Crowdsourced Algorithms in Data Management

DASFAA 2014 Tutorial

Dongwon Lee, Ph.D.

(Thanks to Jongwuk Lee, Ph.D.)

Penn State University, USA
dongwon@psu.edu

PENNSTATE

1 8 5 5

DASFAA
Database Systems for Advanced Applications

Latest Version

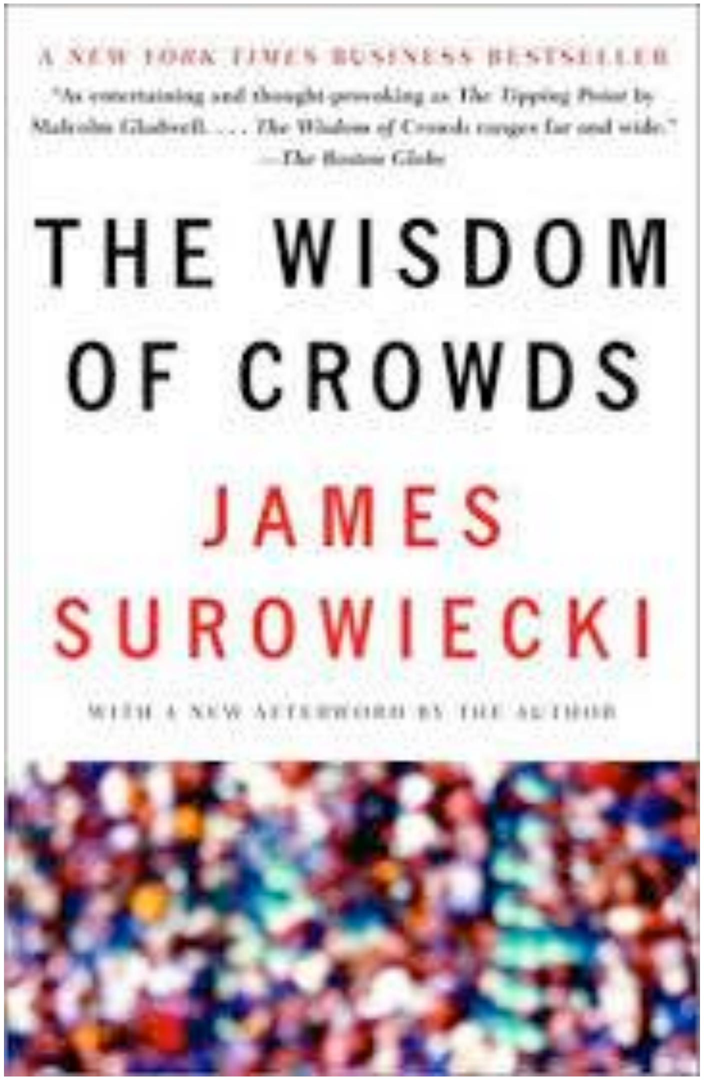
- Latest version of this DASFAA14 tutorial is available at:

http://bit.ly/1h6wHAS

TOC

- Crowdsourcing Basics
  - Definition
  - Apps
- Crowdsourced Algorithms in DB
  - Sort
  - Top-1
  - Top-*k*
  - Select
  - Count
  - Join

James Surowiecki, 2004



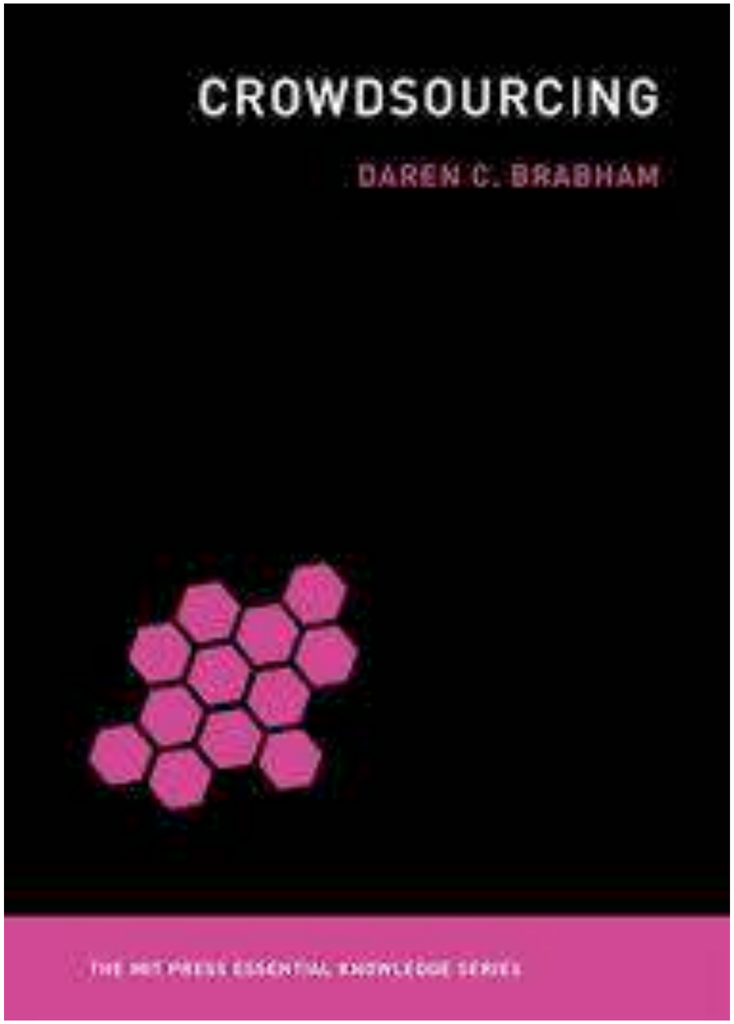
"**Collective intelligence** can be brought to bear on a wide variety of problems, and complexity is no bar… conditions that are necessary for the crowd to be wise: *diversity*, *independence*, and … *decentralization*"

Jeff Howe, WIRED, 2006

"**Crowdsourcing** represents the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call. … The crucial prerequisite is the use of the open call format and the large network of potential laborers…"

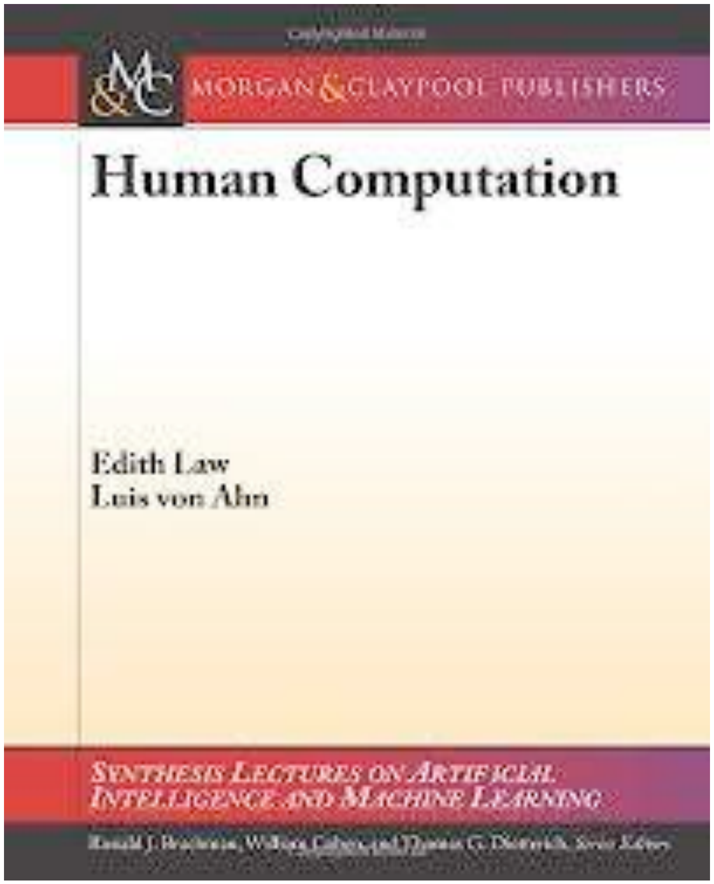
http://www.wired.com/wired/archive/14.06/crowds.html

Daren Brabhan, 2013



"**Crowdsourcing** as an online, distributed problem-solving and production model that leverages the collective intelligence of online communities to serve specific organizational goals"

"Human Computation", 2011



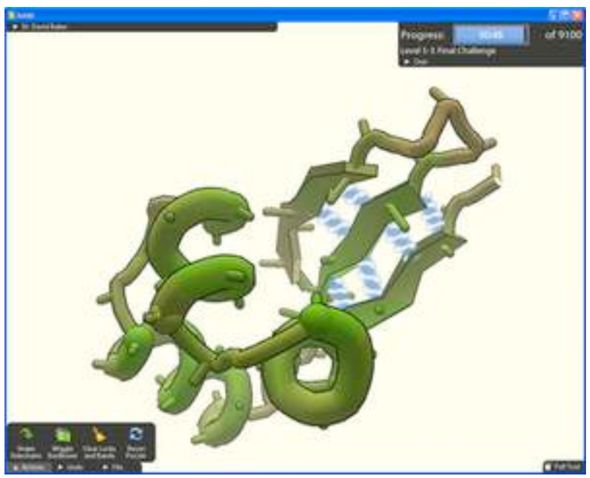
"**Human computation** is simply computation that is carried out by humans… **Crowdsourcing** can be considered a method or a tool that human computation systems can use…"

By Edith Law & Luis von Ahn

Game with a Purpose: GWAP

- Luis von Ahn @ CMU
- Eg,
  - ESP Game → Google Image Labeler

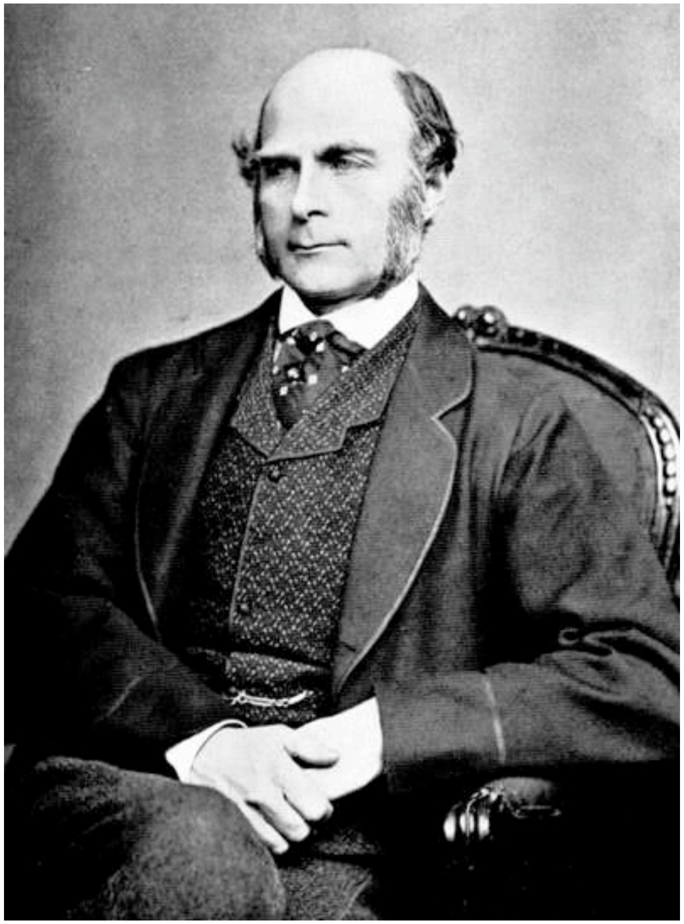
  - Foldit

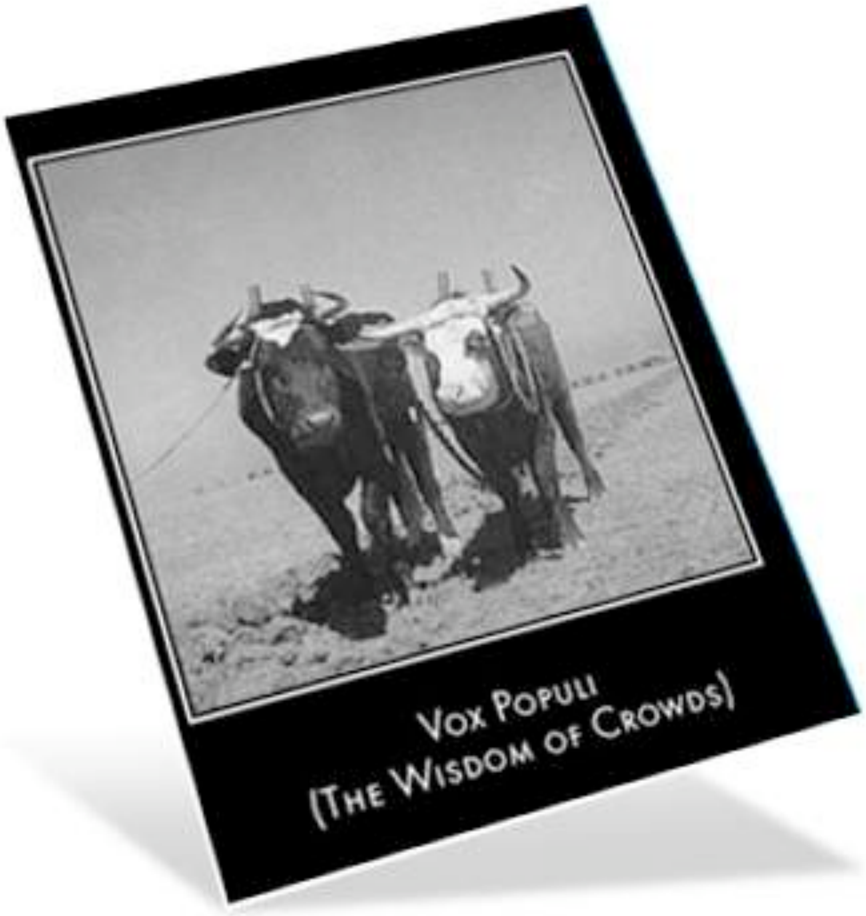


  - Duolingo: crowdsourced language translation

Eg, Francis Galton, 1906

Weight-judging competition:
1,197 (mean of 787 crowds) vs. 1,198 pounds (actual measurement)





Eg, StolenSidekick, 2006

- A woman lost a cellphone in a taxi
- A 16-year-old girl ended up having the phone
  - Refused to return the phone
- Evan Guttman, the woman's friend, sets up a blog site about the incident
  - http://stolensidekick.blogspot.com/
  - http://www.evanwashere.com/StolenSidekick/
  - Attracted a growing amount of attention → the story appeared in Digg main page → NY Times and CNN coverage → Crowds pressure on police …
- NYPD arrested the girl and re-possessed the phone

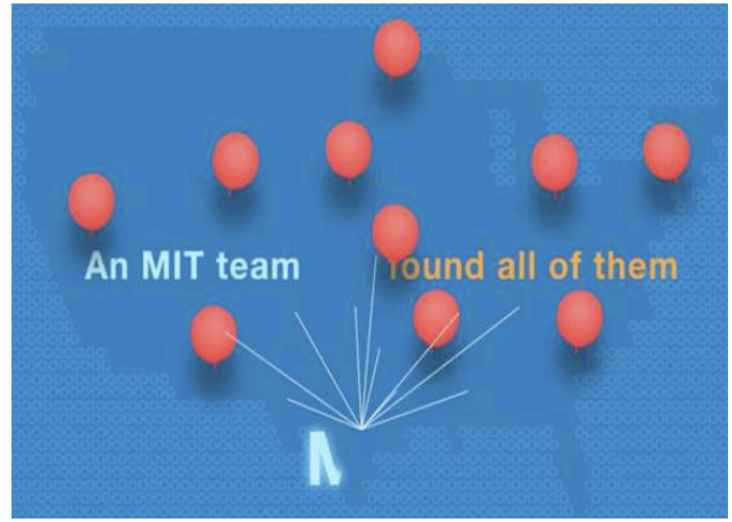
http://www.nytimes.com/2006/06/21/nyregion/21sidekick.html?_r=0

Eg, “Who Wants to be a Millionaire”

Asking the audience usually works ➔ Audience members have diverse knowledge that can be coordinated to provide a correct answer in sum

Eg, DARPA Challenge, 2009

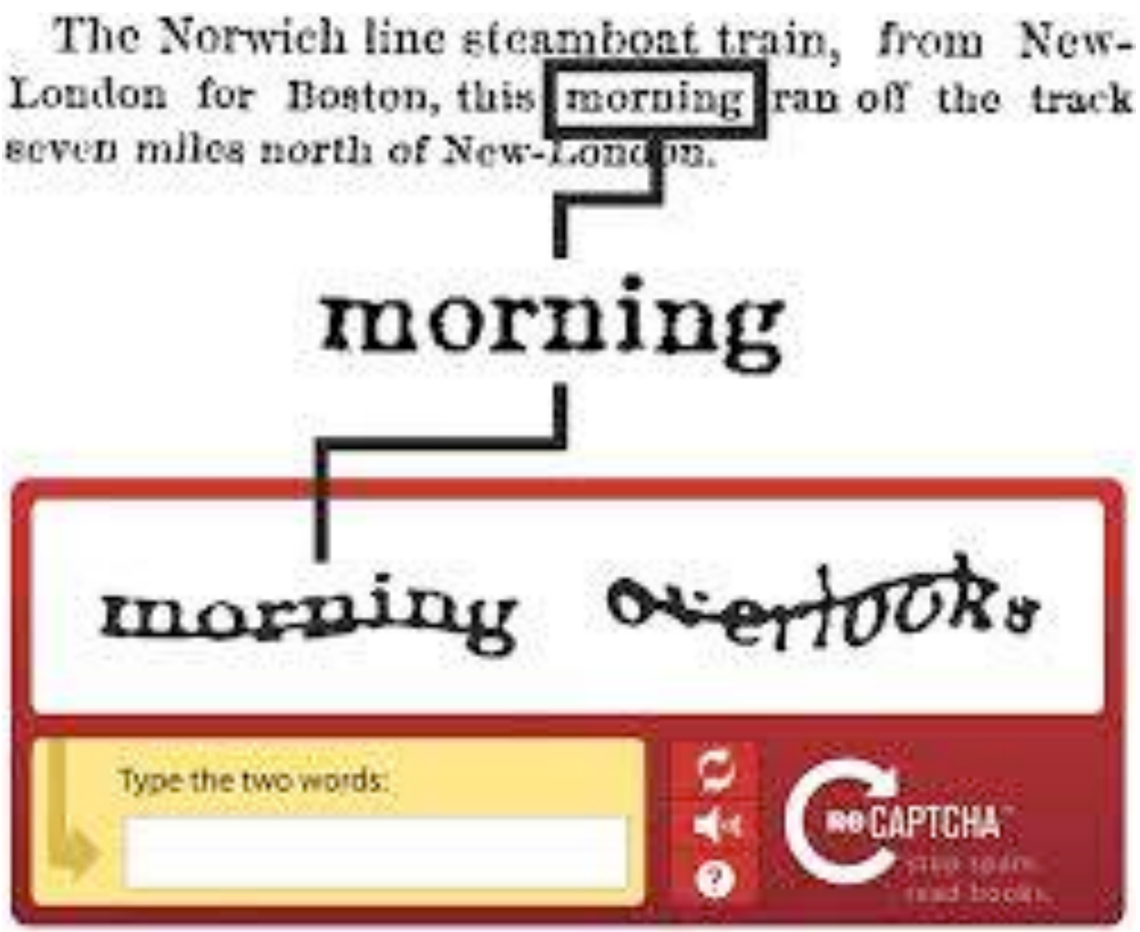
- To locate 10 red balloons in arbitrary locations of US
- Winner gets $40K
- MIT team won the race with the strategy:
  - 2K per balloon to the first person, A, to send the correct coordinates
  - 1K to the person, B, who invited A
  - 0.5K to the person, C, who invited B, …



Eg, Threadless.com

- Sells t-shirts, designed/voted by crowds
- Artists whose designs are chosen get paid

Eg, reCAPCHA



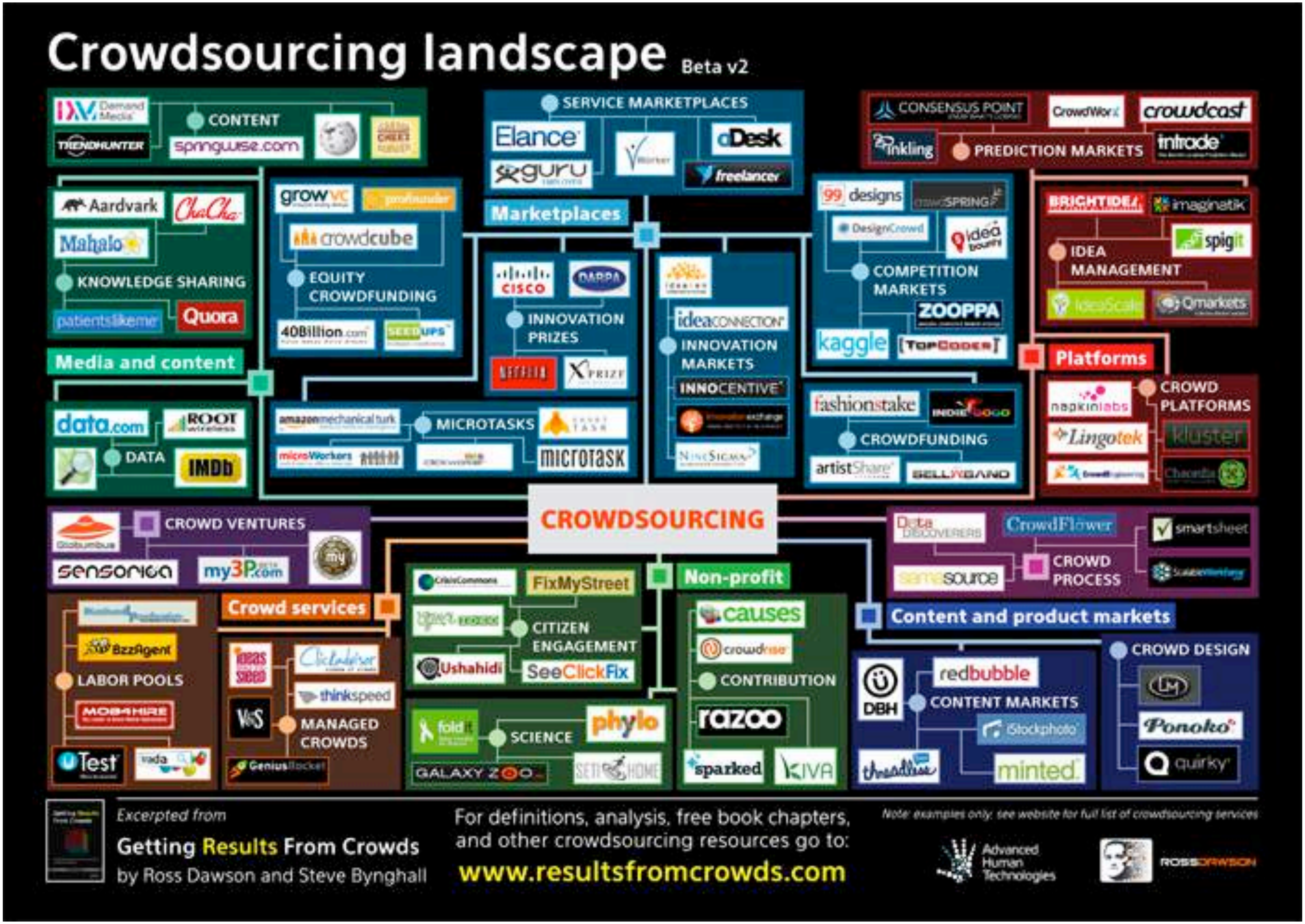
As of 2012

Captcha: 200M every day

ReCaptcha: 750M to date

Eg, KICKSTARTER

- Crowdfunding, started in 2009
- Project creators choose a deadline and a minimum funding goal
  - Creators only from US, UK, and Canada
- Donors pledge money to support projects, in exchange of non-monetary values
  - Eg, t-shirt, thank-u-note, dinner with creators
  - Donors can be from anywhere
- Eg, Pebble, smartwatch
  - 68K people pledged 10M



http://www.resultsfromcrowds.com/features/crowdsourcing-landscape/

What is Crowdsourcing?

- Many definitions
- A few characteristics
  - Online and distributed
  - Open call & right incentive
  - Diversity and independence
  - Top-down & bottom-up

- **Q: What are the computational issues in crowdsourcing?**
  - **Micro-tasks for large crowds**

What is Computational Crowdsourcing?

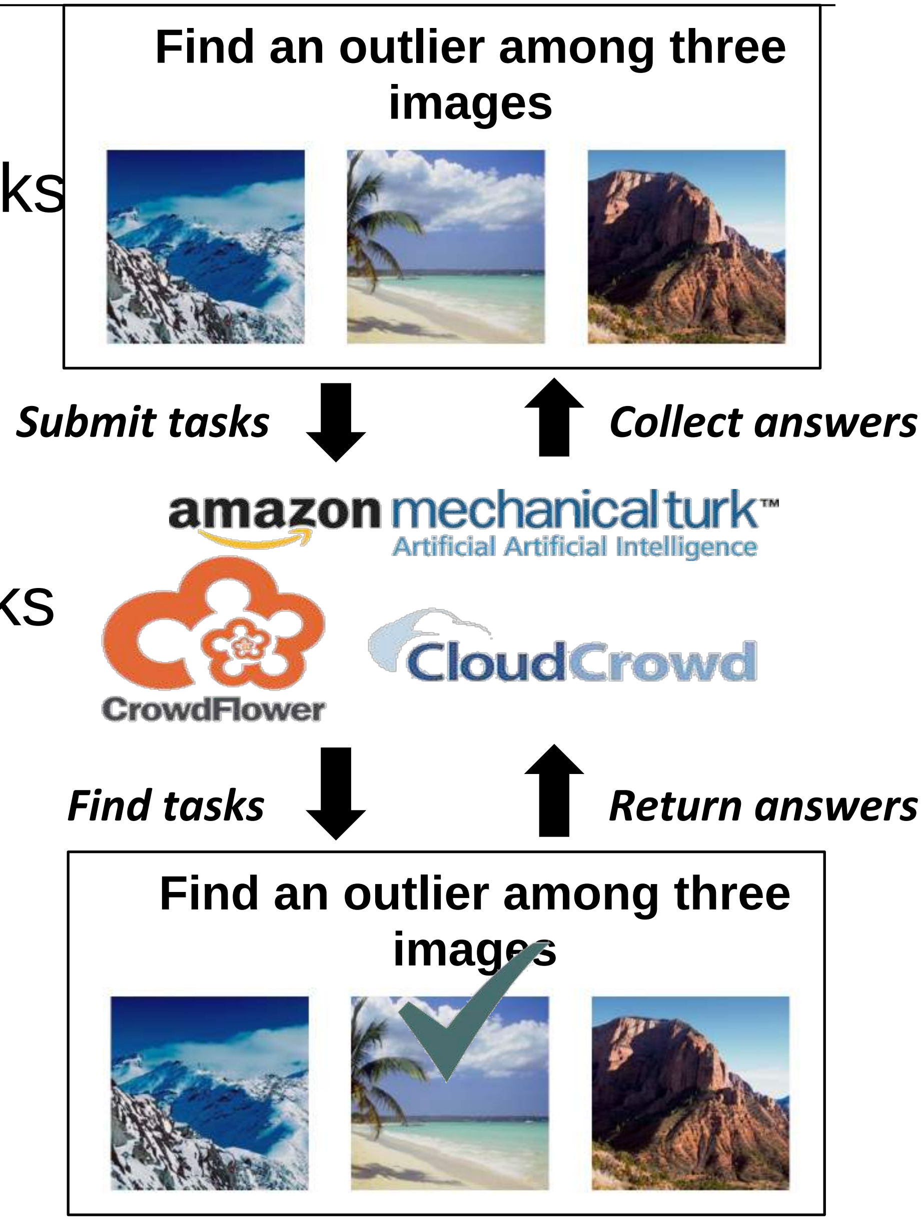
- Focus on computational aspect of crowdsourcing
- Mainly use micro-tasks
- Algorithmic aspect
- Optimization problem with three parameters
- When to use Computational Crowdsourcing?
  - Machine can't do the task well
  - Large crowds can do it well
  - Task can be split to many micro-tasks

Computational Crowdsourcing

- Requesters
  - People submit some tasks
  - Pay **rewards** to workers

- Marketplaces
  - Provide crowds with tasks

- Crowds
  - Workers perform tasks



Crowdsourcing Marketplaces

- Platforms for posting/performing (often micro) tasks

- Those who want to have tasks done via crowdsourcing ➔ Requesters
  - Eg, companies, researchers

- Those who want to perform tasks for monetary profits ➔ Workers
  - Eg, individuals for extra income

Crowdsourcing Platforms

- Notables ones:
  - Mechanical Turk (AMT)
  - CrowdFlower
  - CloudCrowd
  - Clickworker
  - SamaSource

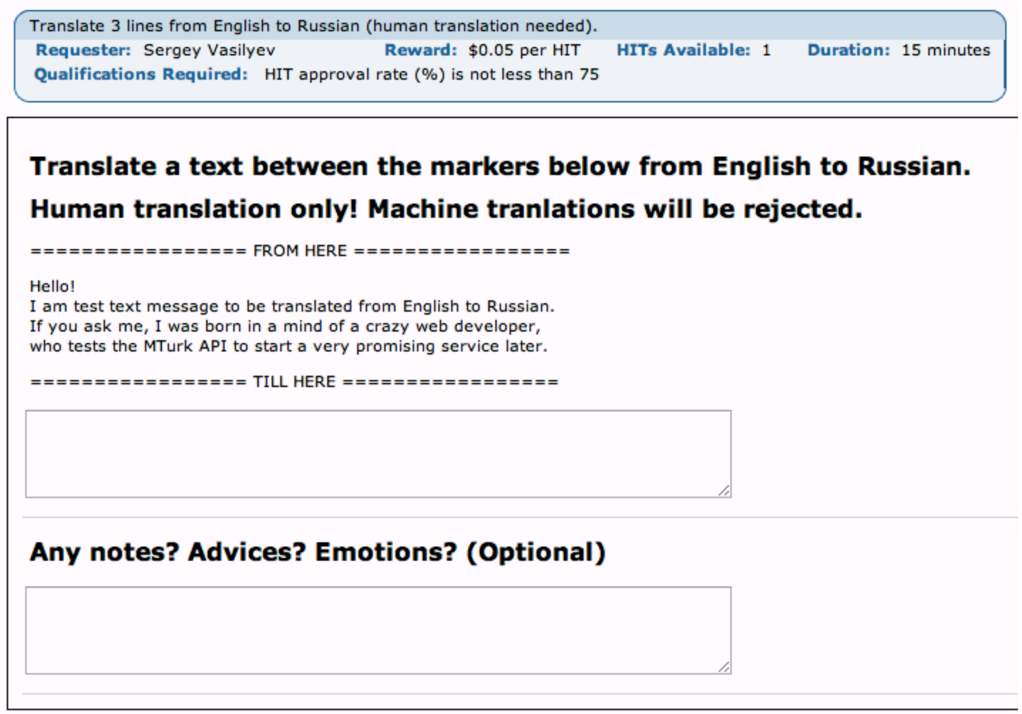
AMT: mturk.com

AMT (cont)

- Workers
  - Register w. credit account (only US workers can register as of 2013)
  - Bid to do tasks for earning money
- Requesters
  - First deposit money to account
  - Post tasks
    - Task can specify a qualification for workers
  - Gather results
  - Pay to workers if results are satisfactory

AMT (cont)

- Tasks
  - Called **HIT** (Human Intelligence Task)
  - Micro-task
- Eg
  - Data cleaning
  - Tagging / labeling
  - Sentiment analysis
  - Categorization
  - Surveying
  - Photo moderation
  - Transcription



Translate 3 lines from English to Russian (human translation needed).

Requester: Sergey Vasilyev **Reward:** $0.05 per HIT **HITs Available:** 1 **Duration:** 15 minutes

Qualifications Required: HIT approval rate (%) is not less than 75

Translate a text between the markers below from English to Russian.

