A Decision Tree Based Quality Control Framework for Multi-phase Tasks in Crowdsourcing

Yili Fang  
State Key Laboratory of Software Development  
Environment, School of Computer Science and Engineering, Beihang University  
Beijing, China  
fangyili@act.buaa.edu.cn

Pengpeng Chen  
State Key Laboratory of Software Development  
Environment, School of Computer Science and Engineering, Beihang University  
Beijing, China  
chenpp@act.buaa.edu.cn

Kai Sun  
State Key Laboratory of Software Development  
Environment, School of Computer Science and Engineering, Beihang University  
Beijing, China  
sunkai@act.buaa.edu.cn

Hailong Sun∗  
State Key Laboratory of Software Development  
Environment, School of Computer Science and Engineering, Beihang University  
Beijing, China  
sunhl@buaa.edu.cn

ABSTRACT

In crowdsourcing, there exists an important category of tasks that comprise an ordered sequence of subtasks, which we refer to as Multi-phase Tasks (MPTs) - e.g. travel planning, translation and micro-writing. Existing result inference methods are ineffective for processing MPTs. The constrained relationships among phase-level subtasks of MPT cannot be ignored for two reasons. First, it is ineffective to conduct a MPT without phase-processing, e.g. for travel planning, recommending a complete route of travel planning, and using existing methods to infer the final result generated by an individual worker can hardly meet various requirements due to the lack of flexibility. Second, although a MPT consists of a set of phase-level subtasks, it is unsuitable to simply split a MPT into subtasks and use top-k methods to recommend final results; because this will not only increase costs but also lose the constrained relationships among the phases. Thus it calls for a new approach to handle MPTs. This research first introduces the concept of MPT to identify these special tasks. Second, a decision tree based framework is provided to control task generation and final result combination in the crowdsourcing cooperative workflow for MPTs. Third, a probabilistic graphical model is proposed to characterize the subtasks of each MPT phase and a maximum likelihood based method is designed for result inference. Finally, extensive experiments were conducted based on real-world travel planning tasks and experimental results demonstrate the superiority of this approach in comparison with the state-of-the-art methods.

1 INTRODUCTION

Crowdsourcing aims at employing the wisdom of crowds to deal with the problems that are still difficult for computers. Its success has been witnessed in various applications, especially in areas of data management [12], ranging from simple tasks (e.g., image labeling [7], character recognition [19]) to complex ones (e.g., text editing [18]). To obtain high quality results of crowdsourcing tasks, especially complex tasks[22], many studies [5, 17] are increasingly seeking for the automatic cooperative workflows, in which a requester submits a set of tasks, and the platform assigns each task to multiple workers. Results are then inferred by aggregating the answers submitted by these workers.

This work investigates a special category of crowdsourcing tasks, such as travel planning [14], micro-writing [18, 22] and translation, which we refer to as Multi-phase Tasks (MPT). Processing a MPT is composed of an ordered sequence of phases, where each single phase is responsible for handling a subtask and depends on its preceding phase. For example, Fig. 1(a) shows a travel planning example, in which a requester is planning a one-day tour in Beijing with some popular tourist attractions. Travel planning consists of multiple scenic spots corresponding to all phases of a MPT. There are two types of result inference methods in crowdsourcing. One treats each MPT as a whole [4], which may not guarantee optimal...
processing of each subtask; while the other performs inference at the subtask level [15], which may not achieve global optimization due to the ignorance of the constraints among subtasks. Therefore, existing result inference methods are not effective for MPTs [3]. Taking travel planning as an example, on one hand, each worker is asked to finish all phases of a task and submit overall results with just one round of crowdsourcing; subsequently, a travel planning contributed by these workers is chosen. No matter which one is chosen, some popular attractions may be lost due to the limited interests of the corresponding workers. Thus, the final results, obtained by the first designed method as shown in Fig. 1(b) hardly meet the requirements of the requesters. On the other hand, as shown in Fig 1(c), workers are asked to provide a set of separate scenic spots and the top-k spots are chosen to generate a travel plan. However, this method does not consider traffic costs among scenic spots. For example, a generated route starting from Tiananmen to the Great Wall of China is generally too far to travel for a one-day tour. Thus these recommendation algorithms [2, 15] are not effective. Consequently, requirement targeting, i.e., make every effort to achieve the goal of the recommendation, and the constrained relationships among phases, e.g., traffic convenience, cannot be ignored in crowdsourcing. Therefore, neither of the two approaches is suitable for MPTs. There are two main challenges to handle MPTs. Under the limited budget, it is non-trivial to generate crowdsourcing tasks to preserve the integrity of the constraint relationships among phases in a task. In addition, how to conduct phase-level result aggregation and final result combination without the absence of compatibility of every phase is also challenging. To address these two challenges, two problems are investigated in this work: task generation and result inference.

