Efficient Task Decomposition in Crowdsourcing

Huan Jiang and Shigeo Matsubara

Department of Social Informatics, Kyoto University, Kyoto 606-8501, Japan
jiang@ai.soc.i.kyoto-u.ac.jp, matsubara@i.kyoto-u.ac.jp

Abstract. In order to facilitate crowdsourcing-based task solving, complex tasks are decomposed into smaller subtasks that can be executed either sequentially or in parallel by workers. These two task decompositions attract a plenty of empirical explorations in crowdsourcing. However the absence of formal study makes difficulty in providing task requesters with explicit guidelines on task decomposition. In this paper, we formally present and analyze those two task decompositions as vertical and horizontal task decomposition models. Our focus is on addressing the efficiency (i.e., the quality of the task's solution) of task decomposition when the self-interested workers are paid in two different ways — equally paid and paid based on their contributions. By combining the theoretical analyses on worker's behavior and simulation-based exploration on the efficiency of task decomposition, our study 1) shows the superiority of vertical task decomposition over horizontal task decomposition in improving the quality of the task's solution; 2) gives explicit instructions on strategies for optimal vertical task decomposition under both revenue sharing schemes to maximize the quality of the task's solution.

Keywords: Task decomposition, task dependence, task difficulty, solution quality, efficient crowdsourcing.

1 Introduction

Crowdsourcing is admired as one of the most lucrative paradigm of leveraging collective intelligence to carry out a wide variety of tasks with various complexity. This very success is dependent on the potential for *decomposing* the complex tasks into smaller pieces of subtasks, such that each subtasks becomes low in complexity, requires little cognitive effort to be completed by an individual.

After decomposing a complex task into multiple small subtasks, a collective of crowds (or workers) execute the subtasks either *independently* or *dependently.* When the subtasks are structured independently, multiple workers are recruited to collaborate *in parallel*, and subtask's quality depends only upon the effort of the worker who performs it. By contrast, when there are dependencies among the subtasks, workers are organized to collaborate *sequentially*, and subtask's quality depends on efforts of multiple workers who jointly produce output. In the sequential process, subtask dependence mainly characterized in the striking feature that one worker's output is used as the starting point for the following worker, which makes the assumption that following worker can do better based

H.K. Dam et al. (Eds.): PRIMA 2014, LNAI 8861, pp. 65–73, 2014.

on quality solution provided by previous worker hold (see [4]). CrowdForge [1] and Soylent [2] delineate case studies on article writing and word processing respectively with sequential process, and they both come up with high quality final outcomes. It is worth noting that, TurKit [3] presents iterative workflow for complex tasks solving. However, such iterative workflows without task decomposition are beyond the scope of this paper.

We define two task decompositions as *horizontal task decomposition* for independent subtasks, and *vertical task decomposition* for dependent subtasks. To illustrate the concepts, we refer to the following crowdsourcing-based *proofreading* task as example, wherein an article containing three paragraphs requires spelling, style and grammar error correction. In this context, the article could be horizontally decomposed into three pieces of subtasks that each has one paragraph and be performed by one worker independently. Meanwhile, the original article could also be vertically decomposed into three sequential subtasks, as "Find-Fix-Verify" proposed by Bernstein et al. [2].

Our research focuses on the complex tasks decomposition in crowdsourcing, wherein the complex tasks can be decomposed and executed in both independent and dependent way. Of particular interest are the work that aim at analyzing workers' strategic behaviors, comparing the *efficiency* of two task decompositions, in terms of *final quality*, and finally generating explicit instructions on strategies for optimal task decomposition. Different from the works that provide efficient solutions for applications with independent subtasks (e.g., [6]), Long et al. [5] first investigates the interdependent subtask allocation in crowdsourcing systems, which has the most relevant background to our research.