Human translation only! Machine tranlations will be rejected.

================= FROM HERE =================

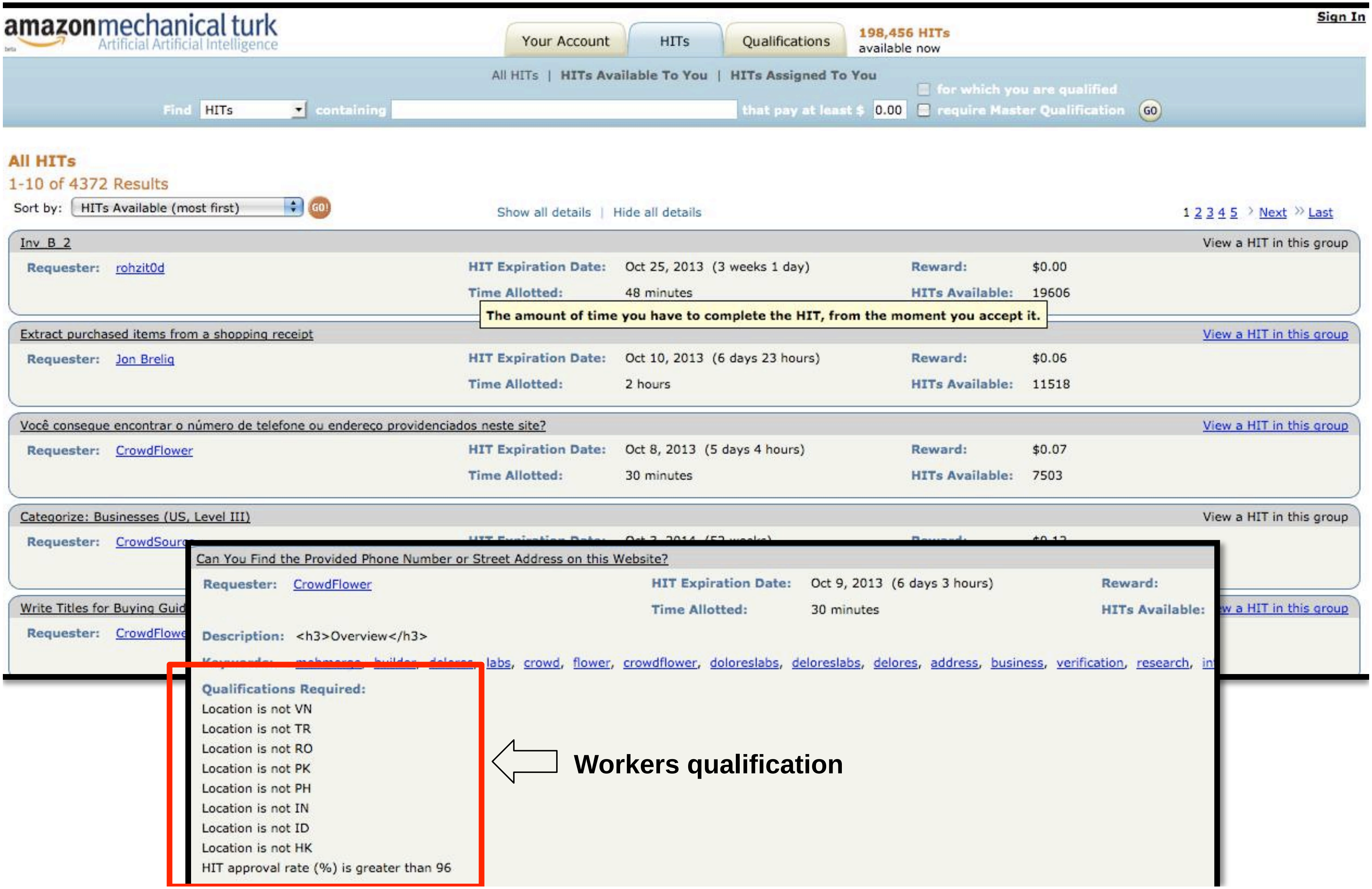
Hello!
I am test text message to be translated from English to Russian.
If you ask me, I was born in a mind of a crazy web developer,
who tests the MTurk API to start a very promising service later.

================= TILL HERE =================

Any notes? Advices? Emotions? (Optional)

Translation task

AMT: HIT List


amazonmechanical turk
Artificial Artificial Intelligence
Your Account
HITs
Qualifications
198,456 HITs available now
Sign In
All HITs | HITs Available To You | HITs Assigned To You
Find HITs containing that pay at least $ 0.00
for which you are qualified
require Master Qualification
GO
All HITs
1-10 of 4372 Results
Sort by: HITs Available (most first)
Show all details | Hide all details
1 2 3 4 5 Next Last
Inv_B_2
View a HIT in this group
Requester: rohzit0d
HIT Expiration Date: Oct 25, 2013 (3 weeks 1 day)
Reward: $0.00
Time Allotted: 48 minutes
HITs Available: 19606
The amount of time you have to complete the HIT, from the moment you accept it.
Extract purchased items from a shopping receipt
View a HIT in this group
Requester: Jon Brelig
HIT Expiration Date: Oct 10, 2013 (6 days 23 hours)
Reward: $0.06
Time Allotted: 2 hours
HITs Available: 11518
Você consegue encontrar o número de telefone ou endereço providenciados neste site?
View a HIT in this group
Requester: CrowdFlower
HIT Expiration Date: Oct 8, 2013 (5 days 4 hours)
Reward: $0.07
Time Allotted: 30 minutes
HITs Available: 7503
Categorize: Businesses (US, Level III)
View a HIT in this group
Write Titles for Buying Guid
Can You Find the Provided Phone Number or Street Address on this Website?
Requester: CrowdFlower
HIT Expiration Date: Oct 9, 2013 (6 days 3 hours)
Reward:
Time Allotted: 30 minutes
HITs Available:
Description: <h3>Overview</h3>
labs, crowd, flower, crowdflower, doloreslabs, deloreslabs, delores, address, business, verification, research,
Qualifications Required:
Location is not VN
Location is not TR
Location is not RO
Location is not PK
Location is not PH
Location is not IN
Location is not ID
Location is not HK
HIT approval rate (%) is greater than 96
Workers qualification

AMT: HIT Example

Can You Find the Provided Phone Number or Street Address on this Website?

Instructions ▲

Overview

In this task, you'll be provided a web page for a business, including its name, address, and phone number. Your goal is to answer a few questions about the business on the web page.

IMPORTANT: Sometimes the business will have multiple locations, and you will have to search the website for the specific business that we provide in order to verify the website.

Step by step instructions:

- Click the link to go to the provided site.
- First, please tell us whether or not the **name** of the business on the provided website is a **close** or **identical** match to the name of the business shown at the top of the page.
- Next, please tell us whether the provided business has
- Please be sure to click the appropriate option if the sit

Wrinkles Day Spa

Phone: +61893455333
Street: Shop 5a Stirling Central Shopping Centre, 478 Wanneroo Rd
City: Westminster
State: WA
Postalcode (Zip): 6061
Country Code: AU

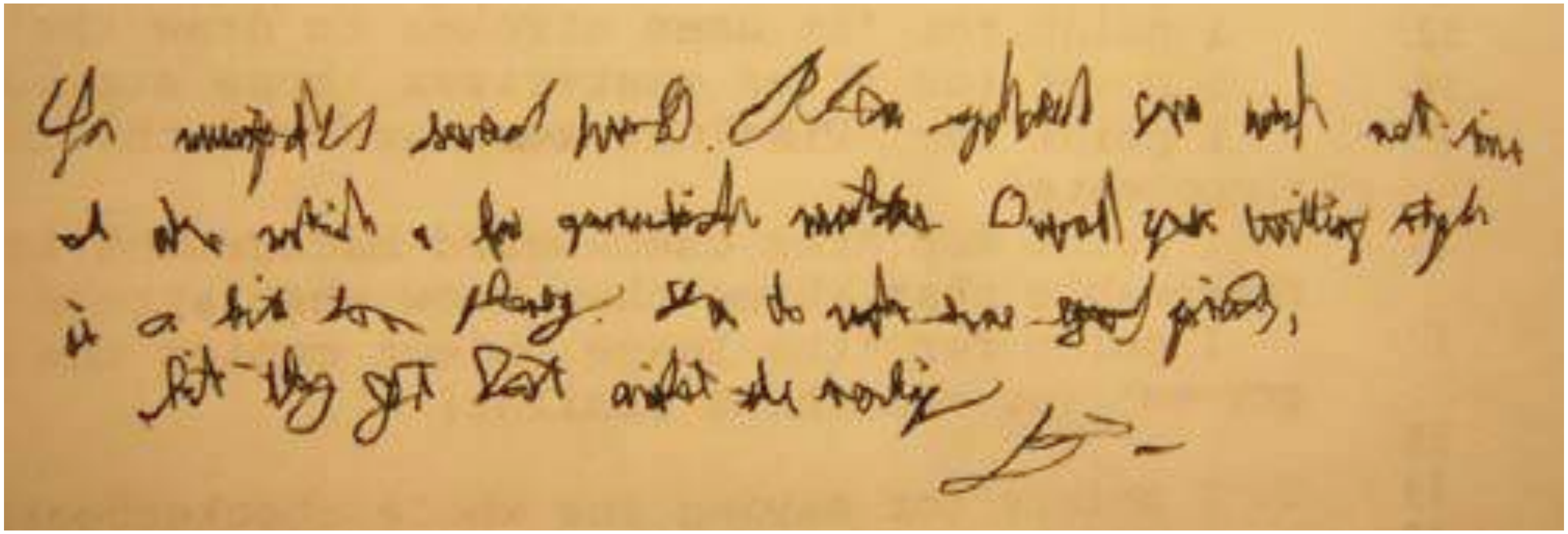
Click here to visit the website.

Is the name of the business on the web page similar or identical to 'Wrinkles Day Spa'?

- Yes: the name of the business is *similar* to *Wrinkles Day Spa*
- Yes: the name of the business is *nearly identical* to *Wrinkles Day Spa*
- No: the name is very different from *Wrinkles Day Spa*

For the first option, the street **number** does not need to match, just the street, **Shop 5a Stirling Central Shopping Centre, 4**

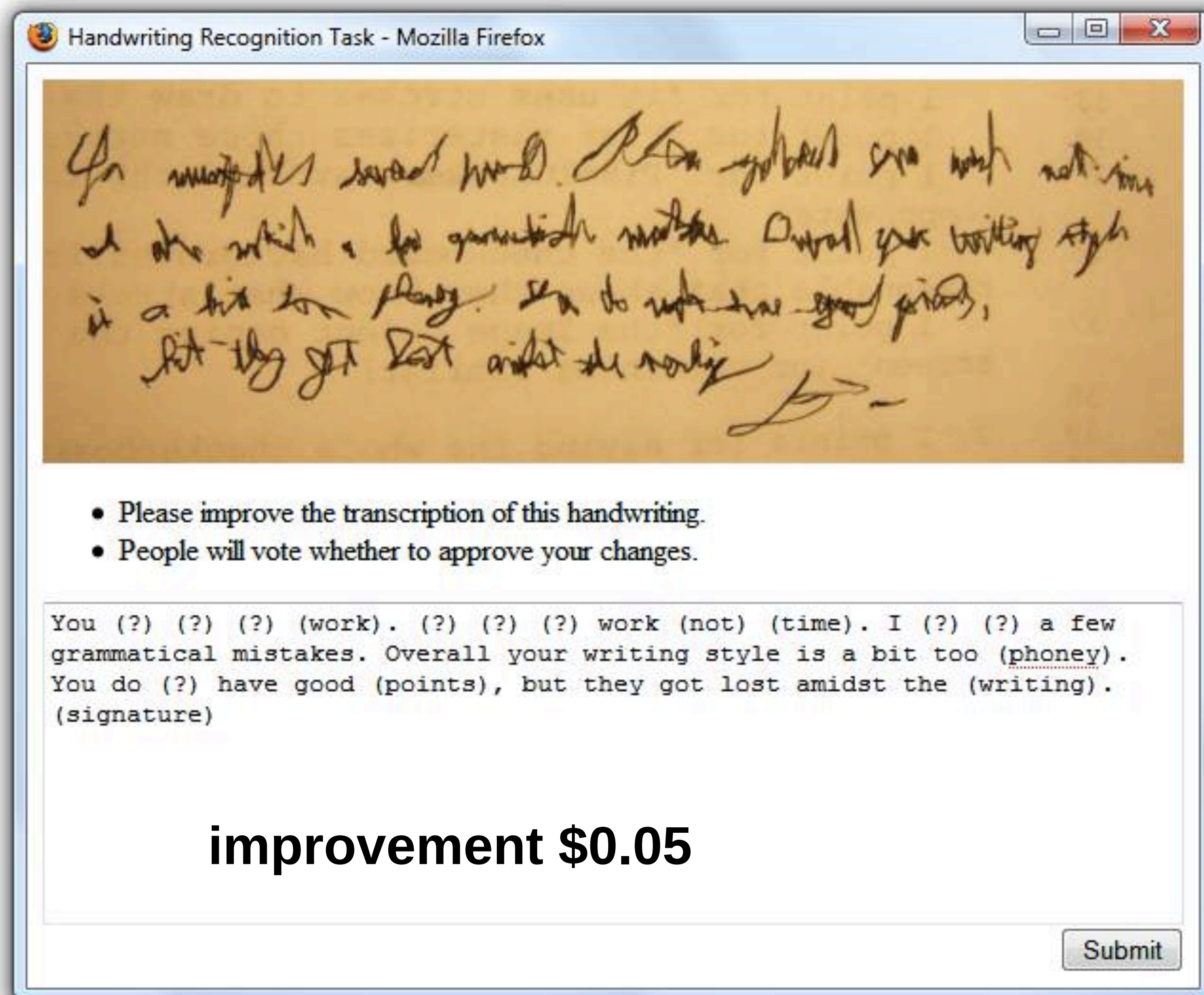
Eg, Text Transcription [Miller-13]



- **Problem**: one person can't do a good transcription
- **Key idea**: iterative improvement by many workers

Greg Little *et al*. "Exploring iterative and parallel human computation processes." HCOMP 2010

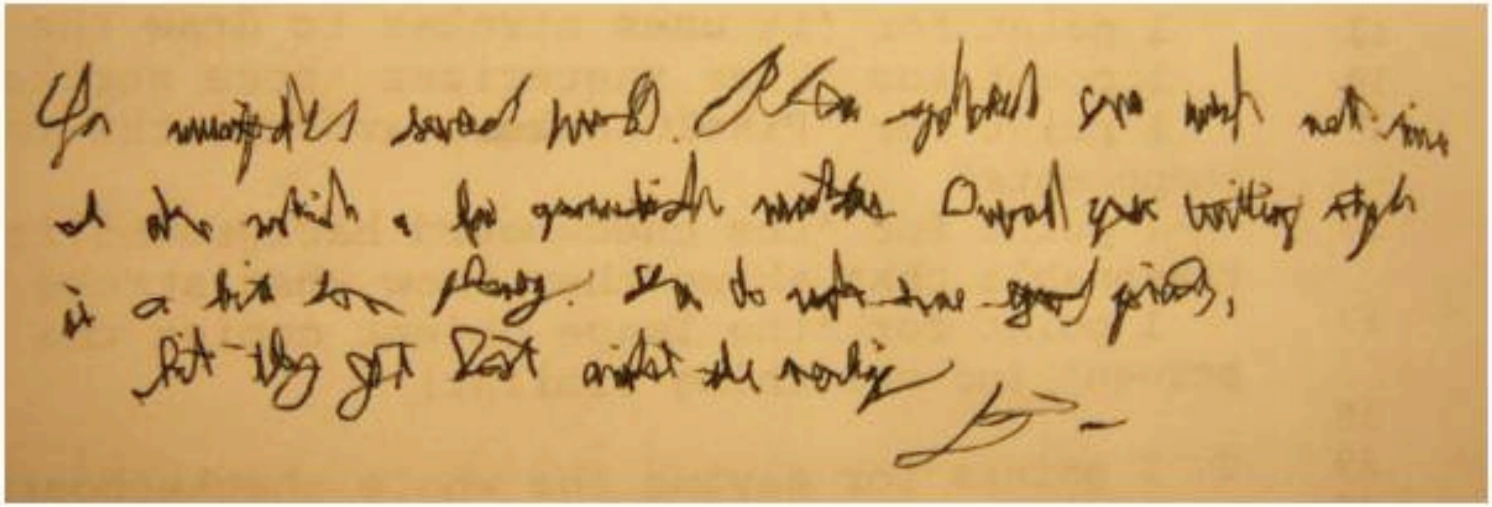
Eg, Text Transcription [Miller-13]



improvement $0.05

Eg, Text Transcription [Miller-13]

MTurk Task - Mozilla Firefox



- Please select the better transcription for this handwriting.
- Differences are highlighted in yellow.

3 votes @ $0.01

\> You (misspelled) (several) (words) (work). (?) (?) (?) work next (time). I also notice a few grammatical mistakes. Overall your writing style is a bit too (phoney). You do (?) have good (points), but they got lost amidst the (writing). (signature)

\> You (?) (?) (?) (work). (?) (?) (?) work (not) (time). I (?) (?) a few grammatical mistakes. Overall your writing style is a bit too (phoney). You do (?) have good (points), but they got lost amidst the (writing). (signature)

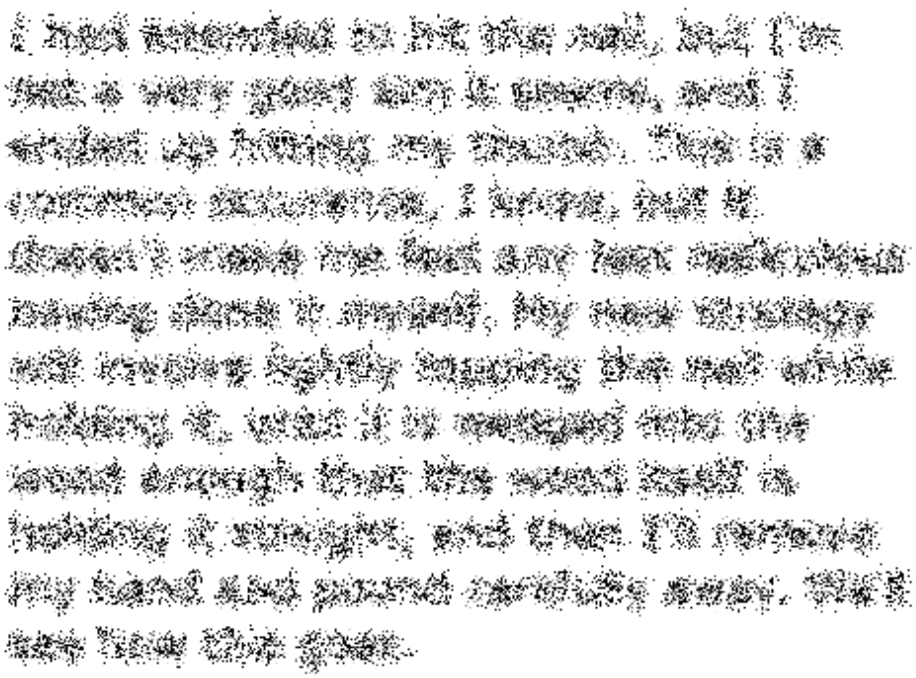
Eg, Text Transcription [Miller-13]

"You (misspelled) (several) (words). Please spellcheck your work next time. I also notice a few grammatical mistakes. Overall your writing style is a bit too phoney. You do make some good (points), but they got lost amidst the (writing). (signature)"

According to our ground truth, the highlighted words should be "flowery", "get", "verbiage" and "B-" respectively.

After 9 iterations

Eg, Text Transcription [Miller-13]



I had intended to hit the nail, but I'm not a very good aim it seems and I ended up hitting my thumb. This is a common occurrence I know, but it doesn't make me feel any less ridiculous having done it myself. My new strategy will involve lightly tapping the nail while holding it until it is embedded into the wood enough that the wood itself is holding it straight and then I'll remove my hand and pound carefully away. We'll see how this goes.

Another example: blurry text
After 8 iterations

Eg, Computer Vision [Li-HotDB12]

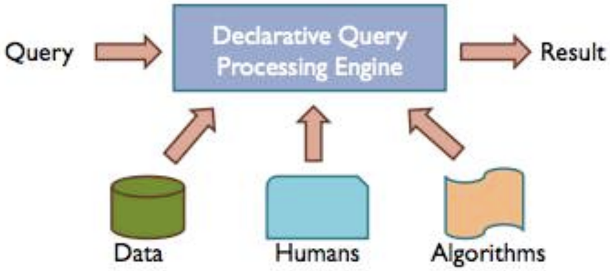
- How similar is the artistic style?

Human and Machine Detection of Stylistic Similarity in Art.
Adriana Kovashka and Matthew Lease. CrowdConf 2010

TOC

Crowdsourcing DB Projects

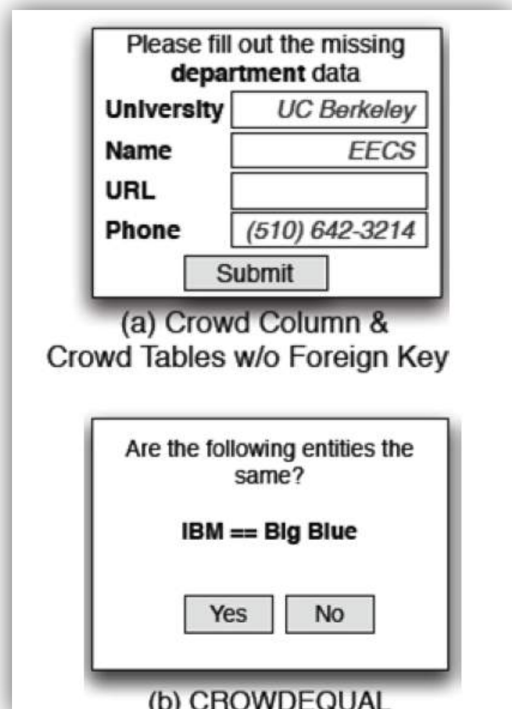
- CDAS @ NUS
- CrowdDB @ UC Berkeley & ETH Zurich
- MoDaS @ Tel Aviv U.
- Qurk @ MIT
- sCOOP @ Stanford & UCSC



Eg, CrowdDB System

- Crowd-enabled databases
  - Hybrid human/machine databases
  - Building a database engine that can dynamically **crowdsource** certain operations

```
CREATE TABLE Department (
  university STRING,
  name STRING,
  url  CROWD STRING,
```



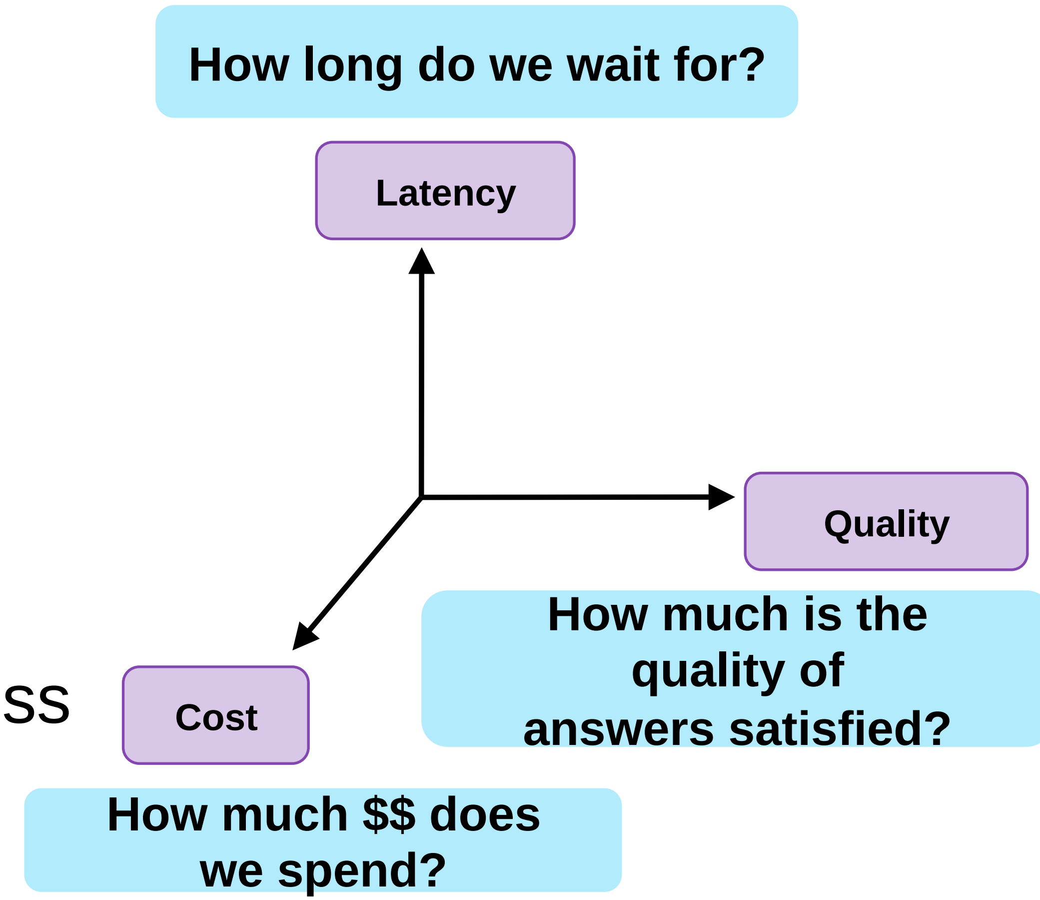
(a) Crowd Column & Crowd Tables w/o Foreign Key

(b) CROWDEQUAL

[Franklin-SIGMOD11]

Preliminaries: 3 Factors

- Latency (or execution time)
  - Worker pool size
  - Job attractiveness
- Monetary cost
  - # of questions
  - # of workers
  - Cost per question
- Quality of answers
  - Worker maliciousness
  - Worker skills
  - Task difficulty



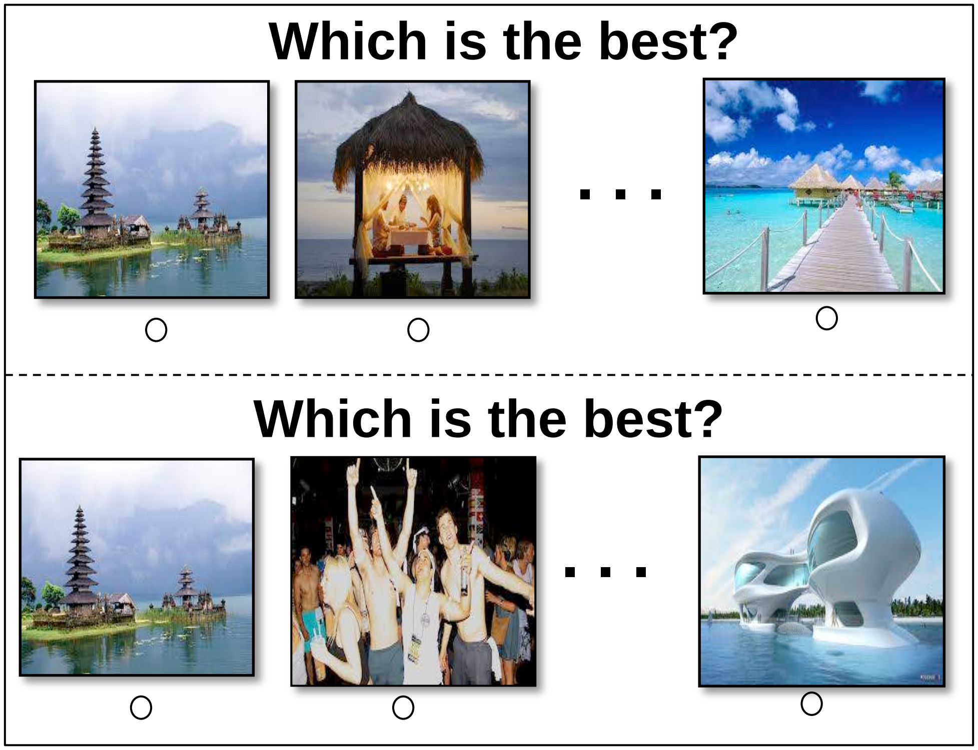
Preliminaries: Size of Comparison

- Diverse forms of questions in a HIT
- Different sizes of comparisons in a question
  - Eg, Binary question
    - $s = 2$

  - Eg, *N*-ary question
    - $s = N$

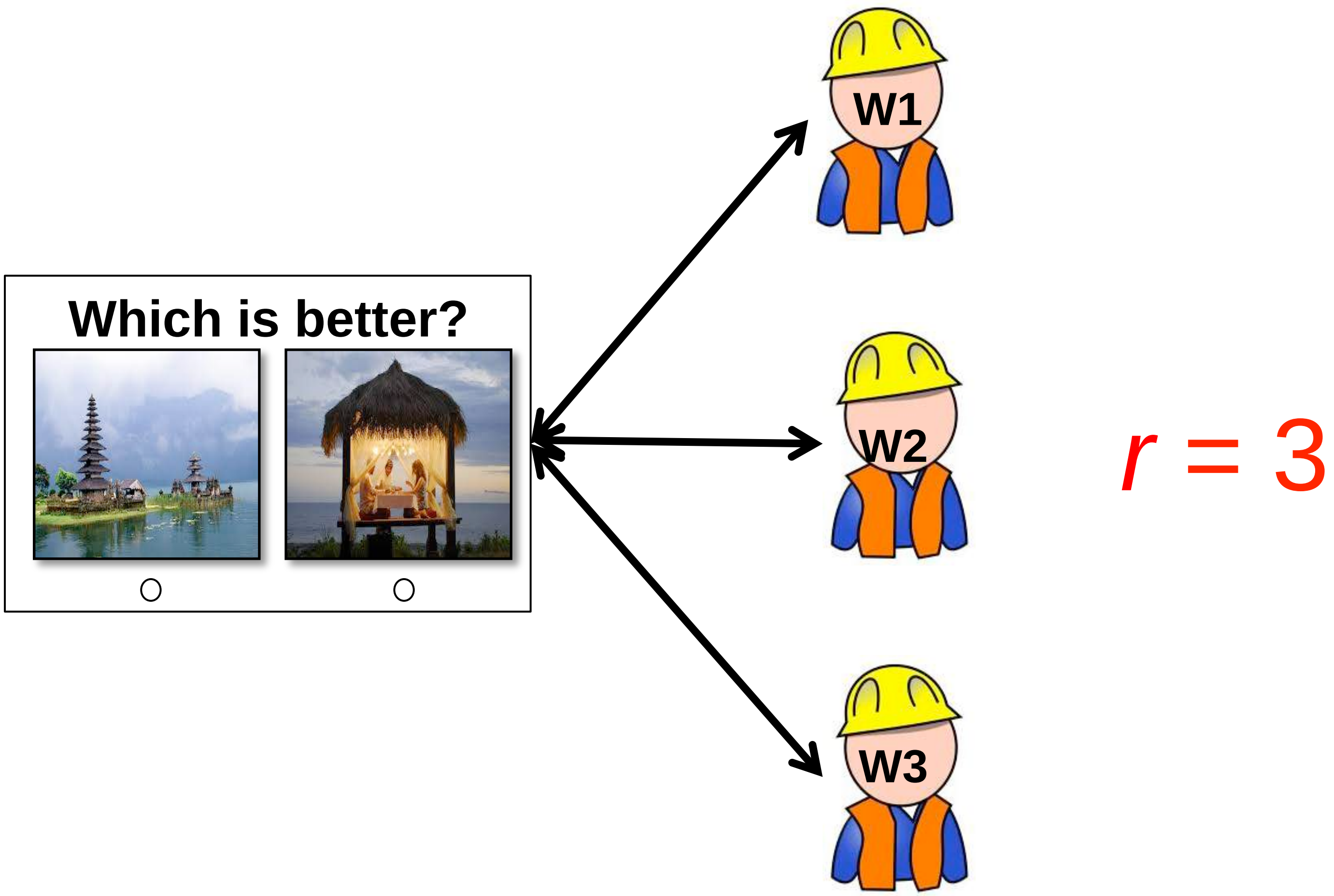
Preliminaries: Batch

- Repetitions of questions within a HIT
- Eg, two *n*-ary questions (batch factor b=2)