Specifically, this research first presents a decision tree based framework for processing MPTs with crowdsourcing, which controls the dynamic generation of subtasks corresponding to the phases of a MPT, as well as the final result combination. Second, a probabilistic graphical model is proposed to infer the result in each phase, which models the task difficulty and the error rate of each worker. Different from the previous work [14], this approach incorporates the constrained relationships among all phases based on the paths of the decision tree, which are orderly distributed. This can reduce the number of required candidate answers in each phase as well as total workers. To summarize, we list the major contributions as follows.

- An important category of crowdsourcing tasks, MPTs, are identified, which consist of an ordered sequence of subtasks. To the best of our knowledge, this is the first attempt to characterize multi-phase tasks as a common task category in crowdsourcing.
- A decision tree based framework is proposed to control the task generation and the final result combination in crowdsourcing workflow for MPTs.
- A probabilistic graphical model is proposed to model the subtasks of each MPT phase and an EM algorithm is designed to infer the result of a subtask based on the answers obtained from crowdsourcing workers.
- A set of experiments were conducted with real-world travel planning tasks. Experimental results show that this approach outperforms existing methods in terms of both result quality and costs.

Figure 1: A MPT task and results of two crowdsourcing methods. The darker the color of the spot is, the more popular the spot is.

The remainder of this paper is organized as follows. Section 2 formalizes the problem and presents an overview of the framework. Section 3 describes the decision tree based controlling method of workflow and result inference method, respectively. The experimental setup and results are described in Section 4. Section 5 discusses related work and conclusions are made in Section 6.
2 PROBLEM FORMULATION & FRAMEWORK

2.1 Problem Formulation

This section formalizes the main problems. A multi-phase task (MPT) contains multiple phase-level subtasks, each of which corresponds to a component of the final result. Several dependencies are involved in associated processes, each phase of which can generate a result. There are two problems to be solved for MPTs: the task generation and the result inference.

For a MPT, the task generation of phase \( k \) depends on the result yielded by the immediate predecessor phase \((k-1)\) and can affect the task generation of immediate successor phase \((k+1)\). If each phase-level task generation yields a set of candidate tasks \( T_k = \{t_{k,l}| l \in I_k \} \), the alternative task set corresponding to \( N \) phases can be denoted as \( T = \{T_1, T_2, ..., T_k, ..., T_N\} \). Then we can denote a MPT generated from \( T \) as \( T = \{T_1, T_2, ..., t_{k,l}, ..., t_{N,N}\} \), here \( t_{k,l} \in T_k \) which is a phase-level subtask. The task generation problem can be defined as follows.

**Definition 2.1 (MPT-Generation MPT-G).** Given a candidate task set corresponding to all phases \( T = \{T_1, T_2, ..., T_k, ..., T_N\} \), where in \( T_k \) = \{\( t_{k,l} \) and \( k,l \) is the \( l \)th candidate subtask of phase \( k \). The MPT-Generation problem is to find a subtask sequence \( T = \{t_{1,l}, t_{2,l}, ..., t_{k,l}, ..., t_{N,l}\} \) from \( T = \{T_1, T_2, ..., T_k, ..., T_N\} \) with the goal of minimizing cost and maximizing quality.

As we have discussed, successful task generation can filter out the subtasks unnecessary to be generated thereby reduce costs and improve quality. As for MPTs, generating a whole MPT and crowdsourcing it can hardly bring a satisfactory result. In addition, although a MPT is composed of a set of atomic phase-level subtasks, it is unsuitable to naively split it into multiple subtasks and publish each subtask because of the loss of constrained relationships among phases. Therefore, To resolve the MPT-Generation problem entails taking the constrained relationship into account.

