# Week 11b - Fairness


Yu Han

han.yu@ntu.edu.sg

*Nanyang Assistant Professor*
*School of Computer Science and Engineering*
*Nanyang Technological University*


# Balancing Fairness and Efficiency in Decision Support Systems

# Decision Support Systems

- After analyzing patterns or making predictions with machine learning algorithms (e.g. deep learning), we still need to make decisions
- This often involves some types of resource allocation
- Reasons:
  - Agents face resource constraints
    - Not enough budget to execute all possible actions
    - Not enough time to try all possible alternatives
    - … …

# Our Scenario: The Sharing Economic

**Sharing of Private Resources**

When I buy a car, how often do I really use it?


The car is not used for over **90%** of the time



**Source:** The future business is the “mesh”

# Sharing Economy (those with resources)

**Sharing of Private Resources**

High Cost + Idling Capacity + Critical Mass

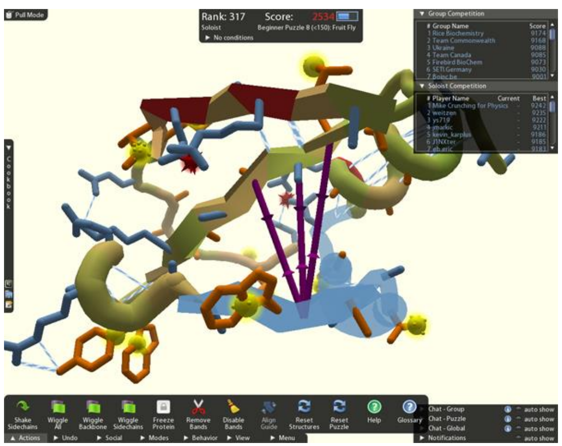
UBER
EVERYONE'S PRIVATE DRIVER™

airbnb

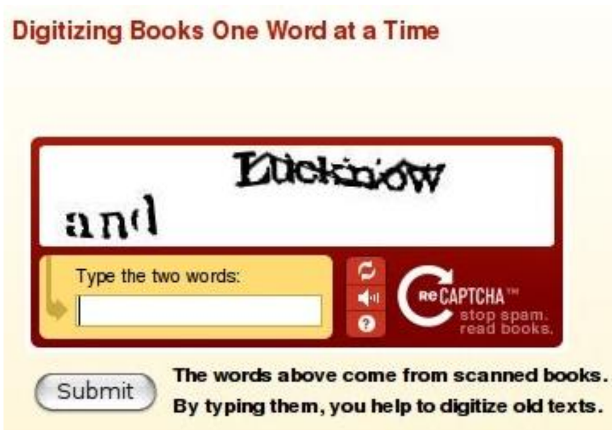
Side•car
a whole new way to get around

GRABTAXI

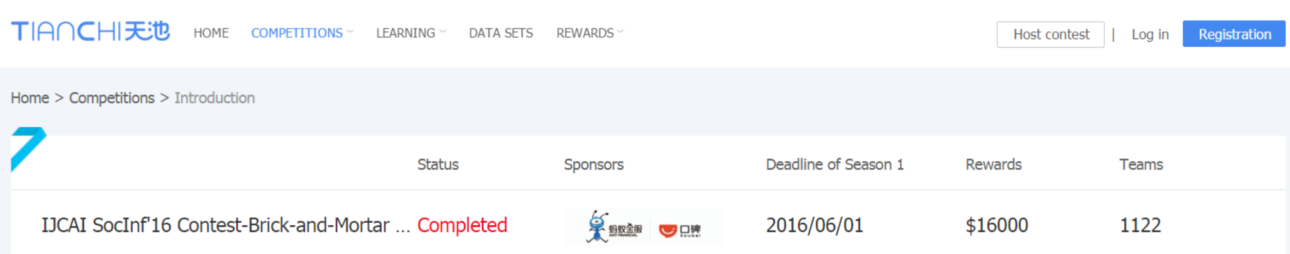
# Sharing Economy (those with skills)

