IEEE 37th International Conference on Distributed Computing Systems
Atlanta, GA, USA

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

Hongwei Wang[1, 2], Song Guo[2], Jiannong Cao[2], Minyi Guo[1]

[1] Dept. of Computer Science and Engineering, Shanghai Jiao Tong University

[2] Dept. of Computing, The Hong Kong Polytechnic University

06/08/2017

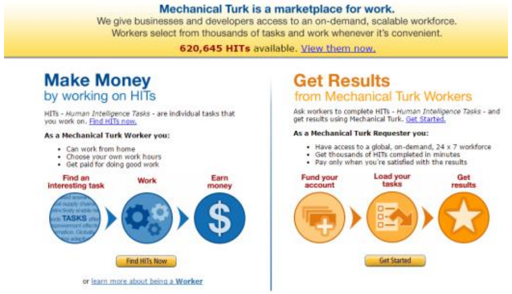
上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學

Background (1/3)

- **Crowdsourcing** allows requesters to allocate tasks to a group of workers on the Internet to make user of their collective intelligence

- Crowdsourcing tasks are typically …
  - small
  - quite difficult or too expensive to automate
  - simple for humans
  - e.g., proofreading and image labeling



Mechanical Turk is a marketplace for work.
We give businesses and developers access to an on-demand, scalable workforce.
Workers select from thousands of tasks and work whenever it's convenient.

620,645 HITs available. View them now.

Make Money
by working on HITs

HITs - *Human Intelligence Tasks* - are individual tasks that you work on. Find HITs now.

As a Mechanical Turk Worker you:

- Can work from home
- Choose your own work hours
- Get paid for doing good work

or learn more about being a **Worker**

Get Results
from Mechanical Turk Workers

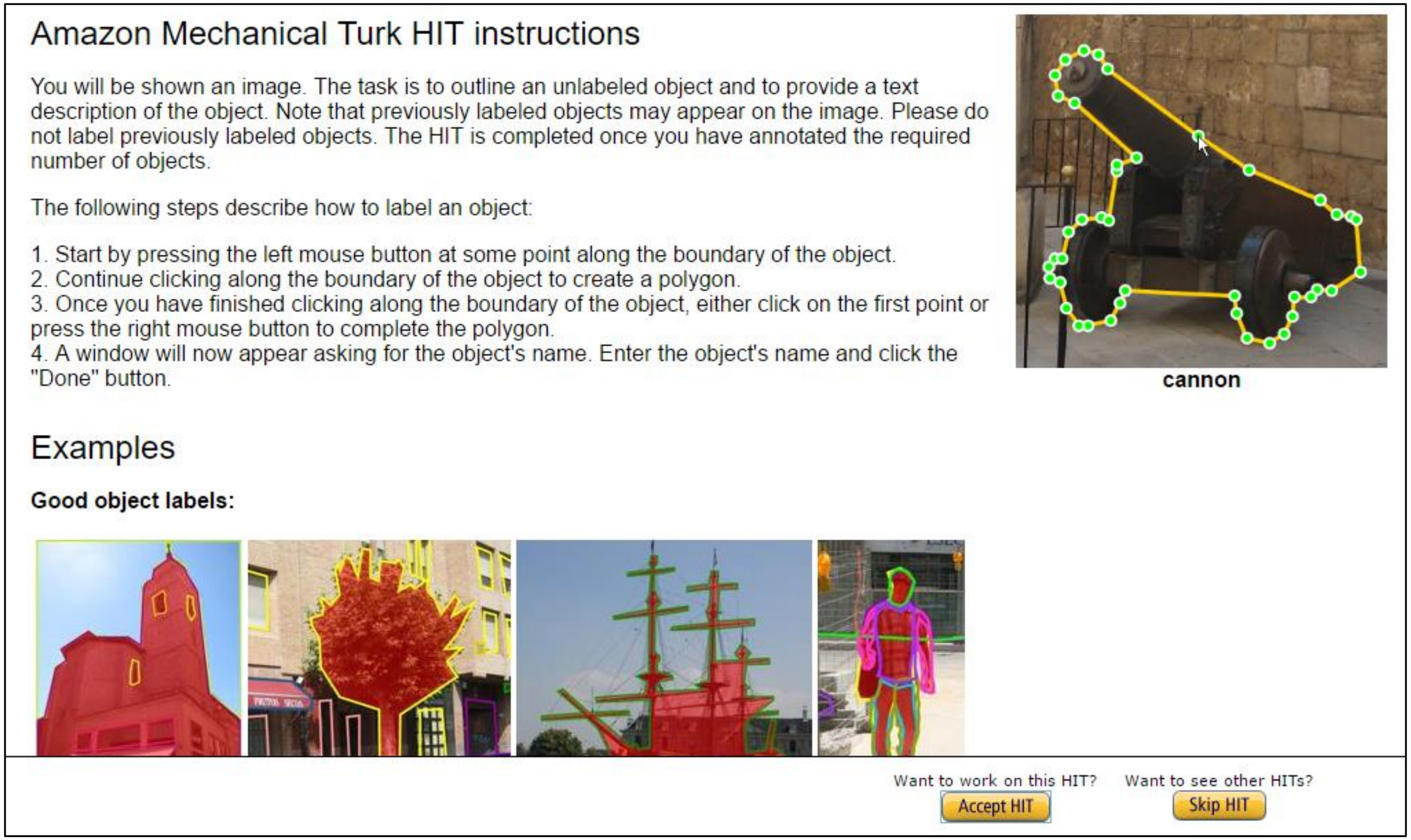
Ask workers to complete HITs - *Human Intelligence Tasks* - and get results using Mechanical Turk. Get Started.

As a Mechanical Turk Requester you:

- Have access to a global, on-demand, 24 x 7 workforce
- Get thousands of HITs completed in minutes
- Pay only when you're satisfied with the results

Background (2/3)

- AMT (Amazon Mechanical Turk)



Amazon Mechanical Turk HIT instructions

You will be shown an image. The task is to outline an unlabeled object and to provide a text description of the object. Note that previously labeled objects may appear on the image. Please do not label previously labeled objects. The HIT is completed once you have annotated the required number of objects.

The following steps describe how to label an object:

1. Start by pressing the left mouse button at some point along the boundary of the object.
2. Continue clicking along the boundary of the object to create a polygon.
3. Once you have finished clicking along the boundary of the object, either click on the first point or press the right mouse button to complete the polygon.
4. A window will now appear asking for the object's name. Enter the object's name and click the "Done" button.

Examples

Good object labels:

Want to work on this HIT?

Want to see other HITs?

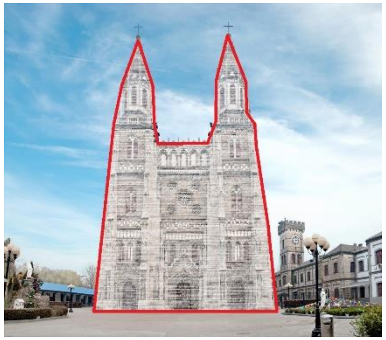
Background (3/3)

- **Quality Control** is one of the key considerations in crowdsourcing

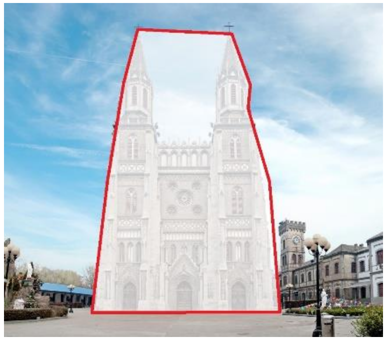
- Quality Control is nontrivial because …
  - crowd workers are at different levels of problem-solving capability, and may vary over time
  - it's hard to incorporate quality control naturally into designs of crowdsourcing platforms