Given a MPT, we aim at inferring a satisfactory result through aggregating all subtask outputs from different workers in each phase. The result inference is indispensable in crowdsourcing cooperative workflows for processing MPTs, which affects the result quality and task generation of subsequent phases. Given the candidate answer set \( \mathcal{R} = \{R_1, R_2, ..., R_k, ..., R_N\} \) of a MPT \( T \), here \( R_k = \{r_{k,l}\} \), we give Definition 2.2 of the result inference as follows.

**Definition 2.2 (Result inference Problem-RIP).** Given a MPT \( T = \{t_{1,l}, t_{2,l}, ..., t_{N,l}\} \) generated from \( T \), and noisy candidate answer set \( \mathcal{R} = \{R_1, R_2, ..., R_k, ..., R_N\} \), here, \( r_{k,l} \in R_k \) denotes candidate answer \( l \) in phase \( k \). The result inference problem is to obtain result \( R' = \{r_{1,l}, r_{2,l}, ..., r_{k,l}, ..., r_{N,l}\} \) from \( \mathcal{R} \) with the goal of inferring the result as high quality as possible via cooperating with the crowd.

Furthermore, result inference method affects the task generation. Thus, it is suitable to develop the result inference method that corresponds to the task generation method.

As for Definition 2.1, we are mainly concerned with reducing the total cost, and give Definition 2.2 with the focus on improving the quality of final results.

2.2 The Crowdsourcing Framework

In this section, we provide an overview of our multi-phase crowdsourcing framework for MPTs, as shown in Fig. 2, where Task Generation, Crowdsourcing, Result Inference and Result Combination are the four major steps.

- **Step 1:** The task generation is implemented in phases. In each phase, the tasks of current phase are generated based on the result of the prior phase.
- **Step 2:** Tasks are submitted to multiple workers from the crowdsourcing platform for multi-phase crowdsourcing processing.
- **Step 3:** After receiving outputs from workers in each phase, the goal of this step is to infer the aggregated result from each phase of MPT. If the current phase is the final phase, then the next task is generated in Step 1 based on the result of current phase. Otherwise, the results of all the phases are combined in Step 4.
- **Step 4:** The result combination generates the optimal final result of MPT which is a associated sequence submitted to the requester.

In this procedure, the task generation, the phase-level result inference and the result combination are all controlled by our decision tree based method. We believe such a framework is general for processing various kinds of MPTs.

3 THE CONSTRAINED DECISION TREE MODEL

In this section, first, we give a probabilistic model to formalize the answering process for each phase-level subtask and give result inference model to formalize the RIP problem (Section 3.1). Second, the constrained relationships among phase-level subtasks are characterized in Section 3.2. Third, we incorporate our result inference method into a decision tree model in which the constrained relationship is considered to control the task generation. Then, we apply this decision tree model to obtain the combined result (Section 3.3).

3.1 The Probabilistic Model for Phase-level Inference

This section introduces a voting based probabilistic model to formalize the crowdsourcing process of a phase-level subtask and presents our phase-level result inference method to compute the satisfaction level of the results of phase-level subtasks.

We start with modeling the process of generating a MPT. Let the selected result of the prior phase be \( R = \{r_{1,k}, r_{2,k}, ..., r_{k-1,k}\} \), ...
My requirement: I want to visit Olympic attractions.
Current path: Olympic Park→Birds’ Nest
Is next path better than current path?
Olympic Park→Birds’ Nest→Palace Museum Yes No
Olympic Park→Birds’ Nest→Water Cube Yes No

Figure 3: The task interface for travel planning.

Figure 4: A probabilistic graph for the subtask $t$ of each phase in a MPT.

$R' = (r_1,t_1, r_2,t_2, \ldots, r_k,t_k)$ be a sequence which consists of the submitted results of all previous phases. Based on the candidate answer set ($R_k = \{r_{k1}, r_{k2}, \ldots, r_{kl}\}$) of current phase, we employ workers to vote which candidate answer $r_{kl}$ in $R_k$ can be combined with the result of prior phase $R$ to generate the result sequence $R'(t) = \{v_1, v_2, \ldots, v_{N_l}\}$ of current phase. Fig. 3 shows a travel planning task which embraces typical characteristics of MPTs. The task interface displays the goal of recommendation and the result of current path. The worker is asked to decide whether a candidate scenic spot in conjunction with the result of the current path is better than the current path or not.

