk. Dai et al. have improved the iterative workflow model from the aspect of workflow control [18]. Their model can autonomously control workflows without human intervention and yield good results. Lin et al. have proposed the idea of multiple workflows based on a probabilistic graphical model and dynamically implemented switches across these workflows [19]. Their experiments demonstrated good results for named-entity recognition. Bernstein et al. introduced a novel idea of multiple-phase workflow and designed a find–fix–verify crowd programming pattern, which split tasks into a series of generation and review stages for complex crowdsourcing writing [20]. Kulkarni et al. developed the PDS (price-divide-solve) algorithm to guide workers by converting large and complex tasks into microtasks that are appropriate for crowd markets. Zheng et al. proposed a general workflow technology by using a state machine based on recursive decomposition approaches, wherein varying types of crowdsourcing applications could be developed using this platform, to meet the requirement of solving diverse tasks [21]. Xiong et al. extended Zheng's work by proposing a workflow framework called the SmartCrowd for complex crowdsourcing tasks [22]. Wu et al. presented the Service4Crowd, which was a highly flexible and extensible process management platform for crowdsourcing on the basis of service-oriented architectures [23]. They indicated that the platform could provide a one-stop solution for requesters. Inspired by the above developments, the authors of this study proposed an alternative method that engages workers in task design to enable requesters to achieve a high-quality design of complex tasks.

The design and the allocation of tasks have achieved remarkable progress but remain as open issues in crowdsourcing workflow scenarios. Task allocation aims to maximize task quality and minimize cost by allocating tasks to appropriate workers [24]. A task allocation method is crucial in allocating tasks to the most appropriate workers. Vaughan et al. explored the problem of allocating heterogeneous tasks to workers with different and unknown skill sets in crowdsourcing markets [25]. Khazankin et al. proposed solutions by extending standards, such as web service-level agreement, to ensure the quality between the crowd consumers and the crowdsourcing platform and provided a skill-based crowd scheduling algorithm on the basis of negotiated agreements [26]. Karger et al. considered a general model of crowdsourcing tasks to minimize the total price [27]. Boutsis et al. explored the most efficient allocation of tasks to workers to achieve their successful completion under real-time constraints [28, 29]. Unlike solving simple tasks, the quality of solving complex tasks needs to ensure the quality of the entire workflow. Several studies have been conducted to determine when and how to publish the tasks on the crowdsourcing platform to complete the workflows with minimum cost and without missing the deadlines. Khazankin et al. formulated the problem as a time-constrained optimization problem [30]. Tang et al. adopted heuristic strategies to improve Khazankin et al.'s work [31].

Fairness in crowdsourcing has been considered in providing fair wages and detecting malicious workers. Franke et al. first highlighted the importance of fairness in the context of crowdsourcing and demonstrated that the workers' expectations on fairness have a strong effect on their decision to participate in solving tasks [7]. Faullant et al. supported this argument by further exploring how the different types of fairness, that is, distributive (a fair amount and distribution of the offered reward) and procedural fairness (fair procedures to determine the winners), affect the workers' intentions to participate in future crowdsourcing tasks and their loyalty toward the platform [14]. Recently, researchers have provided incentive strategies to improve workers' perceived fairness [32, 33]. McInnis et al. have viewed wage discrimination as the wrongful rejection of work, unfair compensation amount, or delayed payment [34]. Studies that address malicious workers through task assignment and workers' reputation focused on the quality, reliability, and total cost of workers' contributions [35]. These schemes are requester-centric and do not guarantee fair task allocation to workers. The problems in some particular crowdsourcing fields have been discussed, but the solutions of fair task allocation of crowdsourcing workflows are very limited [36].

The abovementioned studies show that the research on task allocation has attracted increasing attention by considering the characteristics of workers. All the presented results of the existing literature have provided references and guidelines for the current research on the fair task allocation of crowdsourcing workflows.

3. Problem Formulation

This study considers a crowdsourcing workflow system. Employers (i.e., requesters) recruit employees (i.e., workers or machines) to complete specific tasks and provide them with varying values of wages (i.e., reward). In this section, the fairness-aware task allocation architecture is designed for crowdsourcing workflows, and some preliminary definitions are presented.

3.1. Crowdsourcing Workflow System. A complex task, which comes from a requester, is designed as a workflow that consists of interdependent subtasks. The focus is to distribute these subtasks to workers or machines and consider cost and fairness. In other words, an allocation strategy is designed and proposed. Figure 1 illustrates the basic structure of the system, in which the fairness is considered a

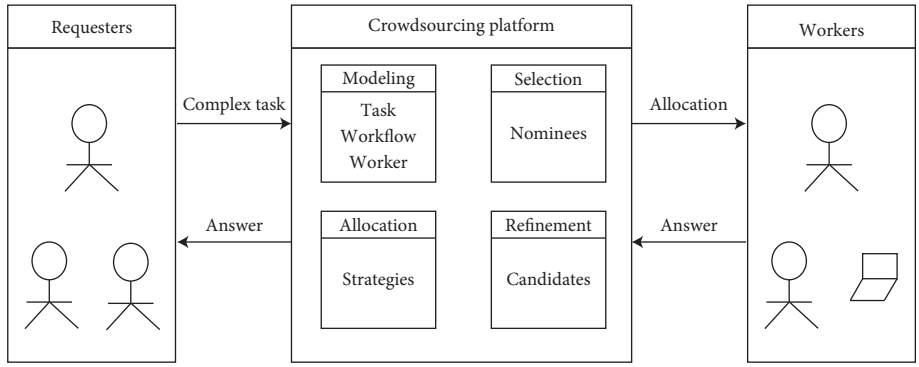

Figure 1: Architecture of fair task allocation model.

major factor in picking workers or machines. In this scenario, a complex task submitted by a requester is first converted to a workflow. Then, the crowdsourcing platform identifies the nominees for each subtask to be allocated. Candidates who come from nominees who volunteer to accept the subtask are confirmed. Finally, the platform allocates the subtask to the candidate in accordance with the allocation strategy. The goal of this study is to optimize the distribution strategy.