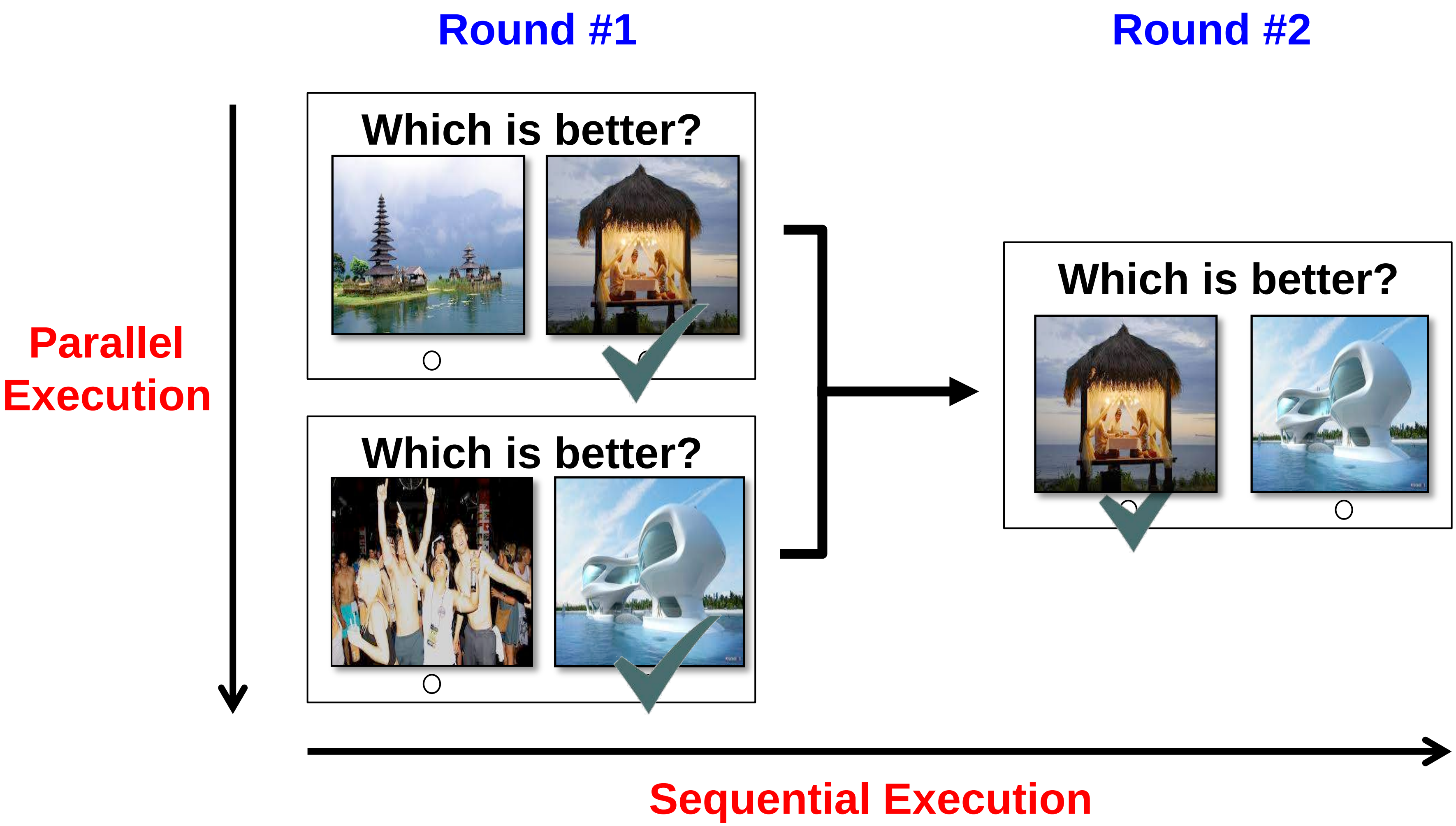
Preliminaries: Response (*r*)

- # of human responses seeked for a HIT



Preliminaries: Round (= Step)

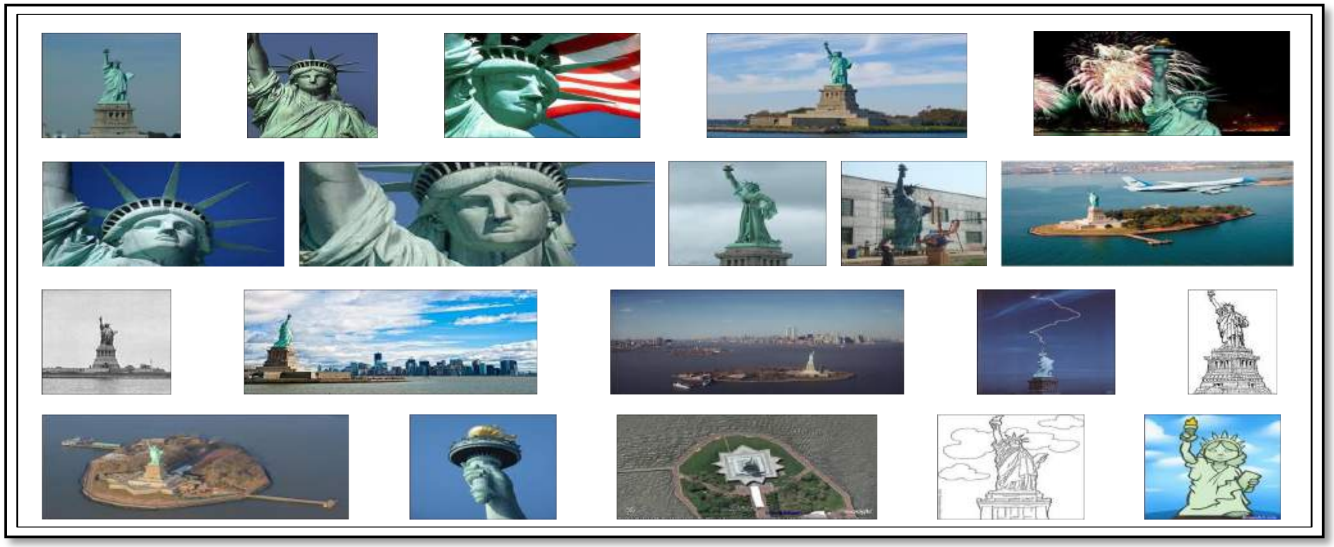
- Algorithms are executed in rounds
- # of rounds ≈ latency



Sort Operation

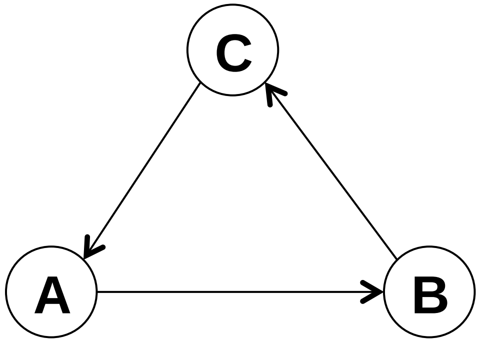
- Rank *N* items using crowdsourcing with respect to the constraint *C*
  - Eg, *C* as “Representative,” “Dangerous,” “Beautiful”

```
SELECT      *
FROM        ImageTable AS I
WHERE       I.Date > 2014 AND I.loc = “NY”
ORDER BY    CrowdOp(“Representative”)
```



Naïve Sort

- Assuming pair-wise comparison of 2 items
  - Eg, “Which of two images is better?”
- Cycle: A > B, B > C, and C > A
- If no cycle occurs
  - Naïve all pair-wise comparisons takes $\binom{N}{2}$ comparisons
  - Optimal # of comparison is *O(N log N)*
- If cycle exists
  - More comparisons are required



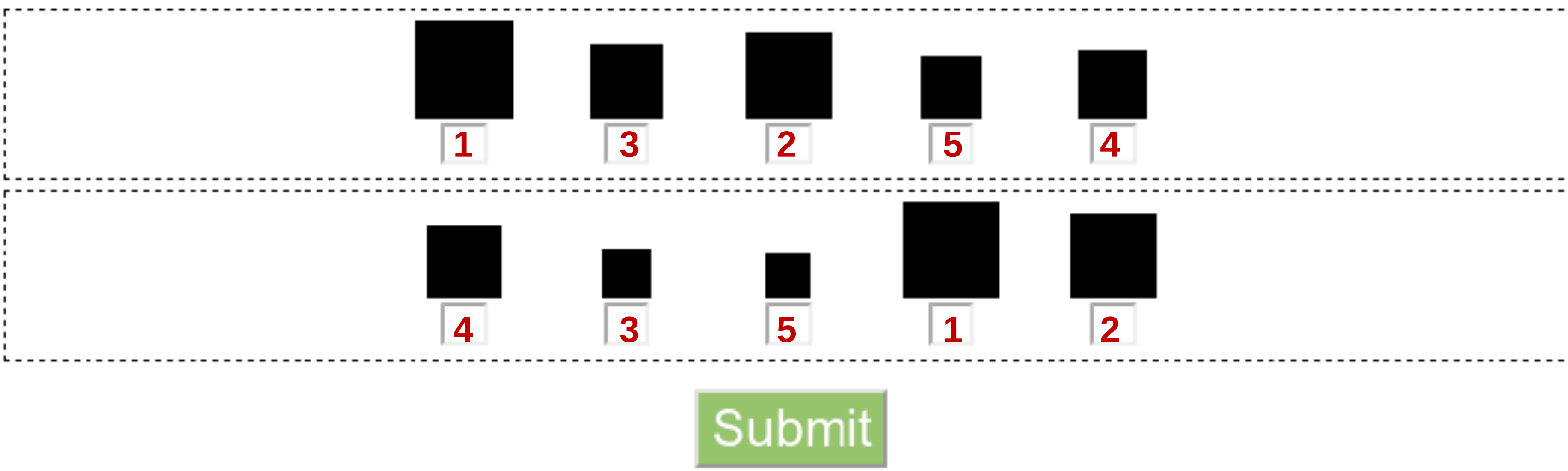
Sort [Marcus-VLDB11]

- Proposed 3 crowdsourced sort algorithms
- #1: Comparison-based Sort
  - Workers rank *S* items ($S \subset N$) per HIT
  - Each HIT yields $\binom{S}{2}$ pair-wise comparisons
  - Build a DAG using all pair-wise comparisons from all workers
    - If *i* > *j*, then add an edge from *i* to *j*
  - Break a cycle in the DAG: “head-to-head”
    - Eg, If *i* > *j* occurs 3 times and *i* < *j* occurs 2 times, keep only *i* > *j*
  - Perform a topological sort in the DAG

Sort [Marcus-VLDB11]

There are 2 groups of squares. We want to order the squares in each group from smallest to largest.

- Each group is surrounded by a dotted line. Only compare the squares within a group.
- Within each group, assign a number from 1 to 7 to each square, so that:
    - 1 represents the smallest square, and 7 represents the largest.
    - We do not care about the specific value of each square, only the relative order of the squares.
    - Some groups may have less than 7 squares. That is OK: use less than 7 numbers, and make sure they are ordered according to size.
    - If two squares in a group are the same size, you should assign them the same number.



Sort [Marcus-VLDB11]

- N=5, S=3

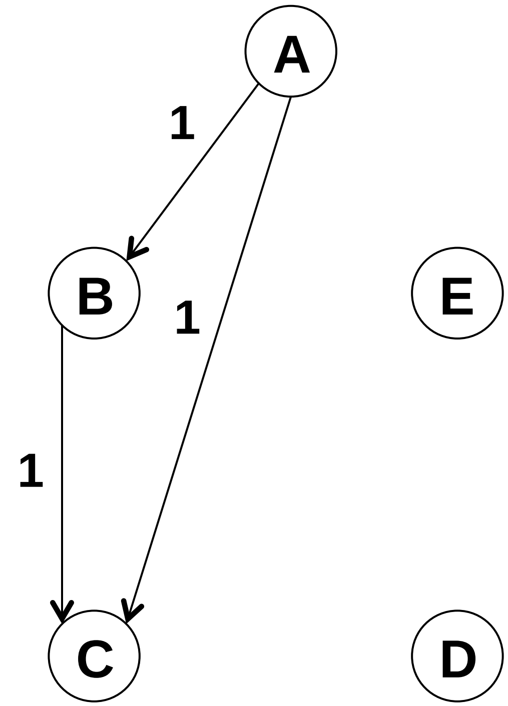
A

B

C

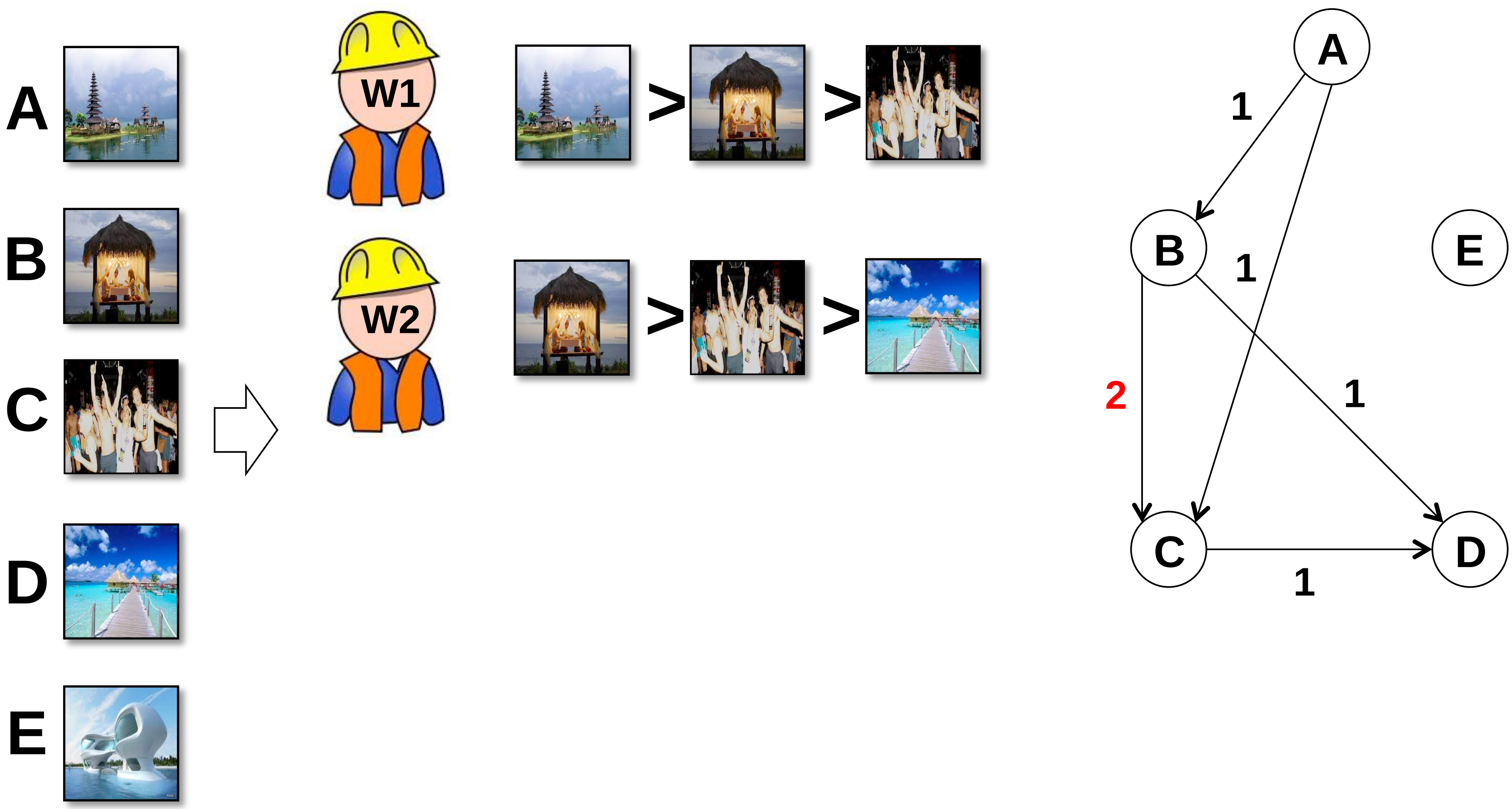
D

E

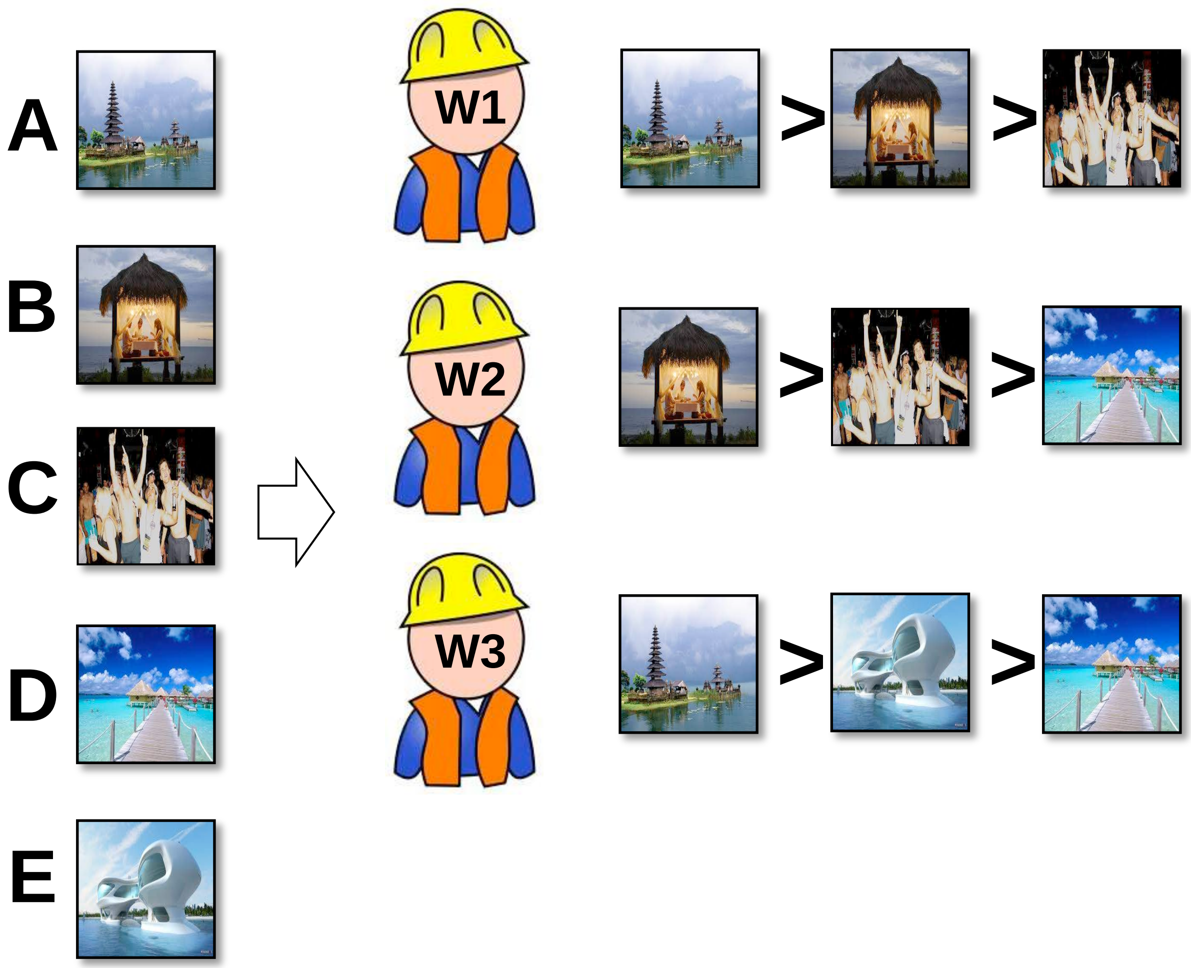
Sort [Marcus-VLDB11]

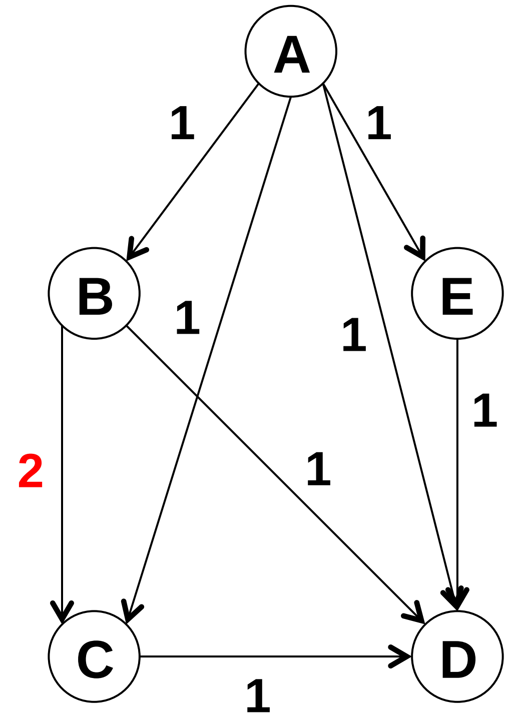
- N=5, S=3



Sort [Marcus-VLDB11]

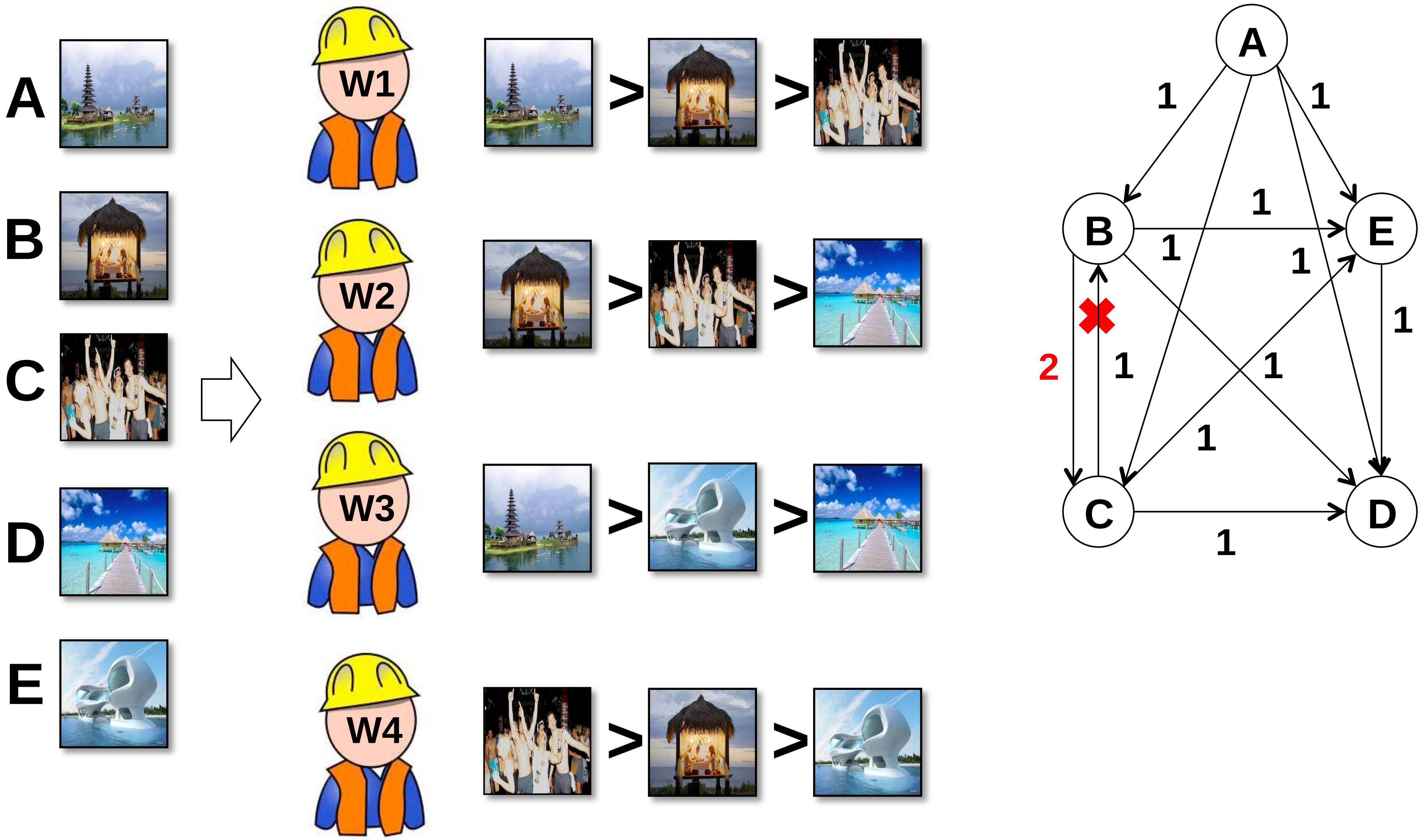
- N=5, S=3





Sort [Marcus-VLDB11]

- N=5, S=3



Sort [Marcus-VLDB11]

- N=5, S=3

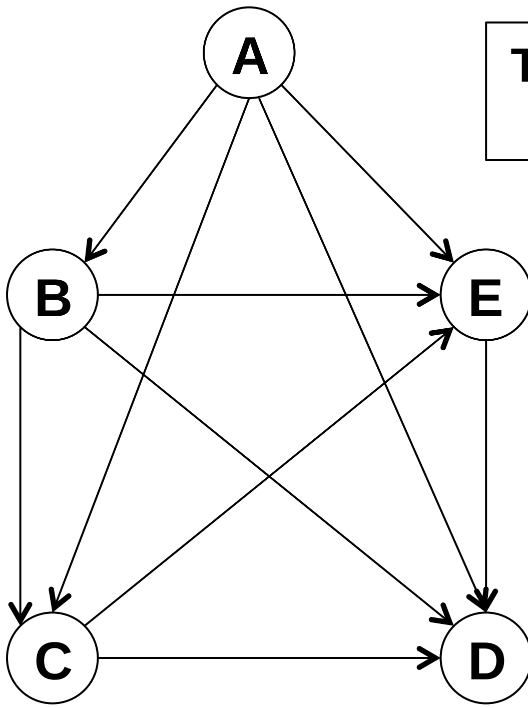
A 

B 

C 

D 

E 



Topological Sort

A > B > C > E > D

Sort [Marcus-VLDB11]

- #2: Rating-based Sort
  - *W* workers rate each item along a numerical scale
  - Compute the mean of *W* ratings of each item
  - Sort all items using their means
  - Requires *W*N* HITs: *O(N)*

Worker	Rating
W1	4
W2	3
W3	4

. . .

Worker	Rating
W1	1
W2	2
W3	1

Mean rating

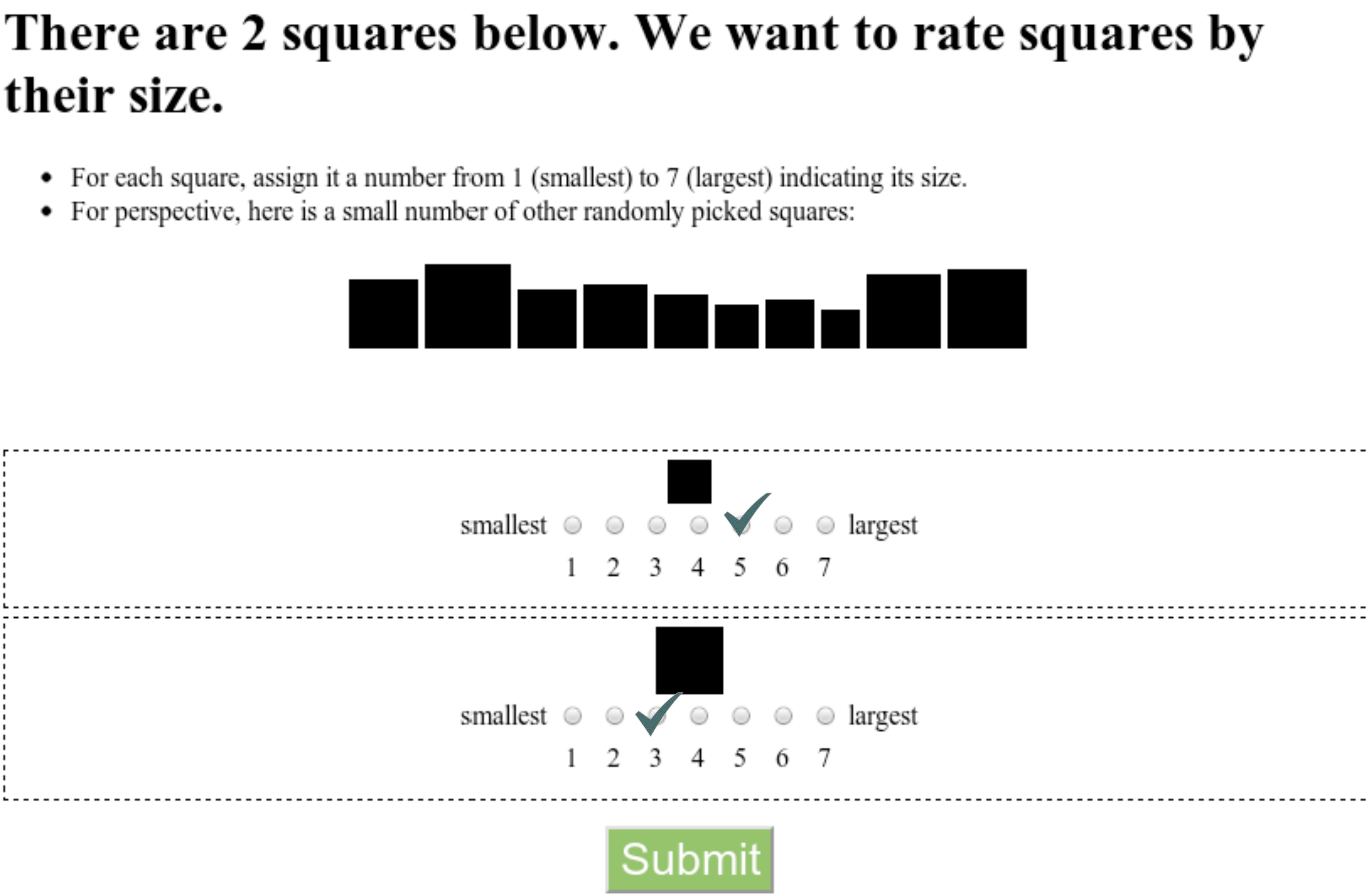
1.3

3.6

. . .