Since one round voting can hardly yield enough outputs for high-quality result inference, we develop an iterative voting method based on the task generation of our workflow. Given the result of current phase $R$ with an unknown quality $q$ that follows the prior $P_R(q)$, and the result $R'$ of the next phase with an unknown quality $q'$ following the prior $P_{R'}(q')$. The voting process will continue until that it is enough to infer which result is better ($R$ or $R'$). Then we model the voting process. First, we give some assumptions: 1) each worker is diligent, that is to say every worker tries his/her best to complete all tasks. Actually, he/she may still make mistakes and show low ability, so it is important to model the error rate; 2) each worker is independent and will not be affected by other answers.

As shown in Fig. 4, we provide a causal structure of the crowdsourcing process for each subtask. Let $t \in T_k$ denote a subtask in phase $k$. An output $v_i$ submitted by worker $i$ of subtask $t$ in $T_k$ mainly depends on three factors: (1) the difficulty of subtask $d \in [0, 1]$. Generally, the higher $d$ is, the harder this subtask is, and thus the higher quality is for a worker to complete; (2) an individual error parameter $\lambda_i \in (0, 1)$ of the answer submitted by worker $i$.

(3) $z$ is the ground truth of subtask $t$, which is a latent value for the probabilistic model. With this causal structure, we can give a probabilistic model to formalize the answering process for each subtask in every phase. First, we define $d$ as follows.

$$d = d(q, q') = 1 - (q - q')^M,$$

where $M$ is a trained constant. It is easy to check $d \in [0, 1]$. Meanwhile, the difficulty of the question directly affects the accuracy of the answer of the worker. We define $\mu(d)$ as the accuracy of a worker on a subtask with difficulty $d$. The accuracy of a worker monotonically decreases with increasing the difficulty $d$ in $t$. Here, $\mu(d) \to 0.5$ as $d \to 1$ and $\mu(d) \to 1$ as $d \to 0$. So generally we give a function to compute the accuracy as follows.

$$\mu(d) = \frac{1}{2}[(1 + (1 - d)^{\lambda_i})].$$

In general, with the training set, we easily obtain the error parameters $\lambda_i$ of all workers by the maximum likelihood estimation. Meanwhile, we infer the probability $P(v_i = 1|q, q')$ that $i^{th}$ worker correctly answers subtask $t$ as follows:

$$\begin{align*}
    &\text{If } q' > q, P(v_i = 1|q, q') = \mu(d(q, q')); \\
    &\text{If } q' \leq q, P(v_i = 1|q, q') = 1 - \mu(d(q, q')).
\end{align*}$$

Let $\overline{v}_i = \langle v_1, \ldots, v_M \rangle$ be the voting outputs given by $M$ workers, where $v_i \in \{0, 1\}$. Based on the probability distribution of $q$ and $q'$ ($P_R(v_i = q)$ and $P_{R'}(v_i = q')$), we can derive the probability $P_{R'|\overline{v}_i+1}(q)$ which denotes the probability that vote $v_{i+1}$ is required. Here, $\overline{v}_i$ denotes that $i$ votes are known and $v_{i+1}$ (currently unknown) will be generated.

$$P_{R'|\overline{v}_i+1}(q) = P(v_{i+1} = 1|q, \overline{v}_i) \times P_{R'}(q')$$

$$= \int_0^1 P(v_{i+1}|q, q') \times P_{R'}(q')dq' \times P_{R'|\overline{v}_i}(q) = \int_0^1 P(v_{i+1}|q, q') \times P_{R'}(q')dq' \times P_{R'|\overline{v}_i}(q).$$

The similar equation can be derived to calculate the posterior of $R'$.

$$P_{R'|\overline{v}_i+1}(q') = P(v_{i+1}|q', \overline{v}_i) \times P_{R'}(q')$$

$$= \int_0^1 P(v_{i+1}|q', \overline{v}_i) \times P_{R'}(q')dq' \times P_{R'|\overline{v}_i}(q') = \int_0^1 P(v_{i+1}|q, q') \times P_{R'}(q')dq \times P_{R'|\overline{v}_i}(q').$$

Now we denote the improvement ratio from sequence $R$ to sequence $R'$ as $imp(R, R') = E(q)/E(q')$, $E(\cdot)$ denotes the mathematic expectation. And we can use it to infer the satisfaction degree of alternative result sequences.

