Proceedings, The Fourth AAAI Conference on
Human Computation and Crowdsourcing

Understanding Crowdsourcing Workflow: Modeling and Optimizing Iterative and Parallel Processes

Shinsuke Goto and **Toru Ishida** and **Donghui Lin**
Kyoto University
s-goto@ai.soc.i.kyoto-u.ac.jp, {ishida,lindh}@i.kyoto-u.ac.jp

Abstract

The advantages and disadvantages of different crowdsourcing workflow structures have been analyzed. Existing studies on crowdsourcing workflow mainly focused on the quality control of the tasks using iterative and parallel processes. On the other hand, the characteristics of workflow considering the various task and crowdsourcing environments are not yet fully analyzed. Therefore, we face two difficulties in making use of workflow: the workflow optimization and the prior quality estimation. This research proposes the crowdsourcing workflow model for the set of improvement tasks, considering the ability distribution of crowdsourcing workers, improvement difficulty of the task, and the preference of the requester. In addition, we show the optimal workflow can be found by the search algorithm on the proposed model. The result of optimization can be used for both constructing the best workflow and estimating the quality of task performance. Experimental results under various conditions indicate that the degree of parallelism of the optimal workflow increases with the variance of worker ability. Also, iterative processes should be used when the average ability of the workers trends away from the middle level. These results include the existing research and therefore the model presented is useful in understanding crowdsourcing workflows.

Introduction

Today, crowdsourcing is being used for a variety of open-ended tasks such as writing, design and translation. However, when performing the open-ended tasks, the quality of the result from a single worker is not guaranteed because of the various abilities of crowdsourcing workers. To ensure the quality, requesters create a workflow, in which the result of one crowd worker is incrementally refined by other workers. Although the importance of crowdsourcing workflow has already been addressed (Kittur et al. 2013), the general characteristics of such workflows are not yet well comprehended.

Until recently, the crowdsourcing workflow for collaboration among workers mainly lies on two processes: the iterative process and the parallel process. In an iterative process, the result of a task conducted by one worker is iteratively improved by other workers (Little et al. 2010a). In a parallel process, the same task is executed by multiple workers and the final result is selected by voting or some other means (Kittur et al. 2011; Luther et al. 2015). Analyses of iterative and parallel processes in crowdsourcing workflows have yielded two main findings: (1) the variety of crowd workers is important in a parallel process (Luther et al. 2015), and (2) prior results can have a negative effect on quality by leading subsequent workers down the wrong path in an iterative process of tasks with high difficulty (Little et al. 2010a). However, previous research mainly focused on workflow analysis for special tasks, and did not provide a comprehensive understanding of crowdsourcing workflows. Even though there exist several studies on workflow optimization (Dai, Lin, and Weld 2013; Tran-Thanh et al. 2015), these works focused on the optimization on the fixed workflow structure such as the number of iteration or parallelism.

To make more use of crowdsourcing workflow, we should solve two main problems. One is the relationship between the characteristics of task and workflow. The optimal process can be changed by the category of tasks. However, parameters to determine the process are not yet found. The other problem is the prior estimation of utility for task requesters. The estimated utility can help them for deciding if they use crowdsourcing or not before the actual request.

In this paper, we aim at understanding the general characteristics of crowdsourcing workflows. We first regard a crowdsourcing workflow as an agglomeration of improvement tasks. The quality of the workflow is predicted based on the model of improvement tasks, where there are three parameters: worker's average ability, variance of worker's ability distribution, and improvement difficulty. To define the requester's utility, we consider the final quality of the workflow and the cost incurred in obtaining the result. Finally, we apply a search algorithm to find the optimal workflow based on the utility of the requester.

To achieve comprehensive understandings of crowdsourcing workflows, we conduct two experiments. First, we analyze how the optimal workflow depends on the characteristics of tasks, quality of workers, and requester preference. Then, we compare the iterative and parallel processes to show that the model can explain the results of previous studies. We will use the results to explain existing researches and to more fully understand the general characteristics of

crowdsourcing workflows.