The basic definitions are as follows.

Definition 1. Requesters are people who submit complex tasks to the crowdsourcing platform. The set of requesters is represented as $\mathscr{R} = \{r_i | i = 1, 2, \ldots, n\}$, where r_i denotes the i-th requester.

Definition 2. Workers are people who complete tasks and earn money. The set of workers is represented as $\mathscr{W} = \{w_i | i = 1, 2, \ldots, n\}$, where w_i denotes the i-th worker. Each worker is associated with a set of attributes $\langle n_1, n_2, p, \mathscr{S}, v \rangle$. $w_i \cdot n_1$ is the number of tasks allocated to worker w_i. $w_i \cdot n_2$ is the number of accepted tasks. $w_i \cdot p$ is the probability of accepting a task. $w_i \cdot \mathscr{S}$ is a skill vector $(s_1, s_2, \ldots, s_m)$, where s_j is a Boolean value that identifies whether the worker w_i has the j-th skill, and $w_i \cdot v$ is the lowest reward for completing a task.

Definition 3. Machines are a group of computing nodes that consist of a series of computing services. The set of machines is represented as $\mathscr{M} = \{m_i | i = 1, 2, \ldots, n\}$, where m_i denotes the i-th machine. Each machine is associated with a set of attributes $\langle n_1, n_2, p, \mathscr{S}, v \rangle$. $m_i \cdot n_1$ is the number of tasks allocated to machine m_i. $m_i \cdot n_2$ is the number of accepted tasks. $m_i \cdot p$ is the probability of accepting a task. $m_i \cdot \mathscr{S}$ is a configuration vector $(s_1, s_2, \ldots, s_m)$, where s_j is a Boolean value that identifies whether the machine m_i has the j-th configuration, and $m_i \cdot v$ is the lowest reward for completing a task.

Definition 4. Tasks can be defined as a set $\mathscr{T} = \{t_i | i = 1, 2, \ldots, n\}$, where t_i denotes the i-th task. Each task is associated with a set of attributes $\langle p_0, p_1, q_0, q_1, q_2, q_3, v, \mathscr{S} \rangle$. $t_i \cdot p_0$ is the type of task t_i used to classify a task as either machine or human (0 or 1, respectively). $t_i \cdot p_1$ is the business type of task t_i, such as *ui-design*, *programming*. $t_i \cdot q_0$ is the start time of task t_i. $t_i \cdot q_1$ is the receiving time of task t_i. $t_i \cdot q_2$ is the time required to finish task t_i. $t_i \cdot q_3$ is the end time of task t_i. $t_i \cdot q_0$ and $t_i \cdot q_1$ are dynamically changing in terms of workflow progress. $t_i \cdot v$ tells the worker the amount of reward he will receive after successfully completing a task on time. Initially, variable v will not be assigned a value until available workers or machines are found. $t_i \cdot \mathscr{S}_1$ is the condition vector which must be satisfied to complete the task t_i.

Definition 5. Fairness can be defined as an indicator that shows two similar workers or machines with similar probability of obtaining the same task. Two different workers or machines are similar if they have similar availabilities. In other words, the parameters $p, \mathscr{S}$, and v of a worker or a machine are similar in context.

3.2. Workflow Model. The workflow can be represented by a directed graph that indicates the dependency of data and the order of execution across tasks. Therefore, the workflow can be defined as a quaternion: $\mathscr{G} = \{\mathscr{T}, \mathscr{E}\}$, where $\mathscr{T} = \{t_i | i = 1, 2, \ldots, n\}$ denotes the collection of task nodes. $|\mathscr{T}| = n$ corresponds to the number of vertices in graph $\mathscr{G}$. Each node in the graph represents a human or machine task, which is the smallest allocation unit. The directional edge $e(i, j) \in \mathscr{E}$ in the graph denotes the order of execution between tasks t_i and t_j. t_j cannot be executed until t_i is completed. Therefore, task t_i is called the predecessor of task t_j, and task t_j is the successor of task t_i. In this study, $\mathrm{pred}(t_i)$ denotes the predecessor set of task t_i, and $\mathrm{ssucc}(t_j)$ denotes the successor set of task t_j. Start time is denoted as $t_i \cdot q_0 = \max_{t_k \in \mathrm{pred}(t_i)} (t_k \cdot q_1) + t_i \cdot q_1$. End time is denoted as $t_i \cdot q_3 = t_i \cdot q_0 + t_i \cdot q_2$.

3.3. Measurement of Fairness. In the context, fairness, which is shown in Definition 5, is the probability that two similar workers or machines accomplish the same task. Thus, two factors, that is, n_1 and n_2, are considered for measurement. Each worker or machine is associated with a local allocation ratio, that is, $w_i \cdot p_0$ or $m_i \cdot p_0$, respectively. The meaning of the two symbols, w_i and m_i, is interpreted in Definitions 2

and 3. A worker's and machine's local allocation ratios can be calculated using the two following equations, respectively.

$$w_i \cdot p_0 = \frac{w_i \cdot n_1}{w_i \cdot n_2}, \tag{1}$$

$$m_i \cdot p_0 = \frac{m_i \cdot n_1}{m_i \cdot n_2}. \tag{2}$$

The fairness of the allocation strategy can be defined as the proximity of local allocation ratios of similar workers or machines. Accordingly, the fairness of each participant can be calculated while crowdsourcing a workflow using the two following equations:

$$\text{Fairness}(w_i) = 1 - \sqrt{\frac{1}{\mathcal{N}-1}\sum(w_i \cdot p_0 - \overline{w_i \cdot p_0})}, \tag{3}$$

$$\text{Fairness}(m_i) = 1 - \sqrt{\frac{1}{\mathcal{N}-1}\sum(m_i \cdot p_0 - \overline{m_i \cdot p_0})}, \tag{4}$$

where $\mathcal{N}$ is the number of similar workers or machines.

