2017 IEEE 37th International Conference on Distributed Computing Systems

MELODY: A Long-term Dynamic Quality-aware Incentive Mechanism for Crowdsourcing

Hongwei Wang*†, Song Guo†, Jiannong Cao†, Minyi Guo*
*Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
†Department of Computing, The Hong Kong Polytechnic University, Hong Kong
Email: wanghongwei55@gmail.com, {song.guo, jiannong.cao}@polyu.edu.hk, guo-my@cs.sjtu.edu.cn

***Abstract*—Crowdsourcing allows requesters to allocate tasks to a group of workers on the Internet to make use of their collective intelligence. Quality control is a key design objective in incentive mechanisms for crowdsourcing as requesters aim at obtaining answers of high quality under a given budget. However, when measuring workers' long-term quality, existing mechanisms either fail to utilize workers' historical information, or treat workers' quality as stable and ignore its temporal characteristics, hence performing poorly in a long run. In this paper we propose MELODY, a long-term dynamic quality-aware incentive mechanism for crowdsourcing. MELODY models interaction between requesters and workers as reverse auctions that run continuously. In each run of MELODY, we design a truthful, individual rational, budget feasible and quality-aware algorithm for task allocation with polynomial-time computation complexity and $O(1)$ performance ratio. Moreover, taking into consideration the long-term characteristics of workers' quality, we propose a novel framework in MELODY for quality inference and parameters learning based on Linear Dynamical Systems at the end of each run, which takes full advantage of workers' historical information and predicts their quality accurately. Through extensive simulations, we demonstrate that MELODY outperforms existing work in terms of both quality estimation (reducing estimation error by $17.6\% \sim 24.2\%$) and social performance (increasing requester's utility by $18.2\% \sim 46.6\%$) in long-term scenarios.**

I. INTRODUCTION

The past decade has witnessed the proliferation of *crowdsourcing*, a new problem-solving platform which outsources small tasks to a large number of online labors and solves the tasks by harnessing their collective intelligence. In crowdsoucing markets such as Amazon Mechanical Turk (AMT)[1], requesters may submit batches of small tasks whose solving is quite difficult or too expensive to automate but simple for humans, such as proofreading and image labeling, along with the amount of money they are willing to pay. Workers then choose to complete tasks and receive payment.

Most of the existing incentive mechanisms for crowdsoucing[2] use auction and pricing [1]–[10] to provide workers with monetary rewards for their participation costs. In the proposed mechanisms [4]–[7], [9]–[13], *quality control* is one of key considerations for platform designers. On the one hand, crowd workers are at different levels of problem-solving capability and may provide answers with varied quality [14]. On the other hand, requesters usually rely on redundancy (i.e., allocating a single task to multiple workers) to identify correct answers [15], [16].

Although many incentive mechanisms concerning workers' quality have been proposed recently, they focus on how to allocate a *single* set of tasks to a group of workers by running a particular task allocation algorithm, and how to measure workers' quality based on their performance in a *single run*. However, in reality, crowdsourcing platforms (e.g., AMT) need to outsource *multiple* sets of tasks continuously for a long time. To fit these *multi-run* scenarios, existing mechanisms [5]–[7], [9], [10], [12], [15] can only perform their algorithms for each set of tasks *independently*, without considering workers' quality of information gathered in previous runs. The independent repetition of the algorithm simply using workers' current information loses prior knowledge of workers' quality and is vulnerable to system noises, which leads to a poor representation of workers' quality known as *over-fitting* [17]. A few existing mechanisms do utilize workers' historical information when measuring and estimating their quality [4], [11]. However, they consider it *stable* (i.e., it is drawn from a specific statistical distribution) and ignore its temporal characteristics (i.e., worker's prior information is treated as a set rather than a sequence). Such models lack sufficient fitting capability and fail to model real workers' long-term quality, which is known as *under-fitting* [17].

To further illustrate these problems, we analyze a public data set[3] publicized by [18] and plot four typical types of worker's time-varying quality curves in Fig. 1, from which we learn two important properties of workers' long-term quality: 1) A worker's quality at a given point is much more related with his recent quality, because workers' latent quality is unlikely to change drastically in a few runs. 2) Except some rare cases (Fig. 1d) where quality exhibits stability[4] in quite a long time, most workers' long-term quality curves present a notable rising, declining and fluctuating as shown in Fig. 1a, 1b, and 1c respectively. Such trends usually result from worker's gradually accumulated change in the level of

[1]http://www.mturk.com

[2]Some of the work is categorized as *crowdsensing* or *participatory sensing* rather than crowdsourcing. However, we do not distinguish these three paradigms since they make no essential difference in this paper.

[3]http://sites.google.com/site/nlpannotations

[4]A worker's quality is called *stable* in this case study if the slope of linear regression of quality curve is within [-0.05, 0.05] and the variance of quality curve is below 100. The percentage of stable workers under the definition is 8.5%.

1063-6927/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDCS.2017.28

TABLE I: Comparison of several popular incentive mechanisms for crowdsourcing with MELODY.

	[2]	[3]	[4]	[5]	[6]	[7]	MELODY
Truthfulness	✓	✓	✓	✓			✓
Individual rationality	✓	✓	✓	✓			✓
Competitiveness	✓	✓		✓			✓
Computational efficiency	✓	✓	✓	✓		✓	✓
Budget feasibility	✓	✓	✓			✓	✓
(short-term) Quality awareness			✓	✓	✓	✓	✓
Long-term quality awareness							✓

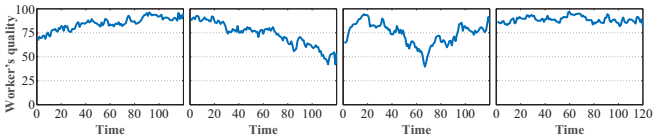
expertise, proficiency and effort rather than accidental error and cannot be characterized by a simple statistical distribution. However, existing mechanisms ignore these two properties of workers' long-term quality, thus fail in maintaining decent performance in a long run as illustrated in Section VII.

To address the above over-fitting and under-fitting problems, in this paper we propose MELODY, a truthful and budget feasible incentive MEchanism considering workers' LOng-term DYnamic quality for crowdsourcing markets. MELODY models the interaction between requesters and workers as continuously running reverse auctions. In each run, a requester posts a set of tasks with a budget and workers submit their bid information. MELODY then determines the task allocation scheme and payment scheme, such that the bid of each worker and quality requirement of each task are conformed, meanwhile the properties of truthfulness, individual rationality, competitiveness and computational efficiency are guaranteed. We compare properties of some popular incentive mechanisms for crowdsourcing with MELODY in Table I.

To characterize the dynamicity of workers' long-term quality, in MELODY we also propose a novel framework for quality inference and parameters learning based on *Linear Dynamical Systems* (LDS) [17], [19], [20]. The LDS-based framework models worker's quality as latent time series which cannot be observed directly by the crowdsourcing platform but dominate worker's performance. At the end of each run, MELODY infers every worker's posterior quality after observing his performance in this run, and then makes prediction about his quality for the next run. In addition, MELODY re-estimates the parameters of every worker regularly to ensure accuracy of quality inference by using Expectation Maximization algorithm. Simulation result shows that compared with prior work, MELODY increases requester's utility by $18.2\% \sim 46.6\%$ and reduce estimation error of workers' quality by $17.6\% \sim 24.2\%$.

Our contributions in this paper are as follows:

- To the best of our knowledge, we are the first to consider long-term characteristics of workers' quality that do matter to the performance of incentive mechanisms for crowdsourcing but have not been well addressed by existing work.
- We propose MELODY for crowdsourcing which models the interaction between requesters and workers as reverse auctions that run continuously. MELODY possesses most of the desired properties such as truthfulness and competitiveness. Meanwhile, we present a novel LDS-based inference and learning approach in MELODY to estimate workers' long-term dynamic quality.
- We conduct extensive simulations and the experimental results demonstrate the effectiveness and excellent performance of our mechanism compared with existing work.



(a) Rising (b) Declining (c) Fluctuating (d) Stable

Fig. 1: Four typical types of workers' long-term quality curves in reality. The data come from crowdsourcing tasks conducted on AMT for affective text analysis. We measure workers' quality as "maximum rating - average error", where the average error is the difference between ground truth and worker's average rating over 10 items.

The rest of this paper is organized as follows. We summarize related work in Section II. Section III proposes system model and problem formulation. Detailed design and theoretical analysis of MELODY are presented in Section IV, V and VI respectively. In Section VII, we evaluate the performance of MELODY with prior work. Section VIII concludes this paper.

II. RELATED WORK

A. Incentive Mechanisms for Crowdsourcing

Except for a few non-monetary incentive mechanisms such as [21] and [22], most incentive mechanisms for crowdsourcing are based on auction and pricing, using monetary reward to incentivize workers. Zhao *et al.* [2] propose a truthful mechanism based on online auction where users arrive in random order. Gao *et al.* [8] design a Lyapunov-based VCG auction policy for online worker selection while providing long-term participation. Anari *et al.* [23] give a budget-feasible mechanism for large scale crowdsourcing markets with the performance that achieves a desirable approximation ratio. However, none of the above mechanisms takes the factor of workers' quality into consideration.