Previous Researches on Crowdsourcing Workflow

Crowdsourcing workflow is widely used for enhancing the quality of difficult tasks. It is originally proposed to complete the tasks whose quality cannot be guaranteed by a single worker. Quality control of the classification or voting task by multiple workers can be regarded as the workflow of parallel process (Sheng, Provost, and Ipeirotis 2008). On the other hand, the iterative process of improvement is proposed to deal with open-ended tasks.

Several workflow processes for special tasks have been proposed considering the issue of quality control. Soylent uses the Find-Fix-Verify crowd programming pattern to improve worker quality; it splits word processing tasks into a series of generation and review stages (Bernstein et al. 2010). Zaidan and Callison-Burch propose a crowdsourced translation workflow, in which high quality translations are obtained by aggregating multiple translations, redundantly editing them, and then selecting the best of the bunch based on machine learning (Zaidan and Callison-Burch 2011).

Various workflow support tools have been proposed for managing the crowdsourcing of complex tasks. For example, TurKit is a toolkit for prototyping and exploring algorithmic human computation (Little et al. 2010b). CrowdForge decomposes and recomposes complex crowdsourcing tasks based on the MapReduce algorithm(Kittur et al. 2011). Turkomatic was developed for supporting task decomposition by crowd workers (Kulkarni, Can, and Hartmann 2012). CrowdWeaver is a system to visually manage complex tasks and revise task decomposition during execution (Kittur et al. 2012). Tool development for modeling and managing workflows is of interest as it actually shares our objective of enhancing the understanding of crowdsourcing workflows. Our research can be regarded as a theoretical basis for workflow design in crowdsourcing.

Some workflow optimization approaches have been proposed to attain excellent cost-quality tradeoffs for crowdsourcing tasks. Dai, Lin, and Weld proposed a POMDP-based workflow control approach for optimizing the iterative improvement process (Dai, Lin, and Weld 2013). Tran-Thanh et al. proposed an effective algorithm to solve the problem of task allocation under a budget constraint in crowdsourcing systems (Tran-Thanh et al. 2015). Different from the previous work, which mainly deals with maximizing the quality of tasks or requester utility, we focus on the comprehensive analysis of crowdsourcing workflows based on the characteristics of the tasks.

Modeling Iterative and Parallel Processes

To understand the crowdsourcing workflow, we first develop the model to estimate the utility from the workflow composed of iterative and parallel processes. A model is featured by the ability distribution of crowdsourcing workers, the improvement difficulty of the task, and the preference of the requester. We first explain the scenario where the model is used, then define the workflow, and show the method to estimate the utility for a given workflow.

Motivating Scenario

Let us suppose a scenario wherein a requester wants to use crowdsourcing to execute an open-ended translation task. There are numerous workers available on the crowdsourcing platform. However, the ability of a worker remains unknown until the task is executed by that worker. Since translation quality cannot be ensured if only one worker is used, it is necessary to consider a translation workflow that consists of several improvement tasks performed by multiple workers. In each iteration, a worker improves the result which is the best result among those output in the previous iteration. However, the requester wants to obtain the best result while balancing cost off against quality. Also, they want to predict the tradeoff of quality and cost for decision-making. Therefore, it is necessary to model crowd workers, tasks and requester utility towards a better understanding of general crowdsourcing tasks.

Worker

High worker ability should yield high quality results. To simplify our model, we assume that the quality of a task execution result is determined uniquely by the ability of the worker who executed the task. Since the ability is unknown before the task is undertaken, we use a beta distribution to model the distribution of worker ability. Its probability density function $f(x|a,v)$ is given by Equation (1).

$$f(x|a,v) = \mathrm{Beta}(\frac{a}{\min(a,1-a)v}, \frac{(1-a)}{\min(a,1-a)v}) \quad (1)$$

Here $a \in (0,1)$ is the normalized value of the average ability of the workers in the crowdsourcing platform. $v \in (0,1)$ is a parameter that determines the variance in worker ability. If v is near 0, then the variance approaches 0. If v is near 1, then the variance is approaches the highest variance with average a. The above model extends the previous work (Dai, Lin, and Weld 2013) by modifying a parameter that describes the variance of worker ability.