3.4. Formalization. In collaborative crowdsourcing scenarios, the procedure of fairness-aware task allocation can be formally described as follows. Given a workflow $\mathcal{G}$, a set of workers $\mathcal{W}$, and a set of machines $\mathcal{M}$, the allocation of tasks in the workflow to workers or machines maximizes the overall fairness of the allocation strategy and minimizes the total cost under the constraint of completing the total workflow before the deadline.

Overall fairness: let $\text{Fairness}(\mathcal{G})$ be the overall fairness, which can be expressed as

$$\text{Fairness}(\mathcal{G}) = \frac{\sum_{i=1}^{n}\text{Fairness}(n_i) + \sum_{j=1}^{n}\text{Fairness}(m_i)}{n+m}. \tag{5}$$

Total cost: Let $\text{Cost}(\mathcal{G})$ be the total cost, which can be expressed as

$$\text{Cost}(\mathcal{G}) = \sum_{k=1}^{m+n} t_k \cdot v. \tag{6}$$

On the basis of equations (5) and (6), we formulate the fundamental research problem of the fair allocation of the workflow among workers and machines as an objective function.

Maximize

$$\Delta(\mathcal{G}) = \text{Fairness}(\mathcal{G}) - \text{Cost}(\mathcal{G}), \tag{7}$$

subject to

$$t_{m+n} \cdot q_3 \le \Theta. \tag{8}$$

4. Allocation Model

In this section, a two-phase allocation model is described in detail. In this case, a requester, whose responsibility can be found in Definition 1, submits a complex task with some constraint conditions. Then, these tasks with conditions are converted as a workflow. The parameters $t_i \cdot p_0, t_i \cdot p_1, t_i \cdot q_1$, and $t_i \cdot \mathcal{S}$, which are interpreted in Definition 4, are estimated by the log of completed tasks. Other parameters change with the completion progress of the workflow. The platform has knowledge on workers' and machines' parameters because the parameters are registered in the platform in advance.

First, the appropriate set of workers and machines needs to be found in accordance with the task's constraint conditions to allocate a task. These workers and machines are called nominees. These workers are nominated to the task t_i if their conditions, the task accept ratio, and the available vector satisfy the task's constraints. The same is true for a nominated machine. In other words, the platform searches participants among the given set of workers and machines under the constraint conditions. Then, these similar nominees are grouped together. A value determined by the distribution of different groups' reward is assigned to the parameter $t_i \cdot v$. The candidates are found by progressive multicasting the task to k similar nominees in batches until at least one candidate is found. The parameter k is determined by the number of the members of two similar groups.

Second, one worker or one machine is picked from the candidates. The main goal of this step is to ensure that the workflow can be completed before its deadline while minimizing the total cost and maximizing the overall fairness. Thus, the longest path of the workflow is calculated first to satisfy the deadline. $\mathcal{P}$ denotes all the paths included in the workflow $\mathcal{G}$ from the start node to the end node. $\mathcal{P}_\ell = (\text{in}, t_1, t_2, \ldots, t_n, \text{out})$ denotes the longest path in the set $\mathcal{P}$. Two virtual nodes, in and out, are added to the path $\mathcal{P}_\ell$ for convenience and represent the start and the end node, respectively. The discovery process of the longest path $\mathcal{P}_\ell$ is described using the pseudocode in Algorithm 1. In the context, Algorithm 1 is used to check $t_{m+n} \cdot q_3 \le \Theta$.

One available constraint solver can always be found to solve the current problem. However, the computational complexity is inevitable when using these solvers. Thus, heuristic strategies are used to deal with this issue instead of the constraint solvers in this study. The simplest allocation strategy is random selection, which is also used as one baseline in the following experimental evaluation. Picking the most proper candidate is beneficial to maximize the overall fairness on the basis of the goal maximizing the local fairness. However, the strategy may result in additional budgets. Similarly, the low-cost strategies may result in the unfairness to workers.

A fairness-aware allocation algorithm called F-Aware is designed to cope with such an issue, which is described in Algorithm 2. The algorithm greedily allocates each task to the most desirable participant. The details are as follows. For each task in the workflow $\mathcal{G} = \{\mathcal{T}, \mathcal{E}\}$, the parameters p_0, p_1, and $\mathcal{S}$ are manually initialized in accordance with the characteristics of the task. q_1 and q_2 are estimated in accordance with the log of completed tasks. The F-Aware first calls Algorithm 1 to obtain the longest path $\mathcal{P}_\ell$ and checks whether the workflow can be completed before the deadline. Second, the algorithm finds the successive nodes of the start

Input: given workflow $\mathcal{G} = \{\mathcal{T}, \mathcal{E}\}$
Output: $\mathcal{P}_\ell = (\text{in}, t_1, t_2, \ldots, t_n, \text{out})$
(1) Let $\mathcal{P}_\ell = (\text{in})$
(2) Let current_node = in
(3) **While** current_node! = out **Do**
(4) Let S = ssucc(current_node)
(5) Let longest_time = 0
(6) Let t be null
(7) **For** $i = 1$ to $|S|$ **Do**
(8) **If** $(t_i \cdot q_1 + t_i \cdot q_2) >$ longest_time **Then**
(9) longest_time = $t_i \cdot q_1 + t_i \cdot q_2$
(10) $t = t_i$
(11) **End If**
(12) **End For**
(13) add t to $\mathcal{P}_\ell$
(14) Let current_node = t
(15) **End while**
(16) return $\mathcal{P}_\ell$

Algorithm 1: FindLongestPath().

node in, searches nominees for each node, and confirms the corresponding candidates. Third, the algorithm sorts the candidates in decreasing order of $\Delta(w)$ or $\Delta(m)$. Finally, the current task is allocated to the top candidate. The parameters of the remaining tasks need to be updated after the task is finished. The process continues until all nodes are visited.