Ho *et al.* [24] study the problem of adjusting quality-contingent payments for tasks and establish a multi-armed bandit model. Zhang *et al.* [4] use Bayesian inference for task allocation and quality estimation. Jin *et al.* [5] model the task allocation problem as the set cover problem and propose a quality-aware mechanism for mobile crowdsensing. Li *et al.* [7] take workers' reliability into consideration for crowdsoucing high quality labels. However, these mechanisms do not consider the long-term characteristics of workers' quality.

B. Quality Control and Measurement for Crowdsourcing

Liu *et al.* [12] propose a network management framework including quality of information (QoI) satisfaction index to

TABLE II: Key notations.

Notation	Definition
$\mathcal{W}_U$	Universal set of workers
$\mathcal{W}^r$	Set of qualified workers in run r
i	The i-th worker in $\mathcal{W}_U$
$\mathcal{T}^r$	Set of tasks publicized by the requester in run r
t_j^r	The j-th task in $\mathcal{T}^r$
B^r	The budget set by the requester in run r
$x_{i,j}^r$	Indicator of whether t_j^r is allocated to i
$p_{i,j}^r$	Payment to i for completing t_j^r
$s_{i,j}^r$	Score for the answer of t_j^r from i
$\mathcal{S}_i^r$	Set of scores that i gets in run r
$\mathbf{S}_i^r$	Sequence $\{\mathcal{S}_i^1, ..., \mathcal{S}_i^r\}$
q_i^r	Latent quality of i in run r
$p(q_i^r \vert q_i^{r-1})$	Transition probability density of i
$p(S_i^r \vert q_i^r)$	Emission probability density of i
a_i	Transition coefficient of i
γ_i, η_i	Variance of transition and emission density of i
$\alpha(q_i^r)$, $\widehat{\alpha}(q_i^r)$	Prior and posterior probability of q_i^r
c_i^r, n_i^r	The cost and frequency that i bids in run r
$\overline{c}_i^r$, $\overline{n}_i^r$	The true cost and frequency of i in run r
$u_{i,j}^r$	The utility of i on t_j^r
u_i^r	The utility of i in run r
U^r	The utility of the requester in run r
Q_j^r	The quality threshold of t_j^r
μ_i^r	Mean of $\alpha(q_i^r)$ (estimated quality)
$\widehat{\mu}_i^r, \widehat{\sigma}_i^r$	Mean and variance of $\widehat{\alpha}(q_i^r)$
$[\Theta_m, \Theta_M]$	The acceptable interval of worker's quality
$[C_m, C_M]$	The acceptable interval of worker's bid of cost

tackle the challenges of supporting QoI requirements in participatory sensing. Yang *et al.* [10] measure data quality as its deviation from the cluster centroid in their quality-based surplus sharing method for participatory sensing. Ho *et al.* [14] show factors that can improve workers' quality by conducting experiments on AMT. Mason *et al.* [25] introduce empirical analysis of quality assurance on AMT. Song *et al.* [16] propose a QoI satisfaction metric to describe the level that collected sensory data can satisfy the requirement of a task. The main difference between our paper and these work is that we focus more on estimation and update methods of workers' long-term quality rather than quality measurement metrics discussed in these work. Therefore, the above mentioned quality measurement metrics can be incorporated naturally into our work.

III. System Model and Problem Formulation

In our system we consider a single *platform*, upon which there are several *requesters* residing in the cloud and a universal set of *workers* $\mathcal{W}_U = \{1, 2, \ldots\}$. In each *run* $r \in \mathbb{N}^+$, one of the requesters publicizes a set of crowdsourcing *tasks* $\mathcal{T}^r = \{t_1^r, t_2^r, \ldots\}$ to a set of qualified workers $\mathcal{W}^r \subseteq \mathcal{W}_U$. We will explain why and how to choose qualified workers in Section IV. A crowdsourcing platform usually covers a wide variety of tasks. Without loss of generality, in this work we only consider tasks that are *homogeneous* in terms of the knowledge and effort needed to solve them, because the model proposed in this paper can be easily extended to the scenario with multiple types of tasks by designing the incentive mechanism for each individual type respectively [21].

We model the interaction between the requester and workers in a given run as a *reverse auction*. *Game-theoretic* workers

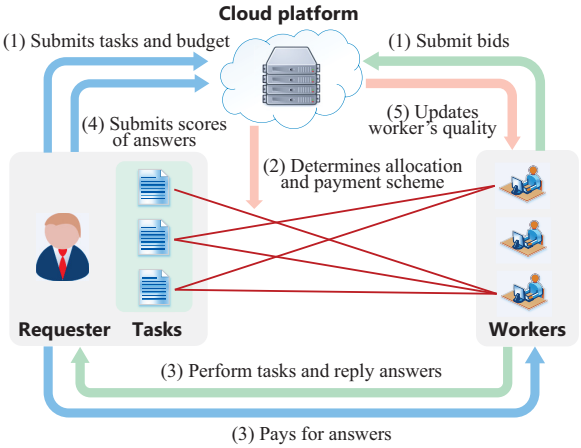


Fig. 2: System workflow in each run unit. The indexes of arrows and captions in the figure are corresponding to those in Section III-A.

sell their wisdom or effort to the requester and seek to make *strategy* to maximize their individual utility, and the requester decides whether and at what price to buy their services.

In our model we assume that a task can be allocated to multiple workers for quality guarantee, and a worker can be assigned multiple tasks in a run. We also assume that the requester holds a *budget* $B^r \in \mathbb{R}^+$ as the maximum amount of total payment he is willing to make in run r.

Table II summarizes the frequently used notations in this paper. Some will be introduced later.

A. System Workflow

Our auction-based system consists of multiple basic run units. In run r, the system workflow is described as follows:

1) Before auction starts, the platform receives a set of tasks $\mathcal{T}^r$ with a budget constraint B^r from one requester, and bids information from all workers. It then determines the set of qualified workers $\mathcal{W}^r \subseteq \mathcal{W}_U$.
2) The platform then determines allocation scheme $\mathcal{X}^r \triangleq \{x_{i,j}^r | i \in \mathcal{W}^r, t_j^r \in \mathcal{T}^r\}$, where the binary variable $x_{i,j}^r \in \{0, 1\}$ indicates whether t_j^r is allocated to worker i in run r.
3) Worker i completes every task t_j^r allocated to him, replies answer to the requester and gets payment $p_{i,j}^r \in \mathbb{R}^+$ for the task according to the payment scheme $\mathcal{P}^r \triangleq \{p_{i,j}^r | i \in \mathcal{W}^r, t_j^r \in \mathcal{T}^r\}$.
4) For the answer of task t_j^r from worker i, a *score* $s_{i,j}^r \in \mathbb{R}$ is given and submitted to the platform.[5] Denote $\mathcal{S}_i^r$ the set of scores worker i gets in run r and $\mathbf{S}_i^r$ the sequence of $\mathcal{S}_i^r$: $\{\mathcal{S}_i^1, ..., \mathcal{S}_i^r\}$.
5) After receiving all scores in this run, the platform then updates every worker's quality for the next run.

Fig. 2 illustrates the above system workflow in each run.

[5]Scores can be given either by the requester manually after answer verification, or by unsupervised algorithms automatically such as majority voting. The choice of rating methods is beyond the scope of this paper.

B. Worker Modeling

Consider the problem of modeling worker's performance $p(\mathbf{S}_i^r) = p(\mathcal{S}_i^1, ..., \mathcal{S}_i^r)$. The easiest way to handle this joint distribution for a sequence of observations would be simply to ignore the sequential aspects and treat all $\mathcal{S}_i^r$ as independent and identical distributions (i.i.d.), i.e., $p(\mathbf{S}_i^r) = \prod_{t=1}^{r} p(\mathcal{S}_i^t)$. However, such an approach would fail to exploit the correlations between observations that are close in the sequence which we have shown before. Therefore, it's natural to turn to the Markov model $p(\mathbf{S}_i^r) = \prod_{t=1}^{r} p(\mathcal{S}_i^t|\mathbf{S}_i^{t-1})$, in which worker's current performance is dependent on previous ones. But a new problem arises as the number of parameters will grow unlimitedly with the increase of length of the sequence. Certainly, we can reduce model complexity by making an assumption that $\mathcal{S}_i^r$ only depends on its most recent d predecessors, but any sort of hand-crafted d is hardly likely to be optimal. Besides, we lose correlations between two observations over d time units under the assumption.