Workflow

An open-ended task consists of iterations of improvement tasks, which is called an iterative process where the high-quality result is acquired by iteratively improving the prior work by a new worker. However, there is also a situation that multiple workers improve the same task in parallel, which is called a parallel process. Examples of improvement tasks implemented as iterative and parallel processes are reported in (Little et al. 2010a).

We formally define a workflow as $w = (p_1, \ldots, p_n)$, where n is the number of improvement tasks in the iterative process and $p_i (1 \leq i \leq n)$ is the number of workers that execute the ith improvement task in parallel. As a result, the total number of the workers in the workflow is given by $m = \sum_{i=1}^{n} p_i$.

After each iteration, the best result will be automatically selected. If none of the results have better quality than the input of the improvement task, the input will be selected as the best result.

Improvement Task

Different types of tasks have different difficulties. For each task, we set a parameter $d \in [0, 1]$ to denote its improvement difficulty. If the improvement difficulty d is 0, then the improvement task is extremely easy. In contrast, the quality of a task with $d = 1$ is extremely difficult to improve. For example, if the task is to describe a picture, then d is near 0 since it is easy for a new worker to improve the quality by adding information. If the task is to improve an illustration, then d might approach 1 since it is always extremely difficult to improve the output of another designer. For most of other types of tasks like translation improvement, the value of d lies between 0 and 1. Given the improvement difficulty d of a task, we use function $q'(a, q)$ to define the quality of the result after executing the improvement task once, where a is the ability of the worker and q is the quality of the input result of the current improvement task.

$$q'(a, q) = q + (1 - q)a - q(1 - a)d \tag{2}$$

The above equation is the sum of three parts. The first part denotes the original quality q of the input result for the current improvement task. The second part means the increase in quality after the execution of the improvement task. The third part shows the penalty in quality if the improvement fails. Let us explain the second part and the third part in more detail. If the original quality of the input result is q, then the room left for quality improvement is $1 - q$. The second part, $(1-q)a$, means that the improvement is proportional to the worker's ability a. On the other hand, $q(1 - a)$ denotes the possibility of improvement failure. When the original quality is high or the worker's ability is low, the possibility of improvement failure is high. The reason why the improvement difficulty d is multiplied in the third part is that it is more probable that quality decreases if d is larger, i.e., the task has higher improvement difficulty. In the situation that the improvement task is executed by just one worker, the expected value of the quality after executing the improvement task is $q'(a, q)$ since the expected value of the worker's ability is a.

Next, we explain the quality improvement for adding the parallel process. If two or more workers perform the improvement task simultaneously, the result with maximum quality will be selected according to the assumption. Therefore, the quality of result is equal to that performed by the worker with maximum ability in the iteration. Here, we note p as the number of workers participated in improvement task in the iteration. The maximum ability among p workers ($a_p^{\max}$) are estimated by the average of the maximum distribution (Equation (4)). Here, $F(x|a, v)$ is the cumulative density function for $f(x|a, v)$. Also, $I_x(y, z)$ is regularized beta function calculated by Equation (3).

$$I_x(y, z) = \frac{\int_0^x t^{y-1}(1 - t)^{z-1}dt}{\mathrm{Beta}(y, z)} \tag{3}$$

$$\begin{aligned} a_p^{\max} &= \int_0^1 xF(x|a, v)^p dx \qquad (4)\\ &= [xF(x|a, v)]_0^1 - \int_0^1 F(x|a, v)^p dx \\ &= 1 - \int_0^1 I_x(\frac{a}{\min(a, 1 - a)v}, \frac{(1 - a)}{\min(a, 1 - a)v})^p dx \end{aligned}$$

Using $a_p^{\max}$ as a, the quality obtained by the parallel process with p workers will be $q'(a_p^{\max}, q)$.