**Foldit** – a game to crowdsource complex protein structures (***Nature* 466**, 756–760, 2010).




**reCAPTCHA** – digitalizing books via crowd-powered online security (***Science* 321**, 1465–1468, 2008).




**US$335 Billion**

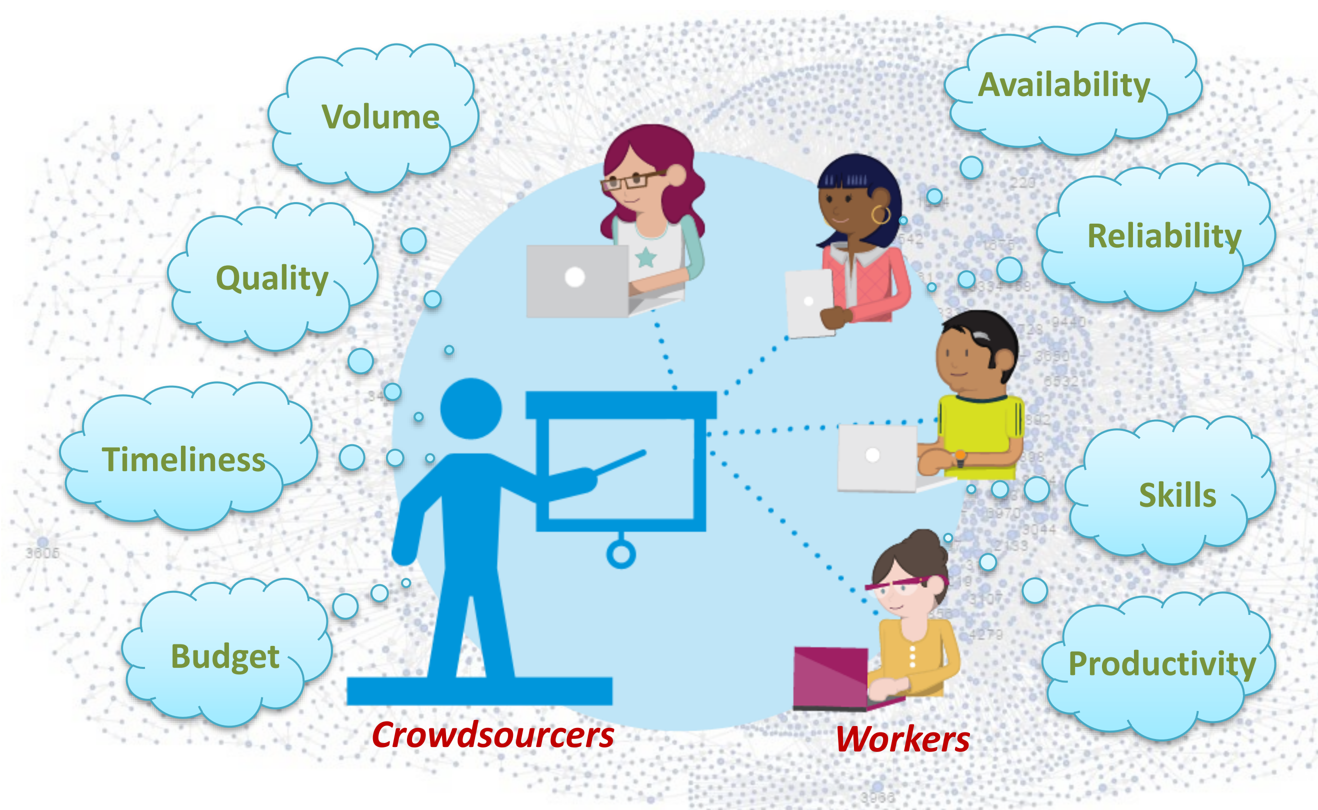
by 2025

# Challenges in Decision-making

amazonmechanical turk
Artificial Artificial Intelligence

**40%** of the results are deemed to be of low quality

**Lack of Efficiency & Quality Control**




**Difficulties**

1. Large-scale congestion game
2. Uncertainty & temporal variations in situational factors
3. Limited time for decision-making

# Challenges in Decision-making

- Crowdsourcing systems need efficiency in order to attract crowdsourcer who pay for their services

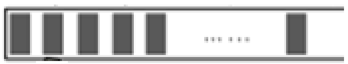
- Crowdsourcing systems need to treat workers fairly in order to attract and retain a large pool of skilled workers to satisfy crowdsourcers' requests

- How to match tasks to workers to achieve these two (possibly conflicting) long-term goals?