This section formalizes the answering process for each phase-level subtask and gives the result inference model in which the satisfaction degree of the results of phase-level subtasks are characterized. It is a preparation for controlling workflow in Section 3.3 by affecting the subtask number and the quality of the result sequence.
3.2 Capturing Constrained Relationship

In this section, we present the definition of constrained relationships among phases, which are critical for the task generation and the result inference. We construct a relationship graph, where nodes are the candidate scenic spots among phases and the edges are the constrained relationship of each candidate pair. In order to quantify the constrained relationship between the candidate scenic spot \( r_{kl} \) of current phase \( T_k \) and the candidate scenic spot \( r_{k+1,l'} \) of the next phase \( T_{k+1} \), we introduce \( \text{tra}_T(r_{kl}, r_{k+1,l'}) \) as follows.

\[
\text{tra}_T(r_{kl}, r_{k+1,l'}) = \sum_{j=1}^{J} \text{weight}_j(r_{kl}, r_{k+1,l'}),
\]

where, \( j \leq J, J \) denotes the factor number of the constrained relationships, \( \text{weight}_j(r_{kl}, r_{k+1,l'}) \) denotes the constrained level from node \( r_{kl} \) to node \( r_{k+1,l'} \) of factor \( j \). It can be obtained from an external knowledge base (mapMicroblog). For example, we can quantify the constrained relationships for travel planning as follows.

\[
\text{tra}_T(r_{kl}, r_{k+1,l'}) = \begin{cases} \prod_{j=1}^{J} \text{weight}_j(r_{kl}, r_{k+1,l'}), & k' = k + 1, \\ 0, & \text{otherwise}, \end{cases}
\]

here, \( \text{weight}_j() \) denotes the normalized transportation distance from node \( r_{kl} \) to node \( r_{k+1,l'} \) that we can obtain from the map (google-map or baidu-map). \( \text{weight}_j() \) denotes the congestion level from node \( r_{kl} \) to node \( r_{k+1,l'} \) obtained from internet (e.g., Traffic Websites).

3.3 Constrained Decision Tree Model for Controlling the Task Generation and Result Combination

In this section, we propose a constrained decision tree (CDT) based model to control the task generation, the phase-level result inference and the result combination, and we give the definition of the decision tree model as follows.

**Definition 3.1 (Constrained Decision Tree Model-CDT).** The CDT is a labeled decision tree \( T = (R, E) \), where \( R = \{R_1, R_2, \ldots, R_K, \ldots, R_N\} \) denotes the nodes of the tree and \( R_k \) denotes the nodes of the stratum \( k \) in the tree that correspond to the candidate answers in phase \( k \). The edge \( (r_{kl}, r_{k+1,l'}) \) in \( E \) denotes the candidate answer \( r_{k+1,l'} \) in phase \( k + 1 \) depending on the candidate answer \( r_{kl} \) in phase \( k \), and its weight is in terms of the satisfaction degree and the constrained relationship.

Based on the CDT, we can transform the result combination problem into the problem of searching optimal sequence from the root to a leaf taking into account the satisfaction degree and constrained relationships. For example, Fig. 5 shows the CDT for travel planning in which the starting point is "Tiananmen". The result combination problem is to select a route which can best satisfy the requester from 6 candidate routes.

To evaluate the satisfaction degree for a complete result sequence, we introduce the satisfaction score \( \text{sat}_T(R) \) of each edge in a tree based on our phase-level inference model as follows.

1. If a sequence only consists of the root, then \( \text{sat}_T(R) = 1 \).
2. If a sequence \( R' \) was generated by adding an item behind a sequence \( R, \text{sat}_T(R, R') = \text{imp}(R, R') \times \text{sat}_T(R) \), where \( \text{imp}(R, R') \) defined in Section 3.1 denotes the improvement ratio from the sequence \( R \) to the sequence \( R' \).