Utility

The utility of the requester in executing workflow U is used as the objective function of workflow optimization. In previous researches, the utility of a workflow is calculated based on the quality of the task and the cost of the execution (Dai, Lin, and Weld 2013; Kamar, Hacker, and Horvitz 2012). We define utility as the weighted sum of quality Q and cost C (Yoon and Hwang 1995). The preference of the requester is denoted by the weight of quality, β, that is used for calculating the utility. Thus, the weight of cost is $1 - \beta$.

$$U = \beta Q + (1 - \beta)C \tag{5}$$

$Q \in [0, 1]$ can be acquired from the predicted quality of workflow w. The cost, $C \in [0, 1]$, is the normalized value given by Equation (6), where m is the number of workers and M is the predefined maximum number of workers. Here we count only the number of the improvement tasks. Therefore, the iterative and parallel processes do not affect the total cost.

$$C = \frac{M - m}{M} \tag{6}$$

Optimal Workflow Search

Based on the process model introduced in previous section, we can predict the utility of a given workflow. Since the number of possible workflows is large (if the number of workers is n, there are 2^n possible workflows), it is necessary to consider an efficient search approach for workflow optimization. We propose a search algorithm that extracts the maximum expected value of utility in a limited search space.

The Search Algorithm

Utility is calculated from cost and quality. We assume that the cost increases in proportional to the number of crowd workers. Therefore, utility will monotonically decrease as the worker number increases when the quality is fixed. On the other hand, the quality will monotonically increase with the worker number. Although there might be occasional failures in the improvement tasks, it is assumed that the result with better quality is selected when comparing the input and output of an improvement task. Therefore, the increase of the worker number does not lead to a drop in quality. From the above assumptions, we can see that the utility will become low if the number of workers is increased excessively since quality always has an upper bound. That is why there always exists an optimal workflow that can maximize the utility.

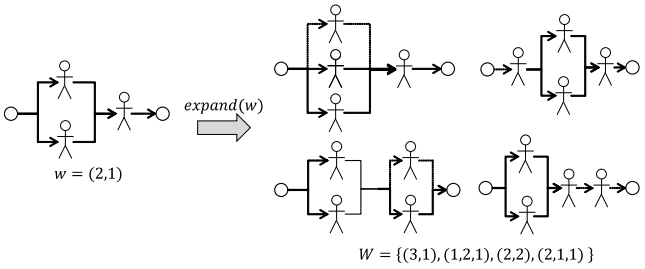

Figure 1: Explanation of $expand$. Given a workflow $w = (2, 1)$, $expand(w)$ returns the set of workflows $W = \{(3, 1), (1, 2, 1), (2, 2), (2, 1, 1)\}$

The algorithm proposed to identify the optimal workflow is written as Algorithm 1. In the algorithm, the state space consists of workflows, each of which is regarded as a state. The initial state is a workflow consists of just one improvement task by one crowd worker, and is denoted as $w = (1)$; it is stored in the state set $OPEN$. The state space is searched by expanding the contents of state set $OPEN$. The expansion process $expand$ is written as Algorithm 2. It uses workflow w as an argument and returns a set of workflows, W, that includes all possible workflows generated by adding one crowd worker to workflow w. Figure 1 shows the example of function $expand$. The function $expand$ will return workflow set $W = \{(3, 1), (1, 2, 1), (2, 2), (2, 1, 1)\}$ if $w = (2, 1)$ is the argument. W is the set of all possible workflows that can be obtained by adding one worker from the workflow $w = (2, 1)$. The function $utility$ in Algorithm 1 uses workflow w as an argument and returns the predicted utility. The search algorithm stores only w' that is in the expanded set of w and has higher utility than workflow w in $OPEN$, which means that the search starts from the center of the crater and stops at the crater rim. In other words, our proposed model is designed to avoid the horizon effect in the state space where workflows are regarded as states.

Algorithm 1 Searching Optimal Workflow $search$

```
w /* workflow */
utility(w) /* utility function for workflow w */
s /* current best workflow */
u /* utility value of the current best workflow */
Closed /* set of workflows already expanded */
Open /* set of workflows to be expanded */
s ← (1)
u ← utility(s)
Open ← {s}
Closed ← {}
while Open ≠ null do
    Select w ∈ Open
    Open ← Open − {w}
    Closed ← Closed ∪ {w}
    for all w' ∈ expand(w) do
        if w' ∉ Closed and utility(w') ≥ utility(w) then
            Open ← Open ∪ {w'}
            if utility(w') ≥ u then
                s ← w'
                u ← utility(w')
            end if
        end if
    end for
end while
return s
```