To address the above shortcomings, instead of directly modeling worker's performance, we introduce a latent random variable $q_i^r \in \mathbb{R}$ to characterize the quality of worker i in run r, which dominates worker's immediate performance. Here *latent* means q_i^r cannot be observed directly and needs to be inferred by the platform. We allow the probability distribution of q_i^r to depend on the state of previous latent variable q_i^{r-1}, thus $p(q_i^r|q_i^{r-1})$ is known as the *transition probability density* from q_i^{r-1} to q_i^r. In addition, latent variable q_i^r and observed variable $\mathcal{S}_i^r$ are connected by the *emission probability density* $p(\mathcal{S}_i^r|q_i^r)$. It is worth noticing that such model is not limited by the Markov assumption to any order d (because there is always a path connecting any two observations via latent variables) and yet can be specified using a limited number of free parameters.

In practice, we concern two types of conditional probability of q_i^r. The first is the prior probability of q_i^r given all previous sets of observed scores:

$$\alpha(q_i^r) \triangleq p(q_i^r|\mathbf{S}_i^{r-1}), \tag{1}$$

and the second is the posterior probability of q_i^r given all sets of observed scores (including the set of scores from run r):

$$\widehat{\alpha}(q_i^r) \triangleq p(q_i^r|\mathbf{S}_i^r). \tag{2}$$

In our proposed model, $\alpha(q_i^r)$ is used to measure worker's quality in task allocation, while $\widehat{\alpha}(q_i^r)$ is used to infer worker's quality in the next run. The above prior and posterior probabilities are connected by transition density:

$$\alpha(q_i^r) = \widehat{\alpha}(q_i^{r-1})p(q_i^r|q_i^{r-1}). \tag{3}$$

To formulate our problem, each worker $i \in \mathcal{W}_U$ in run r associates:

- a bid of *cost* for performing every single task: $c_i^r \in \mathbb{R}^+$;
- a bid of maximum number of tasks (also called *frequency*) he is willing to complete: $n_i^r \in \mathbb{N}^+$;
- the prior probability of q_i^r: $\alpha(q_i^r) \sim \mathcal{D}(q_i^r; \mu_i^r)$, where $\mathcal{D}(q_i^r; \mu_i^r)$ is certain probability distribution with mean μ_i^r.

The above bid of cost and frequency constitute the *bid* $b_i^r = (c_i^r, n_i^r)$ of worker i. A worker may lie about his bid to maximize his utility, so we use $\overline{b}_i^r = (\overline{c}_i^r, \overline{n}_i^r)$ to denote worker i's true bid, which is private information and only known to himself.

Note that though the above model is similar to Hidden Markov Models (HMM), methods for solving HMM cannot be used here as the latent variables of workers' quality are continuous.

C. Utilities of Workers and Requesters

In run r, each worker strategically determines his bid to maximize his utility given his true bid. We formalize a worker's *utility* in Definition 1 as follows.

Definition 1 (A Worker's Utility)**.** *The utility of worker i in run r is the difference between the total payment he receives and his total cost:*

$$u_i^r = \sum_{j=1}^{|\mathcal{T}^r|} u_{i,j}^r, \tag{4}$$

where

$$u_{i,j}^r = \begin{cases} p_{i,j}^r - \overline{c}_i^r, & if\ x_{i,j}^r = 1; \\ 0, & otherwise. \end{cases} \tag{5}$$

To ensure quality of answers (e.g., confidence level of labeling) collected from workers, sufficient amount of collective intelligence and effort are necessary for each task. It is worth noticing that quantifying the *integrated crowd quality* remains a challenging problem in crowdsoucing, since types of answers are in varied forms (e.g., texts, photos, options), and correlation or interaction may exist among answers. Therefore, existing mechanisms all make moderate simplification when measuring the integrated quality of a crowd of people, such as Bayesian assumption [4], additive assumption [5], and proportional assumption [7]. In this paper, we take the additive assumption adopted in [5], in which the integrated quality that all workers generate on task t_j^r is defined as $\sum_{i \in \mathcal{W}^r} x_{i,j}^r \mu_i^r$, and a task t_j^r is *satisfied* if the integrated quality received by this task exceeds a threshold Q_j^r set by the requester. Therefore, we give the definition of satisfied tasks as follows.

Definition 2 (A Satisfied Task)**.** *Task t_j^r is satisfied if*

$$\sum_{i \in \mathcal{W}^r} x_{i,j}^r \mu_i^r \geq Q_j^r. \tag{6}$$

In Definition 2 we use μ_i^r to completely characterize worker i's quality of information. We call μ_i^r the *estimated quality* of worker i.

Based on above conceptions, we present the definition of requester's utility in Definition 3.

Definition 3 (A Requester's Utility)**.** *The utility of the requester in run r is the total number of satisfied tasks:*

$$U^r = \sum_{t_j^r \in \mathcal{T}^r} I(\sum_{i \in \mathcal{W}^r} x_{i,j}^r \mu_i^r \geq Q_j^r), \tag{7}$$

where $I(\cdot)$ is indicator function defined as $I(\mathrm{TRUE}) = 1$ and $I(\mathrm{FALSE}) = 0$.

D. Design Objectives of Platform

We formulate the *Single Run Auction* (SRA) problem as the following linear programming:

Definition 4 (The SRA Problem)**.** *In run r, the platform wishes to maximize requester's utility under the constraints of requester's budget and workers' bids:*

$$\max \quad U^r \tag{8}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{W}^r, t_j^r\in\mathcal{T}^r} p_{i,j}^r \le B^r \tag{9}$$

$$\sum_{t_j^r\in\mathcal{T}^r} x_{i,j}^r \le n_i^r, \qquad \forall i \in \mathcal{W}^r \tag{10}$$

$$x_{i,j}^r \in \{0,1\}, \qquad \forall i \in \mathcal{W}^r, \forall t_j^r \in \mathcal{T}^r \tag{11}$$

Due to the NP-hardness of the SRA problem (see Theorem 1) and to improve practicability of the platform, we aim to design MELODY satisfying the following desirable properties:

- **Truthfulness:** A mechanism is *truthful* if reporting the true bid is a *dominant strategy* for all workers. Formally, A mechanism is truthful if for every worker $i \in \mathcal{W}^r$ with true bid $\overline{b}_i^r$ and any set of bids $\mathcal{B}_{-i}^r$ posted by bidders in $\mathcal{W}^r\backslash\{i\}$, we have $u_i^r(\overline{b}_i^r, \mathcal{B}_{-i}^r) \ge u_i^r(b_i^r, \mathcal{B}_{-i}^r)$ for any bid b_i^r of worker i.
- **Individual rationality:** A mechanism is *individual rational* if each participating worker $i \in \mathcal{W}^r$ will have a non-negative utility: $u_i^r \ge 0$.
- **Competitiveness:** We use notation *OPT* to denote the *optimal solution* that can be computed in the case where the platform has full knowledge about workers' true bids, and *MEL* to denote the outcome of MELODY. A mechanism is *competitive* if the approximation factor $\frac{OPT}{MEL} = O(1)$.
- **Computational efficiency:** A mechanism is *computationally efficient* if both the allocation and the payment schemes run polynomial time.
- **Long-term quality awareness:** A mechanism is *long-term quality-aware* if it is able to utilize workers' historical information according to its long-term characteristics when estimating workers' quality.

In the following sections, when our discussion is only on a fixed run (worker), the superscript r (subscript i) in all notations will be omitted for simplicity.

IV. MECHANISM DESIGN FOR SINGLE RUN AUCTION

In this section, we present the mechanism design of MELODY for the SRA problem. We first prove the NP-hardness of the SRA problem. The superscript r in all notations is omitted in this section.

Theorem 1. *The SRA problem is NP-hard.*

Proof: It is sufficient to demonstrate that the classical NP-hard *dual bin packing* (DBP, also known as *bin covering*) problem introduced in [26]–[28] is polynomial-time reducible to the SRA problem. An instance of the DBP problem is, given a set $\mathcal{I} = \{a_1, a_2, \ldots, a_n\}$ of *items*, a *size function* $s(a) > 0$ for each item a, and a bin capacity C, how to pack the elements of $\mathcal{I}$ into a maximum number of bins so that the sum of sizes in any bin is at least C. We now construct a corresponding instance of the SRA problem specified in Definition 4. We first transform the item set $\mathcal{I}$ and size function $s(a)$ into workers' set $\mathcal{W}^r$ and the estimated quality μ_i^r, respectively. Next we set $p_{i,j}^r = 0$, $n_i^r = 1$, and $Q_j^r = C$ for all $i \in \mathcal{W}^r$ and $t_j^r \in \mathcal{T}^r$. Now it's easy to see that if there is a subroutine to solve the SRA problem, it could be used in polynomial time (excluding the time within the subroutine) to solve the DBP problem. Therefore, every instance of the DBP problem is polynomial-time reducible to an instance of the SRA problem. Since the DBP problem is NP-hard [28], we prove that the SRA problem is also NP-hard. ■