# Balancing Efficiency and Fairness

- Here:
  - We adopt the definition of “Fairness through Unawareness”
  - Develop a smart task allocation algorithm that balances efficiency and fairness considerations
  - Allow “human-over-the-loop” intervention through individual preference statements

# Queueing System Modelling



Pending Task Queue

**Worker**

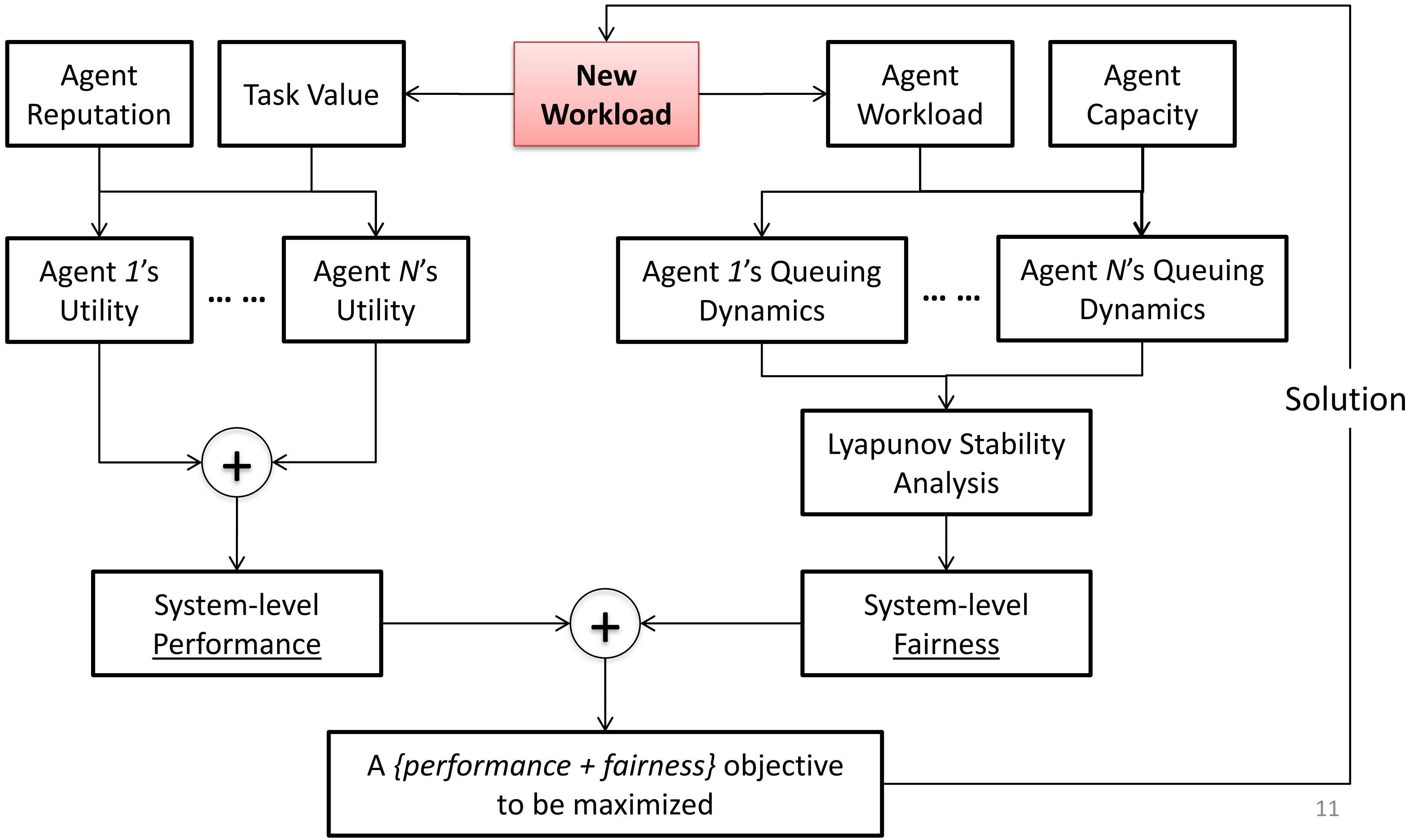
- Reliability
- Productivity
- Availability

No. of tasks accepted by *w* at time *t*.