Algorithm 2 Expanding Workflow $expand$

```
Input: w
p_i /* number of workers that execute the ith improvement task in parallel */
n /* number of iteration */
w = (p_1, ..., p_n) /* workflow */
W = {w_1, ..., w_m} /* the set of created workflows by expansion of w */
m /* number of workflows created by expansion of w */
W ← {}
W ← W ∪ {(1, p_1, ..., p_n)}
for i = 1 to n do
    W ← W ∪ {(p_1, ..., (p_i + 1), ..., p_n)}
    W ← W ∪ {(p_1, ..., p_i, 1, ..., p_n)}
end for
return W
```

Optimality

The optimality of the workflow search algorithm (Algorithm 1) for crowdsourcing tasks is described below: In a crowdsourcing workflow that consists of iterative and parallel processes, let the search algorithm starts from an initial workflow state that consists of just one crowd worker and search the state space that is expanded gradually by adding once crowd worker at each epoch. The search algorithm stops when the workflow state with maximum utility has reached the optimal workflow under the given assumptions.

To prove the termination of our search algorithm, we show that the increase of utility by adding a worker monotonically decreases with higher utility. Let the expected values of the quality and cost of workflow w with m crowd workers be q and c, respectively. Assume that one crowd worker is added and the value of utility changes as follows.

First, we show that incremental quality monotonically decreases if one crowd worker is added for either iteration or parallelism. Assume that the additional crowd worker is used to increase the iteration number, the incremental quality satisfies $\Delta q = a(1 - q) - (1 - a)qd$. Here a and d are constants assuming the additional worker always has the expected quality a. In each iteration, q monotonically increases. Therefore, $a(1 - q)$ monotonically decreases and $(1 - a)qd$ monotonically increases. As a result, incremental quality Δq monotonically decreases. Also, when the additional crowd worker is set to increase the parallelism, the incremental quality Δq depends on the increment of the maximum ability of worker Δa. Since the maximum expected ability is calculated from the regularized beta function and regularized beta function satisfies $I_x(y, z) \leq 1$, therefore, Δa monotonically decreased by the increase of m. There-

Table 1: Optimal workflows w and their utilities U in different variations of v and d ($\beta = 0.5$, $a = 0.5$).

	d=1		d=0.5		d=0	
v	w	U	w	U	w	U
0.9	(2)	0.7313	(2)	0.7257	(1,1,1)	0.7875
0.7	(2)	0.7229	(2)	0.7225	(1,1,1)	0.7875
0.5	(2)	0.7144	(2)	0.7129	(1,1,1)	0.7875
0.3	(2)	0.7024	(1,1)	0.7125	(1,1,1)	0.7875
0.1	(1)	0.7	(1,1)	0.7125	(1,1,1)	0.7875

Table 2: Optimal workflows w and their utilities U in different variations of a and β ($v = 0.5$, $d = 0.5$).

	$\beta = 0.9$		$\beta = 0.5$		$\beta = 0.1$	
a	w	U	w	U	w	U
0.9	(1,2)	0.9378	(1)	0.9	(1)	0.9
0.7	(1,4)	0.8683	(1,1)	0.8025	(1)	0.88
0.5	(8)	0.7517	(2)	0.717	(1)	0.86
0.3	(3,6)	0.5819	(1,1)	0.6025	(1)	0.84
0.1	(2,5)	0.2452	(1)	0.5	(1)	0.82

fore, when repeatedly adding parallelism, Δq monotonically decreases. The increment of maximum value of beta distribution monotonically decreases as it approaches to 1, and therefore incremental quality Δq monotonically decreases by adding a worker. Secondly, incremental cost Δc is always constant if one crowd worker is added. This means the normalized cost C monotonically decreases. As utility is calculated by the weighted summation of quality and cost, the increased amount of utility decreases and turns to minus.

To summarize, the incremental utility will monotonically decrease and finally become a negative value at a certain point. Therefore, the search algorithm will stop under the given assumptions. Moreover, since the expansion of the workflow state space will stop when the incremental utility becomes a negative value, the workflow state with the maximum utility is obtained when the search algorithm stops.