We summarize our main contribution in the following. In Section 2, we formally construct the models for both vertical and horizontal task decompositions. The dependence among subtasks is formalized as the degree to which a subtask's difficulty depends on the qualities of the other subtasks. In Section 3, we rigorously analyze the strategic behaviors of the workers, and find that contribution-based sharing scheme provides more incentives for workers to exert higher efforts on difficult subtasks. In Section 4, we conduct simulations to analyze and compare the efficiency of two task decomposition. We conclude that in general, vertical task decomposition strategy outperforms the horizontal one in improving the quality of the final solution, and give explicit instructions the *optimal strategy* (i.e., arrangement of subtasks with different difficulties for final quality maximization) under vertical task decomposition situation from the task requester's point of view.

2 The Model

In this section we consider the complex task, e.g., proofreading, that can be both vertically and horizontally decomposed into N $(N > 1)$ subtasks. In both situations, N workers contribute their efforts, such as time and resources, to N subtasks respectively. The amount of effort exerted by worker i to subtask i is characterized by e_i, which is normalized to scale $[0, 1]$.

2.1 Vertical Task Decomposition

Find-Fix-Verify: The output of Find stage is the *patches* that may have spelling and grammar errors and need corrections or edits. The quality of patches could be evaluated by how well they cover the true positions [5]. The output of Fix stage is the corrections of the errors in those patches. The quality of fix task could be evaluated by the number and the average validity of the proposed corrections. Last, in Verify stage, workers accept or reject the corrections and edit to improves the proofreading result. The quality of the final solution can be viewed as the *cumulative qualities* obtained from all subtasks.

Definition 1 (Final quality). *The quality of the final solution (Q) to the complex task is the cumulative qualities of all the N decomposed subtasks, i.e.,*

$$Q(\mathbf{e}) = \sum\nolimits_{i=1}^{N} q^{i}_{vertical} \tag{1}$$

where $\mathbf{e} = (e_1, \cdots, e_N)$*, and* $q^{i}_{vertical}$ *is the quality function of subtask i.*

Since each subtask takes the output from the previous subtask as input, the quality of each subtask is also positively related to the quality of the previous subtask. Formally, the subtask quality function is governed by the following form.

Assumption 1. *The quality of subtask i's solution depends not only on the effort exerted by worker i, but also on the quality of previous subtask's solution, i.e.,*

$$q^{i}_{vertical} = f^{i}(q^{i-1}_{vertical}, e_i) \tag{2}$$

where f^i *increases with* e_i *at a decreasing rate, i.e.,*

$$\partial f^i / \partial e_i > 0 \text{ and } \partial^2 f^i / \partial e_i \partial e_i < 0 \tag{3}$$

Remark 1. The recursive definition of f^i directly implies that the quality of subtask i's solution depends also on the efforts from all the prior workers. Hence, we can rewrite Eq. (2) equivalently as $q^{i}_{vertical} = f^i(e_1, \cdots, e_i)$, and any increase in the efforts from the previous workers also leads to an improvement on the quality of subtask i, i.e., $\forall k \in \{1, \cdots, i\}$, $\partial f^i / \partial e_k > 0$. Last, note that, for the first subtask ($i = 1$), the quality function is simplified as $q^{1}_{vertical} = f^1(e_1)$.

Before we illustrate how does $q^{i}_{vertical}$ depend on $q^{i-1}_{vertical}$, we first introduce the concept of *subtask difficulty.* Take Find stage for example, articles that consist more frequent in long and compound sentences, indicate more grammatical errors, and thus require considerable effort for locating the true positions. Thus, low difficulty indicates high marginal contribution based on the same level of effort, which is formalized as follows.

Definition 2 (Subtask difficulty). *We endow subtask i with weight* $\omega_i \in (0, 1)$ *as its difficulty. Subtask i is said to be more difficult than subtask j (i.e.,* $\omega_i > \omega_j$*), iff for any effort level l*

$$f^{i}_{e_i}(e_i, e_{-i} = \mathbf{e}_{N-1}) \mid_{e_i = l} \; < \; f^{j}_{e_j}(e_j, e_{-j} = \mathbf{e}_{N-1}) \mid_{e_j = l} \tag{4}$$

where $\mathbf{e}_{N-1}$ *is the effort levels of all other* $N-1$ *workers. Furthermore,* $\sum_{i=1}^{N} \omega_i = 1$.