Algorithm 1 MELODY Design for the SRA Problem

Input: $\mathcal{W}_U$, $\mathcal{T}$, B;
Output: $\mathcal{X}$, $\mathcal{P}$;
1: $\mathcal{W} \leftarrow \{i \in \mathcal{W}_U | \Theta_m \le \mu_i \le \Theta_M \ and \ C_m \le c_i \le C_M\}$
2: Sort all $i \in \mathcal{W}$ in descending order of μ_i/c_i;
3: Sort all $t_j \in \mathcal{T}$ in ascending order of Q_j;
4: $x_{i,j} \leftarrow 0$, $p_{i,j} \leftarrow 0$, $P_j \leftarrow 0$ **for each** $i \in \mathcal{W}$, $t_j \in \mathcal{T}$;
5: **for all** $t_j \in \mathcal{T}$ **do**
6: Find the smallest k such that $\sum_{i\le k \ and \ n_i>0} \mu_i \ge Q_j$;
7: **if** such k exists **then**
8: **for all** i s.t. $i \le k \ and \ n_i > 0$ **do**
9: $x_{i,j} \leftarrow 1$;
10: $p_{i,j} \leftarrow \frac{c_{k+1}}{\mu_{k+1}}\mu_i$;
11: $n_i \leftarrow n_i - 1$, $P_j \leftarrow P_j + p_{i,j}$;
12: **end for**
13: **end if**
14: **end for**
15: $\mathcal{X} \leftarrow \emptyset$, $\mathcal{P} \leftarrow \emptyset$;
16: Remove all zero P_j and sort the remaining in ascending order;
17: **for all** P_j s.t. $P_j \le B$ **do**
18: $\mathcal{X} \leftarrow \mathcal{X} \cup \{x_{i,j} \mid x_{i,j} = 1, i \in \mathcal{W}\}$;
19: $\mathcal{P} \leftarrow \mathcal{P} \cup \{p_{i,j} \mid p_{i,j} > 0, i \in \mathcal{W}\}$;
20: $B \leftarrow B - P_j$;
21: **end for**
22: **return** $\mathcal{X}$, $\mathcal{P}$;

Due to the NP-hardness of SRA problem, it is inadvisable for us to compute the precise optimal solution in real scenario because of its exponentially increasing execution time with input scale. Therefore, inspired by the basic idea proposed in [29], we design an approximated greedy algorithm to solve the SRA problem in run r, as shown in Algorithm 1.

Algorithm 1 takes workers' set $\mathcal{W}_U$, tasks' set $\mathcal{T}$ and budget B as input, and outputs allocation scheme $\mathcal{X}$ and payment scheme $\mathcal{P}$. Firstly algorithm 1 determines qualified workers' set $\mathcal{W}$ in this run according to an acceptable interval of quality $[\Theta_m, \Theta_M]$ and an acceptable interval of bid of cost $[C_m, C_M]$ (line 1). The reasons for setting such acceptable intervals lie

in that: 1) Θ_m is used to ensure quality of selected workers; 2) Θ_M is implied by the maximum of scores generated by rating methods; 3) C_m and C_M are used to exclude malicious workers whose bids are too low for completing the tasks and to ensure budget feasibility, respectively. Algorithm 1 is subsequently divided into two stages: *pre-allocation* (lines 2-14) and *scheme determination* (lines 15-21).

In *pre-allocation* stage, the algorithm first sorts all workers in $\mathcal{W}$ in descending order of their *estimated quality per unit cost* μ_i/c_i and all tasks in ascending order of their quality requirement Q_j (lines 2-3). The sorted $\mathcal{W}$ is also referred to as *ranking queue*. For each task t_j, the algorithm uses P_j to denote total amount the requester needs to pay for this task if it is finally selected (line 4). In the main loop (lines 5-14), the algorithm iterates among the tasks in sorted $\mathcal{T}$ and determines the scheme greedily based on the order of ranking queue. The algorithm, at the beginning of each iteration, finds top k available workers in ranking queue (if exist) whose total sum of estimated quality covers Q_j exactly (line 6). Then the algorithm chooses these workers as pre-allocation scheme for the task (line 9), determines their pre-payment (line 10), and updates their available frequency n_i and task t_j's P_j (line 11).

In *scheme determination* stage, the algorithm selects tasks as many as possible in ascending order of P_j until the sum of payment exceeds the given budget (lines 16-17). For each selected task, the algorithm adds its associated information to final allocation scheme $\mathcal{X}$ and payment scheme $\mathcal{P}$ (lines 18-19). Note that all $x_{i,j}$ and $p_{i,j}$ not included in final $\mathcal{X}$ and $\mathcal{P}$ equal 0.

It's easy to see that MELODY satisfies constraints (9)-(11) in Definition 4. We will further prove *truthfulness*, *individual rationality*, *competitiveness*, and *computational efficiency* of the mechanism in Section VI.

V. MECHANISM DESIGN FOR DYNAMIC QUALITY ESTIMATION

In this section we discuss the design of updating method for workers' quality between two consecutive runs in MELODY. The subscript i in all notations is omitted in this section.

According to our model, worker's quality in run r is represented as a latent random variable q^r with transition density $p(q^r|q^{r-1})$, and the observed set of scores $\mathcal{S}^r$ that the worker gets in run r follows emission density $p(\mathcal{S}^r|q^r)$. We propose a method of latent variables inference and parameters learning based on *Linear Dynamical Systems* (LDS) framework [17], [19], [20]. LDS is widely used in inference and recognition of temporal hidden states in linear dynamical models. In this paper we choose Gaussian distribution as the form of density, but it is worth noticing that we do not impose a strong assumption on the model since any other distribution in exponential family, e.g., Gamma, Poisson, or Beta distribution, could also be used here to derive the solution [19].

The transition and emission density, in the form of Gaussian distribution, are

$$p(q^r|q^{r-1}) \sim \mathcal{N}(q^r; aq^{r-1}, \gamma), \tag{12}$$

Algorithm 2 EM Algorithm for Parameters Learning

Input: $\mathbf{S^r}$;
Output: $\boldsymbol{\theta}$;
1: Initialize $\boldsymbol{\theta^0}$;
2: $k \leftarrow 0$;
3: **while** $\boldsymbol{\theta}$ not converge **do**
4: Compute $Q(\boldsymbol{\theta}, \boldsymbol{\theta^k}) \triangleq \mathbb{E}_{\mathbf{Q^r} \sim p(\mathbf{Q^r}|\mathbf{S^r};\boldsymbol{\theta^k})}[L(\mathbf{S^r}, \mathbf{Q^r}; \boldsymbol{\theta})]$;
5: $\boldsymbol{\theta^{k+1}} \leftarrow \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta^k})$;
6: $k \leftarrow k + 1$;
7: **end while**
8: **return** $\boldsymbol{\theta^k}$;

$$p(\mathcal{S}^r|q^r) = \prod_j p(s_j^r|q^r) \sim \prod_j \mathcal{N}(s_j^r; q^r, \eta), \tag{13}$$

respectively[6], where $\mathcal{N}(x; \mu, \delta)$ denotes a Gaussian distribution with mean μ and variance δ for variable x, a is *transition coefficient*, γ and η are variance of transition and emission density respectively. We do not set a corresponding *emission coefficient* in equation (13) (or in other words, the emission coefficient is set to 1), as quality and score are modeled in the same space scale. The posterior probability of q^r in run r, therefore, is also a Gaussian distribution

$$\widehat{\alpha}(q^r) \sim \mathcal{N}(q^r; \widehat{\mu}^r, \widehat{\sigma}^r). \tag{14}$$

In addition, we denote $\mathcal{N}(q^0; \widehat{\mu}^0, \widehat{\sigma}^0)$ the initial distribution of $\widehat{\alpha}(q^r)$ preset by the platform.

Given the above formalization, we now turn to two basic problems in our model: 1) The determination of worker parameters $\boldsymbol{\theta} = \{a, \gamma, \eta\}$ given sequence of the worker's score set $\mathbf{S^r} = \{\mathcal{S}^1, ..., \mathcal{S}^r\}$; 2) The inference of the posterior distribution $\widehat{\alpha}(q^r)$ given $\widehat{\alpha}(q^{r-1})$, $\mathcal{S}^r$ and $\boldsymbol{\theta}$. We will discuss the two problems in the following subsections respectively.

A. Parameters Learning

The purpose of parameters learning is to determine the worker parameters $\boldsymbol{\theta} = \{a, \gamma, \eta\}$ according to the sequence of worker's score set $\mathbf{S^r}$. Because the model has latent variables, this can be addressed using the Expectation Maximization (EM) algorithm [30]. The EM algorithm is an iterative method for finding the maximum likelihood estimation of parameters when the model contains unobserved data.