8.2

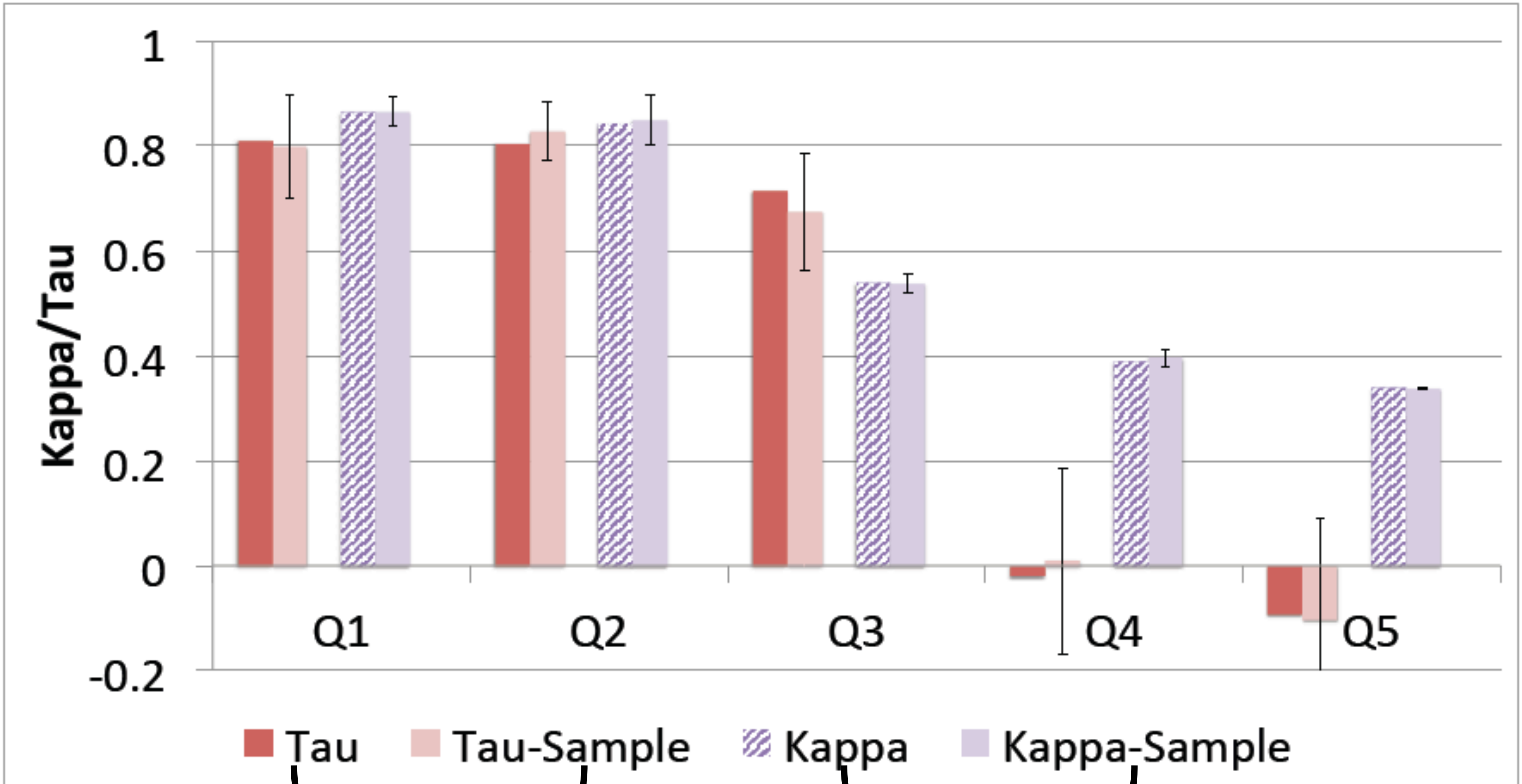
Sort [Marcus-VLDB11]



Sort [Marcus-VLDB11]

- #3: Hybrid Sort
  - First, do rating-based sort → sorted list *L*
  - Second, do comparison-based sort on *S* ($S \subset L$)
  - How to select the size of S
    - Random
    - Confidence-based
    - Sliding window

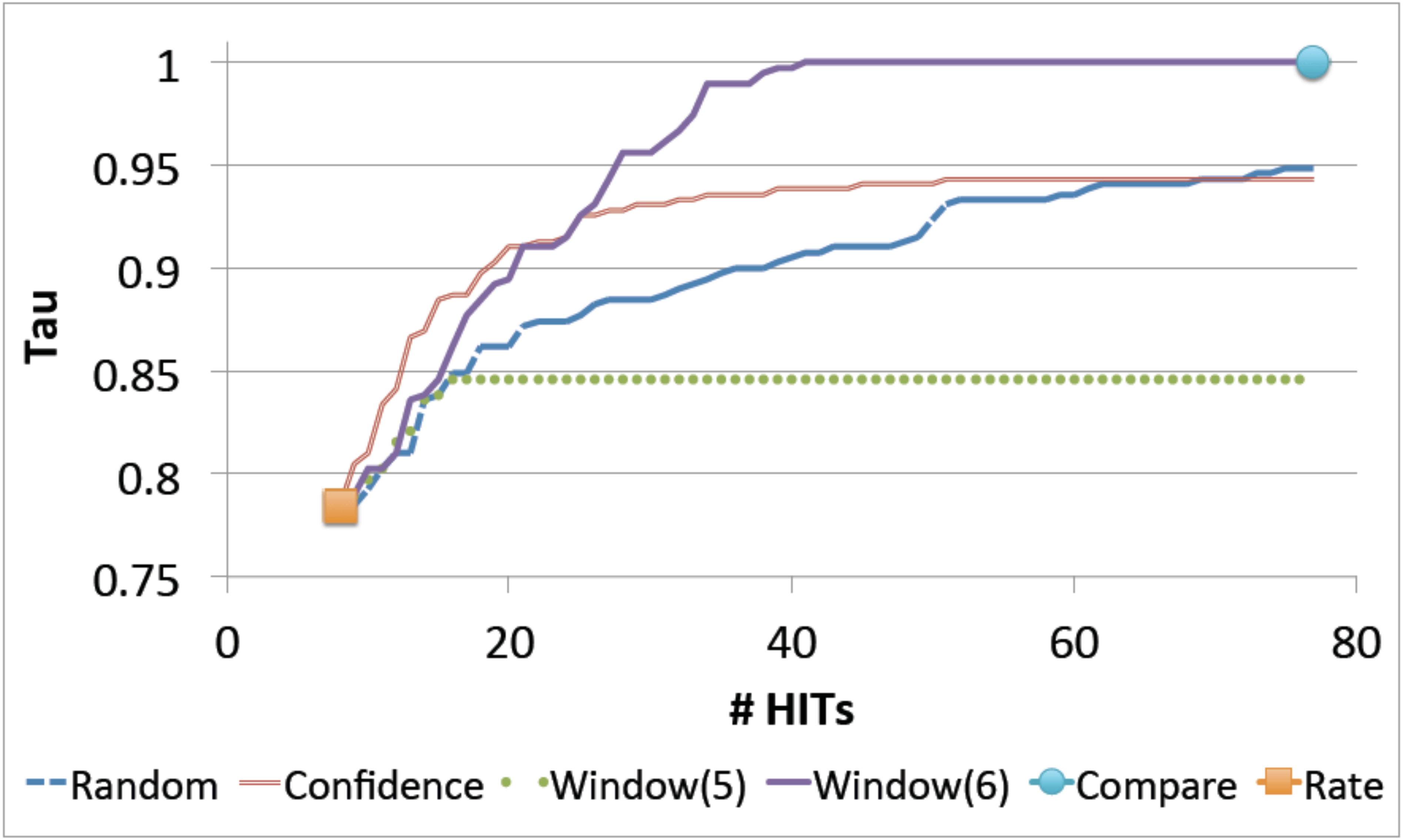
Sort [Marcus-VLDB11]



Rank correlation btw. Comparison vs. rating

Worker agreement

Sort [Marcus-VLDB11]



Top-1 Operation

- Find the top-1, either MAX or MIN, among *N* items w.r.t. “something”

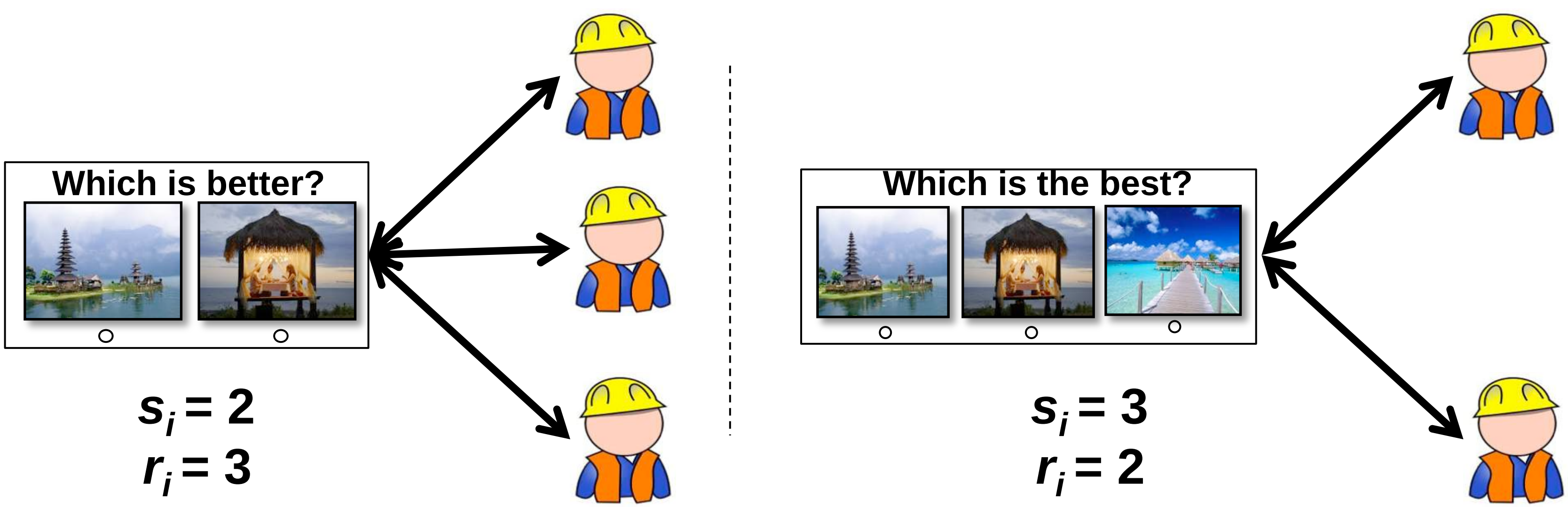
- Objective
  - Avoid sorting all *N* items to find top-1

Top-1 Operation

- Examples
  - **[Venetis-WWW12]** introduces the bubble max and tournament-based max in a parameterized framework
  - **[Guo-SIGMOD12]** studies how to find max using pair-wise questions in the tournament-like setting and how to improve accuracy by asking more questions

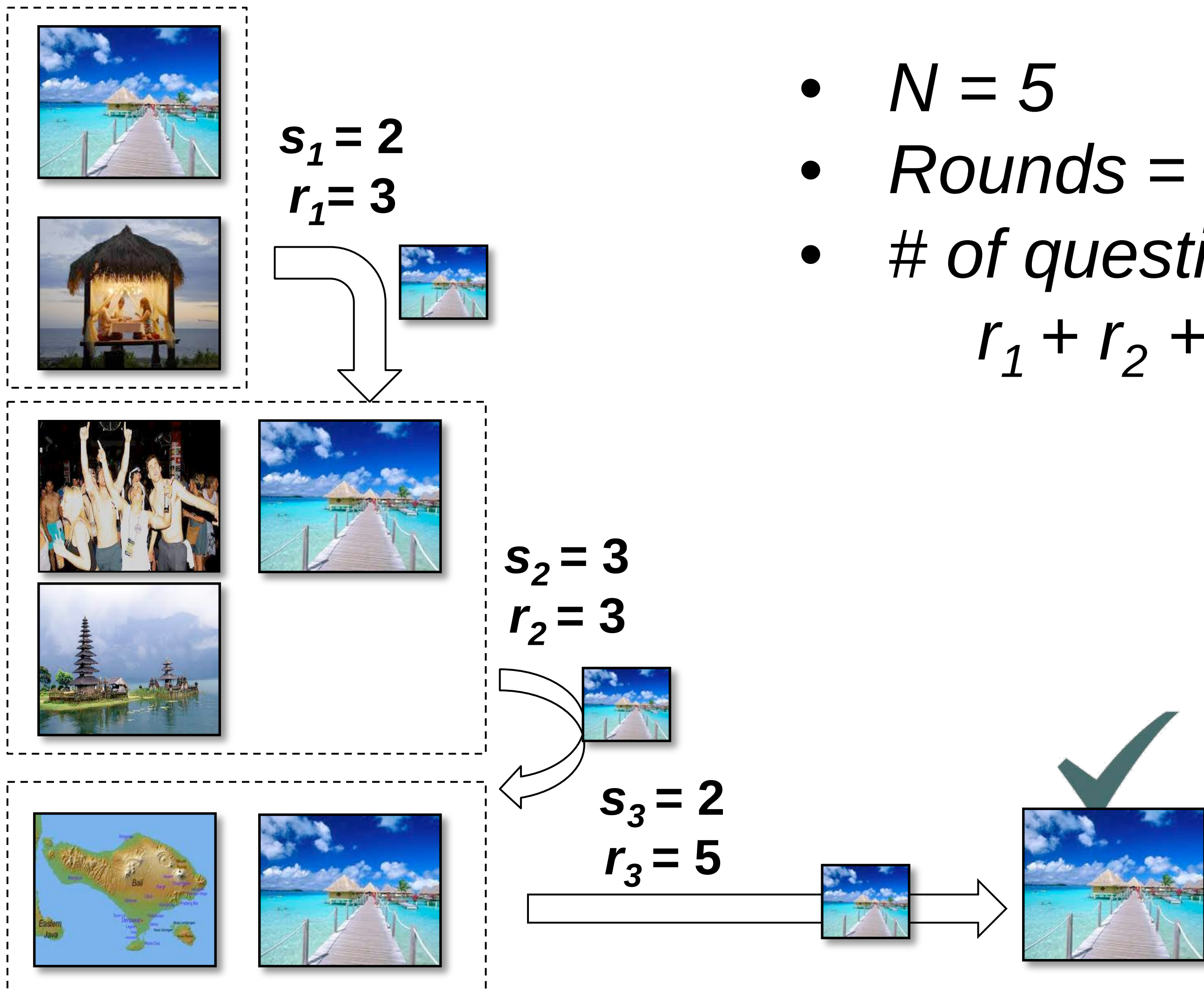
Max [Venetis-WWW12]

- Introduced two Max algorithms
  - Bubble Max
  - Tournament Max
- Parameterized framework
  - s_i: size of sets compared at the i-th round
  - r_i: # of human responses at the i-th round



Max [Venetis-WWW12]

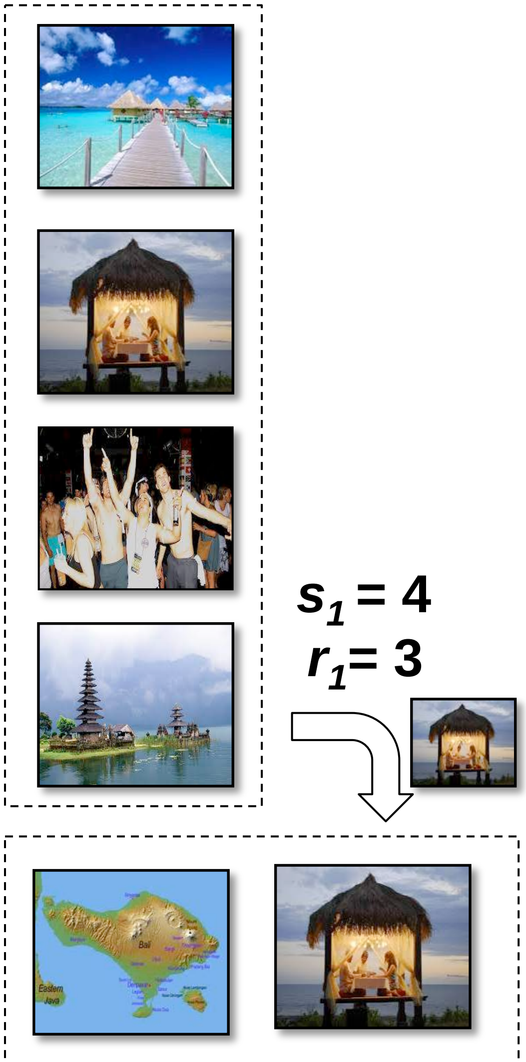
- Bubble Max Case #1



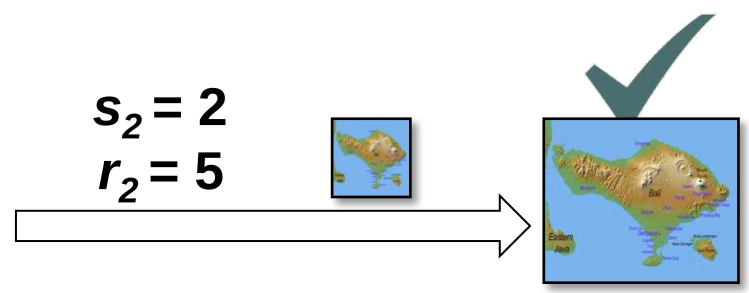
- $N = 5$
- $Rounds = 3$
- $\#\ of\ questions = r_1 + r_2 + r_3 = 11$

Max [Venetis-WWW12]

- Bubble Max Case #2

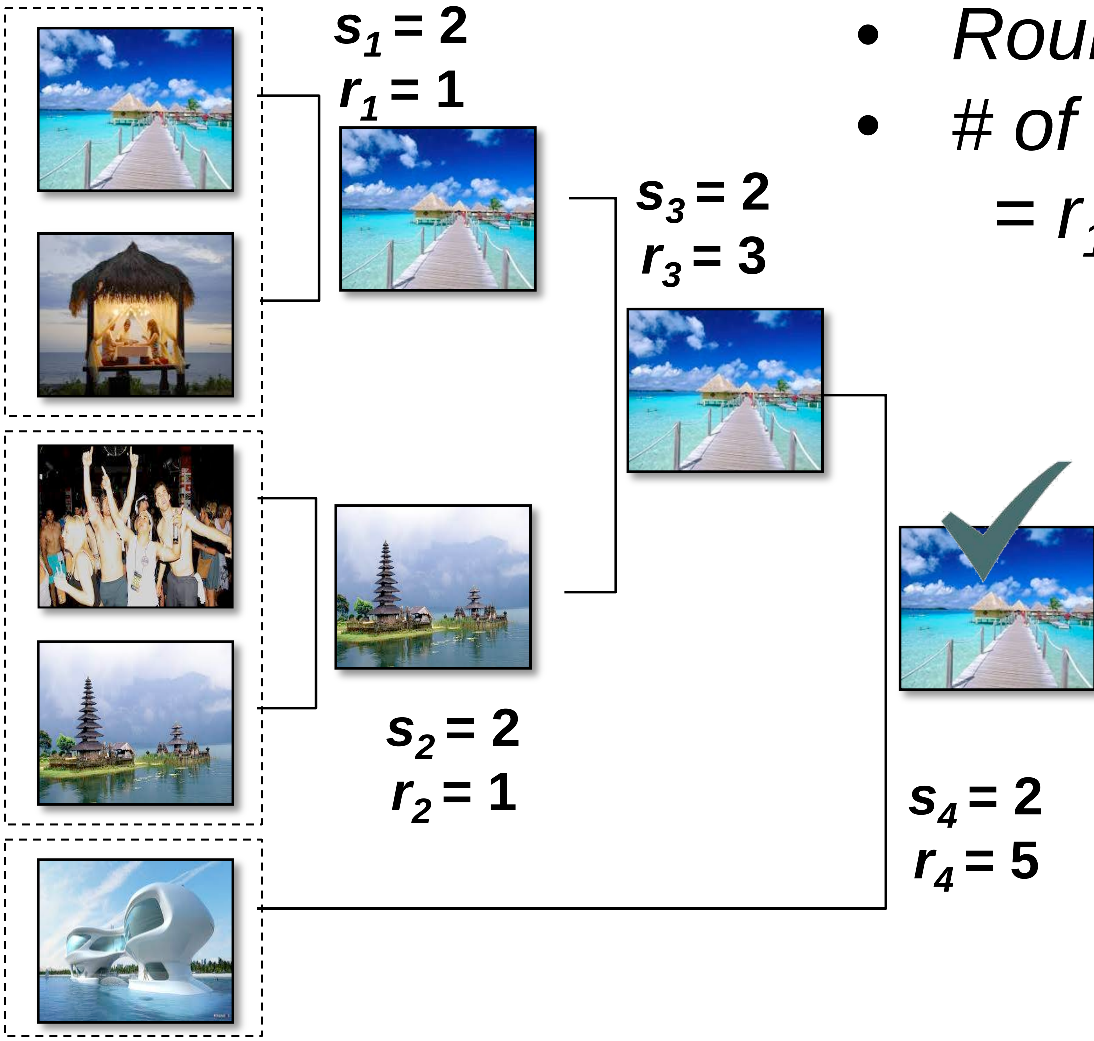


- $N = 5$
- $Rounds = 2$
- $\#\ of\ questions =$
  $r_1 + r_2 = 8$



Max [Venetis-WWW12]

- Tournament Max



- *$N = 5$*
- *Rounds = 3*
- *# of questions*
  $= r_1 + r_2 + r_3 + r_4 = 10$

Max [Venetis-WWW12]

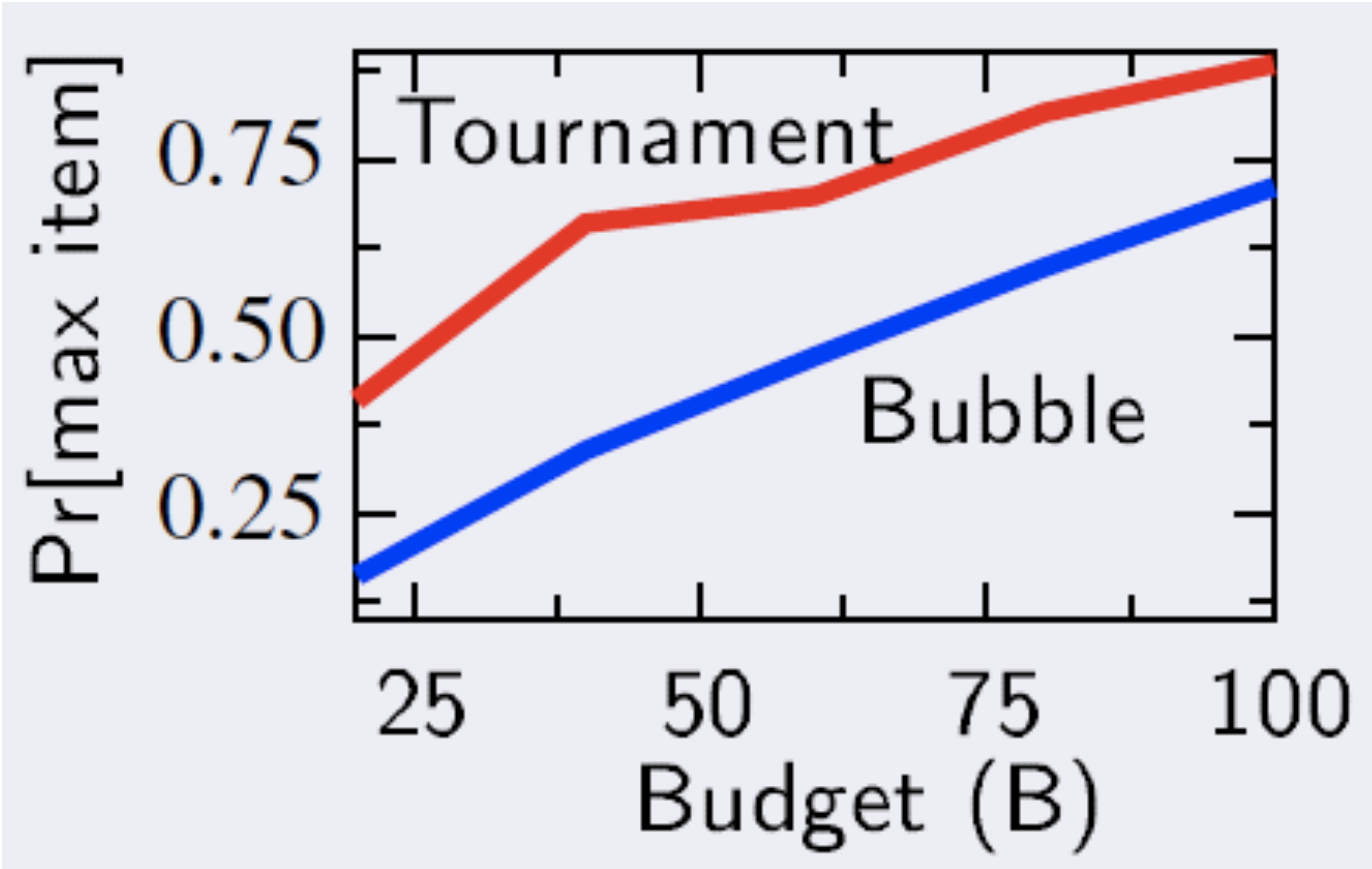
- How to find optimal parameters?: s_i and r_i
- Tuning Strategies (using Hill Climbing)
  - Constant s_i and r_i
  - Constant s_i and varying r_i
  - Varying s_i and r_i

Max [Venetis-WWW12]

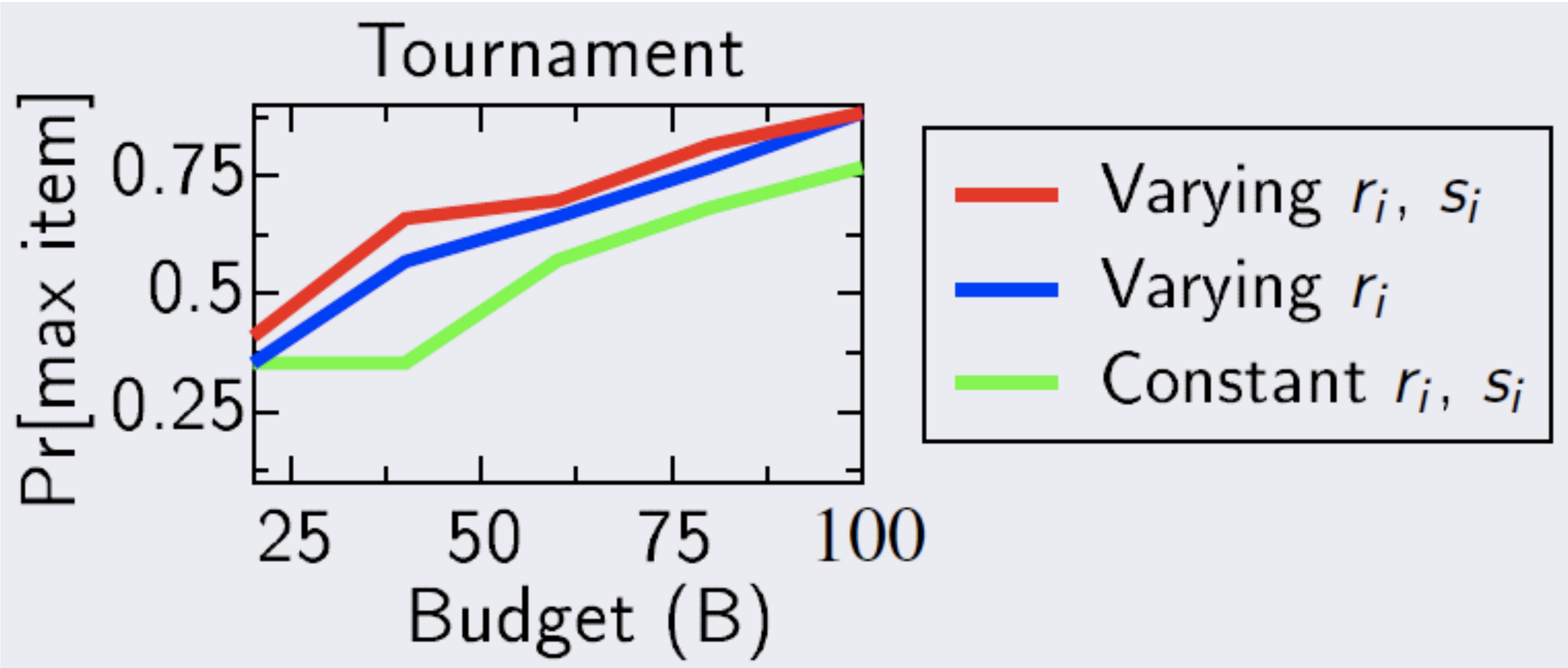
- Bubble Max
  - Worst case: with s_i=2, O(N) comparisons needed
- Tournament Max
  - Worst case: with s_i=2, O(N) comparisons needed

- Bubble Max is a special case of Tournament Max

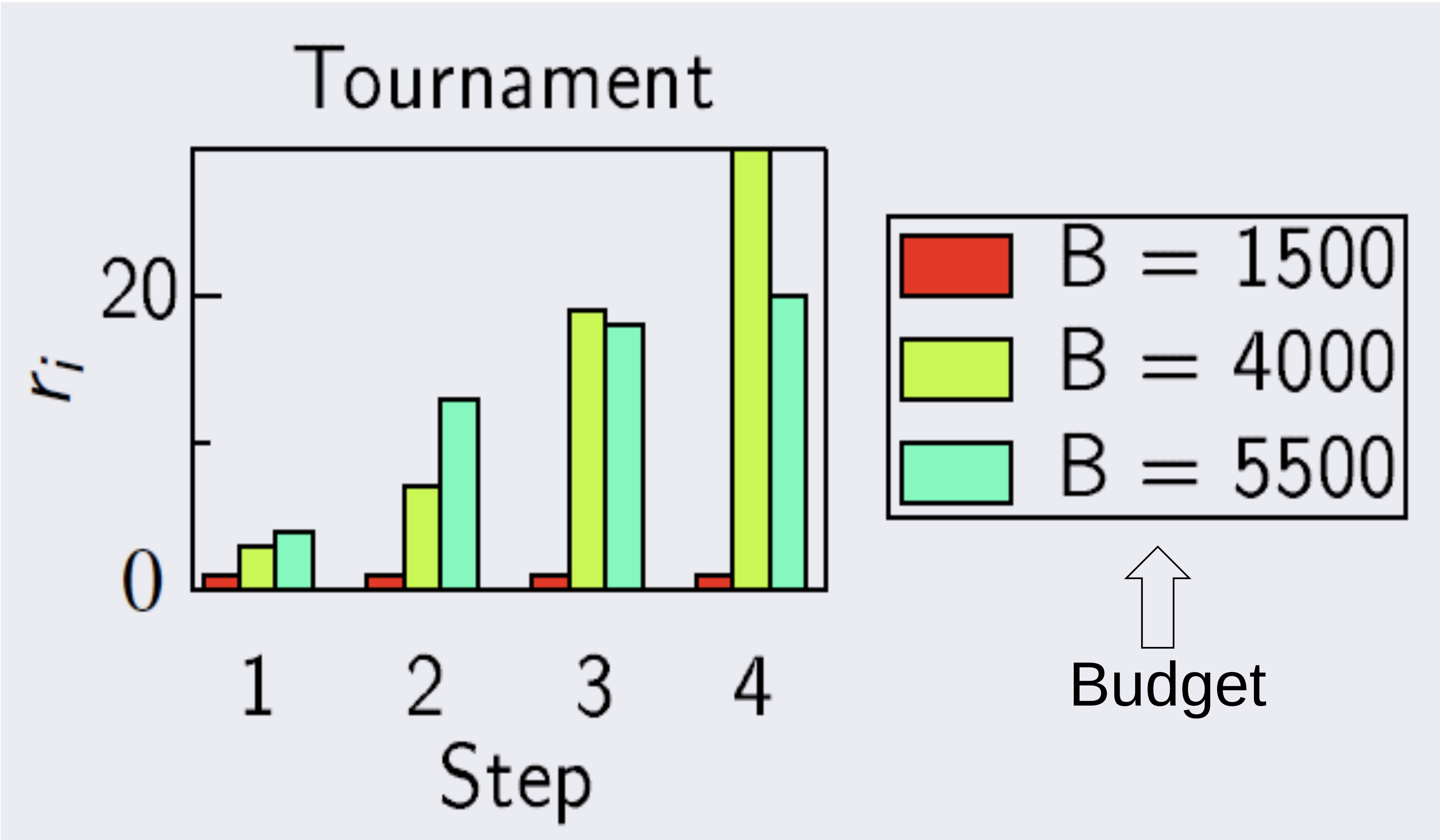
Max [Venetis-WWW12]



Max [Venetis-WWW12]



Max [Venetis-WWW12]



Top-*k* Operation

- Find top-*k* items among *N* items w.r.t. "something"

- Top-*k* list vs. top-*k* set

- Objective
  - Avoid sorting all *N* items to find top-*k*

Top-*k* Operation

- Examples
  - **[Davidson-ICDT13]** investigates the variable user error model in solving top-*k* list problem
  - **[Polychronopoulous-WebDB13]** proposes tournament-based top-*k* set solution