Meanwhile, we also define the transition score \( \text{tra}_T(R) \) to evaluate the constrained relationship corresponding to each edge in the tree. Thus we consider the transition score as another criterion for evaluating a result sequence as follows.

\[
\text{tra}_T(R) = \sum_{n=0}^{k-1} \text{tra}_T(r_{kl}, r_{k+1,l'}).
\]

In Section 3.2, we denote \( \text{tra}_T(r_{kj}, r_{k+1,l'}) \) as the transition score between \( r_{kl} \) and \( r_{k+1,l'} \) corresponding to the constrained degree between the two phases. Here \( \text{tra}_T(R) \) denotes the transition score of the sequence \( R \) calculated in Equation 8.

Thus, we use a synthetical score \( \text{score}_T(R) \) to denote the score of the sequence \( R \) with respect to the satisfaction score and the transition score. \( \text{score}_T(R) \) is calculated as follows.

\[
\text{score}_T(R) = \text{sat}_T(R) - \beta \times \text{tra}_T(R),
\]

where \( \beta \) is the factor to balance the satisfaction of user and the traffic constraint. Take travel planning as an example, \( \beta \) is the balance factor between the satisfaction degree and constrained relationship in terms of phase-level recommended travel routes. Here, we set \( \beta = \frac{1}{N} \), where \( N \) denotes the number of scenic spots that the requester desires to travel. While calculating the score of a recommended travel route with the destination "Water Cube" in a tree \( T \) presented in Fig. 5, for seq = \{Tiananmen, Olympic Park, Bird’s Nest, Water Cube\} which is a recommended travel route, we can calculate \( \text{sat}_T(\text{seq}) = 0.4 \times 0.5 \times 0.3 = 0.06 \) and \( \text{tra}_T(\text{seq}) = (0.3 + 0.4 + 0.4) = 1.1 \). Then we can obtain \( \text{score}_T(\text{seq}) = 0.06 - 1.1 \times 0.25 = -0.215 \).

**Remark 1.** Given CDT \( T = (R, E) \), if the depth of every leaf in the \( T \) is \( N \), then CDT is a complete tree denoted as Compl(T), the optimal results belong to the set which involves all the result sequences corresponding to phase \( N \) of a MPT.

From the requester’s perspective, there is a semantic difference between a complete and a partial sequence, that is, a complete result sequence cannot be extended further.

If the number of phases in MPT is \( N \), the CDT is a complete tree. Given the transition score between items in the set \( R \), we can calculate the potential score of a probable result sequence and the best result sequence with \( N \) items can be obtained as follows.

\[
R = \arg \max_{R' \in \text{Compl}(T)} \text{score}_T(R').
\]

Based on Equation 10, we introduce Algorithm 1 to control our workflow and obtain an optimal result sequence.

Algorithm 1 controls all phases of the workflow for processing a MPT \( T \) corresponding to all nodes in CDT of \( T \), generating the result sequence \( R' \) one phase by one phase. Then we recursively calculate the potential score of a result sequence (line 11-12). After obtaining the transition possibility, it is clear that how to capture the improvement ratio is the key. The satisfaction score is maximized when the sequence \( R' \) gets all of the votes and the sequence \( R \) gets

1. [http://eye.bjyt.gov.cn/Web-T_bjyz_new/Main.html]
zero votes. Here we can get the highest score of a sequence \( R \). In the end, the algorithm returns result sequence with the maximum potential score. Thus, when the length of a sequence reaches \( N \) and the sequence has the highest potential score, the algorithm can be stopped. It is clear that the number of the questions is so large that we can not exhaust every question. Thus, in Algorithm 1, we employ a greedy approach to solve this problem, that is, we ask questions on a result sequence with the highest potential score. This approach is effective and can find the best combination of the result sequence for MPT. However, it will cause the problem of local optimum. Thus, we develop Algorithm 2 to improve Algorithm 1. In Algorithm 2, we choose \( k \) result sequences with the top-\( k \) potential scores to implement the phase-level task generation in order to find the optimal result sequence as soon as possible and reduce the number of subtasks.

