# Max Algorithms in Crowdsourcing Environments

Petros Venetis[1] Hector Garcia-Molina[1]
Kerui Huang[2] Neoklis Polyzotis[2]

[1]Stanford University

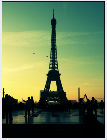
[2]UC Santa Cruz

April 20, 2012

# Crowdsourcing: Getting Tasks done by People

## Why?

- Humans are better than computers in certain tasks

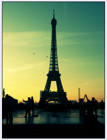


- Human opinions are desired (product and ad design)

# Crowdsourcing: Getting Tasks done by People

## Why?

- Humans are better than computers in certain tasks



- Human opinions are desired (product and ad design)

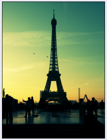
## Positioning

- Worker motivation
- Skills required
- Time for tasks

# Crowdsourcing: Getting Tasks done by People

## Why?

- Humans are better than computers in certain tasks



- Human opinions are desired (product and ad design)

## Positioning

- Worker motivation: payment
- Skills required: no qualifications
- Time for tasks: microtasks/seconds

# Crowdsourcing

## Issues

User Interfaces

Machine Learning

Algorithms

Quality Control

Systems

Spammer Detection

# Max Item Problem: Example

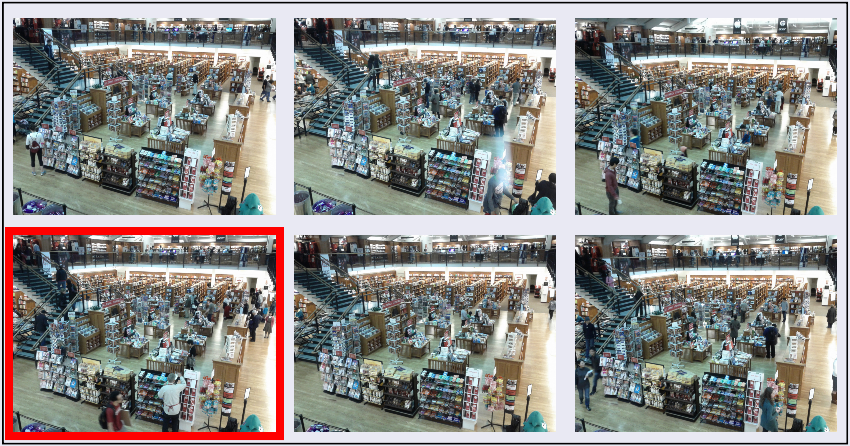