5. Performance Evaluation

In this section, we first analyze the complexity of the allocation algorithm by Big O notation. Then, we evaluate solutions through experiments on the basis of real-world workflows. An object function $\mathcal{O}$ combing all optimization goals is given as equation (9) to compare various solutions easily and effectively.

$$\mathcal{O} = \text{Fairness}(\mathcal{G}) \times e^{-\rho \times \text{Cost}(\mathcal{G})}. \quad (9)$$

The object function has the same tendency with Fairness($\mathcal{G}$) because the goal is maximizing Fairness($\mathcal{G}$). However, the exponential part decreases with the total cost Cost($\mathcal{G}$). The parameter ρ determines the importance of the total cost Cost($\mathcal{G}$), where $0 \le \rho \le 1$. Only the fairness objective is considered when it is assigned to 0. The total cost is paid increasing attention with an increase in this parameter.

Two sets of experiments are conducted. In the first set, the same dataset with different sizes is used to compare the *F-Aware* with three other competitor algorithms from runtime, fairness, cost, as well as the combination of fairness and cost. The random selection is considered one baseline. The strategy, which is most beneficial to the maximum of the fairness objective, is considered as the second competitor. The intuition behind the strategy is finding a reasonable cost. The last strategy is most beneficial to the minimum total cost. The intuition behind the strategy is that an economic and fair allocation solution can be found. The second set of experiments is performed to evaluate their performance on the basis of the different datasets with the same scale.

5.1. Algorithm Complexity. Recall that the allocation algorithm, that is, Algorithm 2, consists of a loop that includes Algorithm 1 and a nested loop. If $|\mathcal{T}| = m$, then the complexity of the outer loop is $\mathcal{O}(m)$ in the worse case. For Algorithm 1, the complexity is $\mathcal{O}(m)$ in the worse case. Thus, the complexity of Algorithm 2 is $\mathcal{O}(m^2)$.

5.2. Experimental Setup. Setting up experimental environments for the crowdsourcing workflow is expensive and difficult with limited resources. Thus, the simulation tool, WorkflowSim, is utilized to achieve our goals. The WorkflowSim is an extension based on the simulation tool, CloudSim [37]. This tool is used to simulate workflow management and scheduling in a dynamic cloud environment. The WorkflowSim has better accuracy and wider support than existing solutions in terms of supporting DAG and simulating scientific workflows in distributed environments [38, 39].

The proposed algorithms are implemented on the basis of the WorkflowSim. The class CondorVM is extended to simulate a worker, in which the name of the corresponding class is Worker. A machine is also implemented on the basis of the class CondorVM, in which the codes are the Machine class. The class Job is extended to simulate a crowd task, and the corresponding class is the class CrowdTask. The class CrowdWorkflowParser, which is used to read real workflow datasets from external files to generate suitable crowd tasks, is rewritten. The package org.workflowsim.crowdsourcing.scheduling includes the above algorithms. The basic crowdsourcing framework is implemented on these classes, that is, CrowdWorkflowPlanner, CrowdWorkflowClusteringEngine, CrowdBasicClustering, CrowdReclusteringEngine, CrowdWorkflowEngine, CrowdWorkflowScheduler, and CrowdFailureGenerator, in the context of crowdsourcing workflows. Readers can find the code at the following URL: https://github.com/hhluci/WorkflowSim-1.0.

Input: given workflow: $\mathcal{G} = \{\mathcal{T}, \mathcal{E}\}$
Output: {totalCost, overallFairness, completedTime}
(1) for $t_i \in \mathcal{T}$, initialize p_0, p_1, $\mathcal{S}$, q_1 and q_2
(2) **While** $|\mathcal{T}| > 0$ **Do**
(3) Get $\mathcal{P}_\ell = (\text{in}, t_1, t_2, \ldots, t_n, \text{out})$ by calling findLongestPath ()
(4) **If** $t_n \cdot q_3 \leq \Theta$ **Then**
(5) Let current_node = in
(6) Let S = ssucc(current_node)
(7) **For** $i = 1$ to $|S|$ **Do**
(8) Get the list of nominees for t_i
(9) Group nominees by similarity
(10) Set $t_i \cdot v$ for each group
(11) Get candidates by multicasting k nominees by group
(12) Calculate the overallFairness
(13) Calculate the indicator $\Delta(w)$ or $\Delta(m)$ for all candidates
(14) Sort the list by $\Delta(w)$ or $\Delta(m)$
(15) Select the top candidate for t_i
(16) Remove t_i from $\mathcal{G}$
(17) totalCost += $t_i \cdot v$
(18) completedTime = $t_i \cdot q_3$
(19) **End For**
(20) Update parameters of remaining tasks after finishing S
(21) **End If**
(22) **End While**
(23) return {totalCost, overallFairness, completedTime}

Algorithm 2: TaskAllocation().

Five real workflow applications in the Pegasus project are chosen to perform two sets of experiments.

CyberShake: this workflow is used by the Southern California Earthquake Center to classify earthquake alarms [40]

Epigenomics: this workflow uses DNA sequence lanes to generate multiple lanes of DNA sequences [41]

Montage: this workflow is created by the NASA/IPAC stitches to gather multiple input images for the creation of custom mosaics of the sky [42]

Inspiral: this workflow is used to generate and analyze gravitational waveforms from the data collected during the coalescing of compact binary systems [43]

Sipht: this workflow is used in bioinformatics to search for small nontranslated bacterial regulatory RNAs [44]

The mentioned datasets are processed to meet the current requirements after obtaining them from external storage.