Now, we continue with the Find-Fix-Verify example to illustrate how does $q^{i-1}_{vertical}$ affect $q^{i}_{vertical}$ by altering the difficulty of subtask i. As Find stage, Fix stage also has its own intrinsic difficulty. Nevertheless, its difficulty can be altered by the quality of the Find task. For example, high quality of Find task due to phrase-level error location reduces the difficulty of Fix task, however, low quality due to the noisy patches can make Fix task more difficult.

Assumption 2 (Quality dependency). *The difficulty of subtask i decreases as the efforts on previous subtasks increase, i.e., $\forall k \leq i-1$, if $e_k < e'_k$, then*

$$f^{i}_{e_i}(e_{-k}, e_k) < f^{i}_{e_i}(e_{-k}, e'_k) \tag{5}$$

Remark 2. It is worth noting that an increase in the effort previous subtasks not only increases the quality of subtask i in quantity (Eq. (3)), but also, according to Eq. (5), enables greater marginal increase on subtask i's quality.

2.2 Horizontal Task Decomposition

In horizontal task decomposition, the complex task is decomposed into N subtasks with no interdependencies, which implies $\partial^2 f^i/\partial e_j \partial e_i{=}0$ $(i \neq j)$. N workers devote efforts independently to their own subtasks for individual utility maximization. The quality function of the final solution is defined as Eq. (1), i.e., $Q(\mathbf{e}) = \sum_{i=1}^{N} q^{i}_{horizontal}$, where $q^{i}_{horizontal}$ is the quality function of subtask i.

In contrast to vertical task decomposition, where each worker concentrates on a single stage of the workflow (take proofreading for example, Find, Fix or Verify), in horizontal task decomposition, each worker gives considerations to all stages. This makes the worker have to divide his effort among all stages. We assume that the effort e_i, exerted by worker i to subtask i, can be viewed as being distributed among N stages as in the vertical task decomposition situation, proportionally to the difficulties of the stages. This assumption simplifies the results without sacrificing much in terms of generality.

Assumption 3 (Horizontal subtask quality function). *The quality of the solution to subtask i only depends on the effort exerted by worker i, which is distributed among N stages proportionally to their difficulties. Hence, $q^{i}_{horizontal} = \sum_{k=1}^{N} f^k(\omega_1 e_i, \cdots, \omega_k e_i)$.*

2.3 Revenue Sharing Schemes

Definition 3 (Group-based revenue sharing). *Under the group-based revenue sharing scheme, each worker receives an equal share of the total revenue given by the task requester, i.e., $R_i = Q(e_i)/N$. Therefore, worker i's utility is $\pi(e_i) = Q(\mathbf{e})/N - c(e_i)$.*

Definition 4 (Contribution-Based revenue sharing). *Under the contribution-based revenue sharing scheme, each worker receives reward determined by his/her marginal contribution to the final task solution, i.e., $R_i(e_i) = Q_{e_i}(e_i) \cdot e_i$. Thus, worker i's utility is $\pi(e_i) = Q_{e_i}(e_i) \cdot e_i - c(e_i)$.*

Assumption 4 (Cost function). *In order to execute subtask i, worker i exerts an effort level e_i with a cost $c(e_i)$. The cost increases with the effort, and it increases at an increasing rate, i.e.,* $c_{e_i} = \frac{\mathrm{d}c}{\mathrm{d}e_i} > 0$ and $c_{e_i e_i} = \frac{\mathrm{d}^2 c}{\mathrm{d}e_i^2} > 0$.

3 Strategic Behaviors

Proposition 1. *Under both vertical and horizontal task decomposition strategies, for individual utility maximization, 1) when group-based sharing scheme is applied, workers devote higher efforts to easy subtasks than difficult subtasks. On the contrary, 2) under contribution-based sharing scheme, workers devote efforts to easy subtasks no less than difficult subtasks.*

According to Proposition 1, although contribution-based revenue sharing may provide workers with more incentives to perform difficult subtasks than group-based revenue sharing, they both indicate the fact that workers are more inclined to perform easy subtasks. This is consistent with findings in worker behavior studies in crowdsourcing, and highlights the need for task decomposition design.