We try to solve the above two challenging issues in this work

Object to be outlined



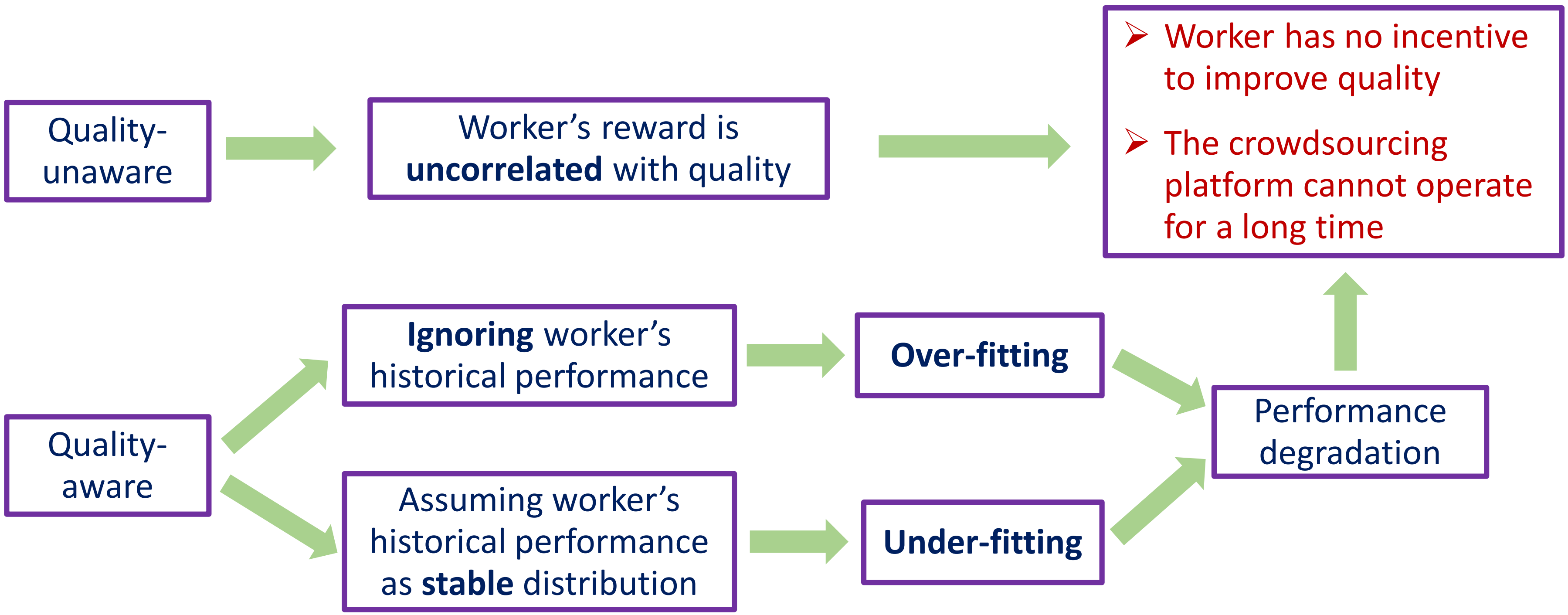
Good job



Bad job

Motivation (1/3)

- Existing incentive mechanisms for crowdsourcing are …



Quality-unaware

Worker's reward is **uncorrelated** with quality

- Worker has no incentive to improve quality
- The crowdsourcing platform cannot operate for a long time

Quality-aware

Ignoring worker's historical performance

Assuming worker's historical performance as **stable** distribution

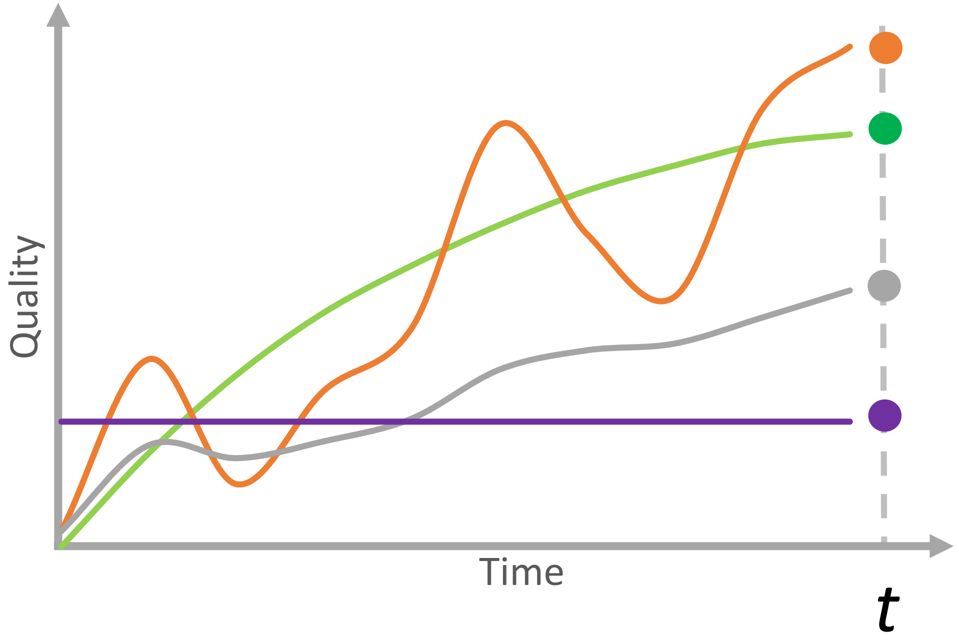
Over-fitting

Under-fitting

Performance degradation

Motivation (2/3)

What is the best prediction of worker's quality at time t based on his observed historical performance?



Observed performance
(over-fitting prediction)

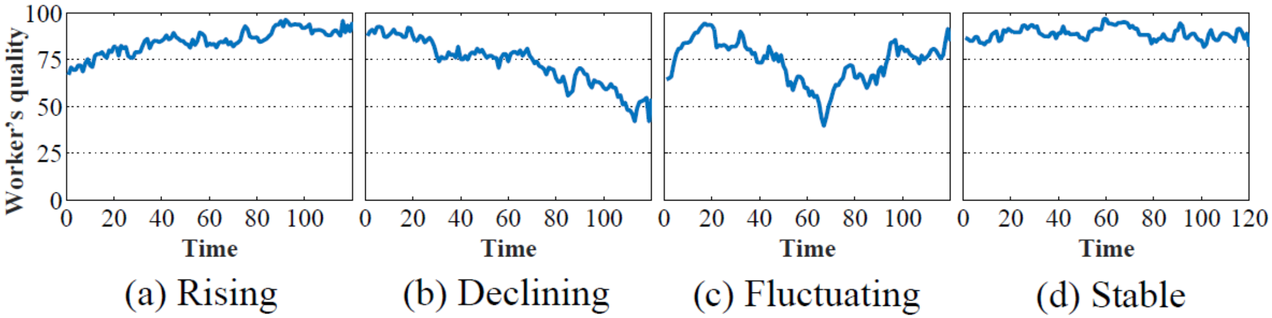
Latent quality
(optimal prediction)

Cumulative average
(under-fitting prediction)

Fixed value
(Quality-unaware prediction)

Motivation (3/3)

Can we really predict workers' quality based on their observed historical performance?



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

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.