## Finding Peak Hours

# Max Item Problem: Example

## Finding Peak Hours

# Max Item Problem: Example

## Finding Peak Hours



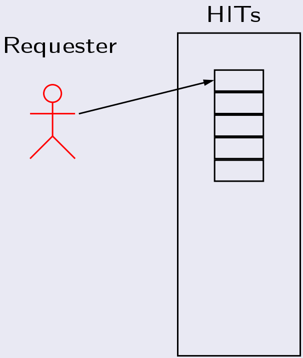
# Crowdsourcing Marketplaces

## Example: Amazon's Mechanical Turk Marketplace

Requester

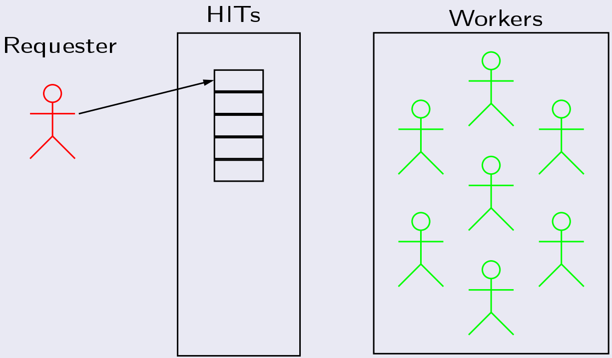
# Crowdsourcing Marketplaces

## Example: Amazon's Mechanical Turk Marketplace



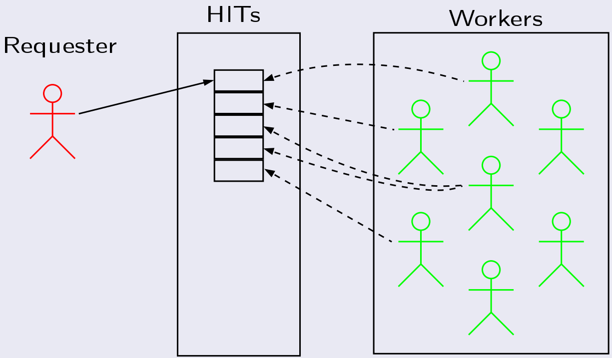
# Crowdsourcing Marketplaces

## Example: Amazon's Mechanical Turk Marketplace



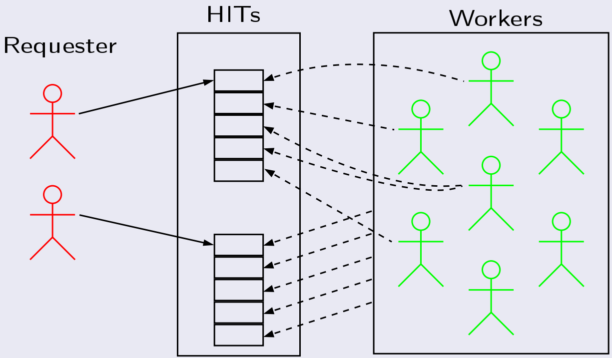
# Crowdsourcing Marketplaces

## Example: Amazon's Mechanical Turk Marketplace



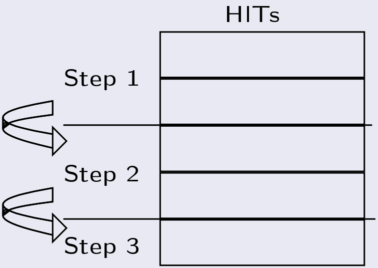
# Crowdsourcing Marketplaces

## Example: Amazon's Mechanical Turk Marketplace



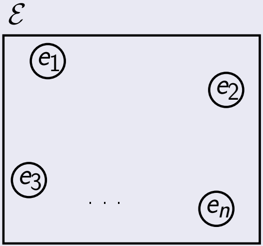
# Crowdsourcing Algorithms

## Notion of steps



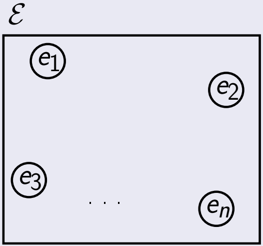
# Max Algorithms

## Model



# Max Algorithms

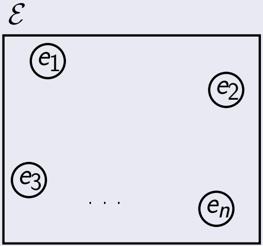
## Model




- Items have inherent quality values

# Max Algorithms

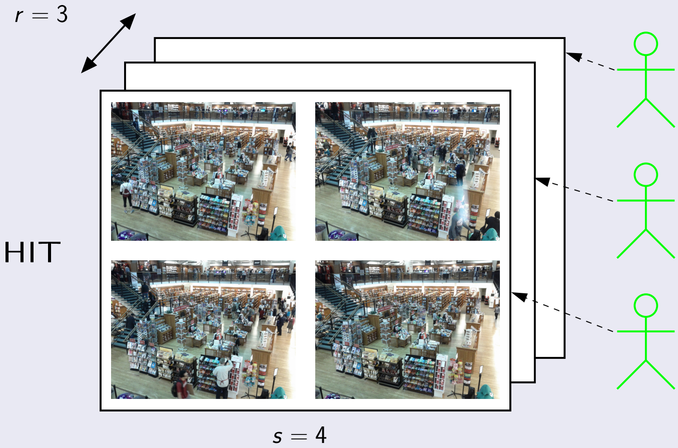
## Model




- Items have inherent quality values
- Max item $e^* \in \mathcal{E}$:
$$e \leq e^* \ \forall e \in \mathcal{E} \setminus \{e^*\}$$