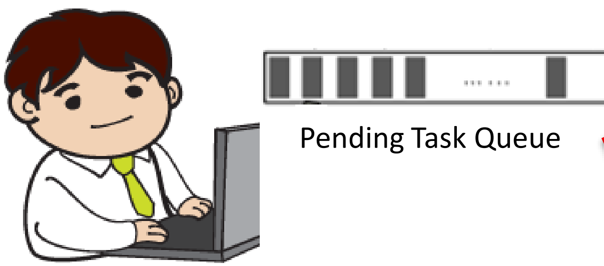
No. of tasks completed by *w* at time *t*.

$$q_w(t+1) = \max[0, q_w(t) + a_w(t) - c_w(t)]$$

# Overall Decision-making Flow



# Formulating Optimization Objective




**Worker**

- Reliability
- Productivity
- Availability

No. of tasks accepted by *w* at time *t*.

No. of tasks completed by *w* at time *t*.

$$\boxed{q_w(t+1)} = \max[0, \boxed{q_w(t)} + \boxed{a_w(t)} - \boxed{c_w(t)}]$$

**Lyapunov functions:** scalar functions that may be used to prove the stability of an equilibrium of an ordinary differential equations.

$$L(t) = \frac{1}{2}\sum_{w=1}^{N} q_w^2(t)$$

# What’s Captured by Lyapunov Function

- Example:
  - Case 1: three workers having been allocated 5, 5 and 65 tasks, respectively.
  - Case 2: three workers having been allocated 25, 25 and 25 tasks, respectively.
  - 5 + 5 + 65 = 25 + 25 + 25 = 75
  - No difference? Tasks in Case 2 is obviously more evenly distributed than in Case 1!

# What’s Captured by Lyapunov Function

- Example:
  - Case 1: three workers having been allocated 5, 5 and 65 tasks, respectively.
  - Case 2: three workers having been allocated 25, 25 and 25 tasks, respectively.
  - With the Lyapunov function, we have:
  - $\frac{1}{2}(5^2 + 5^2 + 65^2) = 2137.5$
  - $\frac{1}{2}(25^2 + 25^2 + 25^2) = 937.5$

# Formulating Optimization Objective

Let $\mathbf{q}(t)$ be a vector of all workers' pending task queues during time slot $t$. Using the *Lyapunov drift*, $\Delta(\mathbf{q}(t))$, the variation in workers' workload can be expressed as:

$$\begin{aligned}
\Delta(\mathbf{q}(t)) &= \mathbb{E}\{L(t+1) - L(t) | \mathbf{q}(t)\} \\
&= \sum_{w=1}^{N} \left( \frac{1}{2} q_w^2(t+1) - \frac{1}{2} q_w^2(t) \right) \\
&\leqslant \sum_{w=1}^{N} \left( q_w(t) a_w(t) - q_w(t) c_w(t) + \frac{1}{2} [a_w^2(t) + c_w^2(t)] \right)
\end{aligned}$$

$$\frac{1}{2}[(a_w^{\max})^2 + (c_w^{\max})^2]$$

# Formulating Optimization Objective

- Let *U*(*t*) be the expected overall utility (i.e., the sum of the expected task success rate) of a strategy which distributes tasks among a given set of *N* workers in a given way during time slot *t*. We have:

$$U(t) = \sum_{w=1}^{N} r_w(t) a_w(t)$$

Worker w's reputation
(i.e. probability of successfully completing a task)

# Formulating Optimization Objective

Weightage

$$\frac{1}{T}\sum_{t=0}^{T-1}(\sigma\mathbb{E}\{U(t)|\mathbf{q}(t)\} - \Delta(\mathbf{q}(t)))$$

Collective Utility        Unfairness

Maximize:

$$\frac{1}{T}\sum_{t=0}^{T-1}\sum_{w=1}^{N} a_w(t)\left[\sigma_w(t)r_w(t) - q_w(t)\right]$$

Subject to:

$$r_w(t) \geqslant r_{\min},\ \forall w,\ \forall t$$

$$a_w(t) \leqslant nc_w^{\max},\ \forall w,\ \forall t$$

# Suitability Ranking Index

Personal eagerness to work

Current workload

$$\Psi_w(t) = \sigma_w(t) r_w(t) - q_w(t)$$

Current reputation

- A simple greedy algorithm
  - greedy in the sense that decisions are made without considering future values of the relevant variables
- leads to a solution that has regret
  - compared to the *optimal solution*
- which increases with increasing **σ**
  - where **σ** > 0 is the average eagerness to work expressed by workers

# Centralized Implementation

**Require:** New tasks in the crowdsourcing task at time slot $t$, $Q(t)$; the average deadline of the new requests, $\bar{d}$; $\sigma_w(t)$, $q_w(t)$, $c_w^{\max}$ and $r_w(t)$ values for all workers.