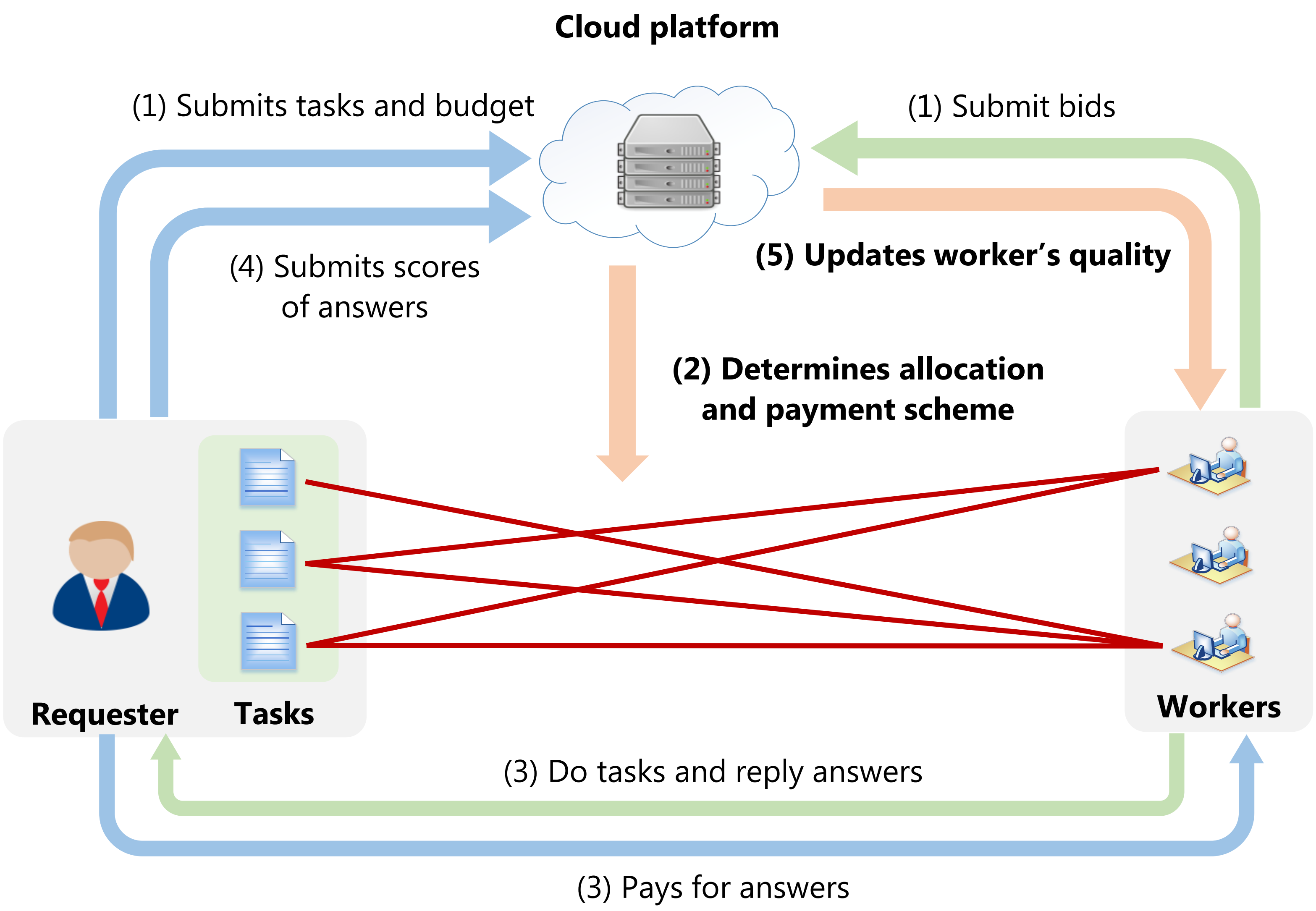
MELODY

So what is MELODY?

MELODY is an incentive **ME**chanism considering workers' **LO**ng-term **DY**namic quality for crowdsourcing markets which satisfies properties of:

- Truthfulness
- Individual rationality
- Competitiveness
- Computational efficiency
- Budget feasibility
- Long-term quality awareness

System Workflow in One Run



Cloud platform

(1) Submits tasks and budget

(1) Submit bids

(4) Submits scores of answers

(5) Updates worker's quality

(2) Determines allocation and payment scheme

Requester

Tasks

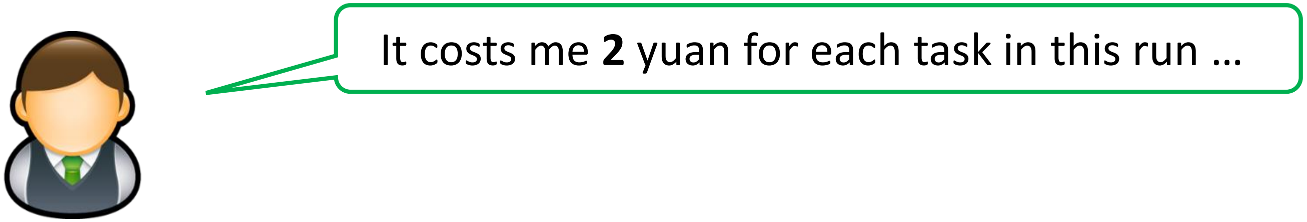
Workers

(3) Do tasks and reply answers

(3) Pays for answers

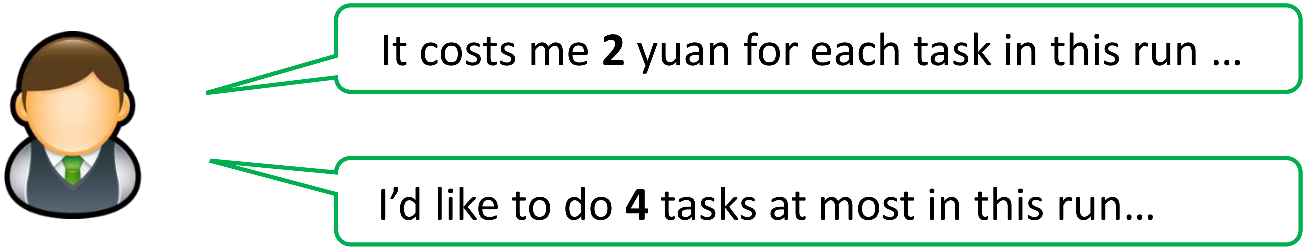
Worker Modeling

- During task allocation in run r, each worker i associates:
  - A bid of **cost** for performing every single task: c_i^r (may not be true)



Worker Modeling

- During task allocation in run r, each worker i associates:
  - A bid of **cost** for performing every single task: $\boldsymbol{c_i^r}$ (may not be true)
  - A bid of maximum number of tasks (also call **frequency**) he is willing to complete: $\boldsymbol{n_i^r}$ (may not be true)



It costs me **2** yuan for each task in this run …

I’d like to do **4** tasks at most in this run…

Worker Modeling

- During task allocation in run r, each worker i associates:
  - A bid of **cost** for performing every single task: $\boldsymbol{c_i^r}$ (may not be true)
  - A bid of maximum number of tasks (also call **frequency**) he is willing to complete: $\boldsymbol{n_i^r}$ (may not be true)
  - A **quality index** given by the platform: $\boldsymbol{\mu_i^r}$

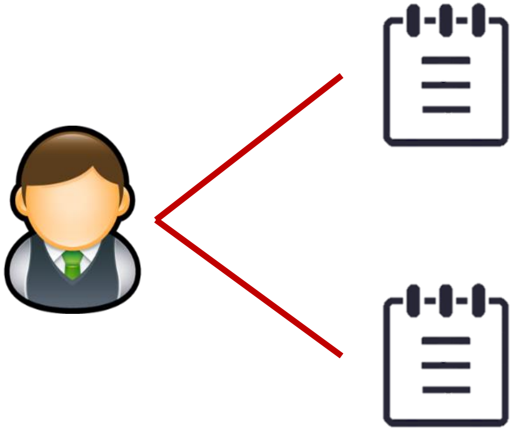
It costs me **2** yuan for each task in this run …

I'd like to do **4** tasks at most in this run…

Quality index: **4.2** / 5

Utility of Worker and Requester (1/3)

- **A worker *i*'s utility** is the difference between total payment he receives and his total cost