The above discussion does not ensure an optimal solution when increasing crowd workers in real crowdsourcing tasks. Rather, it shows that the optimal workflow can be calculated by setting the values in the model. However, we can understand the characteristics of crowdsourcing workflows and use the knowledge in real crowdsourcing task design if the optimal solution search can be conducted efficiently.

Understanding Crowdsourcing Workflow

Based on the model and the optimization algorithm on the model, we can estimate the utility for every workflow on each parameter settings. We conducted two experiments to verify the proposed model. Each experiment analyzes the structure of optimal workflow for various parameter, and the comparison of iterative and parallel processes, respectively.

Analysis of Optimal Workflows

We use the proposed search algorithm to obtain optimal workflow w by varying the combinations of parameters in our model. In addition, we calculate the utility for each w. Settings of the parameters are as follows.

Average ability of workers $a \in (0, 1)$: varied from 0.1 to 0.9 in steps of 0.2 for each variation.

Variance of worker ability $v \in (0, 1)$: varied from 0.1 to 0.9 in steps of 0.2 for each variation.

Improvement difficulty $d \in [0, 1]$: 0 (low), 0.5 (middle) and 1 (high) for three variations.

Preference of the requester over quality β : 0.1, 0.5 and 0.9 for three variations.

We first analyze the optimal workflows obtained by the search algorithm.

Table 1 lists the result of optimal workflows and their utilities with different values of the variance of worker ability and improvement difficulty of tasks. The results in Table 1 show that optimal workflows are more parallel as the variance of worker ability increases. Moreover, the parallelism of optimal workflows also increases with the improvement difficulty. The utility of optimal workflows increases as the improvement difficulty decreases. However, the utility of optimal workflows can also increase as the variance in worker ability increases even when the improvement difficulty is high. This is because the workflow with high degree of parallelism is apt to be an optimal solution and worker ability is more obvious when the variance of worker ability is high.

Table 2 lists the results of optimal workflows and their utilities with different variations of the average worker ability and quality preference of the requester. The results in Table 2 show that the optimal workflows are most parallel when the average worker ability occupies the middle level (i.e., $a = 0.5$). Moreover, the degree of parallelism of optimal workflows falls and iterative improvement becomes more effective when the average worker ability trends away from the middle level (higher or lower). Not surprisingly, an optimal workflow has greater worker number when the quality preference of the requester is high, i.e., cost has low importance. The utility of optimal workflows is more affected by the average worker ability when the requester emphasizes quality.

The above analysis can also explain previous research results. For example, (Kittur et al. 2011) indicated that the variety of crowd workers is important in a parallel process. (Kamar, Hacker, and Horvitz 2012) suggested that it is effective to increase the number of crowd workers when the cost is low. Further, (Little et al. 2010a) showed that prior work with poor quality can have a negative effect on the overall quality of the workflow if the crowdsourcing task is difficult.

Analysis of Utility

Secondly, we compare the iterative and parallel processes to understand the characteristics of workflows. We measured the changes of quality and utility for each processes by adding workers in various combinations of the parameter settings.

Firstly, we analyze the quality of each process. Figure 2

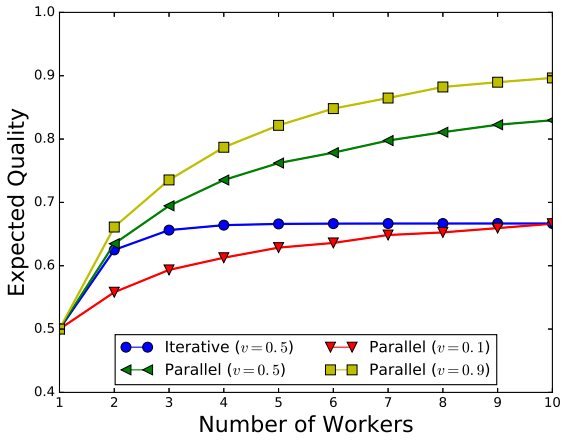

Figure 2: Effects of worker number on expected quality ($a = 0.5$, $d = 0.5$).