Consider the complete-data log likelihood function

$$\begin{aligned} &L(\mathbf{S^r}, \mathbf{Q^r}; \boldsymbol{\theta}) = \log p(\mathbf{S^r}, \mathbf{Q^r}; \boldsymbol{\theta}) \\ &= \sum_{t=1}^{r} \log p(q^t|q^{t-1}; a, \gamma) + \sum_{t=1}^{r} \log p(\mathcal{S}^t|q^t; \eta) + C, \end{aligned} \tag{15}$$

where $\mathbf{Q^r} = \{q^1, ..., q^r\}$ is sequence of worker's latent quality, and C consists of all terms independent of $\boldsymbol{\theta}$. We present the EM algorithm for parameters learning in Algorithm 2.

In iteration k of Algorithm 2, EM algorithm runs the following two steps: E-step computes the expected value of

[6]We assume that all s_j^r are i.i.d. conditioned on q^r for the worker in run r.

Algorithm 3 MELODY Design for Quality Updating

Input: $\widehat{\mu}^{r-1}$, $\widehat{\sigma}^{r-1}$, $\mathbf{S^r}$;
Output: $\widehat{\mu}^{r}$, $\widehat{\sigma}^{r}$, μ^{r+1};
1: **if** i is a new comer **then**
2: $\quad\widehat{\mu}^{r} \leftarrow \widehat{\mu}^{0}$, $\widehat{\sigma}^{r} \leftarrow \widehat{\sigma}^{0}$, $\mu^{r+1} \leftarrow a\widehat{\mu}^{0}$;
3: **else**
4: $\quad$Update $\widehat{\mu}^{r}$, $\widehat{\sigma}^{r}$, and μ^{r+1} according to equation (17)-(19) respectively;
5: **end if**
6: **if** $\boldsymbol{\theta}$ not updated for T runs **then**
7: $\quad\boldsymbol{\theta} \leftarrow$ Algorithm2($\mathbf{S^r}$);
8: **end if**
9: **return** $\widehat{\mu}^{r}$, $\widehat{\sigma}^{r}$, μ^{r+1};

likelihood function with respect to the conditional distribution of $\mathbf{Q^r}$ given observation $\mathbf{S^r}$ under current estimation $\boldsymbol{\theta^k}$ (line 4), and M-step finds the new estimation that maximizes the expectation function (line 5). Algorithm 2 provide a feasible method to learn the parameters $\boldsymbol{\theta} = \{a, \gamma, \eta\}$ of any worker. We can run Algorithm 2 whenever we wish to re-estimate the worker's parameters.

B. Quality Inference

The posterior probability $\widehat{\alpha}(q^r)$ reflects our knowledge about q^r based on observation $\mathbf{S^r}$. Therefore, after a new observation $\mathcal{S}^r$, we wish to update $\widehat{\alpha}(q^r)$ from $\widehat{\alpha}(q^{r-1})$, for given parameters setting. We give the theorem of quality inference in general form as follows:

Theorem 2 (Quality Inference in General Form)**.** *The recursion equation of the posterior probability $\widehat{\alpha}(q^r)$ is*

$$p(\mathcal{S}^r|\mathbf{S^{r-1}})\widehat{\alpha}(q^r) = p(\mathcal{S}^r|q^r)\int \widehat{\alpha}(q^{r-1})p(q^r|q^{r-1})\mathrm{d}q^{r-1}. \tag{16}$$

The proof of Theorem 2 is given in [31].

Theorem 2 provides a general formula for quality inference regardless of the choice of distribution for worker's quality. According to Theorem 2 and making use of equation (3), (12)-(14), we give the following theorem for the inference of worker's latent quality in the Gaussian form.

Theorem 3 (Quality Inference in the Gaussian Form)**.** *For any worker, given $\boldsymbol{\theta} = \{a, \gamma, \eta\}$, observed set of scores $\mathcal{S}^r$ in run r, and posterior probability $\widehat{\alpha}(q^{r-1}) \sim \mathcal{N}(q^{r-1}; \widehat{\mu}^{r-1}, \widehat{\sigma}^{r-1})$ in run $r-1$, the mean and variance of the posterior probability $\widehat{\alpha}(q^r)$ in run r is*

$$\widehat{\mu}^{r} = \frac{a\eta}{NK+\eta}\widehat{\mu}^{r-1} + \frac{K}{NK+\eta}S, \tag{17}$$

$$\widehat{\sigma}^{r} = \frac{K\eta}{NK+\eta}, \tag{18}$$

where $K = a^2\widehat{\sigma}^{r-1} + \gamma$ (here a^2 means a squared), $N = |\mathcal{S}^r|$ and $S = \sum_{s_j^r \in \mathcal{S}^r} s_j^r$. The mean of $\alpha(q_i^{r+1})$ (i.e., the estimated quality for the worker in run $r+1$) is

$$\mu^{r+1} = a\widehat{\mu}^{r}. \tag{19}$$

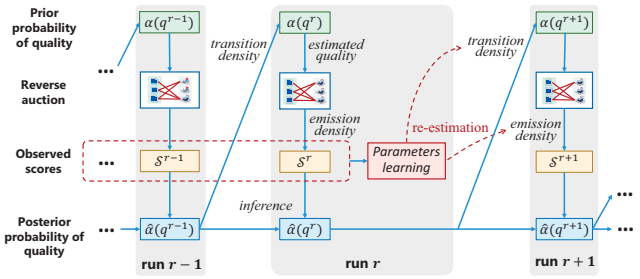


Fig. 3: The overall framework of MELODY. Arrows denote dependency relationship among random variables and algorithms. We assume that the platform decides to re-estimate parameters at the end of run r in this example.

The proof of Theorem 3 is given in [31].

C. MELODY Design Between Two Runs

We propose the updating method for workers' quality between two consecutive runs by combining parameters learning and quality inference in MELODY, as shown in Algorithm 3.

In algorithm 3, we first update every worker's quality based on the updating rules in Theorem 3 (line 1-5). After that, if $\boldsymbol{\theta}$ of this worker is not updated for T runs (T is set by the platform), run Algorithm 2 to re-estimate the worker's parameters (line 6-8). Note that smaller T will bring higher accuracy of the model because we re-estimate $\boldsymbol{\theta}$ more frequently, but meanwhile will increase the time overhead of the algorithm.

Fig. 3 summarizes the overall framework of MELODY. We put more emphasis on quality inference and parameters learning in Fig. 3, while aforementioned mechanism for task allocation and payment determination is miniatured as "reverse auction" in the second row.

VI. MECHANISM ANALYSIS

In this section we prove that the MELODY design for the SRA problem is truthful, individual rational, competitive, and computational efficient. The superscript r in all notations is omitted in this section.

Theorem 4. *The* MELODY *design for the SRA problem shown in Algorithm 1 is truthful.*

Proof: Since a worker's bid consists of cost and frequency, we need to prove the algorithm both *cost-truthful* and *frequency-truthful*:

1) Cost-truthfulness: Given the set of bids $\mathcal{B}_{-i}$ by $\mathcal{W}\backslash\{i\}$, we prove that worker i cannot improve $u_{i,j}$ by reporting untruthful cost whatsoever. That is, $u_{i,j}(\overline{c}_i) \geq u_{i,j}(c_i)$ holds in all cases below.

Case 1 & 2: Worker i wins (or loses) task t_j with both truthful bid $\overline{c}_i$ and untruthful bid c_i. In these two cases, we have $u_{i,j}(\overline{c}_i) = u_{i,j}(c_i)$ according to our payment rule.

Case 3: Worker i wins task t_j with truthful bid $\overline{c}_i$, but loses task t_j with untruthful bid c_i. In this case, we have $u_{i,j}(c_i) = 0$ and $u_{i,j}(\overline{c}_i) = p_{i,j} - \overline{c}_i = \frac{c_{k+1}}{\mu_{k+1}}\mu_i - \overline{c}_i \geq 0$ (because worker i's

winning with bid $\overline{c}_i$ implies $\frac{\mu_i}{\overline{c}_i} \geq \frac{\mu_{k+1}}{c_{k+1}}$ in the ranking queue). Thus we have $u_{i,j}(\overline{c}_i) \geq u_{i,j}(c_i)$.

Case 4: Worker i loses task t_j with truthful bid $\overline{c}_i$, but wins task t_j with untruthful bid c_i. In this case, we have $u_{i,j}(\overline{c}_i) = 0$ and $u_{i,j}(c_i) = p_{i,j} - \overline{c}_i = \frac{c_{k+1}}{\mu_{k+1}}\mu_i - \overline{c}_i \leq 0$ (because worker i's failure with bid $\overline{c}_i$ implies $\frac{\mu_i}{\overline{c}_i} \leq \frac{\mu_{k+1}}{c_{k+1}}$ in the ranking queue). Thus we have $u_{i,j}(\overline{c}_i) \geq u_{i,j}(c_i)$.

In summary, we have $u_{i,j}(\overline{c}_i) \geq u_{i,j}(c_i)$ for all task t_j. After summing up all the inequalities for j, we have $u_i(\overline{c}_i) \geq u_i(c_i)$.