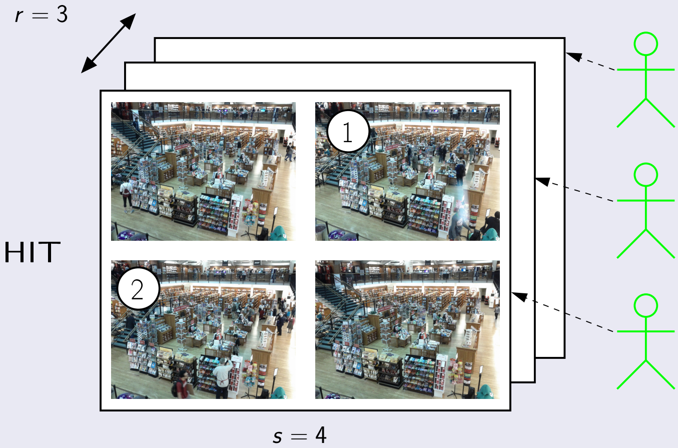
# Worker Tasks

## HITs Used: Comparisons



# Worker Tasks

## HITs Used: Comparisons



# Crowdsourced Max Algorithms

## Structured Algorithms

How to break up the problem to retrieve max item:

- Bubble
- Tournament

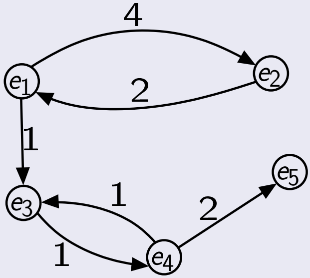
# Crowdsourced Max Algorithms

## Structured Algorithms

How to break up the problem to retrieve max item:

- Bubble
- Tournament

## Unstructured Algorithms (not here)




- Which comparison to perform next?
- Which item is the max?