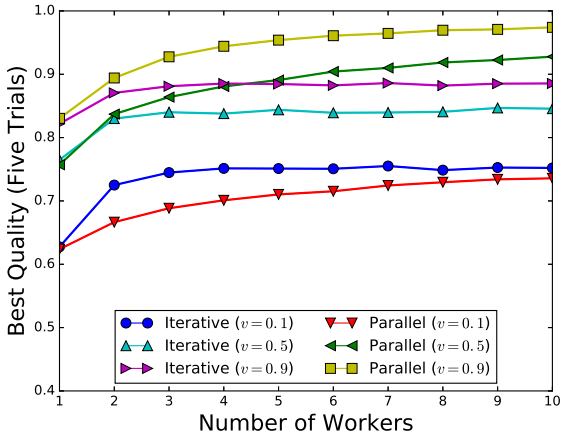

Figure 3: Effects of worker number on best quality ($a = 0.5$, $d = 0.5$).

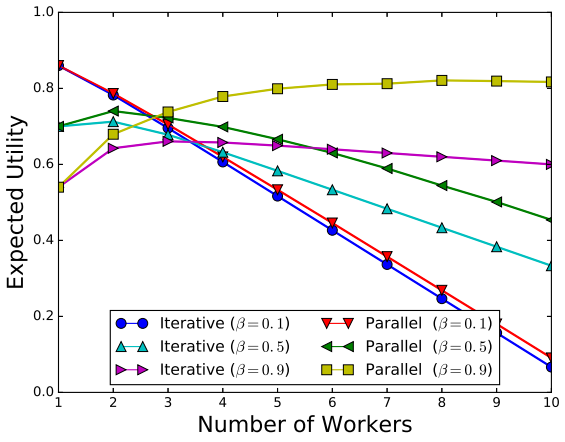

Figure 4: Effects of worker number on expected utility ($a = 0.5$, $d = 0.5$, $v = 0.5$).

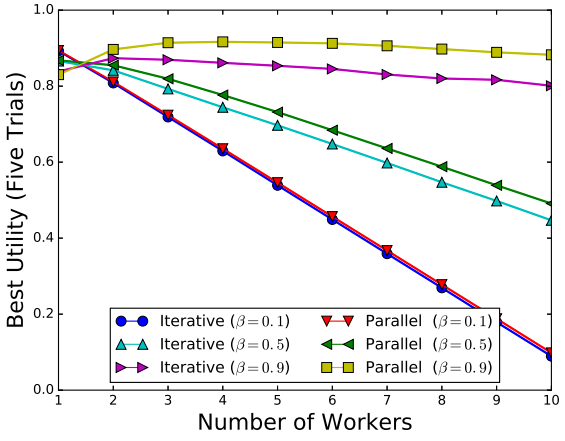

Figure 5: Effects of worker number on best utility ($a = 0.5$, $d = 0.5$, $v = 0.5$).

and Figure 3 show how the expected value of quality and best value of quality depend on worker number. The expected value of quality can be used to evaluate the quality for a large number of tasks, while the best value of quality can be used to evaluate quality when the requester demands excellent results for some special tasks. Figure 2 shows that expected value of quality increases with the number of workers, but eventually saturates. Moreover, the parallel approach can make good use of a wide variety of crowd workers (i.e., worker variance) and so it attains more quality improvement from more crowd workers than the iterative approach. Figure 3 shows that the parallel approach can attain higher quality when worker variance is high.

Secondly, we emphasize cost by varying preferences of the requester over quality β and analyze how the utility is impacted by increasing the number of workers. Figure 4 and Figure 5 plots the expected values of utility and best value of utility, respectively.

The results in Figure 4 and Figure 5 show that increasing the number of workers will not increase the utility if requester preference for cost is high. However, increasing worker number is effective if requester emphasizes quality. In general, the optimal number of crowd workers can be obtained by balancing the trade-off between cost and quality. This analysis result supports the insights from previous research, which suggests incentives for increasing crowd workers are strong when preference of the requester over cost is high (Dai, Lin, and Weld 2013; Kamar, Hacker, and Horvitz 2012).