Top-*k* Operation

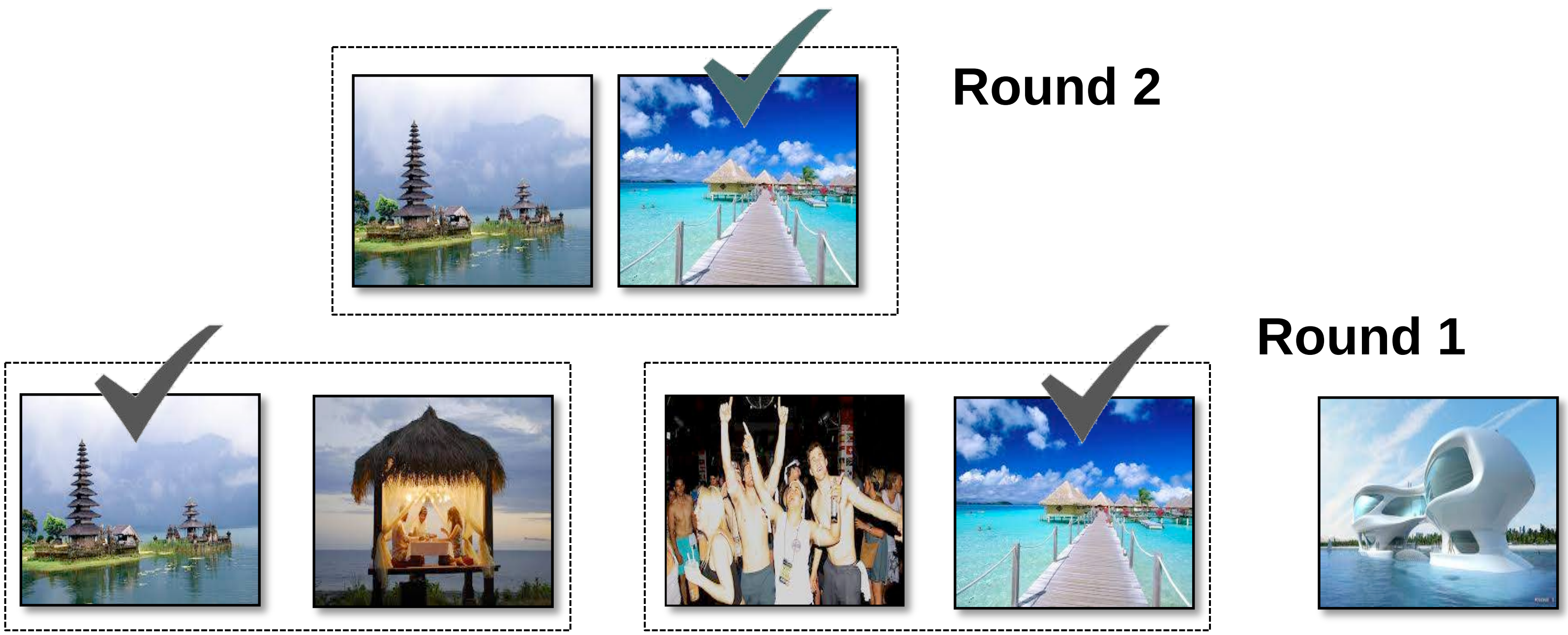
- Naïve solution is to “sort” *N* items and pick top-*k* items
- Eg, *N*=5, *k*=2, “Find two best Bali images?”
  - Ask $\binom{5}{2}$ = 10 pair-wise questions to get a total order
  - Pick top-2 images

Top-*k*: Tournament Solution (*k* = 2)

- Phase 1: **Building a tournament tree**
  - For each comparison, only winners are promoted to the next round

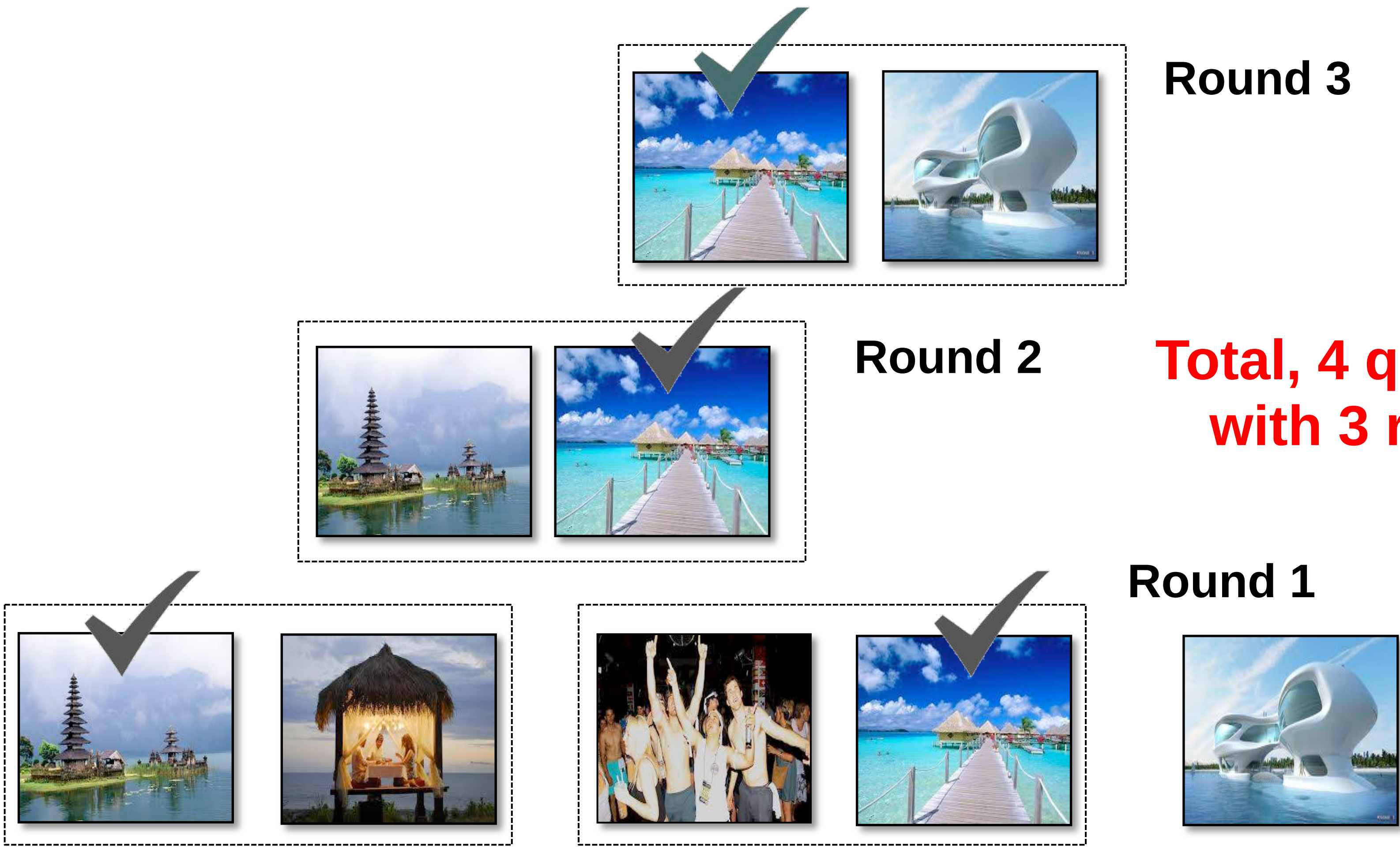
Top-*k*: Tournament Solution (*k* = 2)

- Phase 1: **Building a tournament tree**
  - For each comparison, only winners are promoted to the next round



Top-*k*: Tournament Solution (*k* = 2)

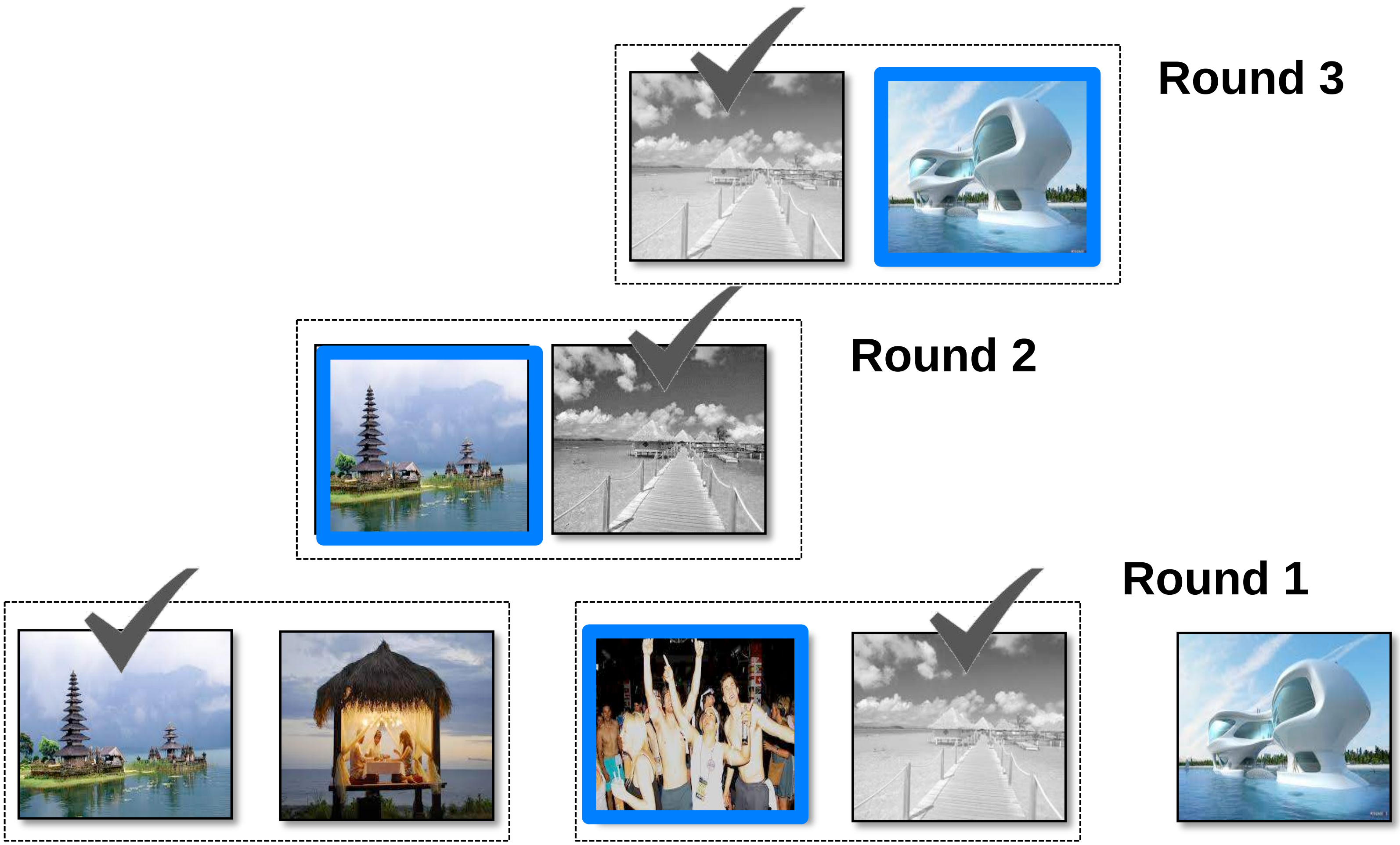
- Phase 1: **Building a tournament tree**
  - For each comparison, only winners are promoted to the next round



Total, 4 questions with 3 rounds

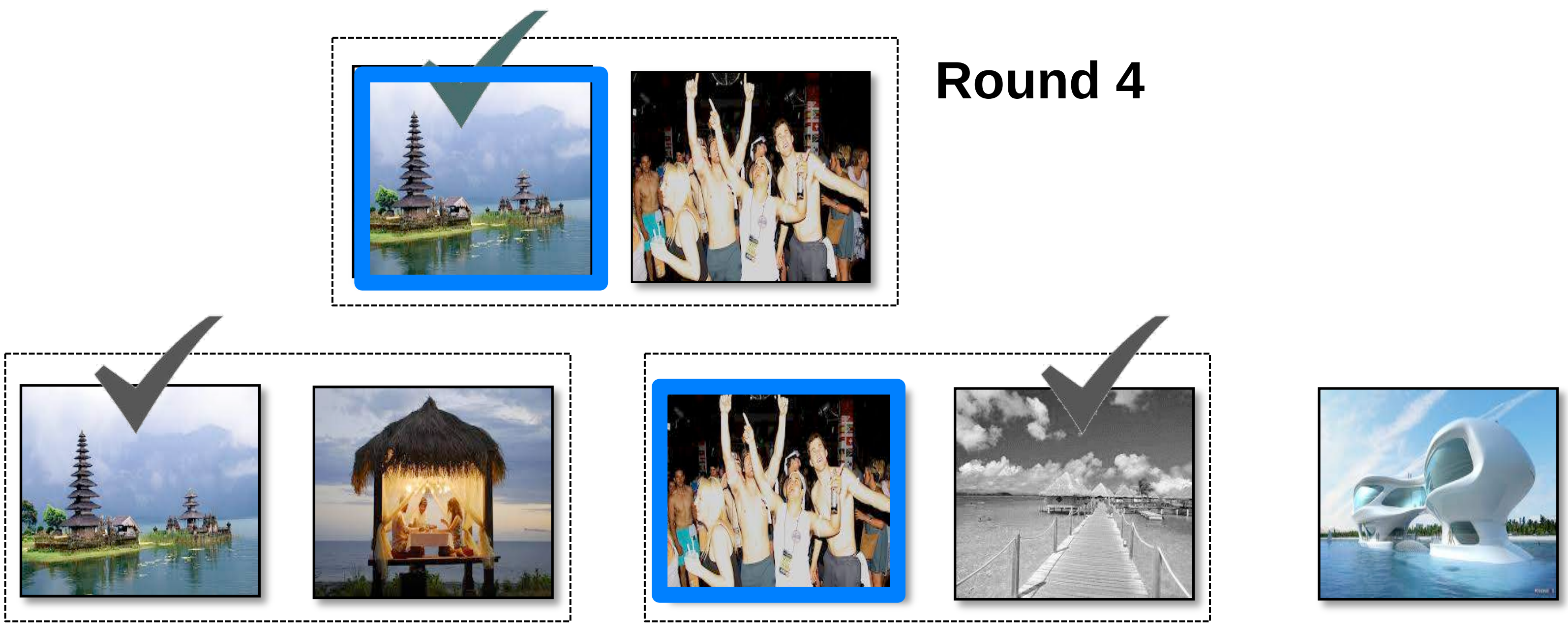
Top-*k*: Tournament Solution (*k* = 2)

- Phase 2: **Updating a tournament tree**
  - **Iteratively** asking pair-wise questions from the bottom level



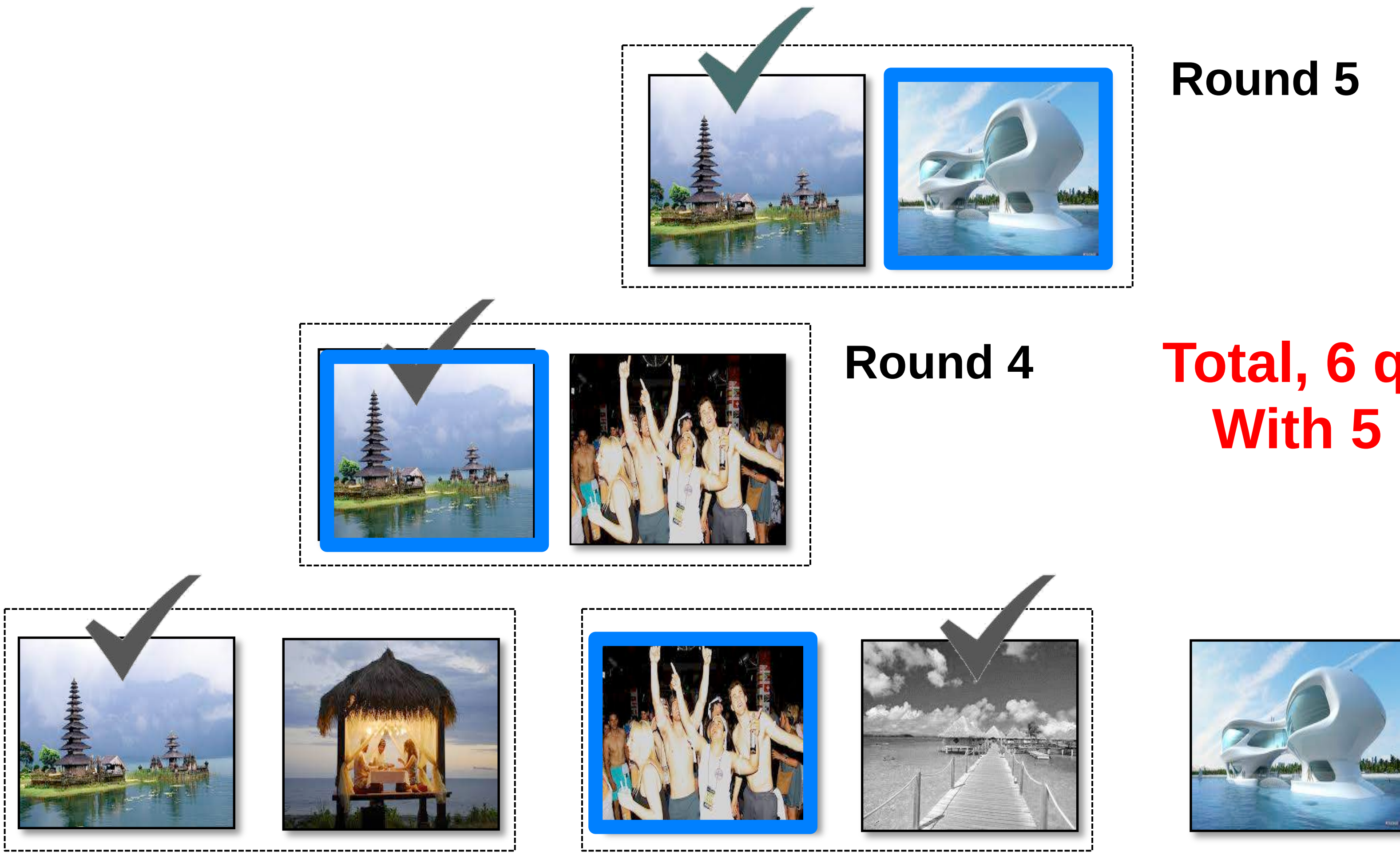
Top-*k*: Tournament Solution (*k* = 2)

- Phase 2: **Updating a tournament tree**
  - **Iteratively** asking pair-wise questions from the bottom level



Top-*k*: Tournament Solution (*k* = 2)

- Phase 2: **Updating a tournament tree**
  - **Iteratively** asking pair-wise questions from the bottom level



Total, 6 questions
With 5 rounds

Top-*k*: Tournament Solution

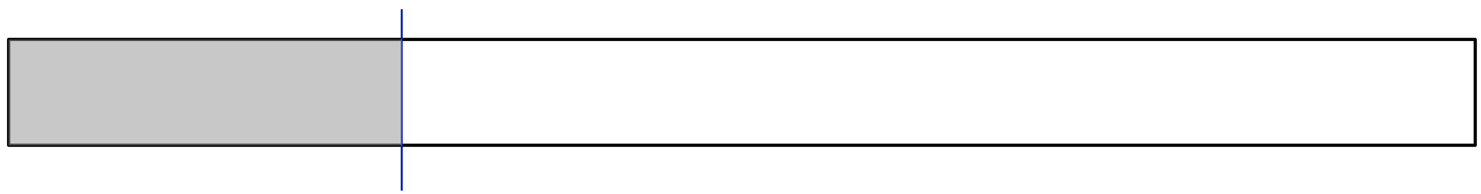
- This is a top-*k* **list** algorithm
- Analysis

	k = 1	k ≥ 2
# of questions	O(n)	$O(n + k \lceil \log_2 n \rceil)$
# of rounds	$O(\lceil \log_2 n \rceil)$	$O(k \lceil \log_2 n \rceil)$

- If there is no constraint for the number of rounds, this tournament sort yields the optimal result

Top-*k* [Polychronopoulous-WebDB13]

- Top-*k* **set** algorithm
  - Top-k items are "better" than remaining items
  - Capture NO ranking among top-*k* items



K items

  - Tournament-based approach
- Can become a Top-*k* **list** algorithm
  - Eg, Top-*k* **set** algorithm, followed by [Marcus-VLDB11] to sort *k* items

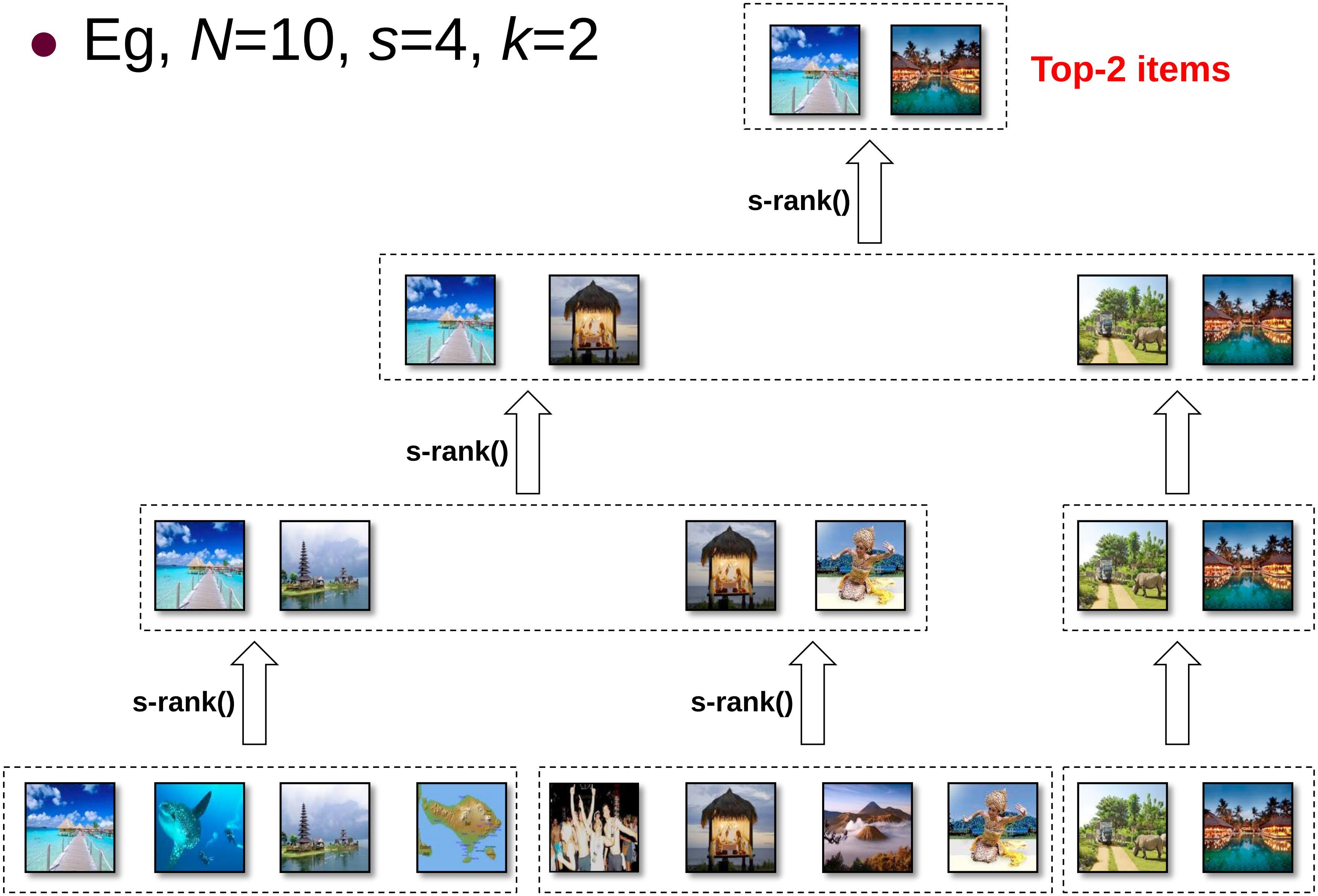
Top-*k* [Polychronopoulous-WebDB13]

- Algorithm
  - Input: *N* items, integer *k* and *s* (ie, *s* > *k*)
  - Output: top-*k* items
  - Procedure:
    - *O* ← *N* items
    - While |*O*| > *k*
      - Partition *O* into disjoint subsets of size *s*
      - Identify top-*k* items in each subset of size *s*: *s-rank(s)*
      - Merge all top-*k* items into *O*
    - Return *O*

- Effective only when *s* and *k* are small
  - Eg, *s-rank*(20) with *k*=10 won't work well

Top-*k* [Polychronopoulous-WebDB13]

- Eg, N=10, s=4, k=2



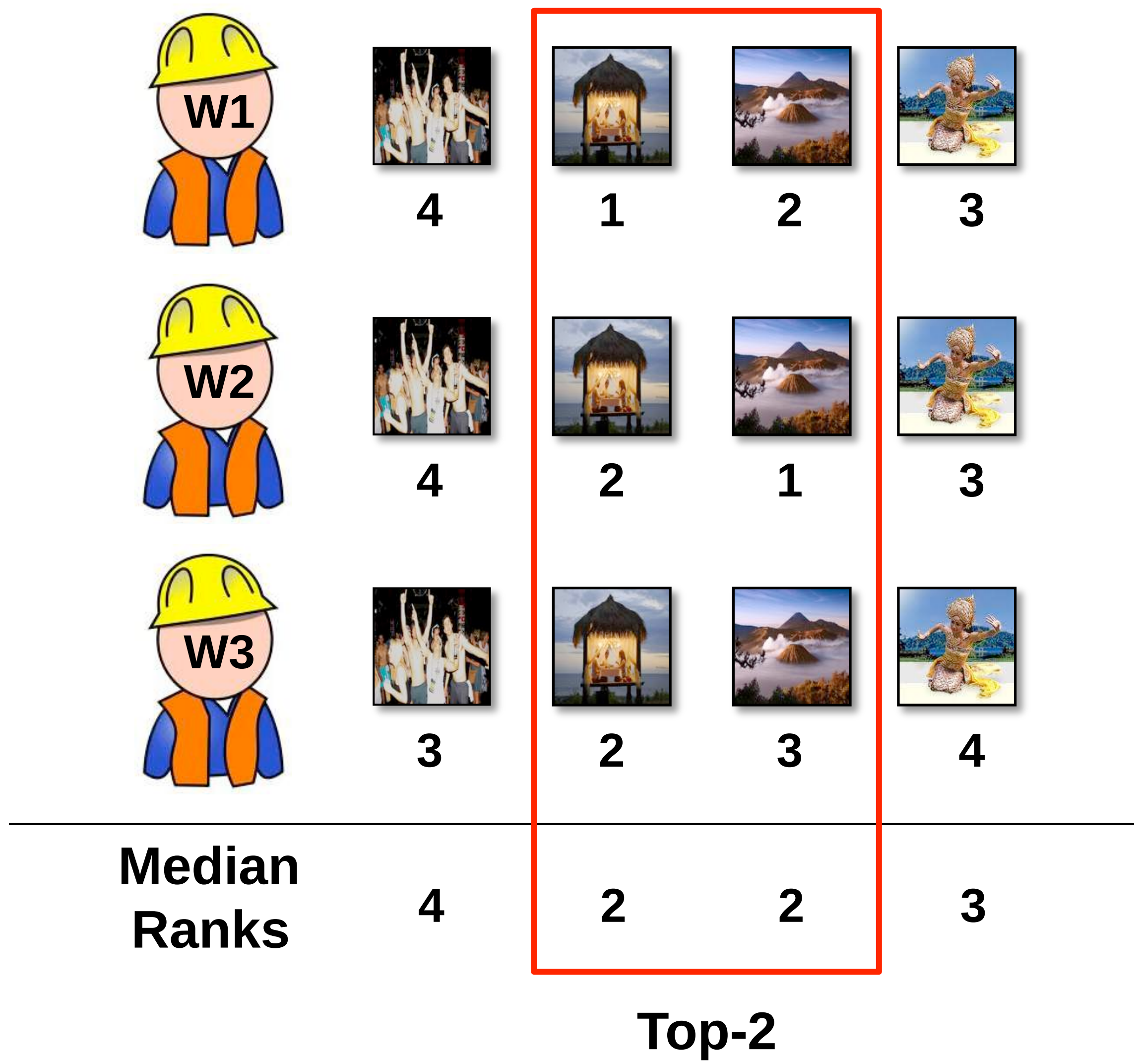
Top-*k* [Polychronopoulous-WebDB13]

- s-rank(s)

  // workers rank *s* items and aggregate
  - Input: *s* items, integer *k* (ie, *s* > *k*), *w* workers
  - Output: top-*k* items among *s* items
  - Procedure:
    - For each of *w* workers
      - Rank *s* items ≈ comparison-based sort [Marcus-VLDB11]
    - Merge *w* rankings of *s* items into a single ranking
      - Use median-rank aggregation [Dwork-WWW01]
    - Return top-*k* item from the merged ranking of *s* items

Top-*k* [Polychronopoulous-WebDB13]

- Eg, s-rank(): s=4, k=2, w=3

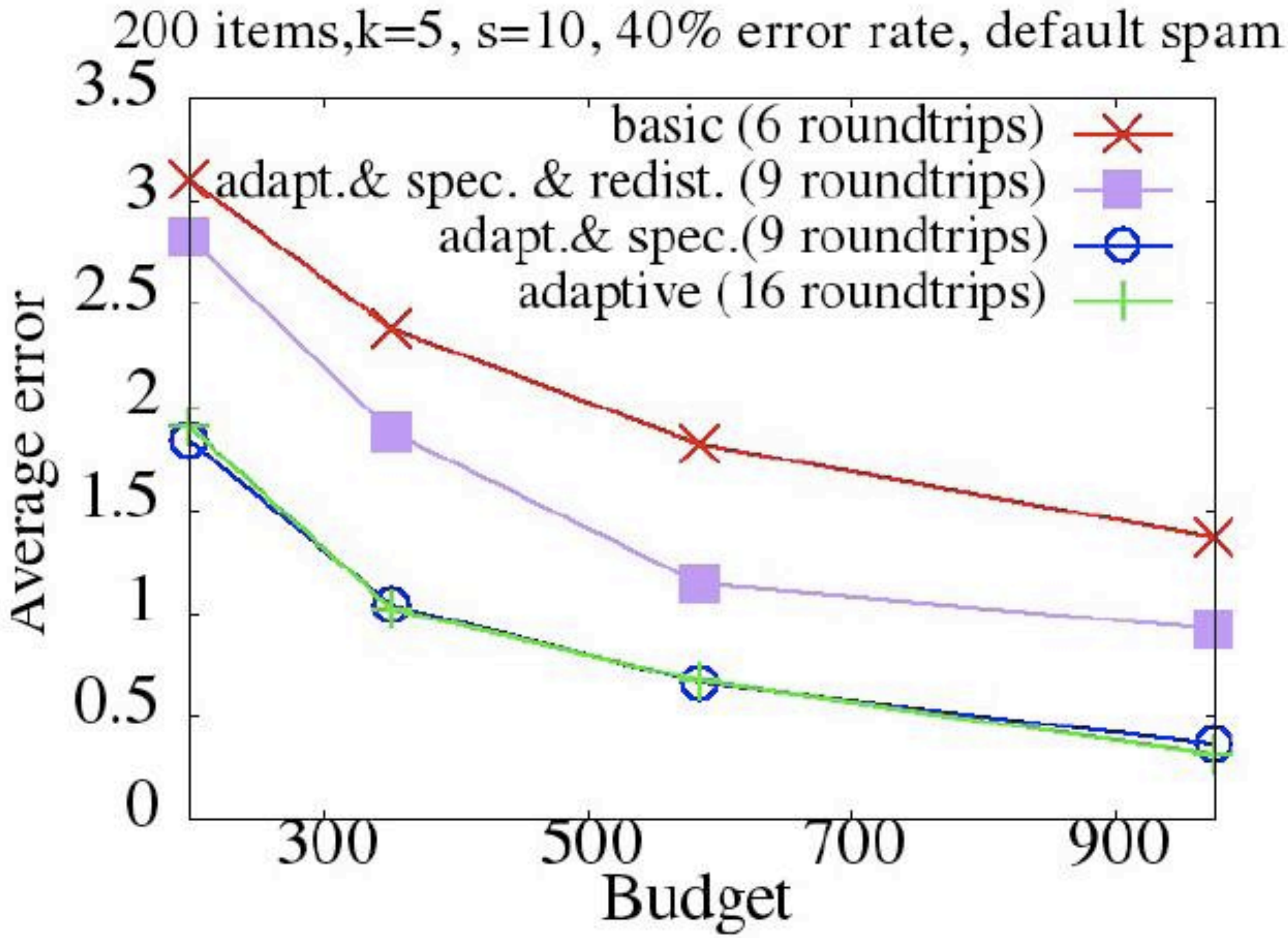


Top-*k* [Polychronopoulous-WebDB13]