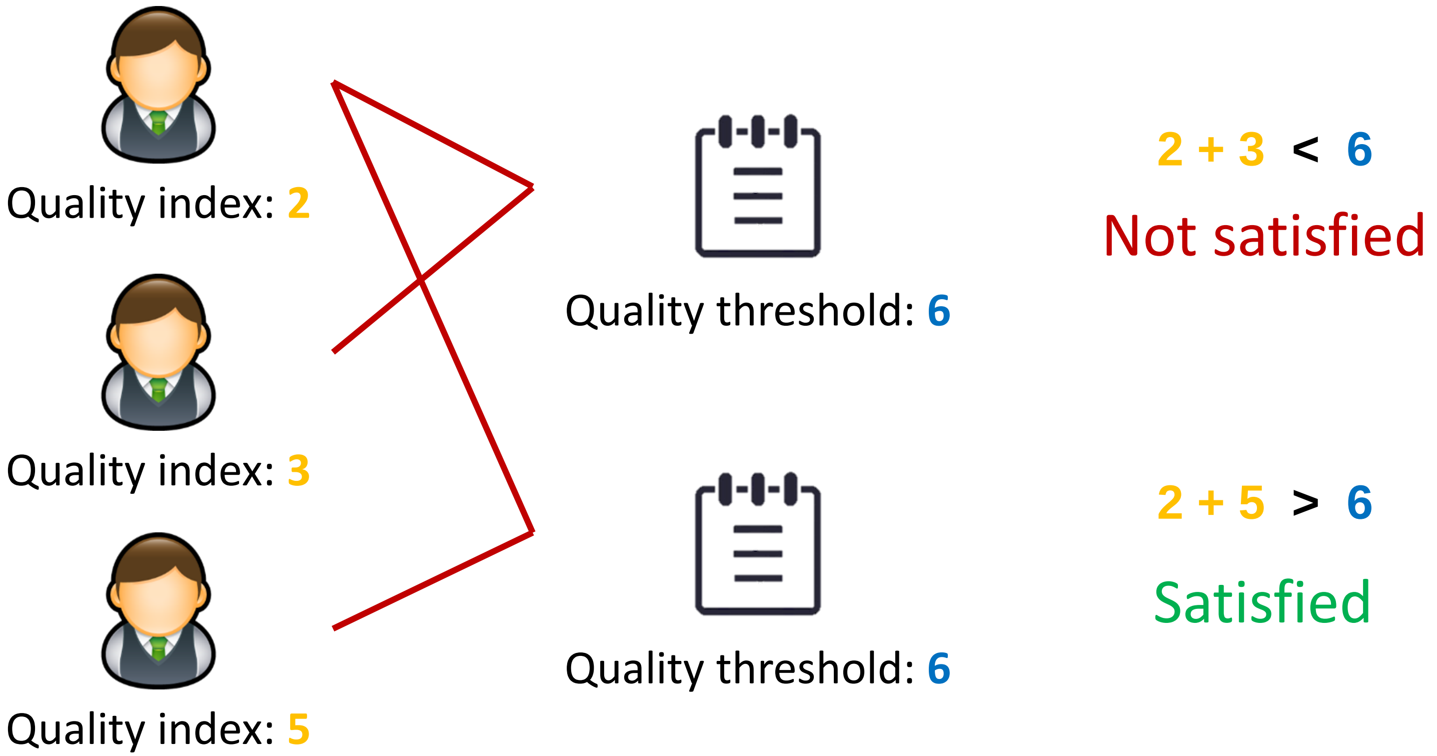


payment		cost		utility per task
4	-	2	=	2
3	-	2	=	1

Utility of this user: 2 + 1 = 3

Utility of Worker and Requester (2/3)

- A task j is **satisfied** if the integrated quality received by this task j is greater than a threshold Q



Quality index: 2

Quality index: 3

Quality index: 5

Quality threshold: 6

$2 + 3 < 6$

Not satisfied

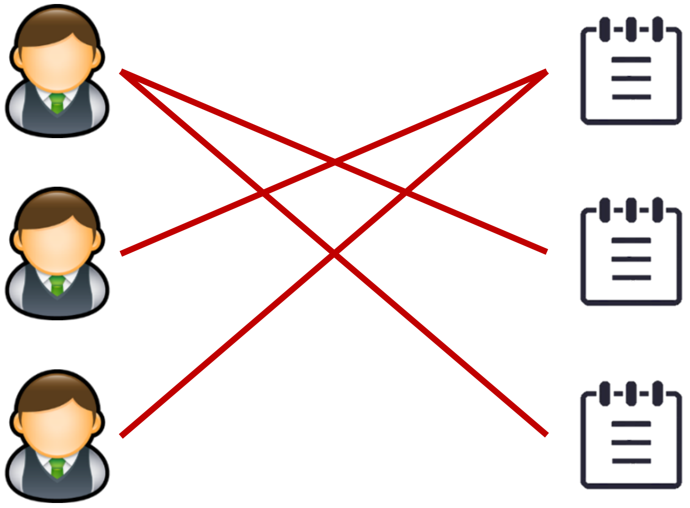
Quality threshold: 6

$2 + 5 > 6$

Satisfied

Utility of Worker and Requester (3/3)

- **A requester's utility** is the number of satisfied tasks



Satisfied

Satisfied

Not satisfied

Utility of the requester: 2

Design Objectives

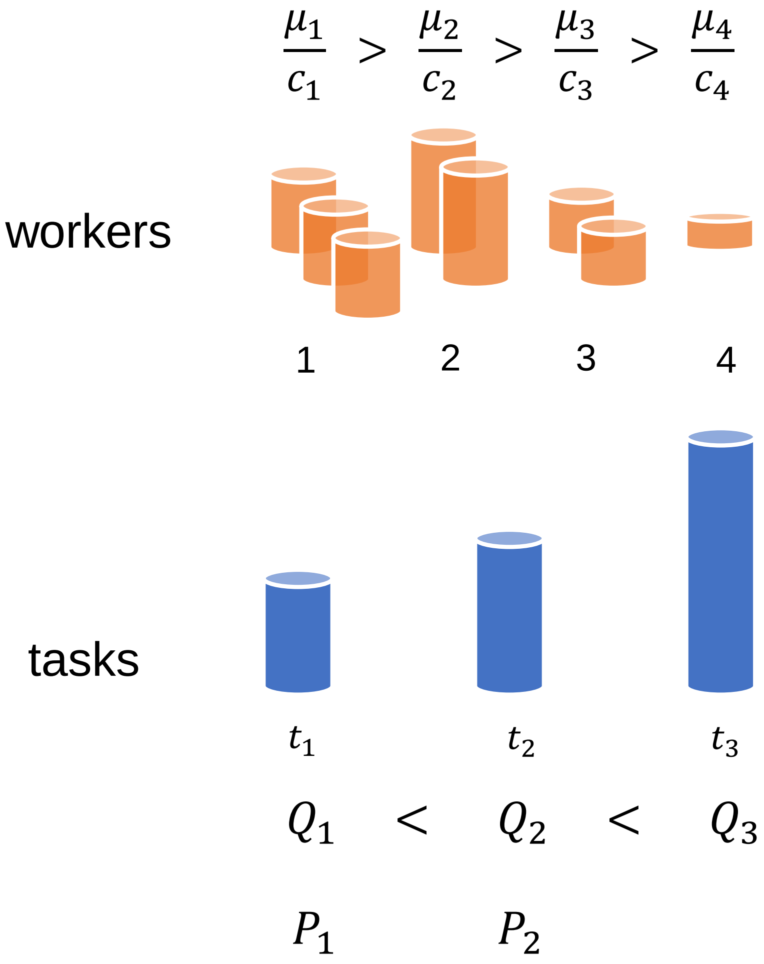
- The **Single Run Auction (SRA)** problem (NP-hard):

$$
\begin{aligned}
\max \quad & U^r \\
\text{s.t.} \quad & \sum_{i\in\mathcal{W}^r, t_j^r\in\mathcal{T}^r} p_{i,j}^r \le B^r \\
& \sum_{t_j^r\in\mathcal{T}^r} x_{i,j}^r \le n_i^r, && \forall i\in\mathcal{W}^r \\
& x_{i,j}^r \in \{0,1\}, && \forall i\in\mathcal{W}^r, \forall t_j^r\in\mathcal{T}^r
\end{aligned}
$$

- We hope to design MELODY satisfying …
  - **Truthfulness** (workers won’t lie about their bids)
  - **Individual rationality** (workers’ utilities are always non-negative)
  - **Competitiveness** (MELODY’s performance is close to optimal solution)
  - **Computational efficiency** (MELODY runs within polynomial time)
  - **Budget feasibility** (Budget constraint is hold)
  - **Long-term quality awareness** (MELODY can predict workers’ future quality according to its long-term characteristics)

MELODY Design for SRA (1/2)

- ❑ **Step 1**: sort all workers in descending order of μ_i/c_i;
- ❑ **Step 2**: sort all tasks in ascending order of Q_j;
- ❑ **Step 3**: Allocate available worker i to task t_j in the corresponding order, and pay $\boldsymbol{\frac{c_{k+1}}{\mu_{k+1}}\mu_i}$ for the allocation; ($k+1$ is the first worker who does not get task t_j)
- ❑ **Step 4**: Calculate the total payment for completing each task, and select as many tasks as possible under the given budget.