Conclusion

We provided a comprehensive analysis of crowdsourcing workflows based on the characteristics of the tasks. A model was proposed that includes the new parameter of requester

preference along with worker's average ability, and the variance, and difficulty of task improvement. The optimal workflows found for various conditions indicate the following:

- The degree of parallelism of optimal workflows increases as the variance of worker ability increases. Also, the degree of parallelism of optimal workflows increases as task improvement difficulty increases.
- The degree of parallelism of optimal workflows is highest when the average ability of workers occupies the middle level. Moreover, the degree of parallelism of optimal workflows becomes lower, and iterative improvement becomes more effective, when the average worker ability deviates from the middle level (higher or lower).

In addition, an analysis of the quality and utility of iterative and parallel processes in various combinations of parameter settings yielded the following results:

- The parallel approach can make good use of the variety of crowd workers, and therefore parallel processes can attain higher quality when worker variance is high.
- Increasing of the number of workers will not increase utility if requester emphasizes cost (the opposite is true if the requester prefers quality). In general, the optimal number of crowd workers can be obtained by balancing cost against quality.

These results are consistent with existing works, and are useful in more fully understanding crowdsourcing workflows.

Future work includes finding the method to obtain the parameters for optimization, such as the ability distribution of workers and the improvement of difficulty. Parameter acquisition based on past performance can provide the framework for the general crowdsourcing workflow. One possible method for acquisition is the crowdsourcing based evaluation method and the estimation of parameters with online learning. In addition, real-world experiments on crowdsourcing could be executed to confirm the tendency of workflow structures. This is a rather practical direction for application, which can show a growth in understanding the crowdsourcing workflow.

Acknowledgments

This research was partially supported by a Grant-in-Aid for Scientific Research(S) (24220002, 20122016) from Japan Society for the Promotion of Science (JSPS).

References

Bernstein, M. S.; Little, G.; Miller, R. C.; Hartmann, B.; Ackerman, M. S.; Karger, D. R.; Crowell, D.; and Panovich, K. 2010. Soylent: a word processor with a crowd inside. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, 313–322. ACM.

Dai, P.; Lin, C. H.; and Weld, D. S. 2013. Pomdp-based control of workflows for crowdsourcing. *Artificial Intelligence* 202:52–85.

Kamar, E.; Hacker, S.; and Horvitz, E. 2012. Combining human and machine intelligence in large-scale crowdsourcing. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, 467–474. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Kittur, A.; Smus, B.; Khamkar, S.; and Kraut, R. E. 2011. Crowdforge: Crowdsourcing complex work. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 43–52. ACM.

Kittur, A.; Khamkar, S.; André, P.; and Kraut, R. 2012. Crowdweaver: Visually managing complex crowd work. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, 1033–1036. New York, NY, USA: ACM.

Kittur, A.; Nickerson, J. V.; Bernstein, M.; Gerber, E.; Shaw, A.; Zimmerman, J.; Lease, M.; and Horton, J. 2013. The future of crowd work. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, 1301–1318. New York, NY, USA: ACM.

Kulkarni, A.; Can, M.; and Hartmann, B. 2012. Collaboratively crowdsourcing workflows with turkomatic. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, 1003–1012. New York, NY, USA: ACM.

Little, G.; Chilton, L. B.; Goldman, M.; and Miller, R. C. 2010a. Exploring iterative and parallel human computation processes. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '10, 68–76. New York, NY, USA: ACM.

Little, G.; Chilton, L. B.; Goldman, M.; and Miller, R. C. 2010b. Turkit: Human computation algorithms on mechanical turk. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, 57–66. New York, NY, USA: ACM.

Luther, K.; Hahn, N.; Dow, S. P.; and Kittur, A. 2015. Crowdlines: Supporting synthesis of diverse information sources through crowdsourced outlines. In *Third AAAI Conference on Human Computation and Crowdsourcing*.

Sheng, V. S.; Provost, F.; and Ipeirotis, P. G. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 614–622. ACM.

Tran-Thanh, L.; Huynh, T. D.; Rosenfeld, A.; Ramchurn, S.; and Jennings, N. 2015. Crowdsourcing complex workflows under budget constraints. In *AAAI Conference on Artificial Intelligence*.

Yoon, K. P., and Hwang, C.-L. 1995. *Multiple attribute decision making: an introduction*, volume 104. Sage publications.

Zaidan, O., and Callison-Burch, C. 2011. Crowdsourcing translation: Professional quality from non-professionals. In *ACL*, 1220–1229.