# Max Algorithms: Bubble

## Example

$e_1$

$e_2$

$e_3$

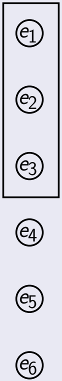
$e_4$

$e_5$

$e_6$

# Max Algorithms: Bubble

## Example

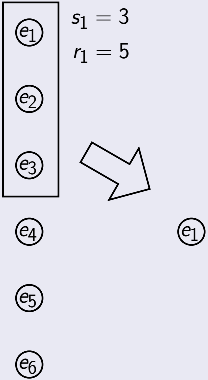



$s_1 = 3$

$r_1 = 5$

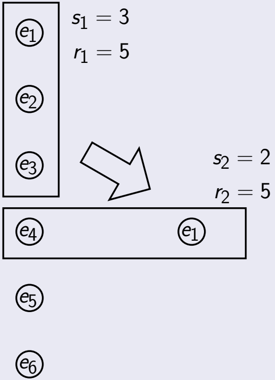
# Max Algorithms: Bubble

## Example



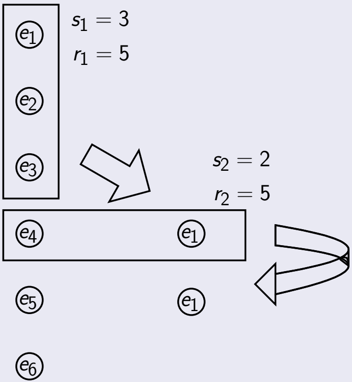
# Max Algorithms: Bubble

## Example



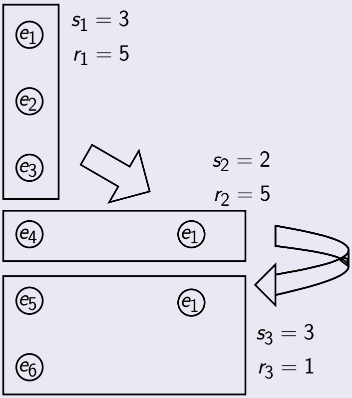
# Max Algorithms: Bubble

## Example



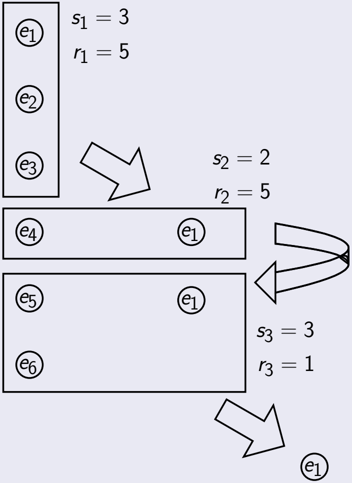
# Max Algorithms: Bubble

## Example



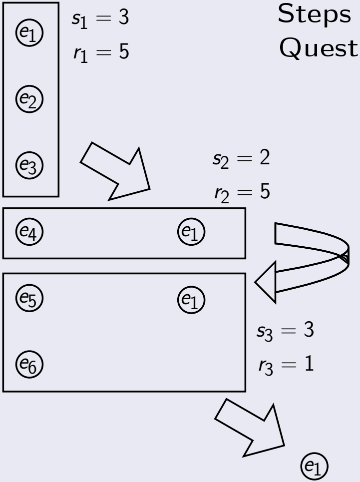
# Max Algorithms: Bubble

## Example



# Max Algorithms: Bubble

## Example




Steps $= 3$

Questions $= 5 + 5 + 1 = 11$

# Max Algorithms: Tournament

## Example

$e_1$

$e_2$

$e_3$

$e_4$

$e_5$

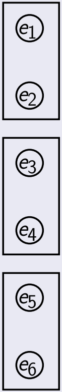
$e_6$

# Max Algorithms: Tournament

## Example

$s_1 = 2$

$r_1 = 5$

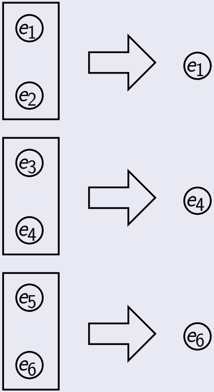


# Max Algorithms: Tournament

## Example

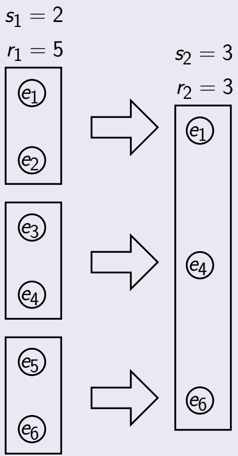
$s_1 = 2$

$r_1 = 5$



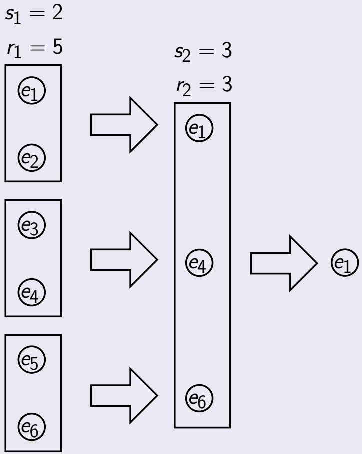
# Max Algorithms: Tournament

## Example



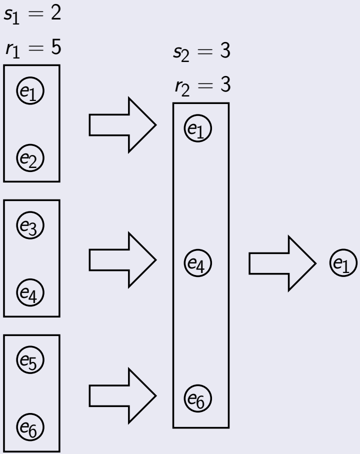
# Max Algorithms: Tournament

## Example



# Max Algorithms: Tournament

## Example




$\text{Steps} = 2$
$\text{Questions} = 3 \cdot 5 + 1 \cdot 3 = 18$

# How to select $\{r_i\}$ and $\{s_i\}$

## Problem Statement

$$\begin{aligned}
\underset{A=\mathcal{A}(\{r_i\},\{s_i\})}{\text{maximize}} \quad & \Pr[A \text{ returns max item from } \mathcal{E}] \\
\text{subject to} \quad & \text{Cost}(A,\mathcal{E}) \leq B \\
& \text{Steps}(A,\mathcal{E}) \leq T
\end{aligned}$$

# How to select $\{r_i\}$ and $\{s_i\}$

## Problem Statement

$$\begin{aligned} \underset{A=\mathcal{A}(\{r_i\},\{s_i\})}{\text{maximize}} \quad & \Pr[A \text{ returns max item from } \mathcal{E}] \\ \text{subject to} \quad & \mathrm{Cost}(A,\mathcal{E}) \leq B \\ & \mathrm{Steps}(A,\mathcal{E}) \leq T \end{aligned}$$

## Tuning Strategies (based on hill climbing)

- Constant $r_i$, $s_i$
- Varying $r_i$, constant $s_i$
- Varying $r_i$, $s_i$

# Models Considered

- Comparison Input $= \{e_1, e_2, \ldots, e_s\}$
- $e_s < \ldots < e_2 < e_1$
- $p_i$: probability $e_i$ is returned by worker