2) Frequency-truthfulness: It is impossible for worker i to bid an untruthful frequency greater than $\overline{n}_i$ since $\overline{n}_i$ itself represents the maximum number of tasks he is willing to complete. On the other hand, worker i will not report an untruthful frequency less than $\overline{n}_i$ because it may reduce the number of tasks he may be assigned, as well as his utility. Hence, the mechanism is frequency-truthful. ■

Since c_i always equals $\overline{c}_i$ for worker i as we proved, c_i and $\overline{c}_i$ do not differ in rest of this paper. Neither do n_i and $\overline{n}_i$.

Theorem 5. *The* MELODY *design for the SRA problem shown in Algorithm 1 is individual rational.*

Proof: From Theorem 4 we know that workers bid truthfully in MELODY. Therefore, if task t_j is finally allocated to worker i with bid c_i and n_i (which implies $\frac{\mu_i}{c_i} \geq \frac{\mu_{k+1}}{c_{k+1}}$), we have $u_{i,j} = p_{i,j} - c_i = \frac{c_{k+1}}{\mu_{k+1}}\mu_i - c_i \geq 0$; otherwise, $u_{i,j} = 0$. Above all, we have $u_{i,j} \geq 0$ for all i and t_j. Then we have $u_i \geq 0$ for all $i \in \mathcal{W}$. ■

Before proving competitiveness, we first introduce several lemmas and notations: P_j is the total pre-payment for task t_j according to payment rule in Algorithm 1; P_j^{OPT} is the total payment of task t_j in optimum solution; $\mathcal{T}^{CAN}$ (candidate tasks set) is the set of tasks t_j with $P_j > 0$ by MELODY; $\mathcal{T}^{MEL}$ is the set of tasks finally chosen by MELODY (clearly $\mathcal{T}^{MEL} \subseteq \mathcal{T}^{CAN}$); $\mathcal{T}^{OPT}$ is the set of tasks chosen by optimum solution; $\mathcal{W}_j \triangleq \{i \in \mathcal{W} \mid x_{i,j} = 1\}$ is the set of workers pre-assigned with task t_j in Algorithm 1. We assume that $\mathcal{T}$ is sorted in ascending order of Q_j.

Lemma 1. *$P_j \leq \frac{C_M\Theta_M}{C_m\Theta_m}\sum_{i\in\mathcal{W}_j} c_i$ for all $t_j \in \mathcal{T}^{CAN}$.*

Proof: If task t_j is allocated to worker i then $p_{i,j} = \frac{c_{k+1}}{\mu_{k+1}}\mu_i$. Thus we have $\frac{p_{i,j}}{c_i} = \frac{c_{k+1}\mu_i}{c_i\mu_{k+1}} \leq \frac{C_M\Theta_M}{C_m\Theta_m}$. So the total payment for task t_j in the mechanism is $P_j = \sum_{i\in\mathcal{W}_j} p_{i,j} = \sum_{i\in\mathcal{W}_j} \frac{p_{i,j}}{c_i} c_i \leq \frac{C_M\Theta_M}{C_m\Theta_m}\sum_{i\in\mathcal{W}_j} c_i$. ■

Lemma 2. *$\sum_{i\in\mathcal{W}_j} c_i \leq \frac{C_M(Q_j+\Theta_M)\Theta_M}{C_mQ_j\Theta_m}P_j^{OPT}$ for all $t_j \in \mathcal{T}^{OPT} \cap \mathcal{T}^{CAN}$.*

Proof: For each task $t_j \in \mathcal{T}^{CAN}$, the total quality that workers may cover is at most $(Q_j + \Theta_M)$ according to our allocation rule, and the "cost density" $\frac{c_i}{\mu_i}$ over all workers is at most $\frac{C_M}{\Theta_m}$. Hence, an upper bound of $\sum_{i\in\mathcal{W}_j} c_i$ can be given as $\sum_{i\in\mathcal{W}_j} c_i = \sum_{i\in\mathcal{W}_j} \mu_i \frac{c_i}{\mu_i} \leq \frac{C_M}{\Theta_m}\sum_{i\in\mathcal{W}_j} \mu_i \leq \frac{C_M(Q_j+\Theta_M)}{\Theta_m}$.

Similarly, for each task $t_j \in \mathcal{T}^{OPT}$, the total quality that workers must cover is at least Q_j, and $\frac{c_i}{\mu_i}$ over all workers is at least $\frac{C_m}{\Theta_M}$. Note that in optimum solution the requester pays worker i his cost c_i exactly to save payment since the requester knows all private information of workers. So P_j^{OPT} has an lower bound $P_j^{OPT} \geq Q_j\frac{C_m}{\Theta_M}$.

In summary, we can derive the inequality in Lemma 2. ■

The final result can be obtained by combining the above two desired inequalities:

Lemma 3. *$P_j \leq \lambda P_j^{OPT}$ for all $t_j \in \mathcal{T}^{OPT} \cap \mathcal{T}^{CAN}$, where*

$$\lambda \triangleq \frac{C_M^2(\Theta_m + \Theta_M)\Theta_M^2}{C_m^2\Theta_m^3}. \tag{20}$$

Proof: After combining Lemma 1 and 2 for each task $t_j \in \mathcal{T}^{OPT} \cap \mathcal{T}^{CAN}$, we have $P_j \leq \frac{C_M\Theta_M}{C_m\Theta_m}\sum_{i\in\mathcal{W}_j} c_i \leq \frac{C_M\Theta_M}{C_m\Theta_m}\frac{C_M(Q_j+\Theta_M)\Theta_M}{C_mQ_j\Theta_m}P_j^{OPT} \leq \lambda P_j^{OPT}$. The last inequality holds because $Q_j \geq \Theta_m$ for all t_j. ■

We use notations $CAN \triangleq |\mathcal{T}^{CAN}|$, $MEL \triangleq |\mathcal{T}^{MEL}|$, and $OPT \triangleq |\mathcal{T}^{OPT}|$ in the following lemma and theorem.

Lemma 4. *$CAN \geq \frac{1}{\beta}OPT$, where β is a constant greater than 1.*

Proof: Let OPT' be the optimum solution of the SRA problem without budget constraint. It is clear that $OPT' \geq OPT$, so we only need to prove $CAN \geq \frac{1}{\beta}OPT'$. Note that now this problem completely degrades into the classical dual bin packing problem, of which the proof can be found in [26]. ■

Theorem 6. *The* MELODY *design for the SRA problem shown in Algorithm 1 has an approximation factor of $\lambda\beta = O(1)$.*

Proof: It's easy to see that $\mathcal{T}^{OPT}$ contains first OPT tasks in $\mathcal{T}$, $\mathcal{T}^{CAN}$ contains first CAN tasks in $\mathcal{T}$, and $\mathcal{T}^{MEL}$ contains top MEL tasks t_j with smallest P_j in $\mathcal{T}^{CAN}$. We discuss in two following cases.

Case 1: $CAN \geq OPT$ (which implies $\mathcal{T}^{OPT} \subseteq \mathcal{T}^{CAN}$). In this case we aim to prove that $MEL \geq \frac{1}{\lambda}OPT$. To achieve this, we take $\lfloor\frac{OPT}{\lambda}\rfloor$ smallest P_j from $\mathcal{T}^{CAN}$ as set S, and it's sufficient to prove that $\sum_{P_j\in S} P_j \leq B$. We further take out $\lfloor\frac{OPT}{\lambda}\rfloor$ smallest P_j from $\mathcal{T}^{OPT}$ as set S'. It is clear that $\sum_{P_j\in S} P_j \leq \sum_{P_j\in S'} P_j$ because $\mathcal{T}^{OPT} \subseteq \mathcal{T}^{CAN}$, so we only need to prove $\sum_{P_j\in S'} P_j \leq B$, which is achieved by contradiction. Suppose $\sum_{P_j\in S'} P_j > B$, then we have $\sum_{j=1}^{OPT} P_j \geq \lambda\sum_{P_j\in S'} P_j > \lambda B$. However, after listing and adding all inequalities in Lemma 3 for all $t_j \in \mathcal{T}^{OPT}$, we have $\sum_{j=1}^{OPT} P_j \leq \lambda\sum_{j=1}^{OPT} P_j^{OPT} \leq \lambda B$, which shows a contradiction.

Case 2: $CAN < OPT$ (which implies $\mathcal{T}^{CAN} \subset \mathcal{T}^{OPT}$). In this case we aim to prove that $MEL \geq \frac{1}{\lambda\beta}OPT$. Similar to the proof in Case 1, we take $\lfloor\frac{OPT}{\lambda\beta}\rfloor$ smallest P_j from $\mathcal{T}^{CAN}$ as set S, and it's sufficient to prove that $\sum_{P_j\in S} P_j \leq B$. Because $CAN \geq \frac{OPT}{\beta}$ according to Lemma 4, we have $\lfloor\frac{CAN}{\lambda}\rfloor \geq \lfloor\frac{OPT}{\lambda\beta}\rfloor$. So we further take $\lfloor\frac{CAN}{\lambda}\rfloor$ smallest P_j from $\mathcal{T}^{CAN}$ as set S'. Since we have $\sum_{P_j\in S} P_j \leq \sum_{P_j\in S'} P_j$, we only need to prove $\sum_{P_j\in S'} P_j \leq B$. This can be also achieved using the similar contradiction approach as in Case 1 expect that all $t_j \in \mathcal{T}^{OPT}$ are replaced by $t_j \in \mathcal{T}^{CAN}$.