4 Task Decomposition Strategy Analysis and Comparison

We construct simulation aiming at explicitly evaluating and comparing the efficiencies, in terms of the final quality (Eq. (1)), of two task decomposition strategies. It is worth noting that, besides the 2-subtask and 3-subtask situation presented in the following, we also explored the situations for $N = 4, \cdots, 9$, which not shown here due to limited space but generate the similar results. It is worth noting that by considering the existing applications such as Soylent, N=9 for the vertical decomposition seems not small.

Subtask Quality Functions. We simulate the task solving process under the assumption of subtask quality functions in the *Cobb-Douglas form*. We assume there are N workers for N subtasks, and for subtask i, the quality functions under vertical and horizontal task decompositions are respectively defined as

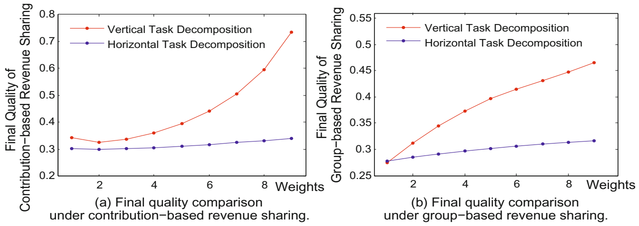
$$q^i_{vertical}(e_1, \cdots, e_i) = \prod_{k=1}^{i} e_k^{\omega_k} \quad \text{and} \quad q^i_{horizontal}(e_i) = \sum_{k=1}^{N} \prod_{r=1}^{k} (\omega_r e_i)^{\frac{\omega_r}{N}}$$

where $\omega_i \in (0,1)$ is the difficulty of subtask i, and $\sum_{i=1}^{N} \omega_i = 1$.

Cost Functions. We specify the cost function for worker i as $c(e_i) = e_i^2$.

4.1 Efficiency Comparison of Two Task Decompositions

Two-subtask situation is depicted in Fig. 1 to illustrate the efficiency difference between vertical and horizontal task decomposition strategies. We endow each of two subtasks with a weight $(\omega_1, \omega_2 \in (0,1))$, which is restricted to one decimal



(a) Final quality comparison under contribution−based revenue sharing.

(b) Final quality comparison under group−based revenue sharing.

X-axis	ω_1	ω_2
1	1	9
2	2	8
3	3	7
4	4	6
5	5	5
6	6	4
7	7	3
8	8	2
9	9	1

Note: weight = $\omega_i/10$

Fig. 1. Comparison of vertical and horizontal task decompositions. Generally, vertical decomposition outperforms horizontal decomposition in terms of final quality.

place. Then, we exhaustively examine the final quality under all the combinations of two weights, as given in the table in Fig. 1. As can be seen in Fig. 1 (a) and (b), vertical task decomposition strategy is superior to the horizontal one, under both group-based and contribution-based revenue sharing schemes.

4.2 Vertical Task Decomposition Strategy

As the qualities of the prior subtasks improve, the positive support they provide become strong, which makes the subsequent subtasks more dependent on the prior ones. Further, in the extreme situation where all subtasks have the highest qualities with all workers exert their highest efforts (e=1), the dependence among subtasks become strongest, which can be viewed as the intrinsic dependence among the sequential subtasks.

Definition 5. *Given a succession of N subtasks, the sequential dependence between subtasks $i-1$ and i is defined as*

$$\partial^2 q^i / \partial e_{i-1} \partial e_i |_{e_1 = \cdots = e_{i-1} = e_i = 1} = \omega_{i-1}\omega_i, \tag{6}$$

and the total dependence among all subtask is $\sum_{i=2}^{N} d(e_{i-1}, e_i)$.