# Models Considered

- Comparison Input $= \{e_1, e_2, \ldots, e_s\}$
- $e_s < \ldots < e_2 < e_1$
- $p_i$: probability $e_i$ is returned by worker

## Worker Error Models

| | |
|---|---|
| Constant | $p_1 = p,\ p_2 = p_3 = \ldots$ |
| Linear | $p_1$ decreases on $s$ |
| Order-based | $p_1 > p_2 > \ldots > p_s$ |
| Distance-based | $p_i$'s depend on value differences of items |

# Models Considered

- Comparison Input $= \{e_1, e_2, \ldots, e_s\}$
- $e_s < \ldots < e_2 < e_1$
- $p_i$: probability $e_i$ is returned by worker

## Worker Error Models

| | |
|---|---|
| Constant | $p_1 = p,\ p_2 = p_3 = \ldots$ |
| Linear | $p_1$ decreases on $s$ |
| Order-based | $p_1 > p_2 > \ldots > p_s$ |
| Distance-based | $p_i$'s depend on value differences of items |

## Worker Compensation Models

| | |
|---|---|
| Constant | $c$ |
| Linear | $c + \lambda \times s$ |

# Experiments

## Why Not Analysis?

- Analysis only for simple models and still is expensive

## Why Not Analysis?

- Analysis only for simple models and still is expensive

Single HIT Accuracy$(s, r; \vec{p}) =$

$$\sum_{l=1}^{s} \frac{1}{l} \cdot \sum_{n=1}^{r} \sum_{L \in \mathcal{L}} \sum_{\substack{0 \le k_i \le n-1, i \in \bar{L} \\ \sum_{i \in \bar{L}} k_i + l \cdot n = r}} \left[ \frac{r!}{(n!)^l \cdot \prod_{j \in \bar{L}} k_j!} \cdot \prod_{z \in L} p_z^n \cdot \prod_{w \in \bar{L}} p_w^{k_w} \right]$$

# Experiments

## Why Not Analysis?

- Analysis only for simple models and still is expensive

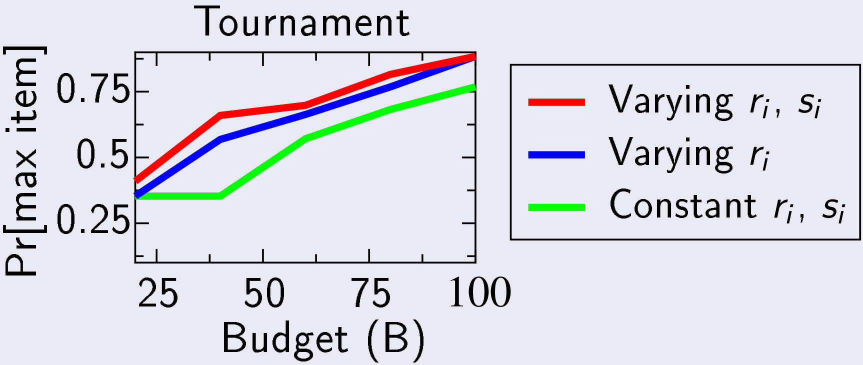
Single HIT Accuracy$(s, r; \vec{p}) =$

$$\sum_{l=1}^{s} \frac{1}{l} \cdot \sum_{n=1}^{r} \sum_{L \in \mathcal{L}} \sum_{\substack{0 \le k_i \le n-1, i \in \bar{L} \\ \sum_{i \in \bar{L}} k_i + l \cdot n = r}} \left[ \frac{r!}{(n!)^l \cdot \prod_{j \in \bar{L}} k_j!} \cdot \prod_{z \in L} p_z^n \cdot \prod_{w \in \bar{L}} p_w^{k_w} \right]$$