$\frac{\mu_1}{c_1} > \frac{\mu_2}{c_2} > \frac{\mu_3}{c_3} > \frac{\mu_4}{c_4}$
workers
1
2
3
4
tasks
t_1
t_2
t_3
$Q_1 < Q_2 < Q_3$
P_1
P_2

MELODY Design for SRA (2/2)

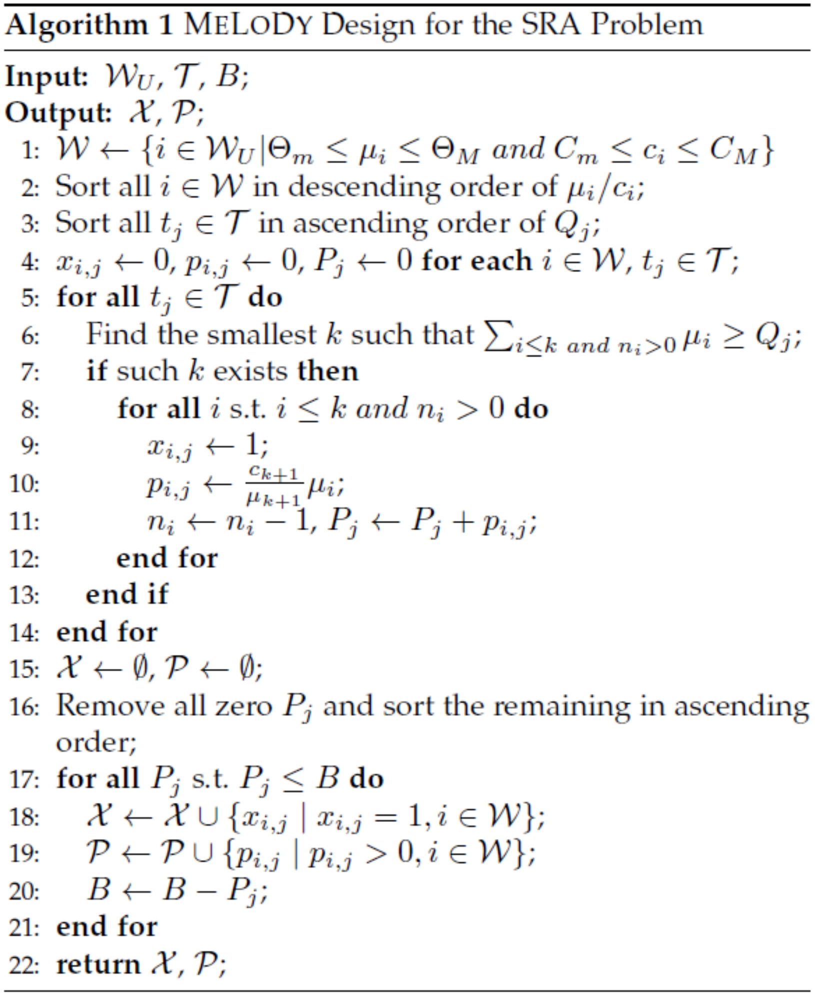
- **Step 1**: sort all workers in descending order of μ_i/c_i;

- **Step 2**: sort all tasks in ascending order of Q_j;

- **Step 3**: Allocate available worker i to task t_j in the corresponding order, and pay $\boldsymbol{\frac{c_{k+1}}{\mu_{k+1}}\mu_i}$ for the allocation; ($k+1$ is the first worker who does not get task t_j)

- **Step 4**: Calculate the total payment for completing each task, and select as many tasks as possible under the given budget.

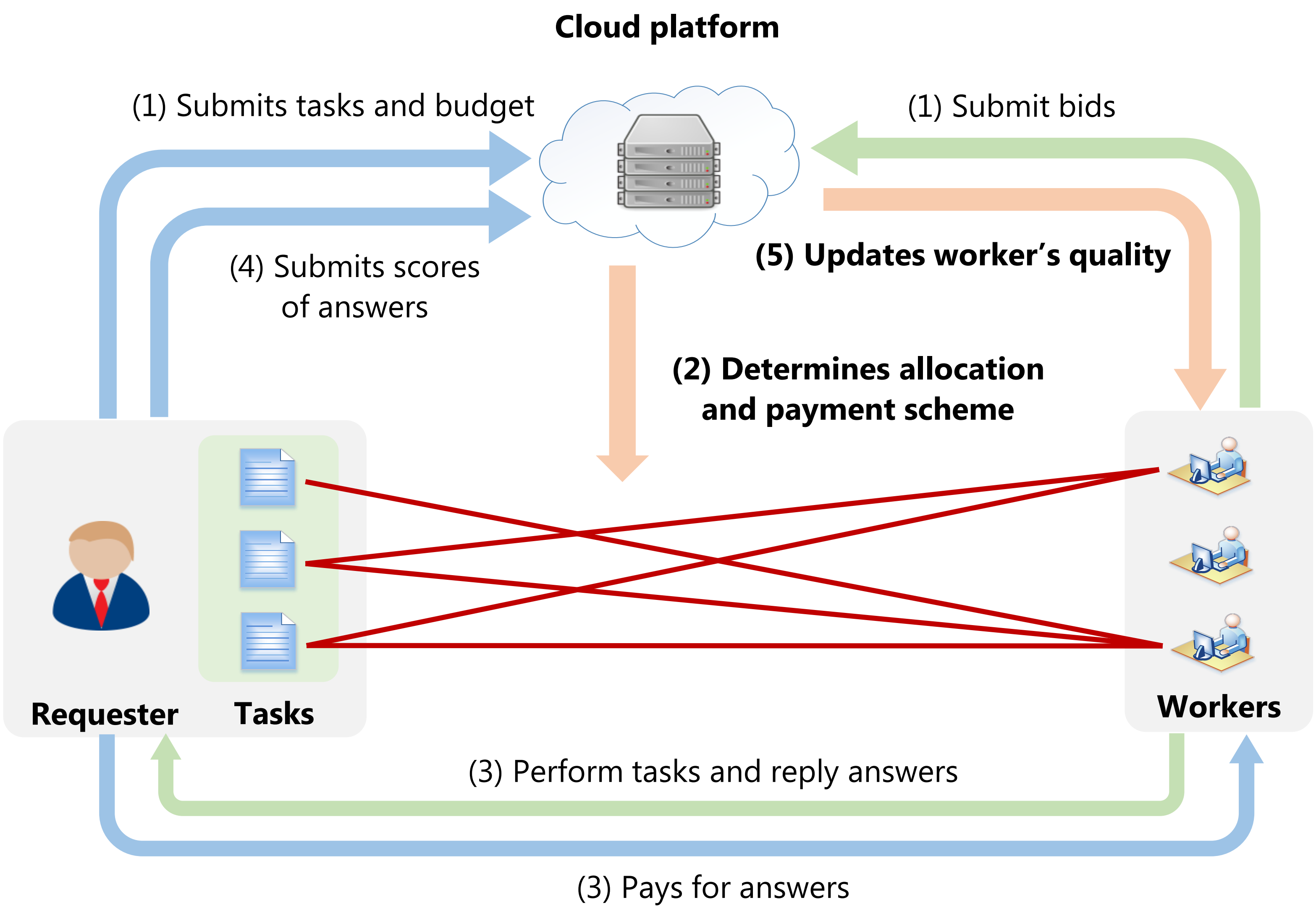
✓ **Truthfulness**
✓ **Individual rationality**
✓ **Competitiveness**
✓ **Computational efficiency**
✓ **Budget feasibility**