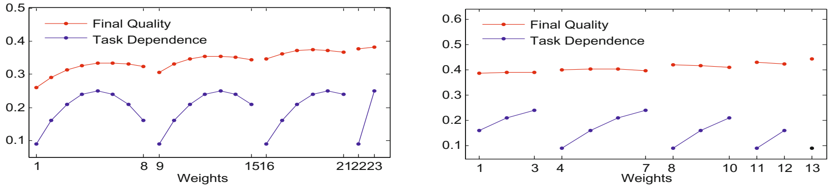
As we do for the two-subtask situation, we respectively endow 3 subtasks with weights ω_1, ω_2, ω_3, which are restricted to one decimal place as well, and then exhaustively examine all combinations of the weights, i.e., a permutation of {0.1,0.2,$\cdots$,0.9} with a restriction that the sum of three weights equals 1. As shown in Table 1, for 3-subtask situation, there are a total of 36 combinations, and they are sorted in a *lexicographic manner*, i.e., $(\omega_1, \omega_2, \omega_3)$ occurs before $(\omega_1', \omega_2', \omega_3')$ iff $\omega_1 < \omega_1'$, or $\omega_1 = \omega_1'$ and $\omega_2 < \omega_2'$, or $\omega_1 = \omega_1'$ and $\omega_2 = \omega_2'$ and $\omega_3 < \omega_3'$.

Lessons Learned on Group-Based Revenue Sharing Scheme. In Fig. 2, we explore 3-subtask situation under group-based revenue sharing scheme.

Table 1. Weight combinations for 3-subtask situation

								(a)												(b)						
X-axis	1	2	···	8	9	10	···	15	16	···	21	22	23	1	···	3	4	5	6	7	8	···	10	11	12	13
ω_1	1	1	···	1	2	2	···	2	3	···	3	4	4	4	···	4	5	5	5	5	6	···	6	7	7	8
ω_2	1	2	···	8	1	2	···	7	1	···	6	1	5	2	···	4	1	2	3	4	1	···	3	1	2	1
ω_3	8	7	···	1	7	6	···	1	6	···	1	5	1	4	···	2	4	3	2	1	3	···	1	2	1	1

Note: weight = $\omega_i/10$



(a) When the first subtask is not the most difficult subtask. (b) When the first subtask is the most difficult subtask.

Fig. 2. Efficiency estimation of vertical task decomposition strategy under group-based revenue sharing scheme. In general, series of subtasks begin with high difficulties generate high efficiency with respect to the final quality.

Lesson 1. *The highest final quality brought by the series of subtasks begin with high difficulty is superior to that brought by the series of subtasks begin with low difficulty under group-based revenue sharing scheme.* (See Fig. 2 (a).)

Lesson 2. *When the first subtask is the most difficult subtask, for the series of subtasks begin with the same difficulty, the highest task dependence given by the convex weights (i.e.,* $\omega_1 \geq \omega_2$ *and* $\omega_2 \leq \omega_3$*) leads to the highest final quality.*

As can be observed in Fig. 2 (b), in four series of subtasks begin with weight 0.5, there are two with convex weights (4.(0.5,0.1,0.4) and 5.(0.5,0.2,0.3), with task dependence 0.09 and 0.16), and the higher task dependence (x-axis value 5), gives us the highest final quality among these four series of subtasks.

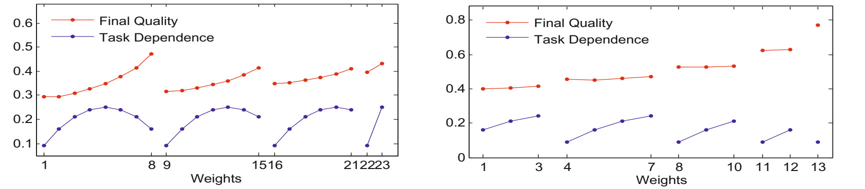
Example 2. Suppose the task requester has to choose among three very similar task decompositions, as (0.3,0.4,0,3), (0.4,0.3,0.3) and (0.4,0.4,0.2). According to Lesson 1, he would prefer the series of subtasks start with a more difficult subtask and eliminate option (0.3,0.4,0,3). Furthermore, according to Lesson 2, convex weights (0.4,0.3,0.3) is the decomposition, among all series of subtasks starts with difficulty 0.4, that leads to the highest final quality. So the task requester can construct his preference as (0.4,0.3,0.3)$\succ$(0.4,0.4,0.2)$\succ$(0.3,0.4,0,3).