Compute $\Psi_w(t)$ for all $w$;<br>
Rank all $w$ in descending order of $\Psi_w(t)$;<br>
**for** each worker $w$ **do**<br>
    **if** $\Psi_w(t) > 0$ **and** $r_w(t) \geqslant r_{\min}$ **then**<br>
        **if** $Q(t) < \bar{d}c_w^{\max} - q_w(t)$ **then**<br>
            $a_w(t) = Q(t)$;<br>
        **else**<br>
            $a_w(t) = \max[0, \lfloor \bar{d}c_w^{\max} - q_w(t) \rfloor]$;<br>
        **end if**<br>
    **else**<br>
        $a_w(t) = 0$;<br>
    **end if**<br>
    $Q(t) \leftarrow Q(t) - a_w(t)$;<br>
**end for**<br>
**return** $(a_1(t), a_2(t), \ldots, a_N(t))$;

# Distributed Implementation

**Require:** New tasks pending worker $w$'s acceptance at time slot $t$, $Q_w(t)$; the average deadline of the new requests for worker $w$, $\bar{d}_w$; $\sigma_w(t)$, $q_w(t)$, $c_w^{\max}$ and $r_w(t)$ values for worker $w$.

Compute $\Psi_w(t)$ for $w$;
**if** $\Psi_w(t) > 0$ **then**
  **if** $Q_w(t) < \bar{d}_w c_w^{\max} - q_w(t)$ **then**
    $a_w(t) = Q_w(t)$;
  **else**
    $a_w(t) = \max[0, \lfloor c_w^{\max} \bar{d}_w - q_w(t) \rfloor]$;
  **end if**
**else**
  $a_w(t) = 0$;
**end if**
**if** $Q_w(t) - a_w(t) > 0$ **then**
  Send the remaining $Q_w(t) - a_w(t)$ task requests to the crowdsourcers;
**end if**
**return** $a_w(t)$;

# Further Reading


Y. Zheng, H. Yu, L. Cui, C. Miao, C. Leung & Q. Yang, "SmartHS: An AI Platform for Improving Government Service Provision," in *Proceedings of the 30th AAAI Conference on Innovative Applications of Artificial Intelligence (IAAI-18)*, pp. 7704–7711, 2018.