Algorithm 1 MELODY Design for the SRA Problem

```
Input: $\mathcal{W}_U$, $\mathcal{T}$, $B$;
Output: $\mathcal{X}$, $\mathcal{P}$;
$\mathcal{W} \leftarrow \{i \in \mathcal{W}_U | \Theta_m \le \mu_i \le \Theta_M \ and\ C_m \le c_i \le C_M\}$
Sort all $i \in \mathcal{W}$ in descending order of $\mu_i/c_i$;
Sort all $t_j \in \mathcal{T}$ in ascending order of $Q_j$;
$x_{i,j} \leftarrow 0$, $p_{i,j} \leftarrow 0$, $P_j \leftarrow 0$ for each $i \in \mathcal{W}$, $t_j \in \mathcal{T}$;
for all $t_j \in \mathcal{T}$ do
    Find the smallest $k$ such that $\sum_{i \le k \ and\ n_i > 0} \mu_i \ge Q_j$;
    if such $k$ exists then
        for all $i$ s.t. $i \le k \ and\ n_i > 0$ do
            $x_{i,j} \leftarrow 1$;
            $p_{i,j} \leftarrow \frac{c_{k+1}}{\mu_{k+1}}\mu_i$;
            $n_i \leftarrow n_i - 1$, $P_j \leftarrow P_j + p_{i,j}$;
        end for
    end if
end for
$\mathcal{X} \leftarrow \emptyset$, $\mathcal{P} \leftarrow \emptyset$;
Remove all zero $P_j$ and sort the remaining in ascending
order;
for all $P_j$ s.t. $P_j \le B$ do
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{x_{i,j} \mid x_{i,j} = 1, i \in \mathcal{W}\}$;
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{p_{i,j} \mid p_{i,j} > 0, i \in \mathcal{W}\}$;
    $B \leftarrow B - P_j$;
end for
return $\mathcal{X}$, $\mathcal{P}$;
```

System Workflow in One Run



Cloud platform

(1) Submits tasks and budget

(1) Submit bids

(4) Submits scores of answers

(5) Updates worker's quality

(2) Determines allocation and payment scheme

Requester

Tasks

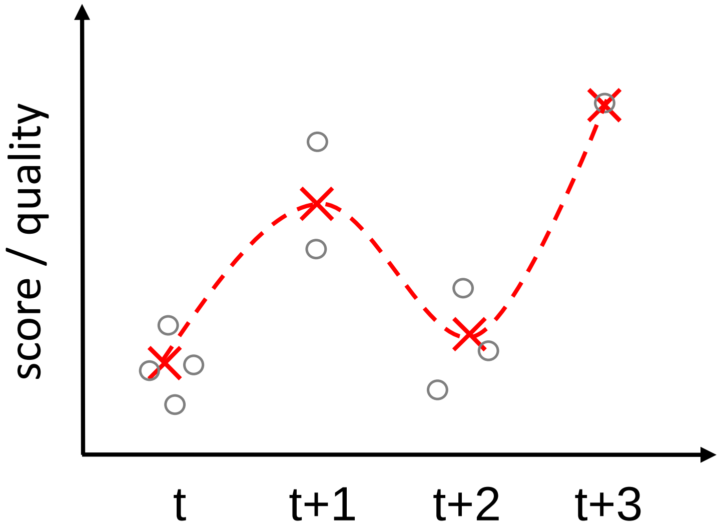
Workers

(3) Perform tasks and reply answers

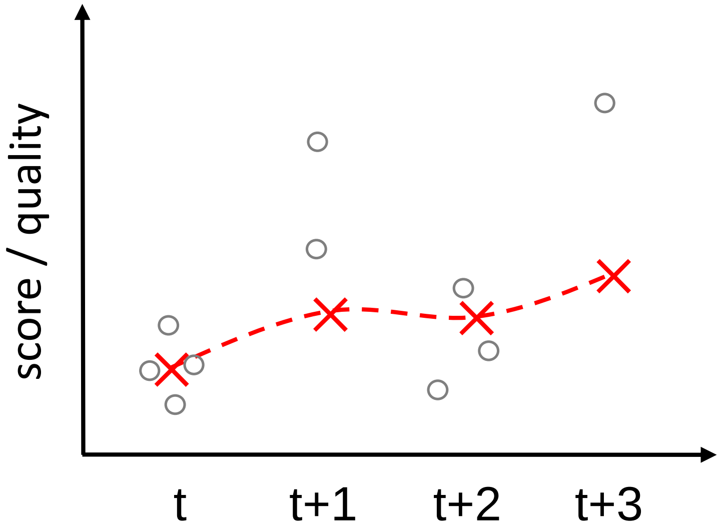
(3) Pays for answers

MELODY Design for Quality Updating (1/3)

Over-fitting and under-fitting



Per-run average

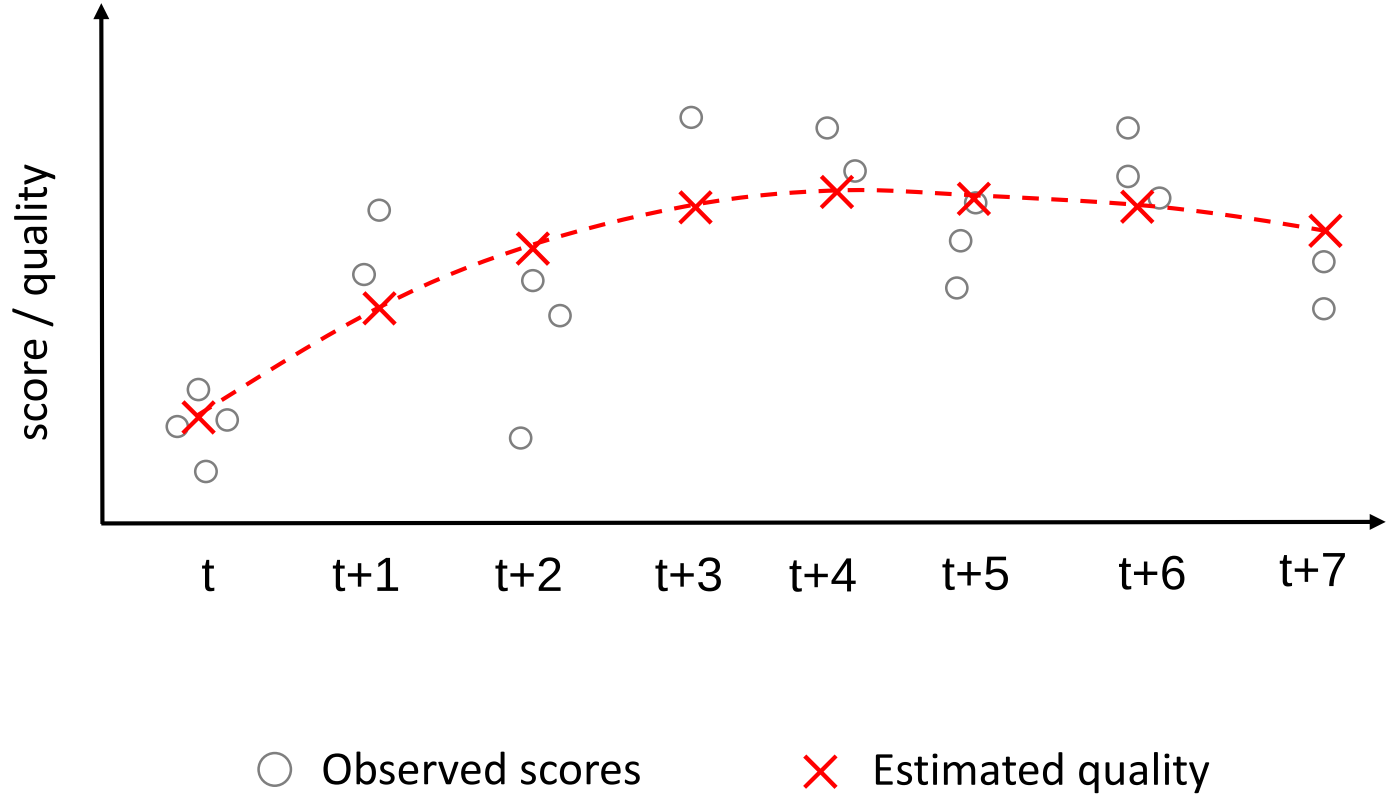


Cumulative average

○ Observed scores ✕ Estimated quality

MELODY Design for Quality Updating (2/3)