5.3. Experimental Results. In the first set of experiments, the runtime, fairness, and cost are observed as a function of the workflow size. The experiments are performed on the dataset, CyberShake workflows, with 30, 50, 100, and 1000 tasks. The least reward v of a worker and a machine is assigned using a normal distribution. The mean is set to 0.3 cents, and the upper and lower limits are 0.1 and 0.6 cents, respectively [45]. The parameters in Definitions 2 and 3 are initialized as follows: $w_i \cdot p$ and $m_i \cdot p$ are initialized using a normal distribution. The mean is set to 0.5, and the upper and lower limits are 0.0 and 1.0, respectively. The parameters $w_i \cdot \mathcal{S}$ and $m_i \cdot \mathcal{S}$ are randomly initialized as a Boolean vector with 10 elements. These workloads are scheduled on a heterogeneous cloud infrastructure with 14 physical hosts, 10 virtual machines, and 10 virtual workers, which are created using different types of virtual machines in terms of MIPS, RAM, and BW. The parameters in Definition 4 are initialized as follows: The parameter p_0 is assigned by a uniform distribution. q_1 is initialized by a uniform distribution, and the upper and lower limits are 0 and 24 h, respectively. The parameter q_2 comes from the data file, and ρ is set to 1.0. The parameter k is determined by the member number of the similar group.

Figure 2 presents the related results. In the figures, the x-axis represents the number of tasks to allocate, whereas the y-axis represents different metrics. Different series represent different allocation algorithms. Figure 2(a) plots the running time as a function of the number of tasks. As shown in Figure 2, the difference between the algorithms is negligible in terms of runtime on the small datasets. Figures 2(b) and 2(c) are discussed together because they are complementary and plot the overall fairness and the total cost, respectively, as a function of the number of tasks. As shown in Figure 2(b), the F-Aware and the Max_Fairness are better than the Random and the Min_Cost. The advantages are highlighted at the large dataset. The F-Aware is lower than the Max_Fairness. For 1000 tasks, the Max_Fairness reaches 0.98, whereas the F-Aware is 0.06 less than its competitor. The F-Aware immensely surpasses the Max_Fairness, as shown in Figure 2(c). The Max_Fairness costs 376 cents to

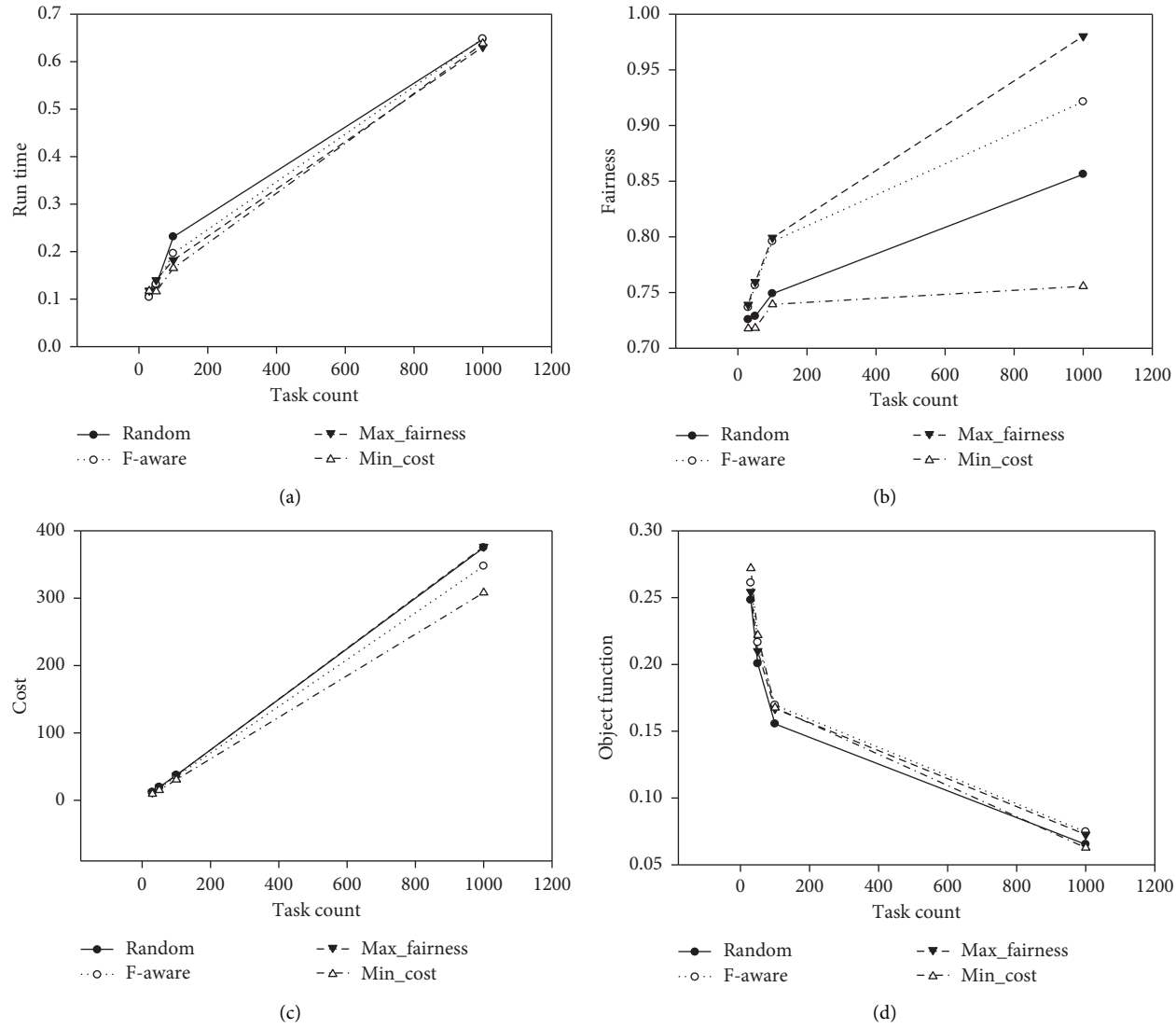

Figure 2: Comparison of the same dataset. (a) Runtime. (b) Fairness. (c) Cost. (d) Object Function.

complete a workflow with 1000 tasks, whereas the F-Aware costs only 347 cents. One hundred tasks can be completed for 29 cents because the average cost of a task is 0.3 cents. This finding exhibits the remarkable difference between the two algorithms on the metric.