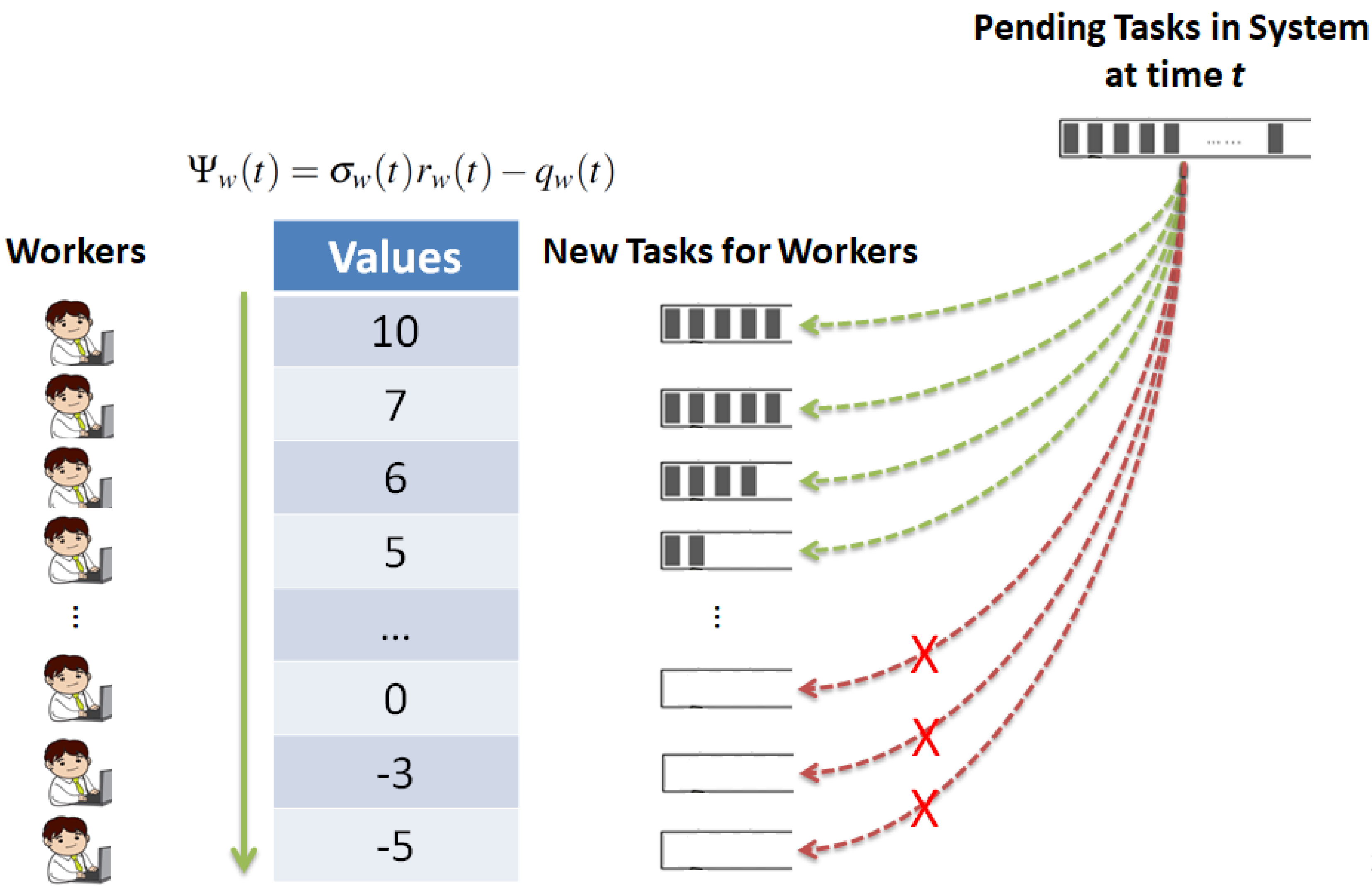
Yu, H., Miao, C., Chen, Y., Fauvel, S., Li, X. & Lesser, V. R. Algorithmic management for improving collective productivity in crowdsourcing. *Scientific Reports*, vol. 7, no. 12541, Nature Publishing Group (2017).

Yu, H., Miao, C., Leung, C., Chen, Y., Fauvel, S., Lesser, V. R. & Yang, Q. Mitigating herding in hierarchical crowdsourcing networks. *Scientific Reports*, vol. 6, no. 4, Nature Publishing Group (2016).

# A Walkthrough Example

# Example

$$\Psi_w(t) = \sigma_w(t) r_w(t) - q_w(t)$$



# Example

- Suppose we have 5 workers, *w1* to *w5*:

| ID | Reputation | Eagerness to Work | Productivity (tasks per round) |
|---|---|---|---|
| w1 | 0.9 | 10 | 5 |
| w2 | 0.8 | 15 | 8 |
| w3 | 0.7 | 10 | 15 |
| w4 | 0.4 | 20 | 20 |
| w5 | 0.2 | 50 | 25 |

# Round 1

- All workers start with 0 workload
- Compute their suitability ranking indices

$$\Psi_w(t) = \sigma_w(t) r_w(t) - q_w(t)$$

| ID | Reputation | Eagerness to Work | Suitability Ranking Index |
|---|---|---|---|
| w1 | 0.9 | 10 | 9 |
| w2 | 0.8 | 15 | 12 |
| w3 | 0.7 | 10 | 7 |
| w4 | 0.4 | 20 | 8 |
| w5 | 0.2 | 50 | 10 |

# Round 1

- Suppose we only want workers with reputation scores above 0.6 to work on our tasks:

| ID | Reputation | Eagerness to Work | Suitability Ranking Index |
|---|---|---|---|
| w1 | 0.9 | 10 | 9 |
| w2 | 0.8 | 15 | 12 |
| w3 | 0.7 | 10 | 7 |
| w4 | 0.4 | 20 | 8 |
| w5 | 0.2 | 50 | 10 |

# Round 1

- Rank the remaining workers in descending order of their suitability ranking indices:

| ID | Reputation | Eagerness to Work | Suitability Ranking Index |
|---|---|---|---|
| w2 | 0.8 | 15 | 12 |
| w1 | 0.9 | 10 | 9 |
| w3 | 0.7 | 10 | 7 |