Linear dynamical systems

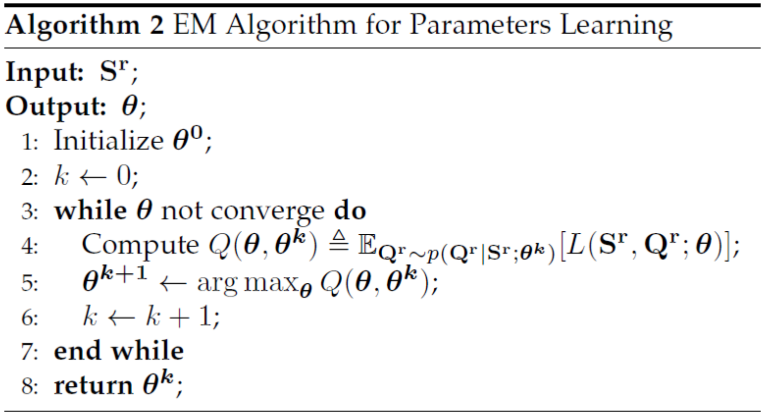


MELODY Design for Quality Updating (3/3)

- Hyper-parameter learning:
  Expectation Maximization (EM)
  - Complete-data likelihood function:

$$L(\mathbf{S^r},\mathbf{Q^r};\boldsymbol{\theta}) = \log p(\mathbf{S^r},\mathbf{Q^r};\boldsymbol{\theta})$$

$$= \sum_{t=1}^{r} \log p(q^t|q^{t-1};a,\gamma) + \sum_{t=1}^{r} \log p(\mathcal{S}^t|q^t;\eta) + C$$



Algorithm 2 EM Algorithm for Parameters Learning

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

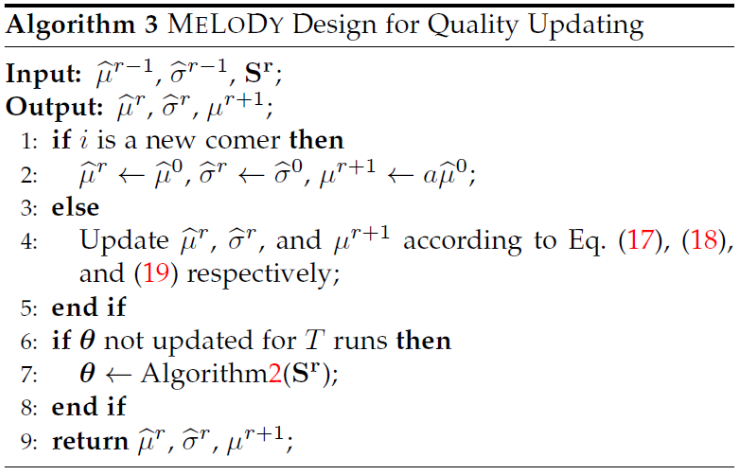
- Quality inference:
  Linear Dynamical Systems (LDS)
  - Updating formula:

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

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

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*

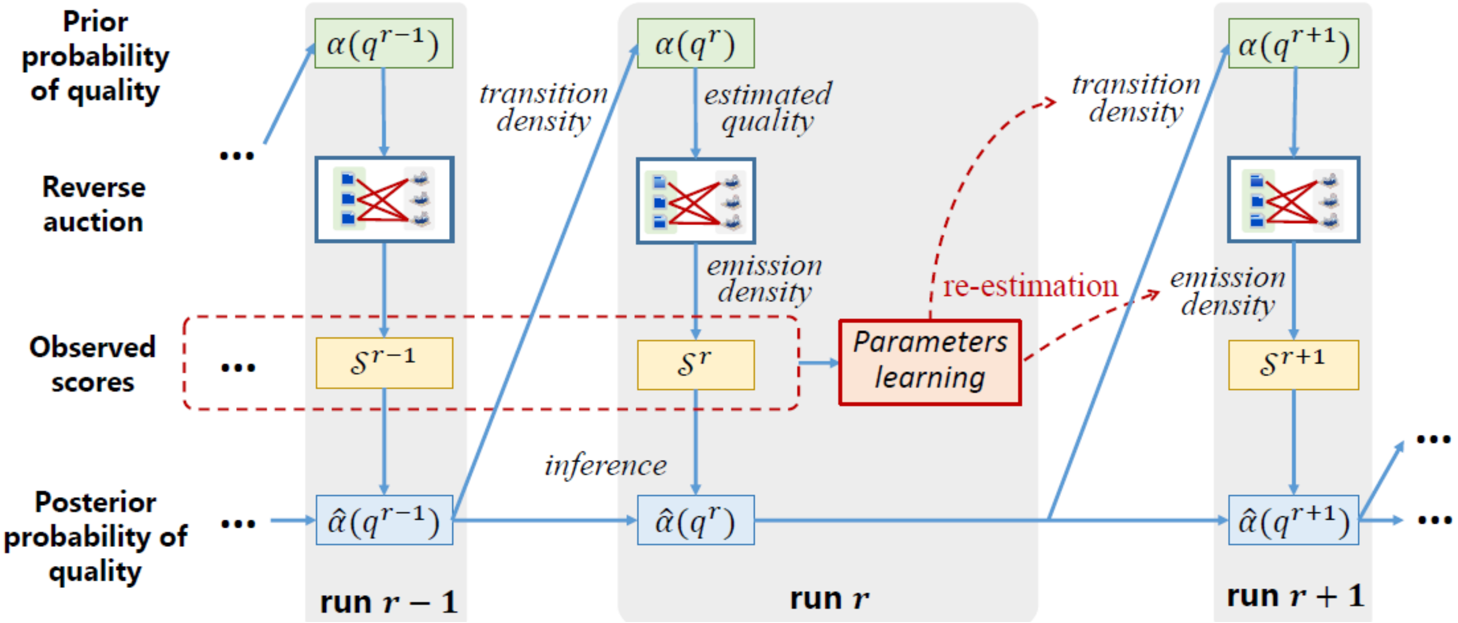
$$\mu^{r+1} = a\widehat{\mu}^r. \qquad (19)$$



Algorithm 3 MELODY Design for Quality Updating

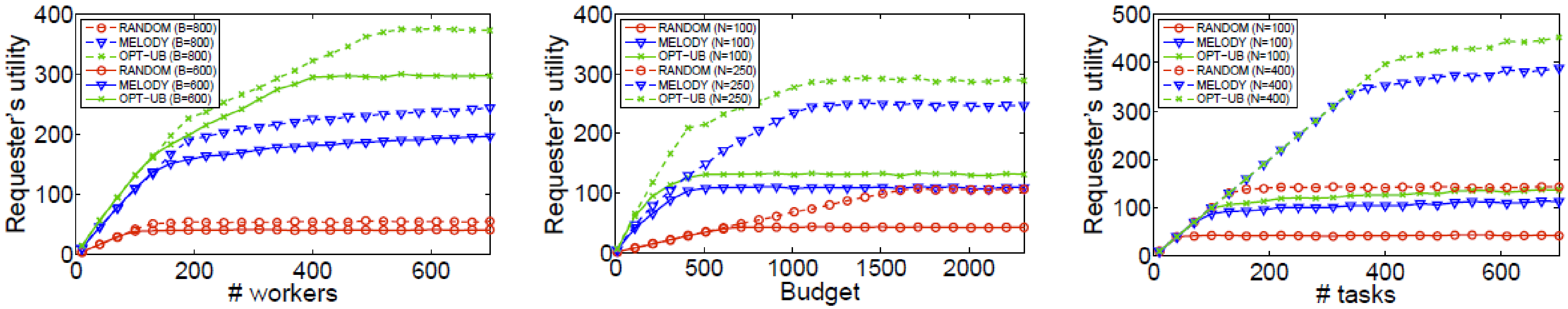
```
Input: $\widehat{\mu}^{r-1}$, $\widehat{\sigma}^{r-1}$, $\mathbf{S^r}$;
Output: $\widehat{\mu}^r$, $\widehat{\sigma}^r$, $\mu^{r+1}$;
if $i$ is a new comer then
    $\widehat{\mu}^r \leftarrow \widehat{\mu}^0$, $\widehat{\sigma}^r \leftarrow \widehat{\sigma}^0$, $\mu^{r+1} \leftarrow a\widehat{\mu}^0$;
else
    Update $\widehat{\mu}^r$, $\widehat{\sigma}^r$, and $\mu^{r+1}$ according to Eq. (17), (18),
    and (19) respectively;
end if
if $\boldsymbol{\theta}$ not updated for $T$ runs then
    $\boldsymbol{\theta} \leftarrow$ Algorithm2($\mathbf{S^r}$);
end if
return $\widehat{\mu}^r$, $\widehat{\sigma}^r$, $\mu^{r+1}$;
```