Figure 2(d) presents the performance of algorithms in terms of the objective, equation (9) as a function of fairness and cost. In the experiment, ρ is set to 1.0 to observe the balanced outcome of fairness and cost. Notably, the object function decreases with the increase in task count in the figure. The cost increases faster than the fairness with increasing task count because the increase in the cost results in the decrease of the object function. As shown in Figure 2, the F-Aware performs better than the Max_Fairness in terms of the objective. The maximum gap between the two algorithms is 1%. The Random and the Min_Cost allocation algorithms show low balance.

The Max_Fairness is the best competitor of F-Aware if fairness is considered. Only the Min_Cost can match the F-Aware when only the cost is considered. The F-Aware is the best option when the two factors are considered.

In the second set of experiments, the same metrics are used as the first set of experiments on the basis of different datasets with the same scale. The CyberShake_1000, the Epigenomics_997, the Inspiral_1000, the Montage_1000, and the Sipht_1000 are applied. Figure 3 presents the related results. Figure 3(a) shows the runtime in different workflows. The difference between the algorithms is small. The F-Aware performs slightly better than the Max_Fairness. The Min_Cost performs poorly in terms of runtime. Figure 3(b) illustrates the fairness in different workflows. The F-Aware and the Max_Fairness are superior to the Min_Cost and the Random on all datasets. The Max_Fairness performs approximately 6% better than the F-Aware on other datasets. However, one requester may pay more by using the Max_Fairness than by using the F-Aware, which is shown in Figure 3(c). Figure 3(d) confirms that the F-Aware can find the tradeoff between fairness and cost, thereby increasing

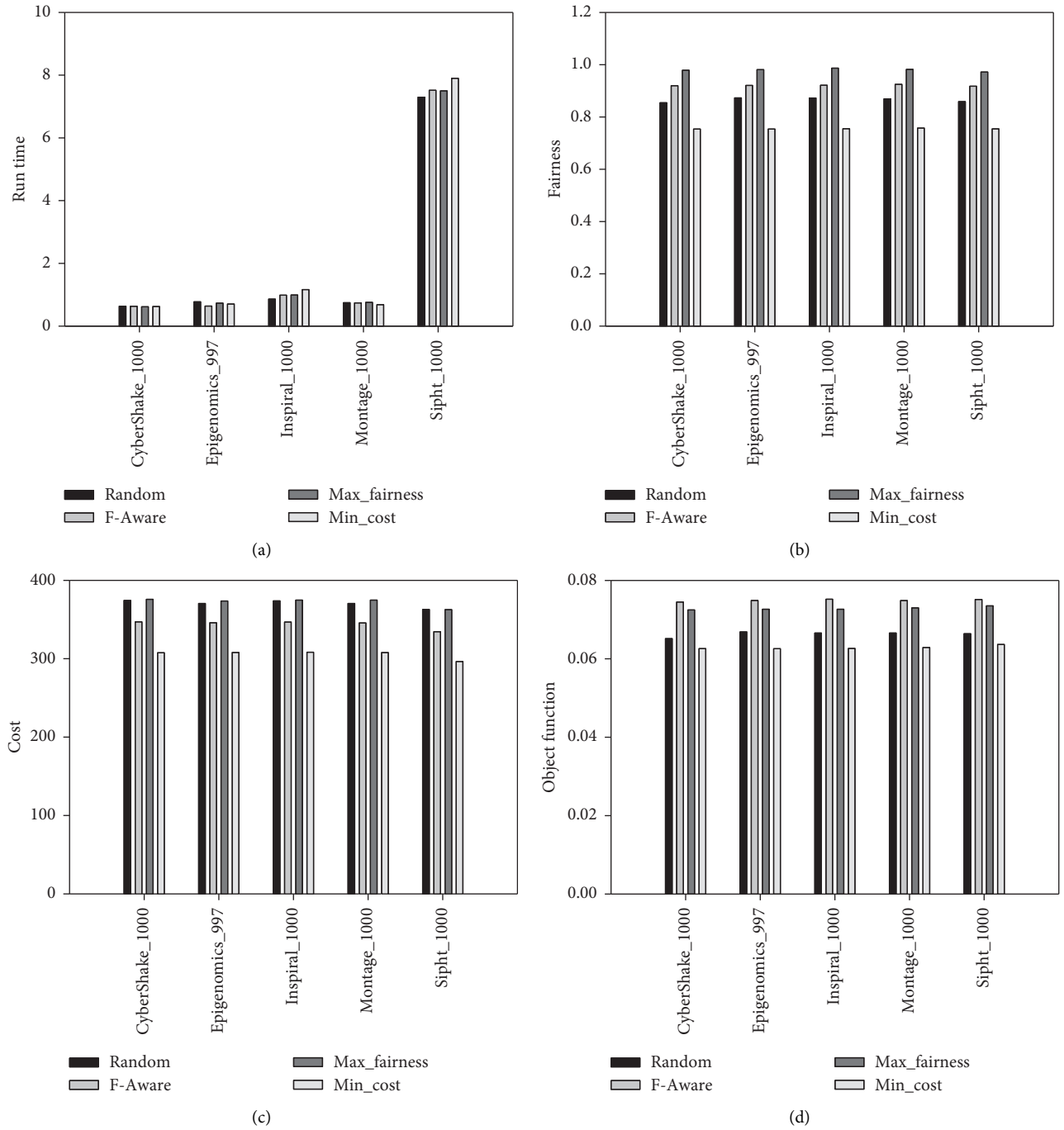

Figure 3: Comparison of different datasets. (a) Runtime. (b) Fairness. (c) Cost. (d) Object Function.

fairness while maintaining low cost. In summary, the F-Aware is better than other algorithms in terms of fairness and cost.