Now we can conclude that $MEL \geq \frac{1}{\lambda\beta} OPT$, i.e., $\frac{OPT}{MEL} \leq \lambda\beta = O(1)$. ■

We denote $N \triangleq |\mathcal{W}|$ and $M \triangleq |\mathcal{T}|$ for Theorem 7.

Theorem 7. *The time complexity of* MELODY *design for the SRA problem shown in Algorithm 1 is* $O(NM)$.

Proof: The time complexity of line 2 is $O(N \log N)$, and the time complexity of lines 3-4 and 16 are all $O(M \log M)$. Consider the first loop in lines 5-14 for M iterations. During each iteration, the algorithm goes through every worker in lines 6 and 7-13 in the worst case. Therefore, the time complexity of this loop is $O(NM)$. Similar to the first loop, the time complexity of the second loop in lines 17-21 is also $O(NM)$. Thus we conclude that the time complexity is $O(N \log N + 3M \log M + 2NM) = O(NM)$. ■

VII. PERFORMANCE EVALUATION

In this section we evaluate the performance of aforementioned properties of MELODY: competitiveness and long-term quality awareness.

A. Competitiveness

To effectively evaluate the performance of our proposed mechanism, we implement MELODY against the following two benchmark mechanisms:

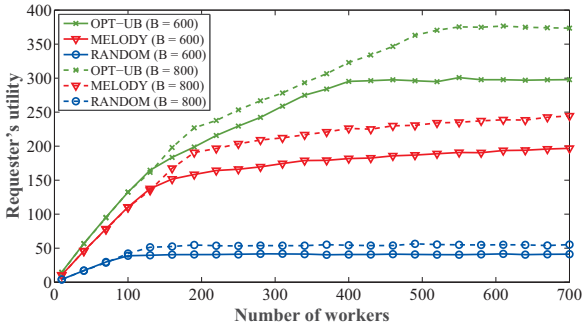
- OPT-UB is an estimated upper bound of the optimal solution of the SRA problem. Due to the fact that computing the optimal solution of the SRA problem is quite time consuming and cannot be computed in rational time as the scale of problem grows, we use an upper bound of optimum rather than optimum itself as the first benchmark.[7]
- RANDOM serves as a baseline solution for the SRA problem which selects tasks in random order. For each selected task t_j, RANDOM selects $k+1$ workers one by one randomly until the total quality of top k workers with higher $\frac{\mu_i}{c_i}$ just exceeds Q_j, i.e., $\sum_{i=1}^{k} \mu_i < Q_j$ and $\sum_{i=1}^{k+1} \mu_i \geq Q_j$ where all μ_i are sorted in descending order of $\frac{\mu_i}{c_i}$. Then these k workers win, each with payment $\mu_i \frac{c_{k+1}}{\mu_{k+1}}$, and the one with lowest $\frac{\mu_i}{c_i}$ is the only loser. The proof of truthfulness of RANDOM is given in [31].

We consider the experiment settings shown in Table III for our simulation of the SRA problem. In the two settings, we set budget (or number of workers) as fixed values and vary the number of workers (or budget). Values of μ_i, c_i, n_i, Q_j are drawn uniformly and randomly from the given ranges for each worker i and task t_j.

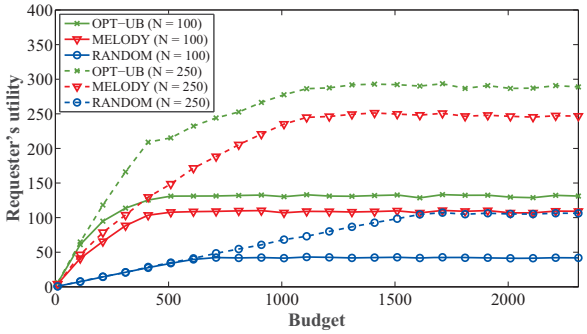
[7] OPT-UB is given as the largest i such that $\sum_{j \leq i} Q_j \leq Q_a$ (all Q_j are sorted in ascending order), where Q_a is the total available quality of workers with respect to budget constraint bounded by following three ways: (1) If $\sum_{i \in \mathcal{W}} n_i c_i \leq B$, then all bids are possibly acceptable and $Q_a = \sum_{i \in \mathcal{W}} n_i \mu_i$ is the corresponding estimated quality of all workers contributed to all bidding tasks. (2) If $\sum_{i \in \mathcal{W}} n_i c_i > B$, then we can modify the above Q_a by subtracting a correction item $(\sum_{i \in \mathcal{W}} n_i c_i - B) \cdot \min_{i \in \mathcal{W}} \frac{\mu_i}{c_i}$, which is the least loss of total available quality due to the budget constraint. (3) $Q_a = B \cdot \max_{i \in \mathcal{W}} \frac{\mu_i}{c_i}$ is an estimated total available quality considering the budget constraint directly. Overall, OPT-UB is the largest i such that $\sum_{j \leq i} Q_j \leq \min \left\{ \sum_{i \in \mathcal{W}} n_i \mu_i - \left[\sum_{i \in \mathcal{W}} n_i c_i - B \right]^+ \cdot \min_{i \in \mathcal{W}} \frac{\mu_i}{c_i}, B \cdot \max_{i \in \mathcal{W}} \frac{\mu_i}{c_i} \right\}$, where $[x]^+$ returns the larger value between x and 0.

TABLE III: Parameter settings for the SRA problem.

Settings	μ_i	c_i	n_i	Q_j	M	N	B
I	[2, 4]	[1, 2]	[1, 5]	[6, 12]	500	10 to 700	600, 800
II	[2, 4]	[1, 2]	[1, 5]	[6, 12]	500	100, 250	10 to 2310



(a) Fixed budget



(b) Fixed number of workers

Fig. 4: Requester's utility of the SRA problem.

The requester's utility of OPT-UB, MELODY and RANDOM is shown in Fig. 4. We observe from Fig. 4a (or 4b) that the requester's utility increases as the number of workers (or budget) grows, until reaching a saturation level due to the limit of budget (or number of workers). We can also learn from Fig. 4 that MELODY is close to optimum and much better than the baseline. More specifically, our simulation result shows that MELODY outperforms RANDOM by 259.2% on average, and achieves an approximation factor of 1.337 at most (as OPT-UB is an upper bound of the optimum), which is much smaller than the theoretical approximation factor of 48β ($\beta > 1$) under the given experiment scenario according to Lemma 3 and Theorem 6. This important observation exhibits the high competitiveness of MELODY in reality.

B. Long-term Quality Awareness

We implement the updating method for workers' quality of MELODY based on Algorithm 2 and 3 and compare MELODY with the following three benchmark methods:

TABLE IV: Parameter settings for workers' long-term quality updating.

Parameters	$s_{i,j}^r$	c_i^r	n_i^r	Q_j^r	M^r	N
Value	[1, 10]	[1, 2]	[1, 5]	[20, 40]	500	300
Parameters	B^r	runs	σ_S	μ_i^0	σ_i^0	
Value	800	1000	3	5.5	2.25	

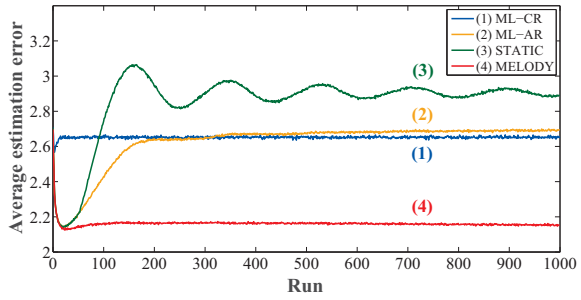
- STATIC is adopted in [7] which treats workers' quality as given values. In our simulation the STATIC mechanism calculates the value of workers' quality based on a few (50 in our simulation) runs at the beginning and keeps them unchanged during the remaining runs.
- ML-CR takes the Maximum Likelihood solution of Current Run as the estimated quality in next run (i.e., it is an over-fitting estimation). ML-CR is adopted in most existing crowdsourcing incentive mechanisms such as [5], [6], [9], [10], [12], [15].
- ML-AR takes the Maximum Likelihood solution of All Runs as the estimated quality in next run, which treats workers' historical scores with equal importance (i.e., it is an under-fitting estimation). ML-AR is used in [4], [11].