- How to set # of workers, *w*, per s-rank()
  - Basic
    - same # to all s-rank()
  - Adaptive
    - 3-level assignments: low, medium, high
    - if rankings from workers disagree, allowing more workers would improve the accuracy
      - Needs more rounds for improvement

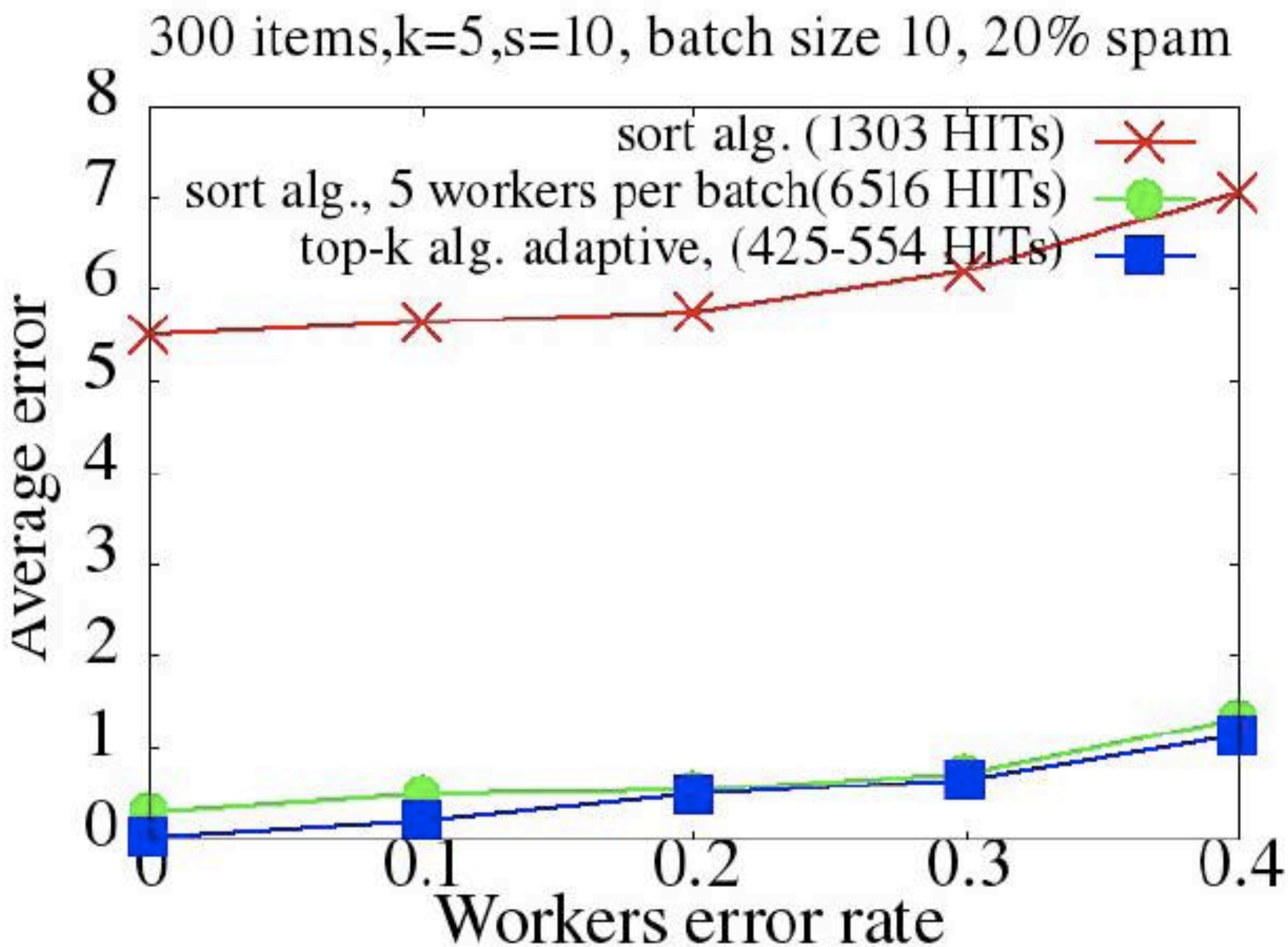
Top-*k* [Polychronopoulous-WebDB13]

- Basic vs. Adaptive



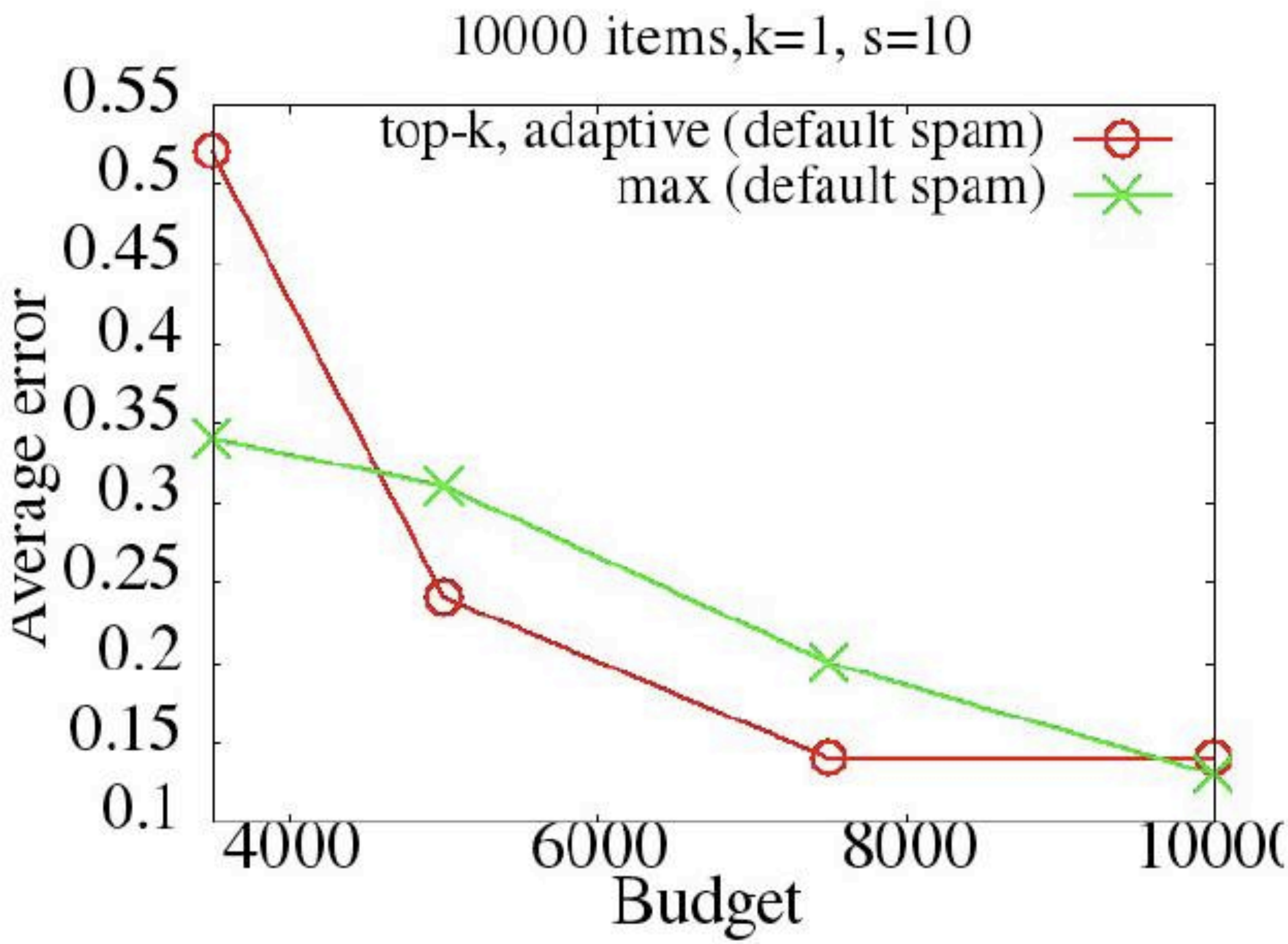
Top-*k* [Polychronopoulous-WebDB13]

- Comparison to Sort [Marcus-VLDB11]



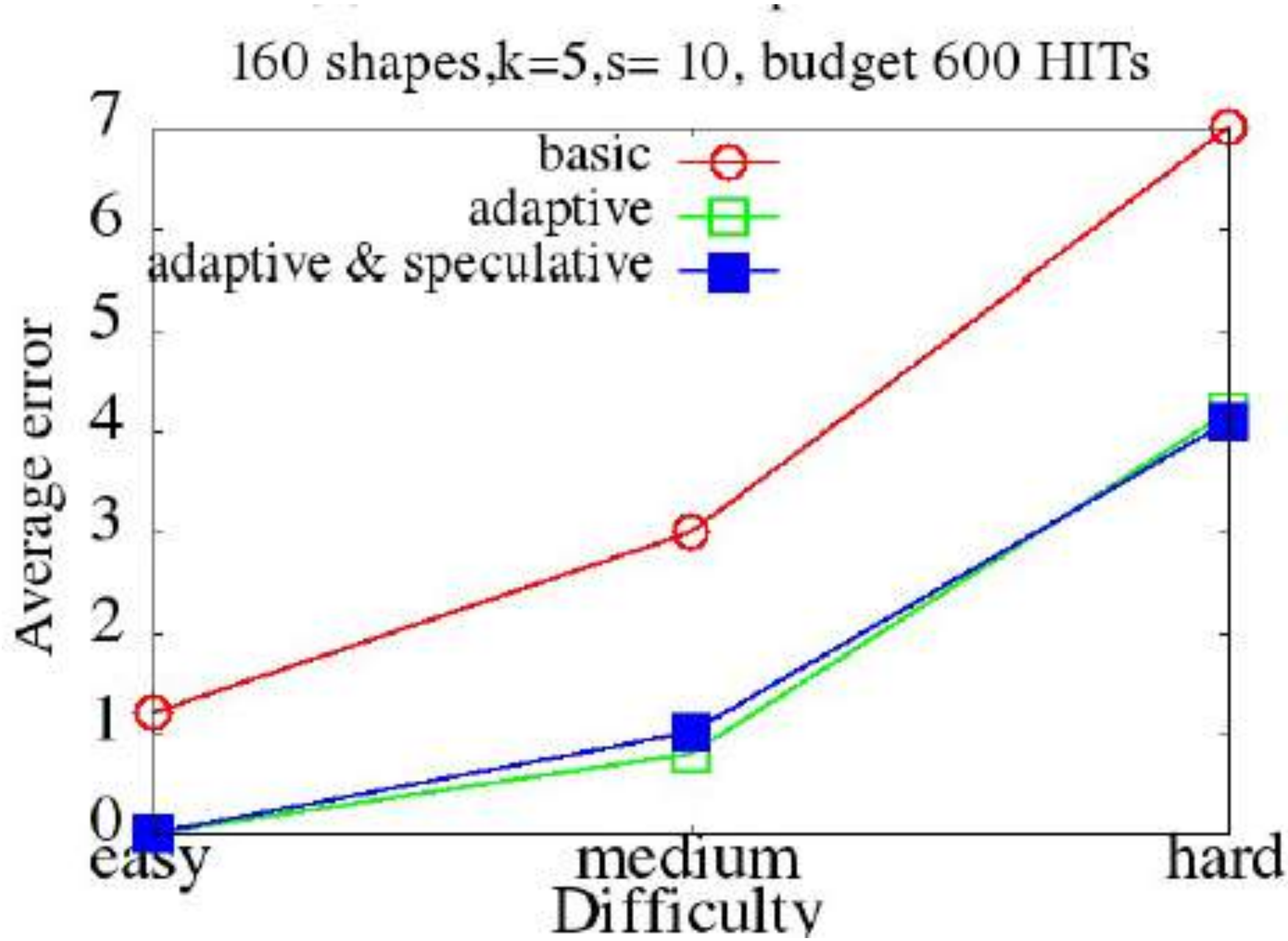
Top-*k* [Polychronopoulous-WebDB13]

- Comparison to Max [Venetis-WWW12]



Top-*k* [Polychronopoulous-WebDB13]

- AMT result



Select Operation

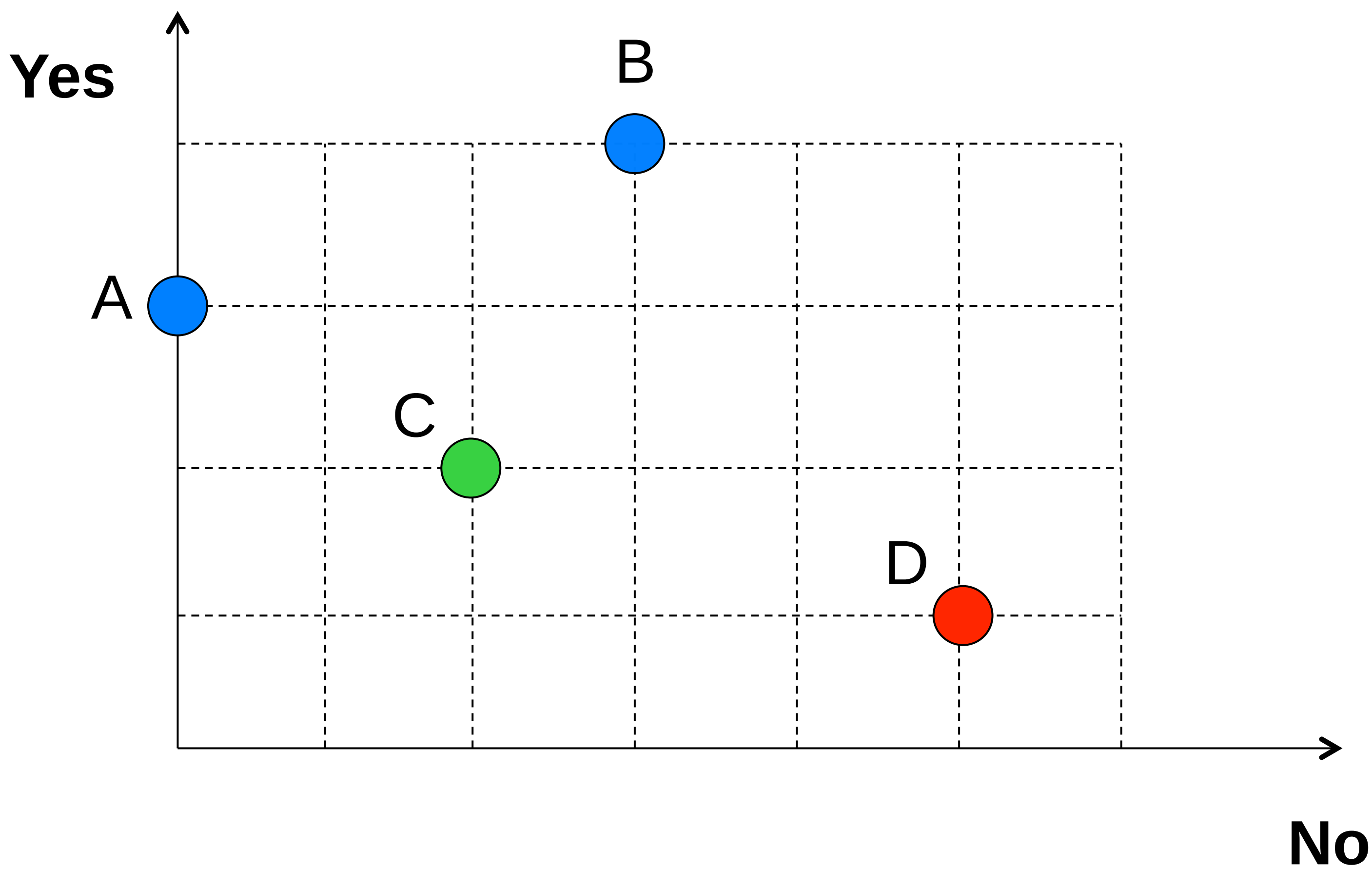
- Given *N* items, select *k* items that satisfy a predicate *P*
- ≈ Filter, Find, Screen, Search

Select Operation

- Examples
  - **[Yan-MobiSys10]** uses crowds to find an image relevant to a query
  - **[Parameswaran-SIGMOD12]** develops ilters
  - **[Franklin-ICDE13]** efficiently enumerates items satisfying conditions via crowdsourcing
  - **[Sarma-ICDE14]** finds a bounded number of items satisfying predicates using the optimal solution by the skyline of cost and time

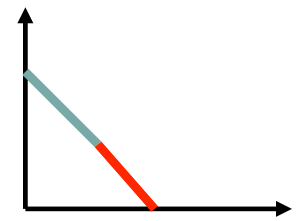
Select [Parameswaran-SIGMOD12]

- Novel grid-based visualization



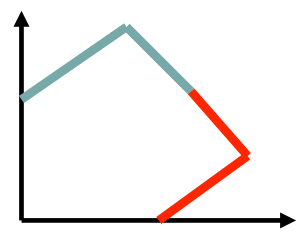
Select [Parameswaran-SIGMOD12]

- Common strategies
  - Always ask X questions, return most likely answer → Triangular strategy



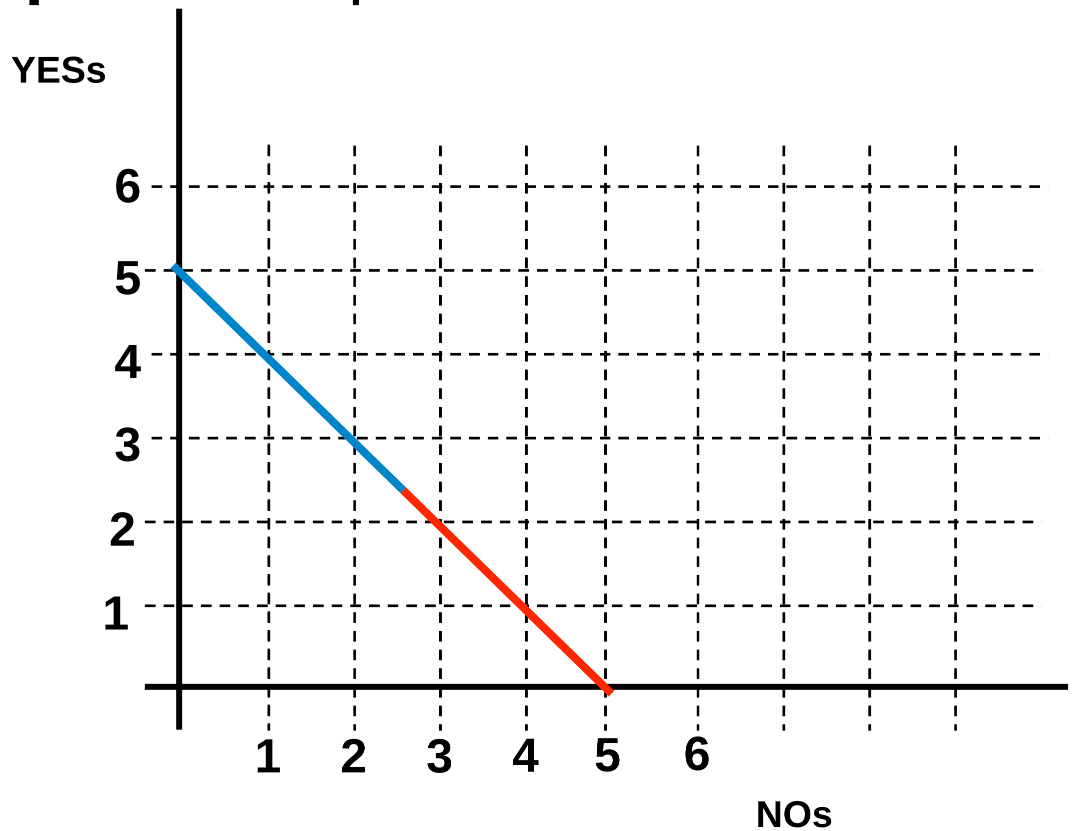
  - If X YES return "Pass", Y NO return "Fail", else keep asking → Rectangular strategy

  - Ask until |#YES - #NO| > X, or at most Y questions → Chopped off triangle



Select [Parameswaran-SIGMOD12]

- What is the best strategy? Find strategy with minimum overall expected cost
  1. **Overall expected error** is less than threshold
  2. **# of questions** per item never exceeds **m**



Count Operation

- Given *N* items, estimate a fraction of items *M* that satisfy a predicate *P*

- Selectivity estimation in DB → crowd-powered query optimizers
- Evaluating queries with GROUP BY + COUNT/AVG/SUM operators
- Eg, "Find photos of females with red hairs"
  - Selectivity("female") ≈ 50%
  - Selectivity("red hair") ≈ 2%
  - Better to process predicate("red hair") first

Count [Marcus-VLDB13]

- Hypothesis: Humans can estimate the frequency of objects' properties in a batch without having to explicitly label each item

- Two approaches
  - #1: Label Count
    - Sampling theory based
    - Have workers label samples explicitly
  - #2: Batch Count
    - Have workers estimate the frequency in a batch

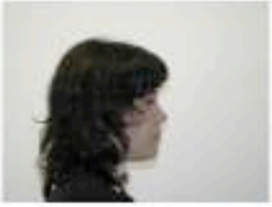
Count [Marcus-VLDB13]

- Label Count (via sampling)

There are 2 people below. Please identify the gender of each.

What is the gender of this person?

○ male ◉ female



What is the gender of this person?

○ male ◉ female

Submit

Count [Marcus-VLDB13]

- Batch Count

There are 10 people below. Please provide rough estimates for how many of the people have various properties.

About how many of the 10 people are male? 4

About how many of the 10 people are female?

Submit

Count [Marcus-VLDB13]

- Findings on accuracy
  - Images: Batch count > Label count
  - Texts: Batch count < Label count

- Further Contributions
  - Detecting spammers
  - Avoiding coordinated attacks

Join Operation

- Identify matching records or entities within or across tables
  - ≈ similarity join, entity resolution (ER), record linkage, de-duplication, …
  - Beyond the exact matching
- [Chaudhuri-ICDE06] similarity join
  - R JOIN$_p$ S, where p=$sim(R.A, S.A) > t$
  - *sim()* can be implemented as UDFs in SQL
  - Often, the evaluation is expensive
    - DB applies UDF-based join predicate after Cartesian product of R and S

Join Operation

- Examples
  - **[Marcus-VLDB11]** proposes 3 types of joins
  - **[Wang-VLDB12]** generates near-optimal cluster-based HIT design to reduce join cost
  - **[Wang-SIGMOD13]** reduces join cost further by exploiting transitivity among items
  - **[Whang-VLDB13]** selects right questions to ask to crowds to improve join accuracy
  - **[Gokhale-SIGMOD14]** proposes the hands-off crowdsourcing for join workflow

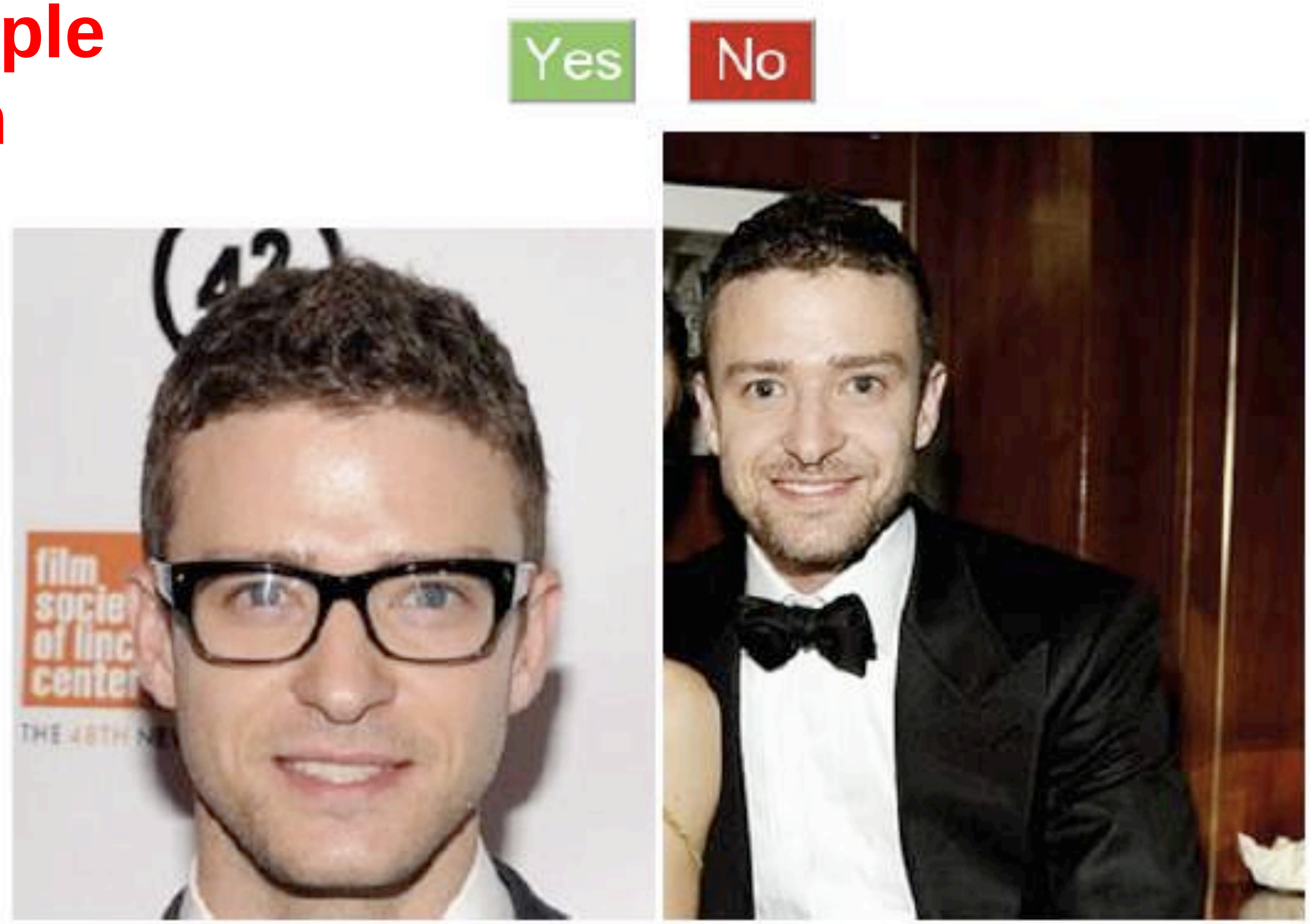
Join [Marcus-VLDB11]

- To join tables *R* and *S*
- #1: Simple Join
  - Pair-wise comparison HIT
  - $|R||S|$ HITs needed
- #2: Naïve Batching Join
  - Repetition of #1 with a batch factor *b*
  - $|R||S|/b$ HITs needed
- #3: Smart Batching Join
  - Show *r* and *s* images from *R* and *S*
  - Workers pair them up
  - $|R||S|/rs$ HITs needed

Join [Marcus-VLDB11]

Is the same celebrity in the image on the left and the image on the right?

#1 Simple Join



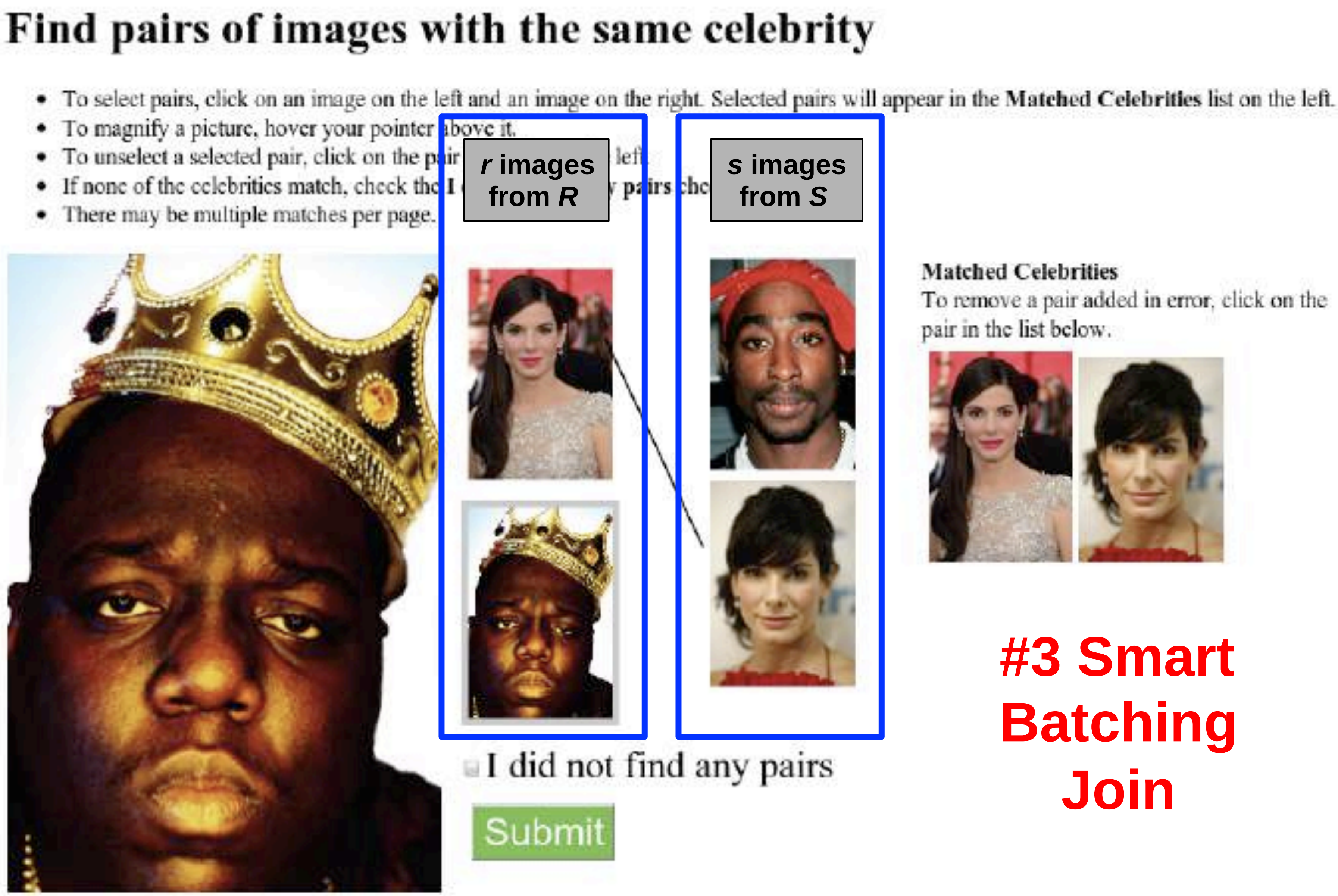
Join [Marcus-VLDB11]

Is the same celebrity in the image on the left and the image on the right?