6. Conclusion

Requester-centric task allocation algorithms are discriminatory for workers. Strategies are presented to bridge such a gap in this study. This study is valuable in addressing crowdsourcing problems, but the strategies presented here are directed to solve the crowdsourcing workflow. A combined optimal function is designed to maximize the overall fairness and minimize the total cost. In the two-phase allocation model, a set of nominees are identified using the availabilities of the participants for each task. The task is allocated to nominees by using a batch progressive strategy for determining the candidates. The batch size is determined by the size of the similar group. The problem can be reduced to a multiobjective optimal problem when the candidates are identified for one task. Four heuristic strategies are designed to solve the problem to avoid the computational complexity trouble. These strategies are tested on the basis of the

WorkflowSim by using scientific workflow benchmarks. The evaluation has shown that the F-Aware outperforms other optimal solutions in finding the tradeoff between fairness and cost. The F-Aware performs better than the best competitor algorithm, Max_Fairness.

The experimental evaluation shows that the tradeoff between fairness and cost can be found for a crowdsourcing workflow. Heuristic strategies can effectively increase the overall fairness and maintain a proper cost. The long-term effects of fairness-aware task allocation strategies in real crowdsourcing workflow platforms should be evaluated in future studies.

Data Availability

The data used to support the findings of this study are included within the article.

Disclosure

Any opinions, findings, conclusions, and recommendations expressed in this publication are from the authors and do not necessarily reflect the views of the sponsors.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors would like to thank everyone, including Song Wenai, Liu Zhongbao, Cai Xingwang, and other members of the research group, for their suggestions and ideas. This work was supported by the Natural Science Foundation of Shanxi Province of China under Grant 201801D121151.

References

[1] Y. Kryvasheyeu, H. H. Chen, N. Obradovich et al., "Rapid assessment of disaster damage using social media activity," *Science Advances*, vol. 2, no. 3, pp. 1–11, 2016.

[2] D. E. Boubiche, M. Imran, A. Maqsood, and M. Shoaib, "Mobile crowd sensing-taxonomy, applications, challenges, and solutions," *Computers in Human Behavior*, vol. 101, no. 10, pp. 352–370, 2019.

[3] T. Abbas, V.-J. Khan, and P. Markopoulos, "Investigating the crowd's creativity for creating on-demand IoT scenarios," *International Journal of Human-Computer Interaction*, vol. 36, no. 11, pp. 1022–1049, 2020.

[4] L. V. Ahn, B. Maurer, C. McMillen et al., "reCAPTCHA: human-based character recognition via web security measures," *Science*, vol. 321, no. 5895, pp. 1465–1468, 2008.

[5] T. P. Robinson and M. E. Kelley, "Renewal and resurgence phenomena generalize to amazon's mechanical turk," *Journal of the Experimental Analysis of Behavior*, vol. 113, no. 1, pp. 206–213, 2020.

[6] A. M. Brawley and C. L. S. Pury, "Work experiences on MTurk: job satisfaction, turnover, and information sharing," *Computers in Human Behavior*, vol. 54, no. 08, pp. 531–546, 2016.

[7] N. Franke, P. Keinz, and K. Klausberger, ""Does this sound like a fair deal?": antecedents and consequences of fairness expectations in the individual's decision to participate in firm innovation," *Organization Science*, vol. 24, no. 5, pp. 1495–1516, 2013.

[8] F. Basik, B. Gedik, H. Ferhatosmanoglu, and K.-L. Wu, "Fair task allocation in crowdsourced delivery," *IEEE Transactions on Services Computing*, p. 1, 2018.

[9] P. Michelucci and J. L. Dickinson, "The power of crowds," *Science*, vol. 351, no. 6268, pp. 32-33, 2016.

[10] J. C. R. Licklider, "Man-computer Symbiosis," *IRE Transactions on Human Factors in Electronics*, vol. HFE-1, no. 1, pp. 4–11, 1960.

[11] J. Hendler and T. Berners-Lee, "From the semantic web to social machines: a research challenge for AI on the world wide web," *Artificial Intelligence*, vol. 174, no. 2, pp. 156–161, 2010.

[12] J. Howe, M. Tech, and P. Reviews, "The rise of crowd sourcing," *Wired Magazine*, vol. 14, no. 06, pp. 1–6, 2006.

[13] A. Kulkarni, M. Can, and B. Hartmann, "Collaboratively crowdsourcing workflows with turkomatic," in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, pp. 1003–1012, Seattle, WA, USA, February 2012.

[14] R. Faullant, J. Fuller, and K. Hutter, "Fair play: perceived fairness in crowdsourcing communities and its behavioral consequences," *Academy of Management Proceedings*, vol. 2013, no. 1, 2013.

[15] A. Kittur, B. Smus, R. Kraut et al., "CrowdForge: crowdsourcing complex work," in *Proceedings of Annual Acm Symposium on User Interface Software & Technology*, pp. 1801–1806, Santa Barbara, CA, USA, October 2011.

[16] G. Little, L. B. Chilton, M. Goldman et al., "TurKit: tools for iterative tasks on mechanical Turk," in *Proceedings of the ACM SIGKDD Workshop on Human Computation*, pp. 29-30, Paris, France, June 2009.

[17] G. Little, L. B. Chilton, M. Goldman et al., "Turkit: human computation algorithms on mechanical turk," in *Proceeding of the 23rd Annual ACM Symposium on User Interface Software and Technology*, pp. 57–66, New York, NY, USA, October 2010.

[18] P. Dai, N. Mausam, and D. S. Weld, "Decision-theoretic control of crowd-sourced workflows," US Patent US20110313933 A1, 2011.

[19] C. H. Lin, M. Mausam, and D. S. Weld, "Dynamically switching between synergistic workflows for crowdsourcing," in *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pp. 87–93, Toronto, Canada, July 2012.

[20] M. S. Bernstein, G. Little, R. C. Miller et al., "Soylent: a word processor with a crowd inside," in *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology*, pp. 331–322, New York, NY, USA, October 2010.