MELODY Framework

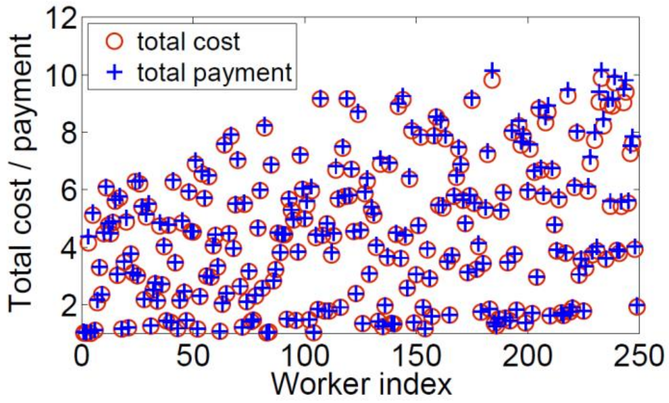


Performance Evaluation (1/3)

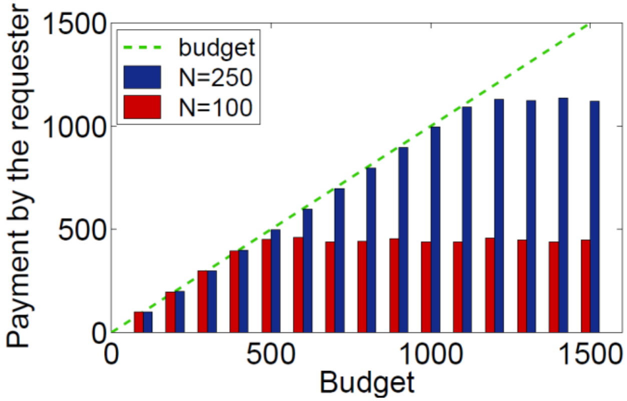
- Competitiveness



- Individual rationality check

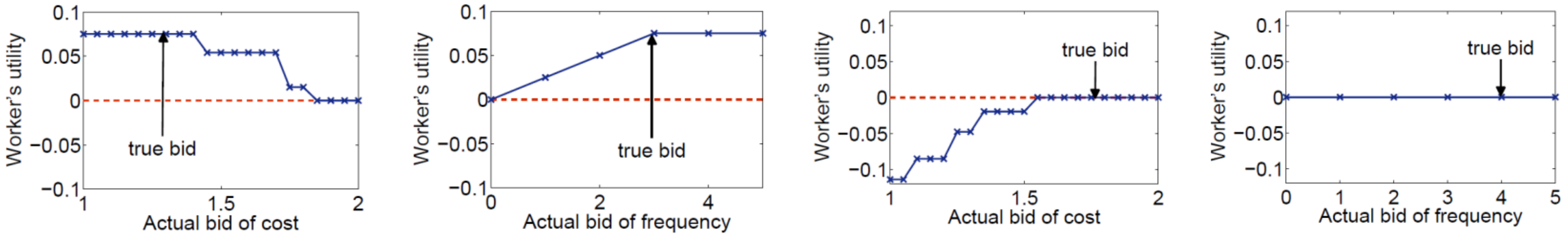


- Budget feasibility check



Performance Evaluation (2/3)

- Truthfulness check



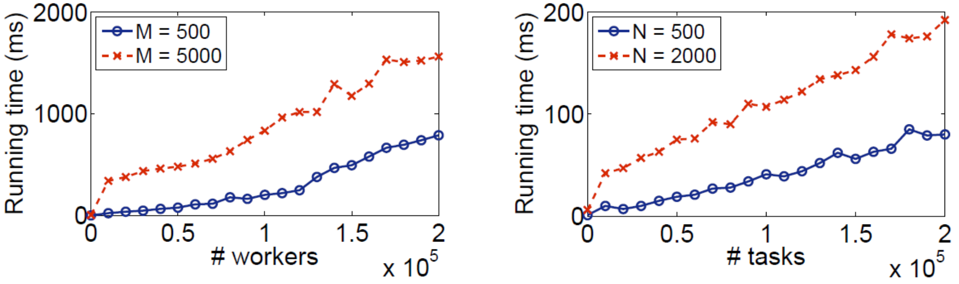
(a) Cost-truthfulness of a winner i

(b) Frequency-truthfulness of a winner i

(c) Cost-truthfulness of a loser j

(d) Frequency-truthfulness of a loser j

- Computational efficiency check

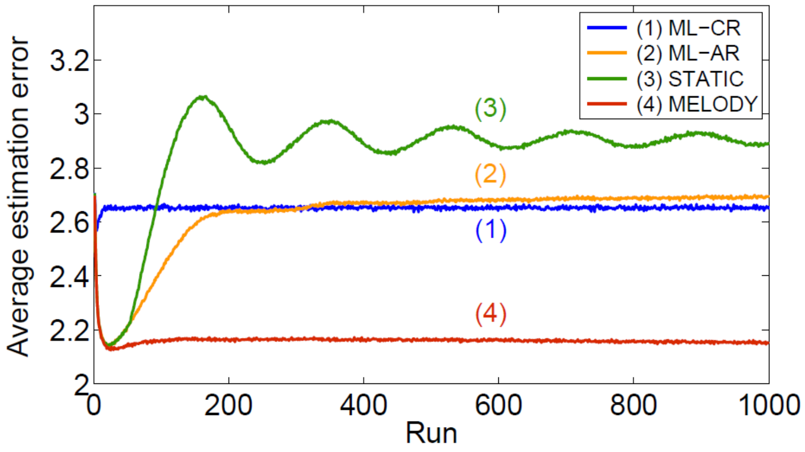


(a) Running time changing with the number of workers

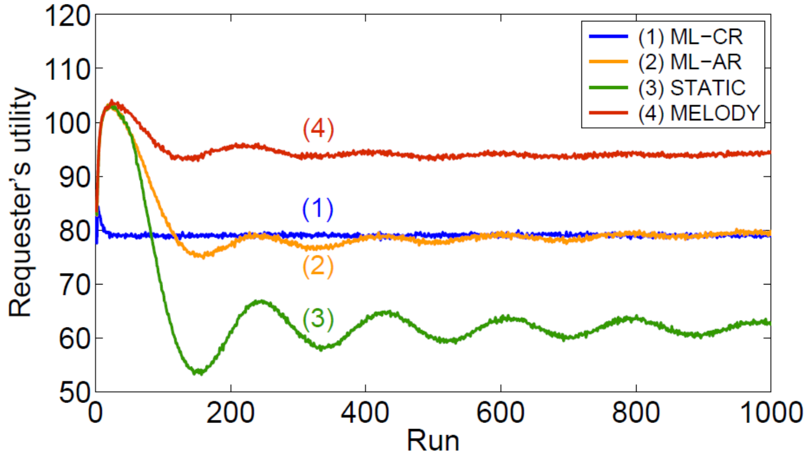
(b) Running time changing with the number of tasks

Performance Evaluation (3/3)

- Long-term quality awareness



(a) Average estimation error of quality per run



(b) Requester's utility per run

Q & A

Thanks!