# Round 1

- Rank the remaining workers in descending order of their suitability ranking indices.
- Suppose we have 20 new tasks, all with deadlines = 2 rounds:

| ID | Suitability Ranking Index | Productivity (tasks per round) | New Tasks Assigned |
|---|---|---|---|
| w2 | 12 | 8 | 16 |
| w1 | 9 | 5 | 4 |
| w3 | 7 | 15 | 0 |

# Round 2

- Suppose w2 completed 8 tasks during round 1, and w1 completed 4 tasks during round 1:

| ID | Reputation | Eagerness to Work | Remaining Tasks | Suitability Ranking Index |
|---|---|---|---|---|
| w1 | 0.9 | 10 | 0 | 9 |
| w2 | 0.8 | 15 | 8 | 4 |
| w3 | 0.7 | 10 | 0 | 7 |
| w4 | 0.4 | 20 | 0 | 8 |
| w5 | 0.2 | 50 | 0 | 10 |

# Round 2

- Suppose we only want workers with reputation scores above 0.6 to work on our tasks:

| ID | Reputation | Eagerness to Work | Remaining Tasks | Suitability Ranking Index |
|---|---|---|---|---|
| w1 | 0.9 | 10 | 0 | 9 |
| w2 | 0.8 | 15 | 8 | 4 |
| w3 | 0.7 | 10 | 0 | 7 |
| w4 | 0.4 | 20 | 0 | 8 |
| w5 | 0.2 | 50 | 0 | 10 |

# Round 2

- Rank the remaining workers in descending order of their suitability ranking indices.
- Suppose we have 10 new tasks, all with deadlines = 1 round:

| ID | Suitability Ranking Index | Productivity (tasks per round) | New Tasks Assigned |
|---|---|---|---|
| w1 | 9 | 5 | 5 |
| w3 | 7 | 15 | 5 |
| w2 | 4 | 8 | 0 |

# Measuring Fairness

# Jain's Fairness Index

- Raj Jain's equation:

$$\mathcal{J}(x_1, x_2, \dots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} {x_i}^2}$$

- $x_i$ denotes the amount of resources allocated to *i*.
- It is useful to evaluation fairness of resource allocation among recipients.
- The range of values for *J* is between 0 and 1. The larger the value, the fairer the allocation.

# Example

- [Case 1]:
  - Consider workers *a*, *b* and *c*.
  - Their reputation scores (r) are 0.9, 0.5 and 0.2, respectively.
  - The values of the tasks (v) being allocated to them are 25, 12 and 20, respectively.
  - How fair is the allocation?

- Solution using Jain's Fairness Index:
  - In this case, we want to see if more reputable workers are allocated more valuable tasks
  - Thus, we set $x_i = \frac{v_i}{r_i}$
  - $J = \frac{\left(\frac{25}{0.9}+\frac{12}{0.5}+\frac{20}{0.2}\right)^2}{3\left[\left(\frac{25}{0.9}\right)^2+\left(\frac{12}{0.5}\right)^2+\left(\frac{20}{0.2}\right)^2\right]} = \frac{23036.49}{34042.81} = 0.677$

# Example

- [Case 2]:
  - Consider workers *a*, *b* and *c*.
  - Their reputation scores (r) are 0.9, 0.5 and 0.2, respectively.
  - The values of the tasks (v) being allocated to them are 25, 20 and 12, respectively.
  - How fair is the allocation?

- Solution using Jain's Fairness Index:
  - In this case, we want to see if more reputable workers are allocated more valuable tasks
  - Thus, we set $x_i = \frac{v_i}{r_i}$
  - $J = \frac{\left(\frac{25}{0.9}+\frac{20}{0.5}+\frac{12}{0.2}\right)^2}{3\left[\left(\frac{25}{0.9}\right)^2+\left(\frac{20}{0.5}\right)^2+\left(\frac{12}{0.2}\right)^2\right]} = \frac{16327.16}{17914.81} = 0.911 > 0.677$
  - Thus, Case 2 is fairer than Case 1.

# Example

- What about [Case 3]:
  - Consider workers *a*, *b* and *c*.
  - Their reputation scores (r) are 0.9, 0.5 and 0.2, respectively.
  - The values of the tasks (v) being allocated to them are 12, 20 and 25, respectively.
  - How fair is the allocation?

- Try it yourself

# Fairness


Yu Han

han.yu@ntu.edu.sg

*Nanyang Assistant Professor*
*School of Computer Science and Engineering*
*Nanyang Technological University*