[21] Q. Zheng, W. Y. Wang, Y. Yu et al., "Crowdsourcing complex task automatically by workflow technology," in *Proceedings of the International Workshop on Management of Information, Process and Cooperation*, pp. 17–30, Singapore, October 2017.

[22] T. H. Xiong, Y. Yu, M. L. Pan et al., "SmartCrowd: a workflow framework for complex crowdsourcing tasks," in *Proceedings of the International conference on Business Process Management*, pp. 387–398, Vienna, Austria, September 2019.

[23] S. J. Wu, H. L. Sun, P. P. Chen et al., "Service4Crowd: a service oriented process management platform for crowdsourcing," in *Proceedings of the Computer Supported Cooperative Work and Social Computing*, pp. 37–40, New Jersey, NY, USA, November 2018.

[24] H. Rahman, S. B. Roy, S. Thirumuruganathan et al., "Task assignment optimization in collaborative crowdsourcing," in *Proceedings of the IEEE International Conference on Data Mining*, pp. 949–954, Atlantic City, NJ, USA, November 2015.

[25] C. J. Ho and J. W. Vaughan, "Online task assignment in crowdsourcing markets," in *Proceedings of the Twenty-Sixth AAAI Conf. Artificial Intelligence*, pp. 45–51, Toronto, Canada, July 2012.

[26] R. Khazankin, H. Psaier, D. Schall et al., "QoS-based task scheduling in crowdsourcing environments," in *Proceedings of the 9th international conference on Service-Oriented Computing*, pp. 297–311, Paphos, Cyprus, January 2011.

[27] D. R. Karger, S. Oh, and D. Shah, "Budget-optimal task allocation for reliable crowdsourcing systems," *Operations Research*, vol. 62, no. 1, pp. 1–24, 2011.

[28] I. Boutsis and V. Kalogeraki, "Crowdsourcing under real-time constraints," in *Proceedings of the 27th IEEE Int. Symp. Parallel & Distributed Processing*, pp. 753–764, Cambridge, MA, USA, January 2013.

[29] I. Boutsis and V. Kalogeraki, "On task assignment for real-time reliable crowdsourcing," in *Proceedings of the 34th IEEE Int. Conf. Distributed Computing Systems*, pp. 1–10, Madrid, Spain, July 2014.

[30] R. Khazankin, B. Satzger, and S. Dustdar, "Optimized execution of business processes on crowdsourcing platforms," in *Proceedings of the 8th IEEE International Conference on Collaborative Computing*, pp. 443–451, Pittsburgh, PA, USA, October 2012.

[31] W. J. Tang, R. Chen, and S. K. Guo, "A novel approach to publishing tasks for collaboratively crowdsourcing workflows," *International Journal of Software Engineering and Knowledge Engineering*, vol. 29, no. 6, pp. 763–790, 2019.

[32] X. Y. Gan, Y. Q. Li, W. W. Wang et al., "Social crowdsourcing to friend: an incentive mechanism for multi-resource sharing," *IEEE JSAC*, vol. 35, no. 3, pp. 795–808, 2017.

[33] C. C. Xu, Y. W. Li, W. Y. Zhang et al., "Towards greater perceived fairness: crowdsourcing moderation work to online deliberation participants," in *Proceedings of the CHI-Workshop: Crowd Dynamics: Exploring Conflicts and Contradictions in Crowdsourcing*, San Jose, CA, USA, May 2016.

[34] B. McInnis, D. Cosley, C. Nam et al., "Taking a hit: designing around rejection, mistrust, risk, and workers' experiences in amazon mechanical turk," in *Proceedings of the CHI*, pp. 2271–2282, ACM, San Jose, CA, USA, May 2016.

[35] C. J. Ho and J. W. Vaughan, "Online task assignment in crowdsourcing markets," in *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pp. 45–51, Toronto, Canada, July 2012.

[36] Q. C. Ye, Y. Q. Zhang, and R. Dekker, "Fair task allocation in transportation," *Omega*, vol. 68, no. 05, pp. 1–16, 2017.

[37] W. W. Chen and E. Deelman, "Workflowsim: a toolkit for simulating scientific workflows in distributed environments," in *Proceedings of the 2012 IEEE 8th International Conference on E-Science*, pp. 1–8, Chicage, IL, USA, October 2012.

[38] F. Abazari, M. Analoui, H. Takabi et al., "MOWS: Multi-objective workflow scheduling in cloud computing based on heuristic algorithm," *Simulation Modelling Practice and Theory*, vol. 93, no. 10, pp. 119–132, 2019.

[39] R. N. Calheiros, R. Ranjan, A. Beloglazov et al., "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software*, vol. 41, no. 1, pp. 23–50, 2011.

[40] H. Magistrale, S. Day, R. W. Clayton et al., "The scec southern California reference three-dimensional seismic velocity model version 2," *Bulletin of the Seismological Society of America*, vol. 90, no. 6B, pp. S65–S76, 2000.

[41] Usc Epigenome Center, accessed (2020), http://epigenome.usc.edu.

[42] J. C. Jacob, D. S. Katz, T. Prince et al., "The montage architecture for grid-enabled science processing of large, distributed datasets," in *Proceedings of the 2004 Earth Science Technology Conference*, Pasadena, CA, USA, June 2004.

[43] D. A. Brown, P. R. Brady, A. Dietz et al., "A case study on the use of workflow technologies for scientific analysis: gravitational wave data analysis," *Workflows for E-Science*, pp. 39–59, Springer, Berlin, Germany, 2007.

[44] J. Livny, H. Teonadi, M. Livny et al., "High-throughput, kingdom-wide prediction and annotation of bacterial noncoding rnas," *PLoS One*, vol. 3, no. 9, 2008.

[45] K. Goel, S. Rajpal, and M. Mausam, "Octopus: a framework for cost-quality-time optimization in crowd sourcing," in *Proceedings of the HCOMP*, Québec City, Canada, October 2017.