The performance metrics include average estimation error of quality per run and requester's utility per run. We first generate all latent quality q_i^r for each worker i in each run r, while the quality sequence of each worker follows a specific global pattern (rising, declining, fluctuating or stable) as shown in Fig. 1 with random noises. Based on each of the above four mechanisms, we perform the allocation and payment scheme in Algorithm 1 and update workers' quality in each run. We subsequently compute the average estimation error of quality as the average difference between workers' latent and estimated quality: $\frac{1}{|\mathcal{W}^r|}\sum_{i\in\mathcal{W}^r}|q_i^r-\mu_i^r|$, and requester's real utility as the number of tasks whose total obtained latent quality exceeds the threshold: $\sum_{t_j^r\in\mathcal{T}^r} I(\sum_{i\in\mathcal{W}^r} x_{i,j}^r q_i^r \geq Q_j^r)$.

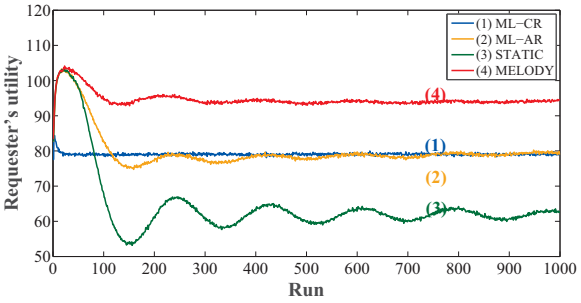
Experiment settings of workers' long-term quality updating are described in Table IV. We perform our simulation for 1000 runs. All scores $s_{i,j}^r$ are generated according to equation (13) within range [1, 10]. Finally, we set $T = 10$ in Algorithm 3 for MELODY.

The average estimation error of quality and the requester's utility per run are plotted in Fig. 5a and Fig. 5b, respectively. We observe that the performance of the four mechanisms are roughly the same at the beginning due to the same initial settings, and differs significantly in the long term. Our proposed MELODY outperforms the other three in terms of both the average estimation error of quality and the requester's utility per run.

To further evaluate MELODY quantitatively, we compute the average estimation error of quality and requester's utility of all runs for STATIC, ML-CR, ML-AR and MELODY. According to the results shown in Fig. 5, MELODY achieves an average requester's utility at 94.6, which outperforms the other three mechanisms by 46.6%, 19.7%, and 18.2%, respectively. Meanwhile, the estimation error of workers' quality is reduced by 24.2%, 18.5%, and 17.6%, respectively. Experimental results show the significant improvement of performance brought by MELODY.



(a) Average estimation error of quality per run



(b) Requester's utility per run

Fig. 5: Average estimation error of quality and requester's utility per run of workers' long-term quality updating.

VIII. CONCLUSION

In this paper we propose MELODY, a truthful and long-term quality-aware incentive mechanism for crowdsourcing, where workers' quality changes dynamically over time. We design an approximation algorithm for task allocation and payment scheme within a single run, and propose a quality inference and parameters learning framework for workers' long-term quality between two consecutive runs based on Linear Dynamical Systems and EM algorithm. We prove that MELODY has strong theoretical guarantees such as truthfulness and competitiveness. Furthermore, simulation results show that MELODY outperforms baseline method and existing work significantly.

ACKNOWLEDGMENT

We thank our anonymous reviewers for their feedback and suggestions. This work was partially sponsored by the National Basic Research 973 Program of China under Grant 2015CB352403, the National Natural Science Foundation of China (NSFC) (61602301), the NSFC/RGC Joint Research Scheme (RGC No: N_PolyU519/12), the NSFC Key Grant (No. 61332004), and the funding for Project of Strategic Importance provided by The Hong Kong Polytechnic University (Project Code: 1-ZE26).

REFERENCES

[1] Y. Singer and M. Mittal, "Pricing mechanisms for crowdsourcing markets," in *Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2013, pp. 1157–1166.

[2] D. Zhao, X.-Y. Li, and H. Ma, "How to crowdsource tasks truthfully without sacrificing utility: Online incentive mechanisms with budget constraint," in *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 1213–1221.

[3] K. Han, C. Zhang, J. Luo, M. Hu, and B. Veeravalli, "Truthful scheduling mechanisms for powering mobile crowdsensing," *Computers, IEEE Transactions on*, vol. 65, no. 1, pp. 294–307, 2016.

[4] Q. Zhang, Y. Wen, X. Tian, X. Gan, and X. Wang, "Incentivize crowd labeling under budget constraint," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015, pp. 2812–2820.

[5] H. Jin, L. Su, D. Chen, K. Nahrstedt, and J. Xu, "Quality of information aware incentive mechanisms for mobile crowd sensing systems," in *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. ACM, 2015, pp. 167–176.

[6] D. Peng, F. Wu, and G. Chen, "Pay as how well you do: A quality based incentive mechanism for crowdsensing," in *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. ACM, 2015, pp. 177–186.

[7] Q. Li, F. Ma, J. Gao, L. Su, and C. J. Quinn, "Crowdsourcing high quality labels with a tight budget," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 2016, pp. 237–246.

[8] L. Gao, F. Hou, and J. Huang, "Providing long-term participation incentive in participatory sensing," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015, pp. 2803–2811.

[9] I. Koutsopoulos, "Optimal incentive-driven design of participatory sensing systems," in *Infocom, 2013 proceedings ieee*. IEEE, 2013, pp. 1402–1410.

[10] S. Yang, F. Wu, S. Tang, X. Gao, B. Yang, and G. Chen, "Good work deserves good pay: A quality-based surplus sharing method for participatory sensing," in *Parallel Processing (ICPP), 2015 44th International Conference on*. IEEE, 2015, pp. 380–389.

[11] Y. Liu and M. Liu, "An online learning approach to improving the quality of crowd-sourcing," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 1. ACM, 2015, pp. 217–230.

[12] C. H. Liu, P. Hui, J. W. Branch, C. Bisdikian, and B. Yang, "Efficient network management for context-aware participatory sensing," in *Sensor, mesh and ad hoc communications and networks (secon), 2011 8th annual ieee communications society conference on*. IEEE, 2011, pp. 116–124.

[13] J. Wang, J. Tang, D. Yang, E. Wang, and G. Xue, "Quality-aware and fine-grained incentive mechanisms for mobile crowdsensing," in *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*. IEEE, 2016, pp. 354–363.

[14] C.-J. Ho, A. Slivkins, S. Suri, and J. W. Vaughan, "Incentivizing high quality crowdwork," in *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015, pp. 419–429.

[15] P. G. Ipeirotis, F. Provost, and J. Wang, "Quality management on amazon mechanical turk," in *Proceedings of the ACM SIGKDD workshop on human computation*. ACM, 2010, pp. 64–67.

[16] A. G. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh, and J. Widom, "Crowdscreen: Algorithms for filtering data with humans," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, 2012, pp. 361–372.

[17] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[18] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng, "Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks," in *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2008, pp. 254–263.

[19] T. Minka, "From hidden markov models to linear dynamical systems," Citeseer, Tech. Rep., 1999.

[20] Z. Ghahramani and G. E. Hinton, "Parameter estimation for linear dynamical systems," Technical Report CRG-TR-96-2, University of Totronto, Dept. of Computer Science, Tech. Rep., 1996.

[21] Y. Zhang and M. Van der Schaar, "Reputation-based incentive protocols in crowdsourcing applications," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 2140–2148.

[22] S. Jain, Y. Chen, and D. C. Parkes, "Designing incentives for online question and answer forums," in *Proceedings of the 10th ACM conference on Electronic commerce*. ACM, 2009, pp. 129–138.

[23] N. Anari, G. Goel, and A. Nikzad, "Mechanism design for crowdsourcing: An optimal 1-1/e competitive budget-feasible mechanism for large markets," in *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*. IEEE, 2014, pp. 266–275.

[24] C.-J. Ho, A. Slivkins, and J. W. Vaughan, "Adaptive contract design for crowdsourcing markets: Bandit algorithms for repeated principal-agent problems," *Journal of Artificial Intelligence Research*, vol. 55, pp. 317–359, 2016.

[25] W. Mason and S. Suri, "Conducting behavioral research on amazons mechanical turk," *Behavior research methods*, vol. 44, no. 1, pp. 1–23, 2012.

[26] J. Csirik, J. Frenk, S. Zhang, and M. Labbé, "Two simple algorithms for bin covering," *Acta Cybernetica*, vol. 14, no. 1, pp. 13–25, 1999.

[27] V. V. Vazirani, *Approximation algorithms*. Springer Science & Business Media, 2013.

[28] S. Assmann, D. S. Johnson, D. J. Kleitman, and J.-T. Leung, "On a dual version of the one-dimensional bin packing problem," *Journal of algorithms*, vol. 5, no. 4, pp. 502–525, 1984.

[29] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic game theory*. Cambridge University Press Cambridge, 2007, vol. 1.

[30] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal processing magazine*, vol. 13, no. 6, pp. 47–60, 1996.

[31] https://drive.google.com/file/d/0B5i-E0ujGnCoS2VNeThvbmtOV0U/view?usp=sharing.