#2 Naïve Batching Join

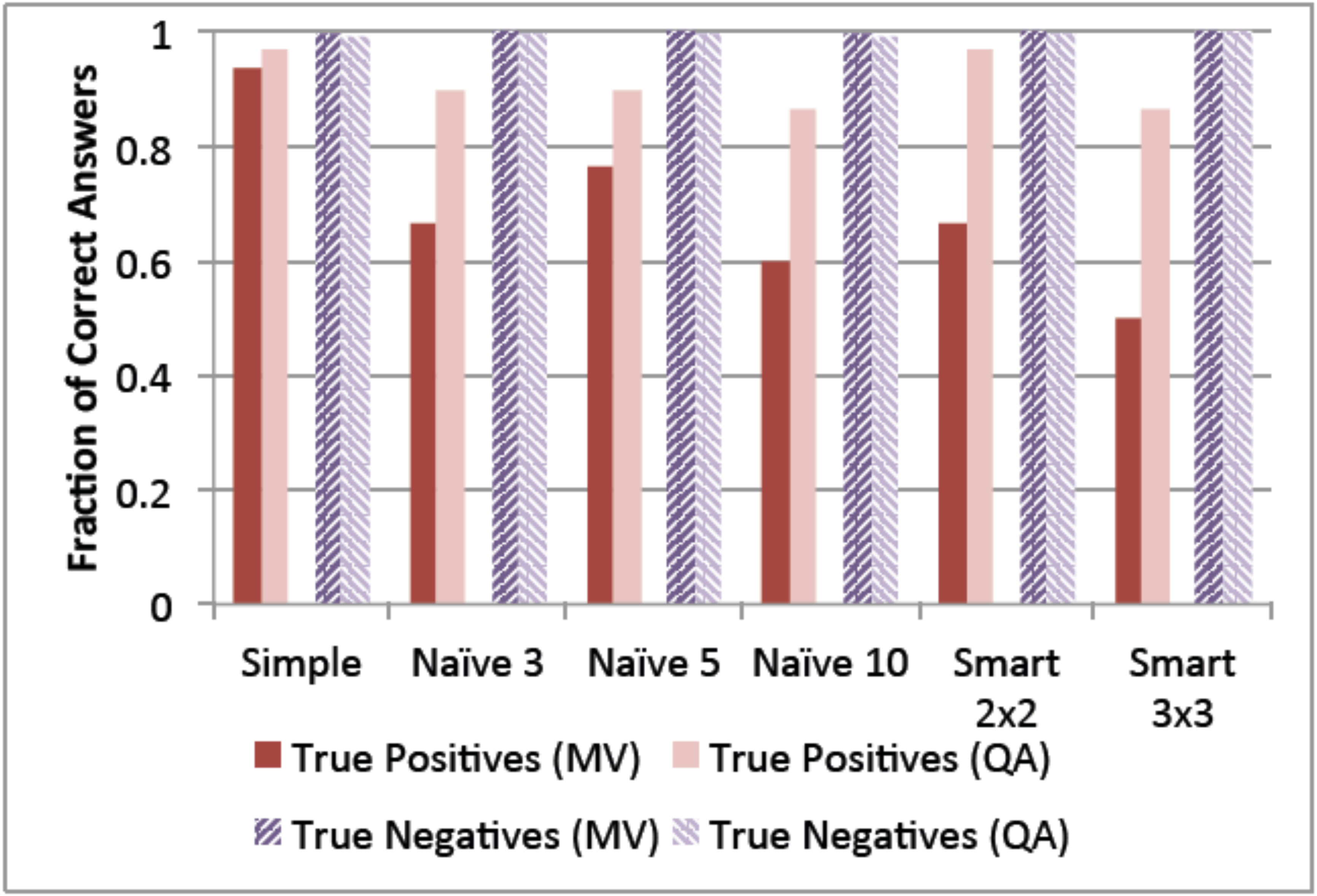
Batch factor
***b* = 2**

Join [Marcus-VLDB11]

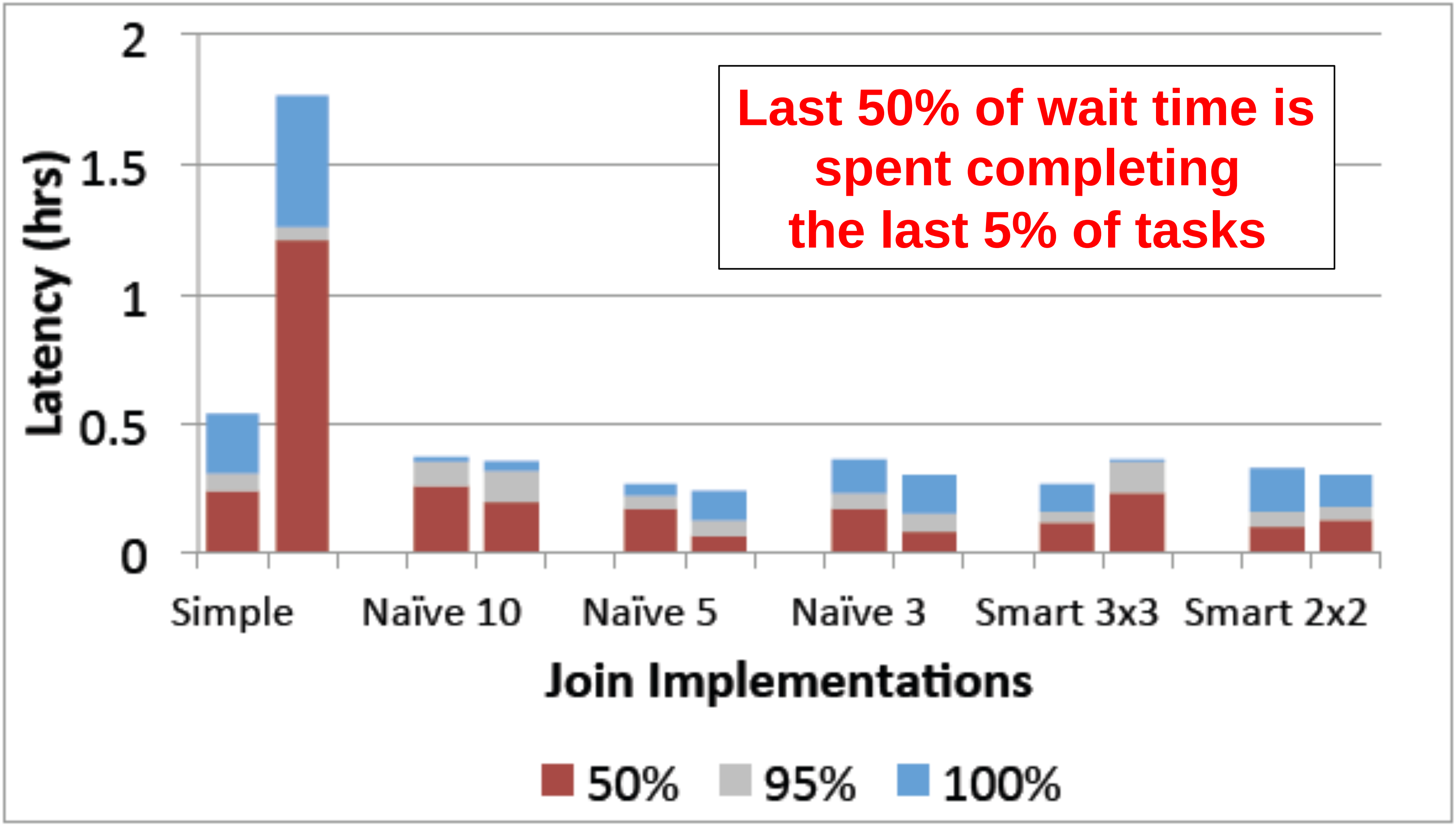


#3 Smart Batching Join

Join [Marcus-VLDB11]



Join [Marcus-VLDB11]



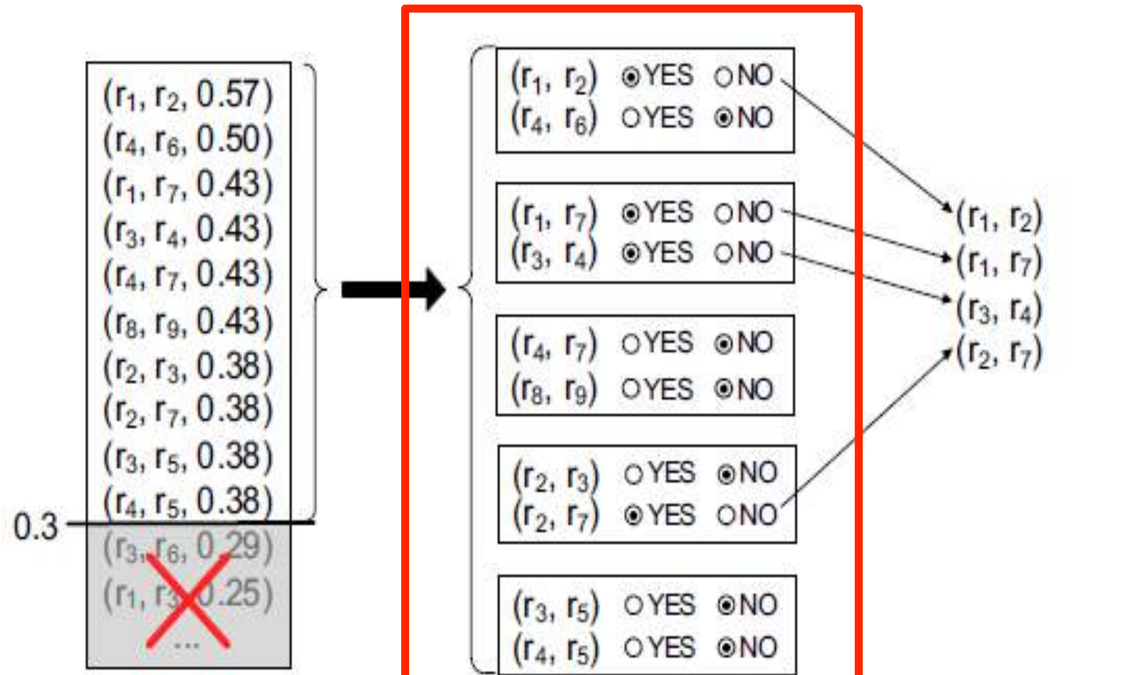
Join [Wang-VLDB12]

- [Marcus-VLDB11] proposed two batch joins
  - More efficient smart batch join still generates $|R||S|/rs$ # of HITs
  - Eg, (10,000 X 10,000) / (20 x 20) = 250,000 HITs → Still too many !
- [Wang-VLDB12] contributes CrowdER:
  1. A hybrid human-machine join
     - #1 machine-ER prunes obvious non-matches
     - #2 human-ER examines likely matching cases
       - Eg, candidate pairs with high similarity scores
  2. Algorithm to generate min # of HITs for step #2

Join [Wang-VLDB12]

- Hybrid idea: generate candidate pairs using existing similarity measures (eg, Jaccard)

ID	Product Name	Price
r_1	iPad Two 16GB WiFi White	$490
r_2	iPad 2nd generation 16GB WiFi White	$469
r_3	iPhone 4th generation White 16GB	$545
r_4	Apple iPhone 4 16GB White	$520
r_5	Apple iPhone 3rd generation Black 16GB	$375
r_6	iPhone 4 32GB White	$599
r_7	Apple iPad2 16GB WiFi White	$499
r_8	Apple iPod shuffle 2GB Blue	$49
r_9	Apple iPod shuffle USB Cable	$19



(a) Remove the pairs whose likelihood < 0.3

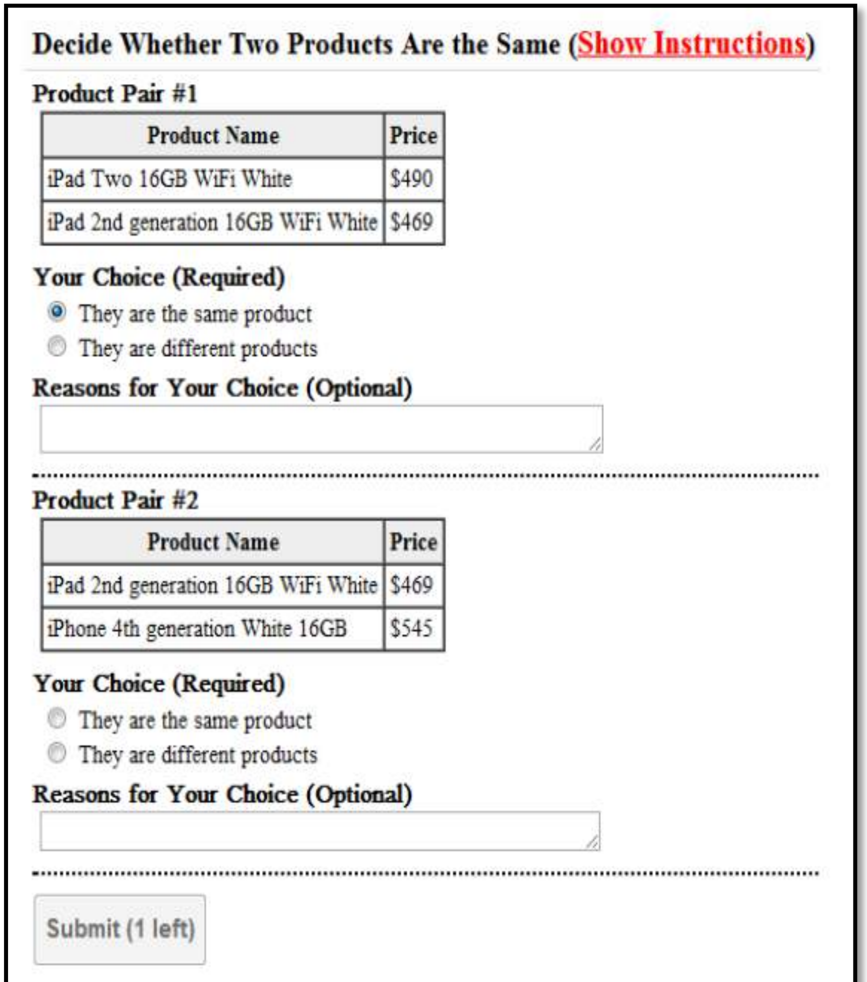
(b) Generate HITs to verify the pairs of records

(c) Output matching pairs

Main Issue: HIT Generation Problem

Join [Wang-VLDB12]

Pair-based HIT Generation = Naïve Batching in [Marcus-VLDB11]



Decide Whether Two Products Are the Same (Show Instructions)

Product Pair #1

Product Name	Price
iPad Two 16GB WiFi White	$490
iPad 2nd generation 16GB WiFi White	$469

Your Choice (Required)

- They are the same product
- They are different products

Reasons for Your Choice (Optional)

Product Pair #2

Product Name	Price
iPad 2nd generation 16GB WiFi White	$469
iPhone 4th generation White 16GB	$545

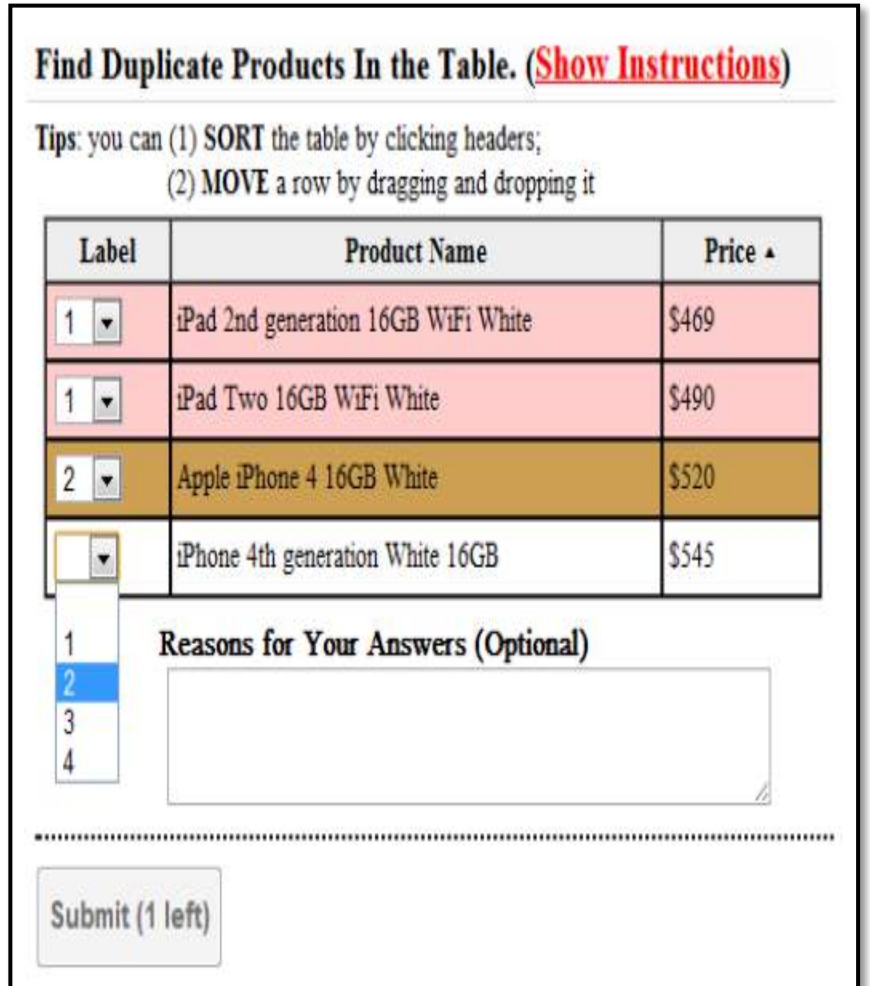
Your Choice (Required)

- They are the same product
- They are different products

Reasons for Your Choice (Optional)

Submit (1 left)

Cluster-based HIT Generation = Smart Batching in [Marcus-VLDB11]



Find Duplicate Products In the Table. (Show Instructions)

Tips: you can (1) SORT the table by clicking headers;
(2) MOVE a row by dragging and dropping it

Label	Product Name	Price ▲
1	iPad 2nd generation 16GB WiFi White	$469
1	iPad Two 16GB WiFi White	$490
2	Apple iPhone 4 16GB White	$520
	iPhone 4th generation White 16GB	$545

1
2
3
4

Reasons for Your Answers (Optional)

Submit (1 left)

Join [Wang-VLDB12]

- HIT Generation Problem
  - Input: pairs of records *P*, # of records in HIT *k*
  - Output: minimum # of HITs s.t.
    - All HITs have at most *k* records
    - Each pair $(p_i, p_j) \in P$ must be in at least one HIT

1. Pair-based HIT Generation
   - Trivial: *P/k* # of HITs s.t. each HIT contains *k* pairs in *P*
2. Cluster-based HIT Generation
   - NP-hard problem → approximation solution

Join [Wang-VLDB12]

ID	Product Name	Price
r_1	iPad Two 16GB WiFi White	\$490
r_2	iPad 2nd generation 16GB WiFi White	\$469
r_3	iPhone 4th generation White 16GB	\$545
r_4	Apple iPhone 4 16GB White	\$520
r_5	Apple iPhone 3rd generation Black 16GB	\$375
r_6	iPhone 4 32GB White	\$599
r_7	Apple iPad2 16GB WiFi White	\$499
r_8	Apple iPod shuffle 2GB Blue	\$49
r_9	Apple iPod shuffle USB Cable	\$19

$(r_1, r_2, 0.57)$
$(r_4, r_6, 0.50)$
$(r_1, r_7, 0.43)$
$(r_3, r_4, 0.43)$
$(r_4, r_7, 0.43)$
$(r_8, r_9, 0.43)$
$(r_2, r_3, 0.38)$
$(r_2, r_7, 0.38)$
$(r_3, r_5, 0.38)$
$(r_4, r_5, 0.38)$

***k* = 4**

Cluster-based HIT #1

r_1, r_2, r_3, r_7

Cluster-based HIT #2

r_3, r_4, r_5, r_6

Cluster-based HIT #3

r_4, r_7, r_8, r_9

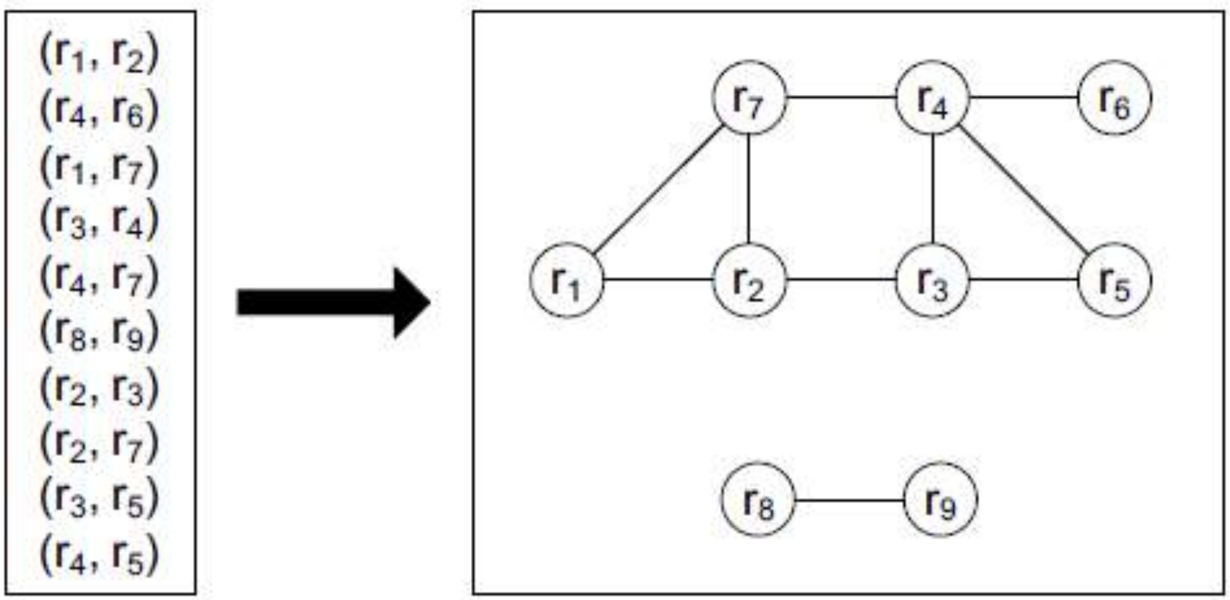
This is the minimal # of cluster-based HITs satisfying previous two conditions

Join [Wang-VLDB12]

- Two-tiered Greedy Algorithm
  - Build a graph *G* from pairs of records in *P*
  - CC ← connected components in *G*
    - LCC: large CC with more than *k* nodes
    - SCC: small CC with no more than *k* nodes
  - Step 1: Partition LCC into SCCs
  - Step 2: Pack SCCs into HITs with *k* nodes
    - Integer programming based

Join [Wang-VLDB12]

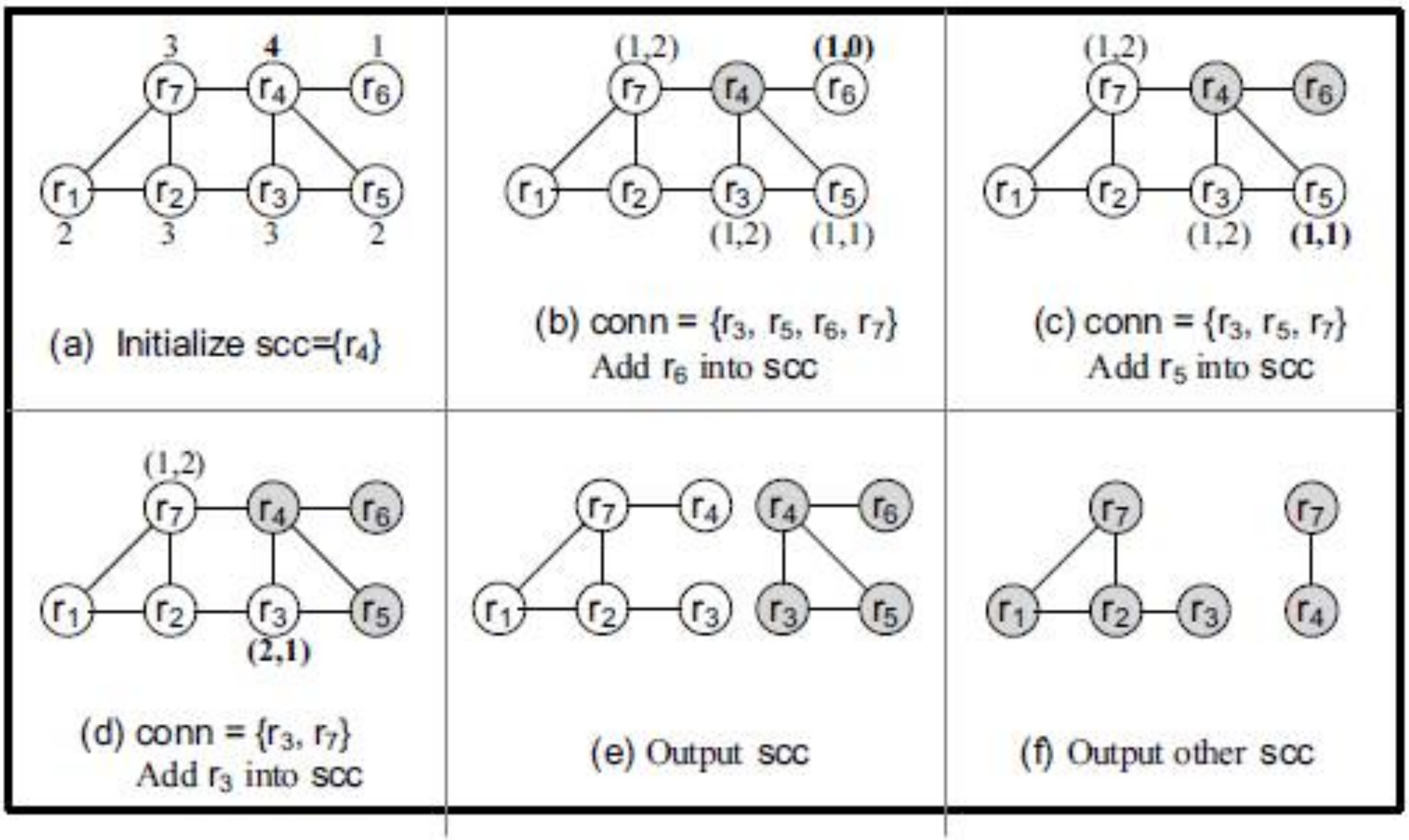
- Eg, Generate cluster-based HITs (k = 4)
  1. Partition the LCC into 3 SCCs
     - $\{r_1, r_2, r_3, r_7\}$, $\{r_3, r_4, r_5, r_6\}$, $\{r_4, r_7\}$
  2. Pack SCCs into HITs
     - A single HIT per $\{r_1, r_2, r_3, r_7\}$ and $\{r_3, r_4, r_5, r_6\}$
     - Pack $\{r_4, r_7\}$ and $\{r_8, r_9\}$ into a HIT



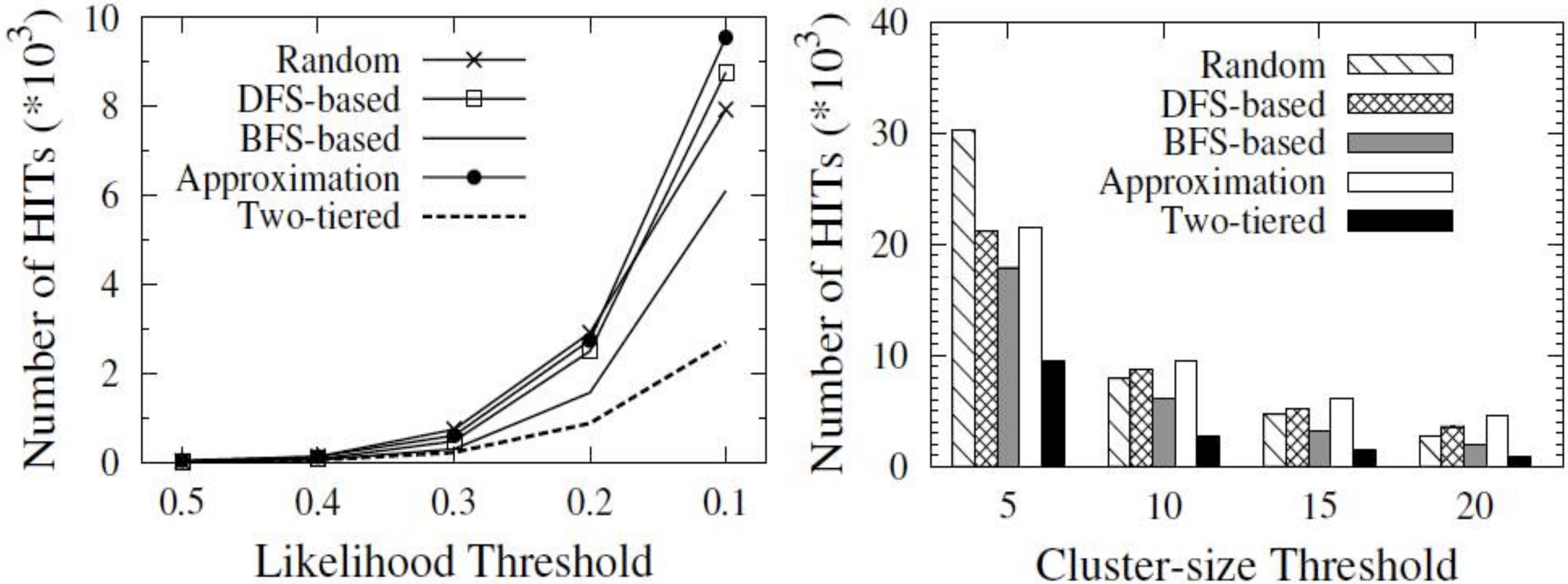
Join [Wang-VLDB12]

- Step 1: Partition
  - Input: LCC, *k*   Output: SCCs
  - r_{max} ← node in LCC with the max degree
  - scc ← $\{r_{max}\}$
  - conn ← nodes in LCC directly connected to r_{max}
  - while |scc| < k and |conn| > 0
    - r_{new} ← node in conn with max indegree (# of edges to scc) and min outdegree (# of edges to non-scc) if tie
    - move r_{new} from conn to scc
    - update conn using new scc
  - add scc into SCC

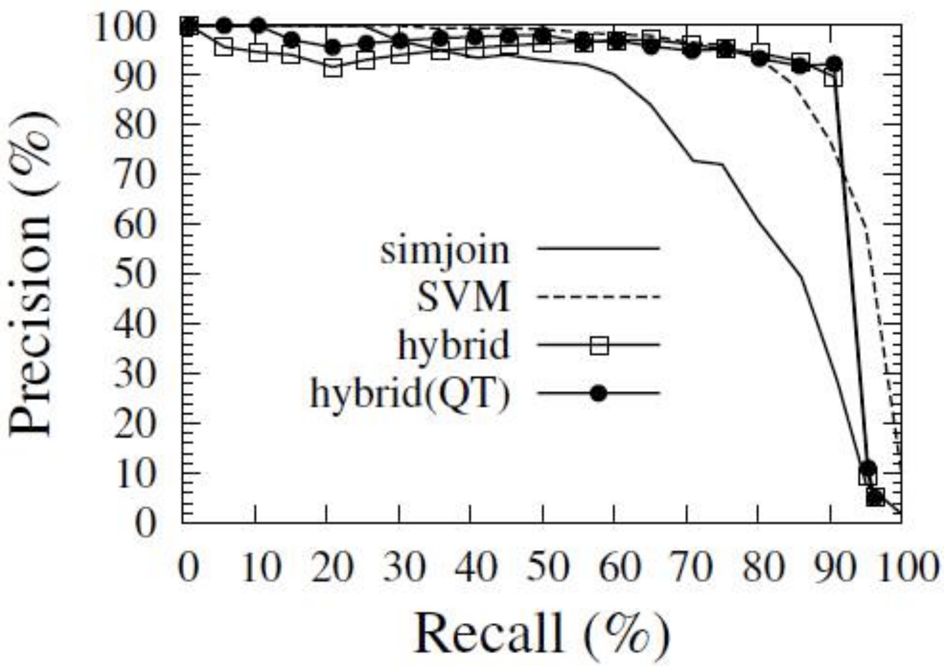
Join [Wang-VLDB12]