Algorithm 1. Task generation and result combination

**Input**: Decision tree \( T' \) without satisfaction score

**Output**: Result sequence \( R \) with MaxScore

1. **Initialization**: \( R \leftarrow \{ r_{11} \} \) is root of \( T' \); // Initialization
2. **while** \( |R| \leq N \) do
   
   3. **while** \( l \) is smaller than the successor size of \( R \) do // Phase \( k \)
      
      4. \( R' \leftarrow R \cup \{ r_{kl} \} \); // \( r_{kl} \) is a successor of \( R \) in phase \( k \)
      
      5. **if** MaxScore < score\(_T'(R')\) **then**
         
         6. \( R'' \leftarrow R' \); // Result combination
         
         7. **if** imp ≤ threshold **then**
            
            8. Ask a question on node \( r_{kl} \);
            
            9. Update score\(_T'(R')\);
            
            10. **else**
                
                11. \( l = l + 1 \); // To ask the next question in phase \( k \)
                
                12. \( R = R'', \ k = k + 1 \); // To ask a question in the next phase

13. **return** Result sequence \( R \);

4 EXPERIMENTS

In this section, we conducted an extensive set of real-world experiments on a set of representative MPTs and we present the evaluation results of our proposed method in comparison to the state-of-the-art methods.

4.1 Experimental Settings

In our experiments, we chose a set of tasks: travel planning for one-day tour in Beijing concerning the popular scenic spots. It embraces the typical characteristics of MPTs, where each task is composed of a set of subtasks and constrained relationships among subtasks which are critical to obtain high-quality results from the crowd. In addition, we developed a travel-planing system named CrowdTP, which is open-source and available.

The evaluation is performed in terms of three perspectives: 1) how the value of \( k \) in Algorithm 2 affects the number of generated subtasks in a MPT; 2) whether our voting method can reduce the costs; 3) whether the result sequence yielded by our algorithm satisfies the requester better than the existing methods.

The worker pool of our system consisted of 60 local university students and the results were evaluated by some other students. To obtain unknown parameters, such as the error rate \( \hat{\lambda} \) and the difficulty \( \omega \), we introduced 100 attraction tasks and asked all the workers from our pool to learn the parameters of our voting model, then we obtained that the task difficulty \( \omega \in (0.5, 0.7) \) and the error rate of every worker \( \hat{\lambda} \in (0.6, 0.7) \). In general, we assume the
4.2 The Influence of \( k \) on Cost

In this experiment, first, we explore the impacts of \( k \) in top-\( k \) on the number of subtasks generated in a MPT. \( k \) was set to be different numbers (i.e., 2, 3, 4). The results are shown in Fig. 6. As the number of generated subtasks is closely related to the costs of crowdsourcing, more generated subtasks means more costs. We believe it is fair to measure the costs of a MPT in terms of the number of generated subtasks when the reward of every subtask is fixed. We also can observe that generating more subtasks causes more scenic spots of the final result sequence. And the number of generated subtasks is also related to the value of \( k \) in our method: a larger \( k \) in our method generates more subtasks. To be specific, when \( k = 2 \), our method generates 3,500 and 550 less subtasks than our method \( k = 3 \) and \( k = 4 \) in average respectively.

4.3 The Total Costs of Different Methods

Next, we evaluate the performance of two methods in terms of the total costs which are related to the number of the subtasks in all phases. We give a comparison between our method and the previous work [8] named CrowdPlanr. We respectively measure the number of scenic spots (i.e., the number of generated subtasks) in the two running systems. The results are shown in Fig. 7. No matter how the number of the scenic spots varies, the subtasks generated by our system CrowdTP are much less than those generated by CrowdPlanr. Because, in every phase-level task generation of our method, the requirement and traffic costs are considered to filter out some subtasks which are not necessary to be generated. In particular, we can achieve \( 1.92 \times 10^4 \) less subtasks than those of CrowdPlanr.

4.4 Users’ Satisfaction Evaluation

This experiment aims to evaluate whether the plan generated by our system can better meet the requirement than the existing method or not. Generally, it is difficult to measure the usefulness of the generated plans due to lack of ground truth. Thus, we invited additional 20 workers to evaluate the routes generated with CrowdPlanr and CrowdTP. This is a binary classification task, which asks workers to judge which route can better meet the specific requirements. Fig. 8 shows that our system CrowdTP can achieve higher users’ satisfaction ratio of different tasks compared with CrowdPlanr. In particular, for task 4, the satisfaction score of our method is 63.3% higher than that of CrowdPlanr on average. Because, our method takes into account the traffic costs and the preference of the requester in each phase of a MPT.