Lessons Learned on Contribution-Based Revenue Sharing Scheme. We explore the 3-subtask situation under contribution-based sharing scheme in Fig. 3.

Lesson 3. *When the first subtask is not the most difficult subtask, given a segment of weights on the first k (k < N) subtasks, the highest weight of all the possible weights on the* $(k{+}1){-}th$ *subtask leads to the highest final quality.*

As in Fig. 3 (a), given the weight on the first subtask equaling 0.1 (X axis scale is 1 to 8), all possible weights on the second subtask are 0.2, 0.3, $\cdots$ 0.8, then the highest weight on the second subtask which equals 0.8 gives us the highest final quality among all the series of subtasks begin with weight 0.1.

Lesson 4. *When the first subtask is the most difficult subtask, 1) the more difficult the first subtask is, the more efficient is the contribution-based revenue sharing scheme; 2) for the series of subtasks begin with the same difficulty, highest task dependence leads to the highest final quality.* (See Fig. 3 (b).)



(a) When the first subtask is not the most difficult subtask. (b) When the first subtask is the most difficult subtask.

Fig. 3. Efficiency estimation of vertical task decomposition strategy under contribution-based revenue sharing scheme

From the incentive viewpoint, it is true that decomposing a task into subtasks is worse than assigning the whole task to one worker. However, the latter makes difficult to find a worker who is willing to choose this task due to its limited resources. If we incorporate worker availability into discussions, Lesson 4 1) does not necessarily reduce the demand of crowdsourcing.

Example 3. Suppose the task requester is restricted to start with the subtask of a given difficulty. When the first subtask is not the most difficult subtask, (e.g., with difficulty 0.3), according to Lesson 3, the optimal decomposition is the one whose second subtask's difficulty is the highest among all possible difficulties (in this case, 0.1,$\cdots$, 0.6), so the optimal decomposition is (0.3,0.6,0.1). When the first subtask is the most difficult subtask, (e.g., with difficulty 0.5), according to Lesson 4, the optimal decomposition is (0.5,0.4,0.1) with the highest dependence 0.24 among all series of subtasks start with difficulty 0.5.

5 Conclusion

In this paper we have formally presented and analyzed vertical and horizontal task decomposition models which respectively specify the relationship between subtask quality and the worker's effort level in the presence of positive and none dependence among subtasks. We conclude that in general, vertical task decomposition strategy outperforms the horizontal one in improving the quality of the final solution, and furthermore give explicit instructions the optimal strategy under vertical task decomposition from the task requester's point of view.

Acknowledgments. The work is partially supported by a Grant-in-Aid for Scientific Research (S) (24220002, 2012-2016) from Japan Society for the Promotion of Science (JSPS).

References

1. Kittur, A., Smus, B., Khamkar, S., Kraut, R.E.: Crowdforge: crowdsourcing complex work. In: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, pp. 16–19 (2011)
2. Bernstein, M.S., Little, G., Miller, R.C., Hartmann, B., Ackerman, M.S., Karger, D.R., Crowell, D., Panovich, K.: Soylent: a word processor with a crowd inside. In: Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology, pp. 313–322 (2010)
3. Little, G., Chilton, L.B., Goldman, M., Miller, R.C.: Exploring iterative and parallel human computation processes. In: Proceedings of the ACM SIGKDD Workshop on Human Computation, p. 25 (2010)
4. Kulkarni, A., Can, M., Hartmann, B.: Collaboratively crowdsourcing workflows with turkomatic. In: Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work, pp. 1003–1012 (2012)
5. Tran-Thanh, L., Huynh, T.D., Rosenfeld, A., Ramchurn, S.D., Jennings, N.R.: Budgetfix: budget limited crowdsourcing for interdependent task allocation with quality guarantees. In: AAMAS, pp. 477–484 (2014)
6. Tran-Thanh, L., Venanzi, M., Rogers, A., Jennings, N.R.: Efficient budget allocation with accuracy guarantees for crowdsourcing classification tasks. In: AAMAS, pp. 901–908 (2013)