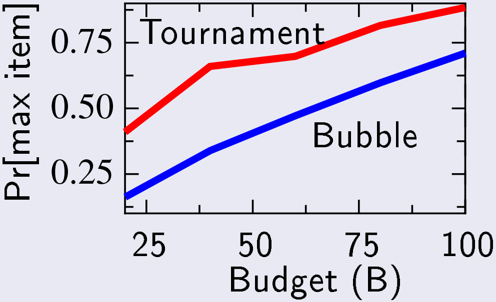
## Simulations

- Used linear error and compensation model
- 100,000 simulations per data point

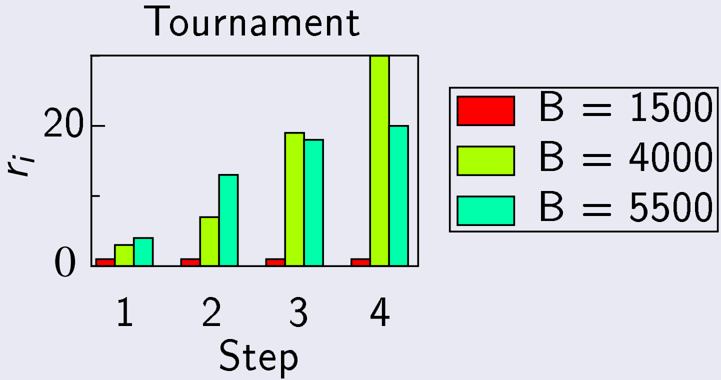
# It pays off to vary $r_i$, $s_i$



# Tournament is better

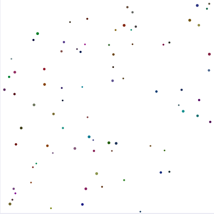


# It pays off to have more responses towards the end



# MTurk Experiment

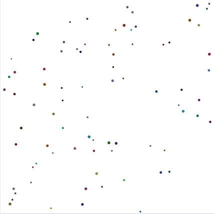
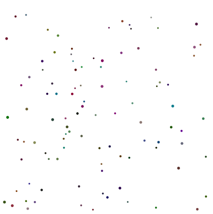
## Dataset

80 and 100 dots

# MTurk Experiment

## Dataset

80 and 100 dots

## Setting

- Learned distance-based model ($s_i = 4$)
- Images with 5, 10, . . . , 320 dots
- Tournaments
- $B$ enough for 105 comparisons
- 200 runs
- \$0.01 per comparison

# MTurk Experiment (cont'd)

**Constant $r_i$**

- $r_1 = r_2 = r_3 = 5$

**Varying $r_i$**

- $r_1 = 3$
- $r_2 = 5$
- $r_3 = 37$

# MTurk Experiment (cont'd)

### Constant $r_i$

- $r_1 = r_2 = r_3 = 5$

Predicted Pr[max item]: 0.67

### Varying $r_i$

- $r_1 = 3$
- $r_2 = 5$
- $r_3 = 37$

Predicted Pr[max item]: 0.84

# MTurk Experiment (cont'd)

### Constant $r_i$

- $r_1 = r_2 = r_3 = 5$

Predicted Pr[max item]: 0.67
Measured Pr[max item]: 0.69

### Varying $r_i$

- $r_1 = 3$
- $r_2 = 5$
- $r_3 = 37$

Predicted Pr[max item]: 0.84
Measured Pr[max item]: 0.80

# MTurk Experiment (cont'd)

## Constant $r_i$

- $r_1 = r_2 = r_3 = 5$

Predicted Pr[max item]: 0.67
Measured Pr[max item]: 0.69

## Varying $r_i$

- $r_1 = 3$
- $r_2 = 5$
- $r_3 = 37$

Predicted Pr[max item]: 0.84
Measured Pr[max item]: 0.80

## Observations

- Experiments match predictions
- Varying repetitions ($r_i$) improves results significantly

# More in paper

## Experimental

- Algorithms are robust
- Repetitive algorithms not helpful
- Relaxing step bound increases accuracy
- Finding the top-1 item is usually hard, but top-$k\%$ is easy

## Analysis

- How to analyze tournament and bubble algorithms (for some models)

# Conclusions

## Summary

- It pays off to vary the size of a task ($s_i$)
- It pays off to optimize the number of repetitions ($r_i$)
- Tournament performs significantly better than bubble
- Tuning tournaments improves results in practice

# Conclusions

## Summary

- It pays off to vary the size of a task ($s_i$)
- It pays off to optimize the number of repetitions ($r_i$)
- Tournament performs significantly better than bubble
- Tuning tournaments improves results in practice

## Current Work

- Spammer detection and appropriate actions
- Dynamic adjustments to account for comparison difficulty