5 RELATED WORK

In the crowdsourcing market, since answers offered by unknown crowd workers are often noisy and unreliable, two core challenges in crowdsourcing are to design the workflow and to guarantee the result quality. Subsequently, a brief summary is provided of existing works from two perspectives.

In terms of the methods of tasks processing in crowdsourcing, existing crowdsourcing applications span a broad range of tasks, which can be roughly put into two categories: atomic tasks and compound tasks. Atomic tasks cannot be split into subtasks, and their results, processed by workers, cannot be divided into components such as image labeling [10], peer grading [3] and sentiment analysis. Compound tasks are usually composed of a set of subtasks, and final results consist of a sequence of answers. Furthermore, this research classifies compound tasks into two subcategories: the combined task and MPT. A combined task can be converted into a set of subtasks, each of which needs crowdsourcing. Typical combined tasks include handwriting recognition, video recognition, and any kind of grouping of atomic tasks. The MPT, however, cannot be solved in the same way. It is hard to be solved as a whole or as a part, because the result consists of a set of components with some inner relationships. A large number of crowdsourcing tasks fall into this category, e.g., travel planning [14], micro-writing [22] and translation [20] et.al. Previous methods always design this task as...
a whole to collect final results. To the best of our knowledge, little work previously exists to control MPT quality reasonably.

To obtain a better quality of crowdsourcing results, previous works used to control crowdsourcing with the workflows. For the atomic tasks, there is a general workflow to control the quality of results. For the complex tasks, the workflows contain two categories: the distributed workflow [9, 11] and iteration workflow [5]. The distributed workflow aims at processing the combination tasks. It splits a task into several subtasks, each of which is processed by the general workflow, and finally combines them into a whole result submitting to the requester. The iteration workflow mainly processes some complex tasks, which cannot reach the high quality by running a round general workflow, and run some iteration of general workflows to improve the quality of results.

In contrast to previous studies, this research focuses on the workflow for processing MPTs. Although exciting previous work successfully processes special cases of MPT - e.g., FFV is a special situation of MPT (micro-writing) [1, 13] - proposed to correct and shorten text in three phases: find, fix, verify. However, there is no general workflow to process MPTs. Therefore, this research proposes a novel workflow to process these tasks.

In various kinds of workflows, the result inference is also important to control quality. Majority voting is a well-known method to infer results from answers of multiple workers. Early work with majority voting does not consider the differences among worker accuracy, but was effective for simple tasks that most workers are able to process correctly. However, this method can lead to low-quality results in the case of difficult tasks or less capable workers. To consider worker accuracy, weighted majority voting is proposed [16]. Various solutions like EM and Bayesian methods are proposed to estimate worker accuracy [16].

To further improve the majority-voting strategy, there are two lines of studies related to the specific types of crowdsourcing tasks. For isolated tasks, some literatures [6, 15] propose result inference methods based on probabilistic model to deal with inaccuracies and minimize cost, i.e. different workers are asked to perform the same task until a consensus is achieved on the outcome. For complex tasks, some researchers give other inference methods based on probabilistic graphical models to utilize a multiple iteration strategy to process tasks [13] using a quality-sensitive answering model. AskIt utilizes entropy-like techniques [2]. QASCA studies the quality-aware inference method [21]. Compared with these result inference methods, this research considers the dependencies between multiple phases in crowdsourcing, which can enable the workflow to achieve better quality.

6 CONCLUSIONS

This research studies the quality-control problem in crowdsourcing for complex tasks. We identify a category of tasks, i.e., multi-phase tasks, for which the constrained relationships are of great importance to achieve better result quality. A decision tree based framework is given to control task generation and the final result combination in the crowdsourcing workflow for MPTs. A probabilistic graphical model is proposed to characterize the phase-level subtask processing and the result inference. A series of experiments were conducted with real-world travel planning tasks. The experimental results demonstrate that this approach can achieve higher quality with less costs than that of the state-of-the-art methods.

7 ACKNOWLEDGMENTS

This work was supported partly by the National Key Research and Development Program of China under Grant No.2016YFB1000804, the National Basic Research 973 Program of China under Grant Nos.2014CB340304, 2015CB358700, and the State Key Laboratory of Software Development Environment under Grant No. SKLSE-2017ZX-14.

REFERENCES