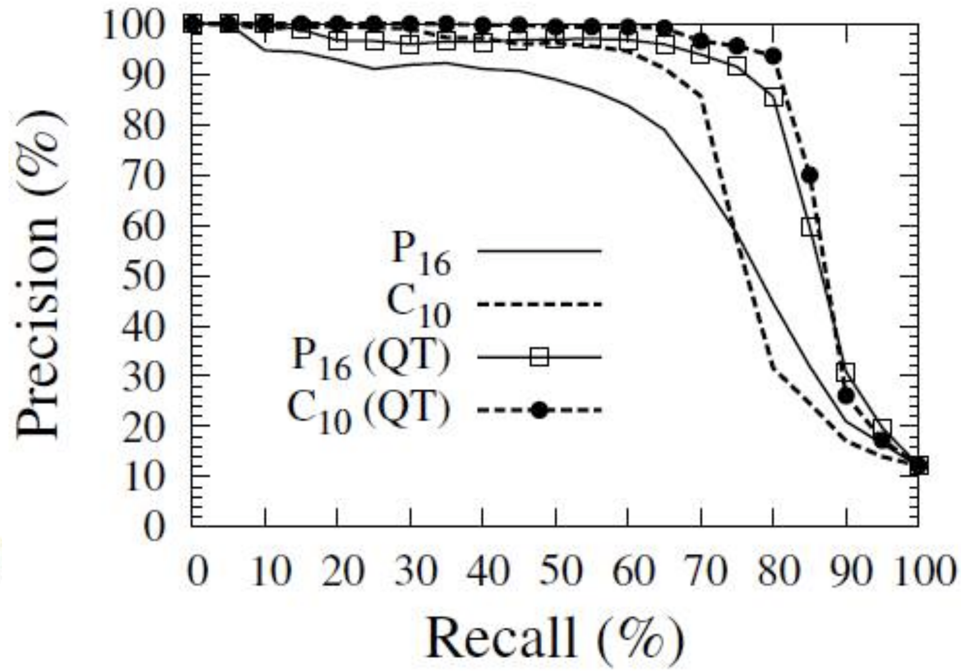


Join [Wang-VLDB12]



Join [Wang-VLDB12]





Join [Wang-SIGMOD13]

- Use the same hybrid machine-human framework as [Wang-VLDB12]
- Aim to reduce # of HITs further
- Exploit transitivity among records

Join [Wang-SIGMOD13]

- Positive transitive relation
  - If a=b, and b=c, then a=c

iPad 2nd Gen = iPad Two

iPad Two = iPad 2

➡ *iPad 2nd Gen = iPad 2*

- Negative transitive relation
  - If a = b, b ≠ c, then a ≠ c

iPad 2nd Gen = iPad Two

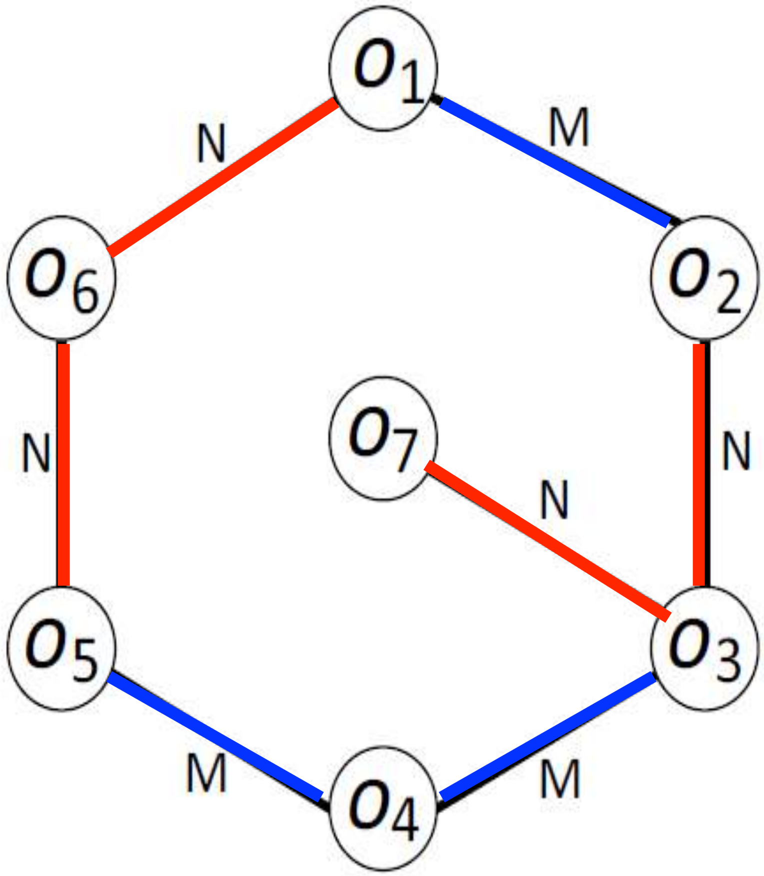
iPad Two ≠ iPad 3

➡ *iPad 2nd Gen ≠ iPad 3*

Join [Wang-SIGMOD13]

- Three transitive relations
  - If there exists a path from o to o’ which only consists of **matching pairs**, then (o, o’) can be deduced as a **matching pair**
  - If there exists a path from o to o’ which only contains **a single non-matching pair**, then (o, o’) can be deduced as a **non-matching pair**
  - If any path from o to o’ contains **more than one non-matching pairs**, (o, o’) **cannot** be deduced.

Join [Wang-SIGMOD13]



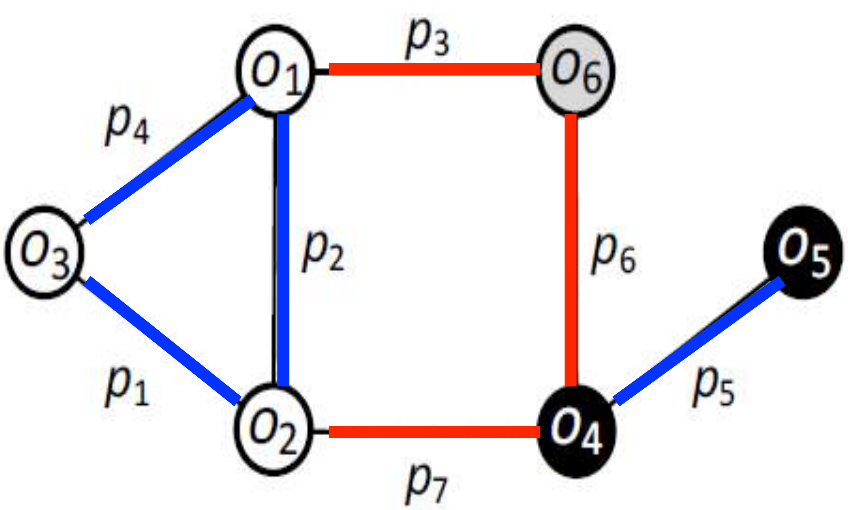
(o_3, o_5) → match

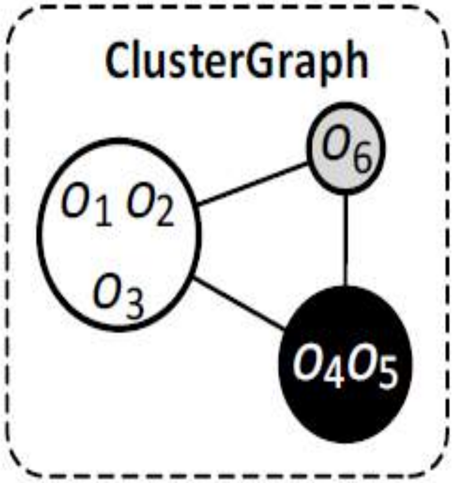
(o_5, o_7) → non-match

(o_1, o_7) → ?

Join [Wang-SIGMOD13]

- Given a pair (o_i, o_j), to check the transitivity
  - Enumerate path from o_i to o_j → exponential !
  - Count # of non-matching pairs in each path
- Solution: Build a cluster graph
  - Merge matching pairs to a cluster
  - Add inter-cluster edge for non-matching pairs





(o_5, o_6) → ?

(o_1, o_5) → ?

Join [Wang-SIGMOD13]

- Problem Definition:
  - Given a set of pairs that need to be labeled, **minimize the # of pairs** requested to crowd workers based on **transitive relations**

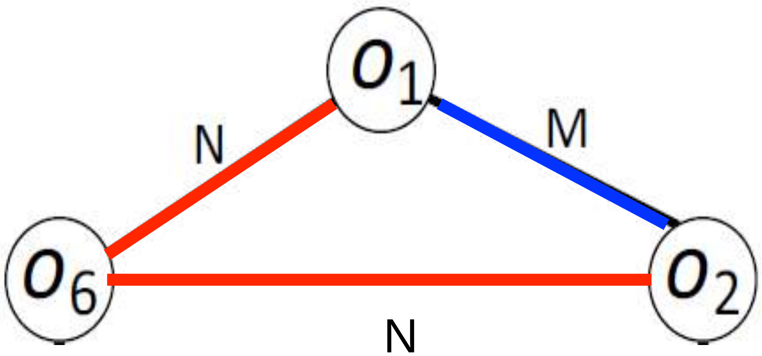
ID	Object
o_1	iPhone 2nd Gen
o_2	iPhone Two
o_3	iPhone 2
o_4	iPad Two
o_5	iPad 2
o_6	iPad 3rd Gen

ID	Object Pairs	Likelihood
p_1	(o_2, o_3)	0.85
p_2	(o_1, o_2)	0.75
p_3	(o_1, o_6)	0.72
p_4	(o_1, o_3)	0.65
p_5	(o_4, o_5)	0.55
p_6	(o_4, o_6)	0.48
p_7	(o_2, o_4)	0.45
p_8	(o_5, o_6)	0.42

?

Join [Wang-SIGMOD13]

- Labeling order matters !



$(o_1, o_2), (o_1, o_6), (o_2, o_6)$

vs.

$(o_1, o_6), (o_2, o_6), (o_1, o_2)$

➔ Given a set of pairs to label, how to order them affects the # of pairs to deduce using the transitivity

Join [Wang-SIGMOD13]

- Optimal labeling order

  $w = \langle p_1, \ldots, p_{i-1}, p_i, p_{i+1}, \ldots, p_n \rangle$

  $w' = \langle p_1, \ldots, p_{i-1}, p_{i+1}, p_i, \ldots, p_n \rangle$

  - If p_i is a matching pair and p_{i+1} is a non-matching pair, then $C(w) \le C(w')$
    - *C(w)*: # of crowdsourced pairs required for *w*
- That is, always better to first label a matching pair and then a non-matching pair

- In reality, optimal label order cannot be achieved

Join [Wang-SIGMOD13]

- Expected optimal labeling order
  - $w = <p_1, p_2, ..., p_n>$
  - $C(w)$ = # of crowdsourced pairs required for w

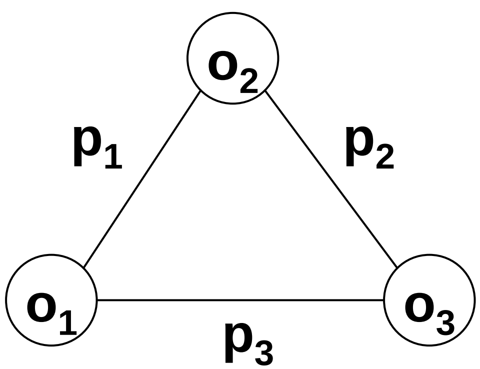
$$\mathrm{E}\left[\mathcal{C}(\omega)\right] = \sum_{i=1}^{n} \mathbb{P}(p_i = \text{crowdsourced})$$

  - $P(p_i$ = crowdsourced$)$
    - Enumerate all possible labels of $<p_1, p_2, ..., p_{i-1}>$, and for each possibility, derive whether p_i is crowdsourced or not
    - Sum of the probability of each possibility that whether p_i is crowdsourced

Join [Wang-SIGMOD13]

- Expected optimal labeling order
  - $w_1 = <p_1, p_2, p_3>$
  - $E[C(w_1)] = 1 + 1 + 0.05 = 2.05$
    - P_1: $P(P_1 = crowdsourced) = 1$
    - P_2: $P(P_2 = crowdsourced) = 1$
    - P_3: $P(P_3 = crowdsourced) = P(both\ P_1\ and\ P_2\ are\ non\text{-}matching) = (1\text{-}0.9)(1\text{-}0.5) = 0.05$

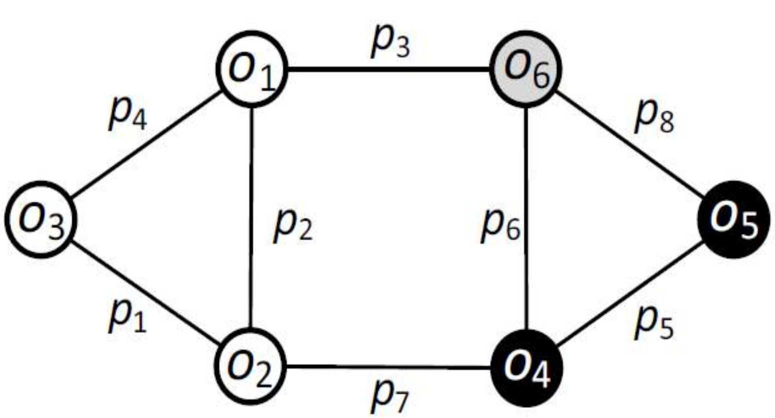
Probability of matching	
P_1	0.9
P_2	0.5
P_3	0.1



Expected value	
$w_1 = <p_1, p_2, p_3>$	**2.05**
$w_2 = <p_1, p_3, p_2>$	2.09
$w_3 = <p_2, p_3, p_1>$	2.45
$w_4 = <p_2, p_1, p_3>$	**2.05**
…	…

Join [Wang-SIGMOD13]

- Expected optimal labeling order
  - Label the pairs in **the decreasing order of the probability** that they are a matching pair
  - Eg, p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8



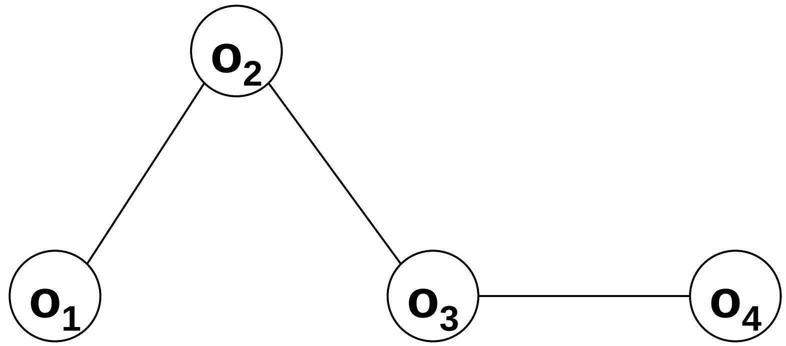
$$\mathrm{E}[C(\omega)] = \sum_{i=1}^{n} \mathbb{P}(p_i = \text{crowdsourced})$$

ID	Object Pairs	Likelihood
p_1	(o_2, o_3)	0.85
p_2	(o_1, o_2)	0.75
p_3	(o_1, o_6)	0.72
p_4	(o_1, o_3)	0.65
p_5	(o_4, o_5)	0.55
p_6	(o_4, o_6)	0.48
p_7	(o_2, o_4)	0.45
p_8	(o_5, o_6)	0.42

High

Join [Wang-SIGMOD13]

- Parallel labeling
  - To reduce the rounds → smaller latency
  - Eg, w=<(o_1, o_2), (o_2, o_3), (o_3, o_4)>
    - (o_1, o_2) has to be crowdsourced
    - (o_2, o_3) has to be crowdsourced whether the label of (o_1, o_2) is matching or not
    - (o_3, o_4) has to be crowdsourced no matter which labels (o_1, o_2) or (o_2, o_3) has
    - ➔ All three pairs can be crowdsourced concurrently



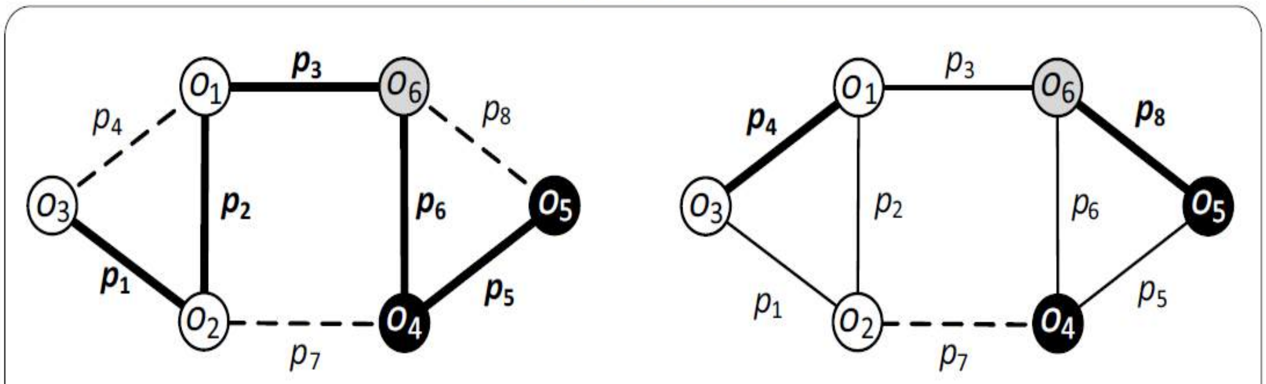
Join [Wang-SIGMOD13]

- Parallel labeling
  - $w=<p_1, p_2, \ldots, p_h, \ldots, p_{i-1}, p_i, \ldots, p_n>$
  - p_i needs to be crowdsourced iff:
    - p_i cannot be deduced from $<p_1, p_2, \ldots, p_{i-1}>$
    - $<p_1, p_2, \ldots, p_{i-1}>$ has more than one non-matching
  - If an intermediate pair p_h is unlabeled
    - Assume p_h as matching and check the transitivity
  - For each pair p_i $(1 \leq i \leq n)$
    - Output p_i as a crowdsourced pair if p_i cannot be deduced from $<p_1, p_2, \ldots, p_{i-1}>$ concurrently
    - Based on results, deduce pairs via transitivity
    - Iterate until all pairs are labeled

Join [Wang-SIGMOD13]

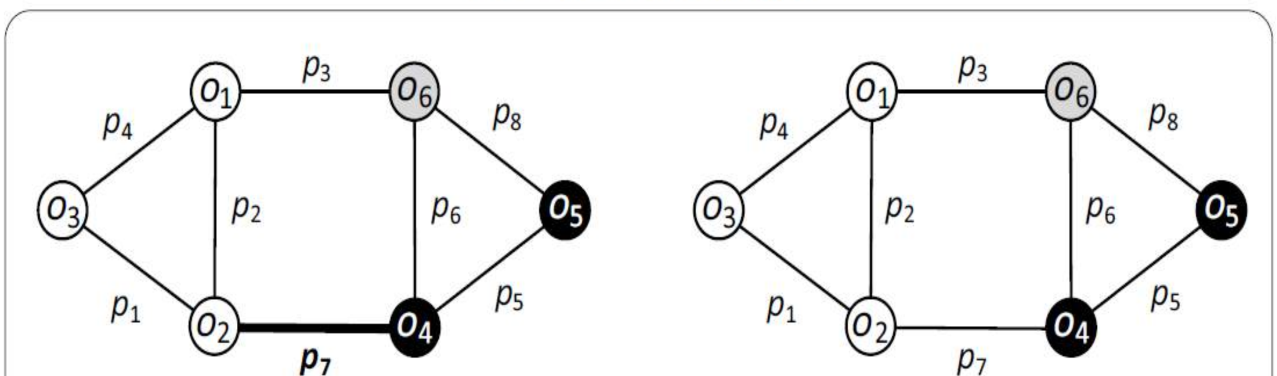
- Parallel labeling
  - Iterative algorithm

The first iteration



(a) Crowdsourcing $\{p_1, p_2, p_3, p_5, p_6\}$ (b) Deducing $\{p_4, p_8\}$

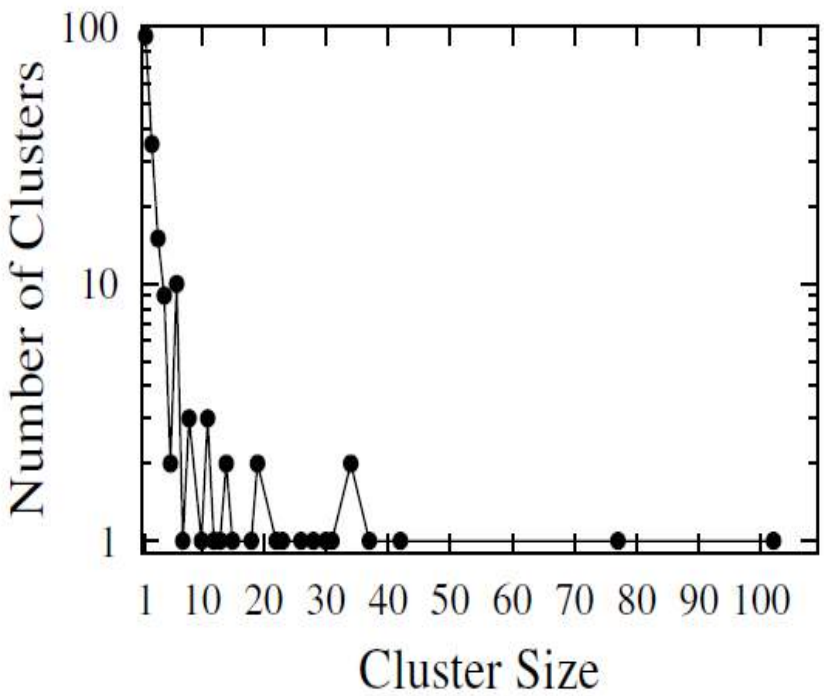
The second iteration



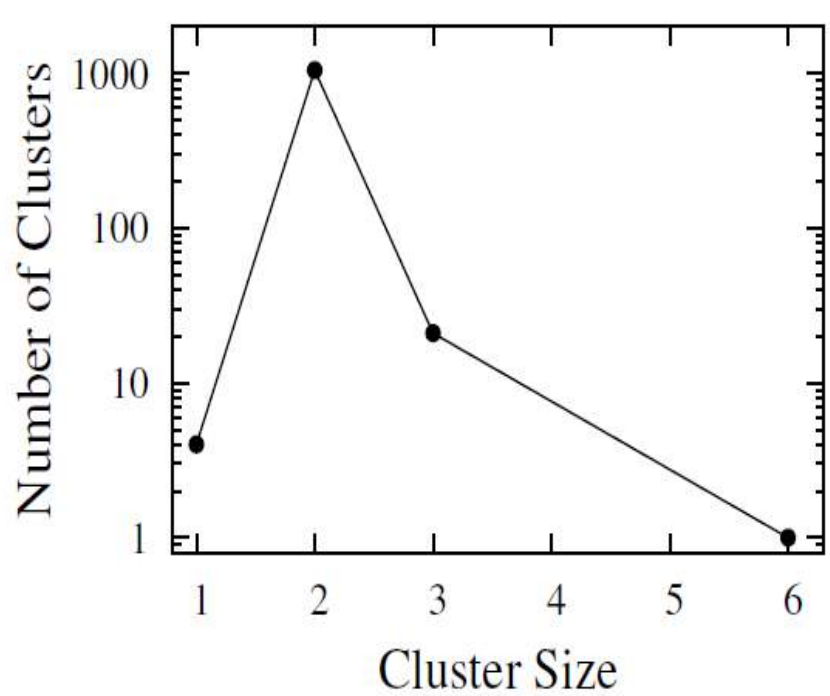
(c) Crowdsourcing $\{p_7\}$ (d) Terminate

Join [Wang-SIGMOD13]

- Two data sets
  - Paper: 997 (author, title, venue, date, and pages)
  - Product: 1081 product (abt.com), 1092 product (buy.com)



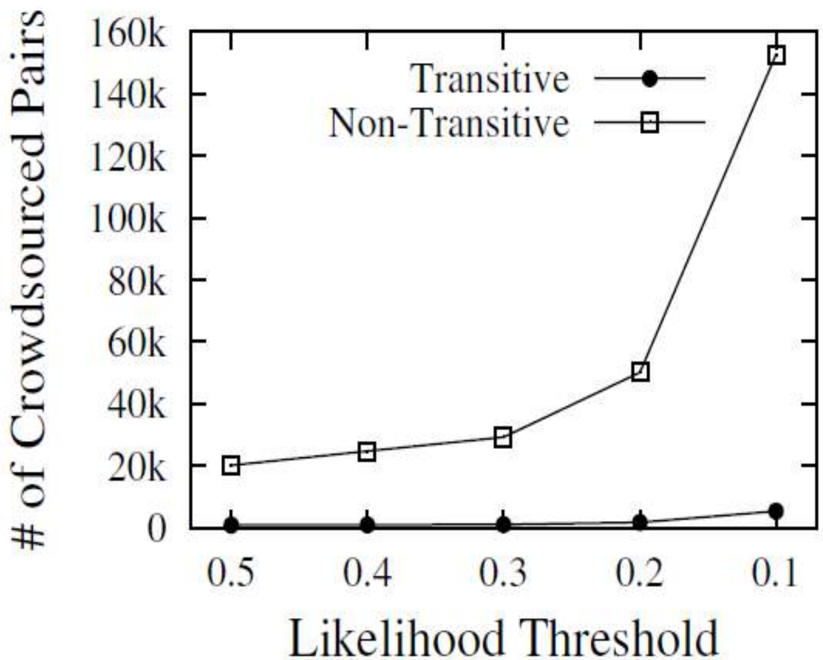
(a) Paper



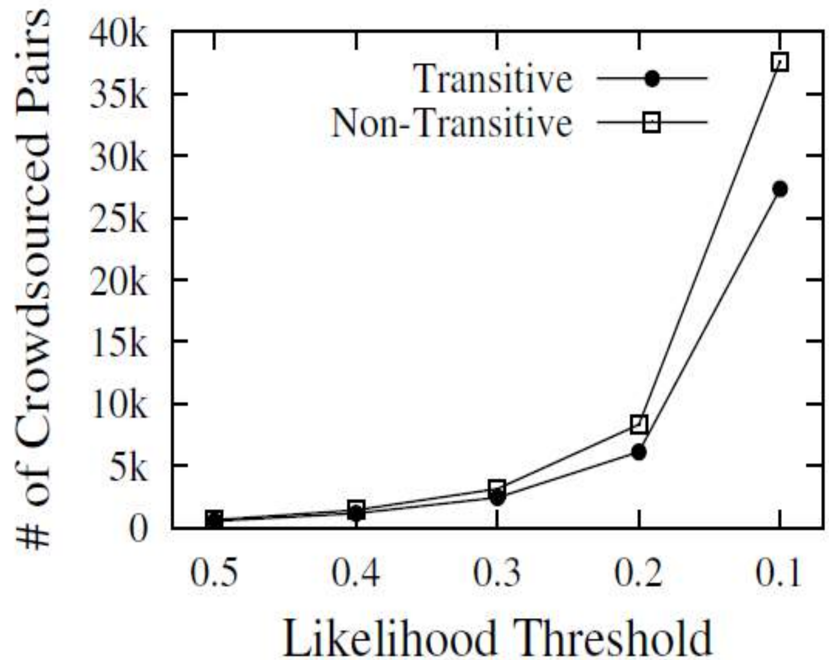
(b) Product

Join [Wang-SIGMOD13]

- Transitivity



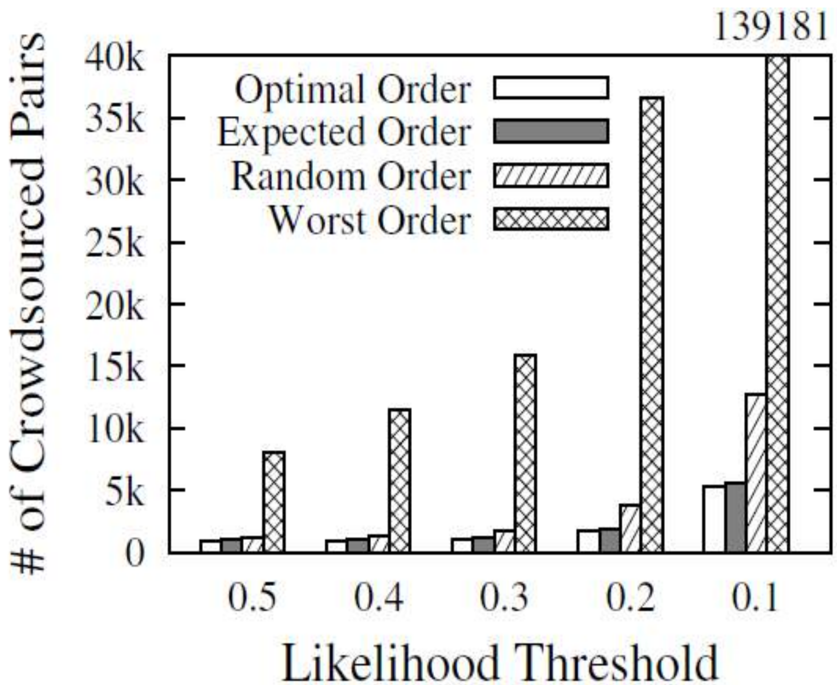
(a) Paper



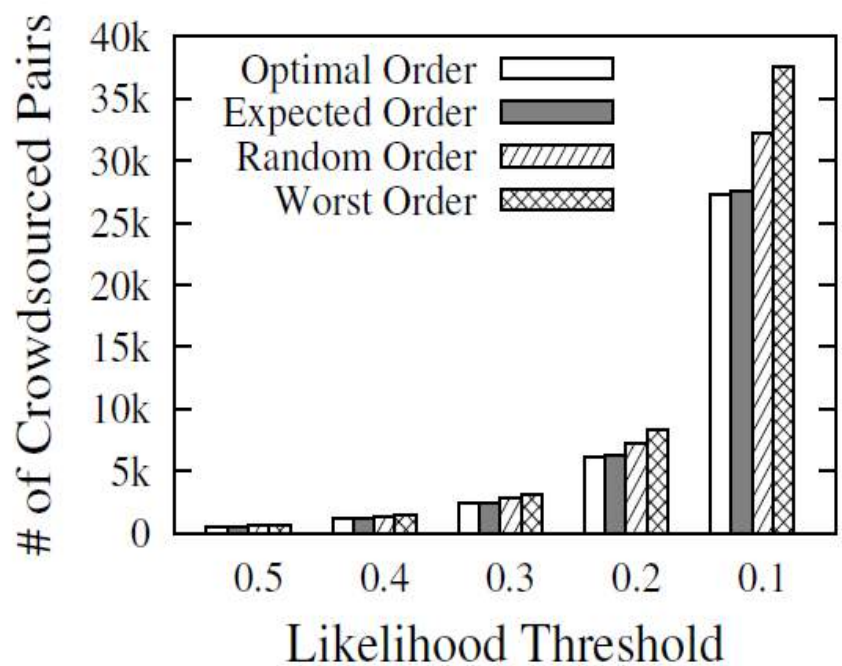
(b) Product

Join [Wang-SIGMOD13]

- Labeling order



(a) Paper



(b) Product

Conclusion

- Sampled a few representative human-powered DB operations
- Exciting field with lots of opportunities

Reference

- **[Brabham-13]** *Crowdsourcing*, Daren Brabham, 2013
- **[Cao-VLDB12]** *Whom to Ask? Jury Selection for Decision Making Tasks on Microblog Services*, Caleb Chen Cao et al., VLDB 2012
- **[Chaudhuri-ICDE06]** A Primitive Operator for Similarity Join in Data Cleaning, Surajit Chaudhuri et al., ICDE 2006
- **[Davidson-ICDT13]** *Using the crowd for top-k and group-by queries*, Susan Davidson et al., ICDT 2013
- **[Dwork-WWW01]** *Rank Aggregation Methods for the Web*, Cynthia Dwork et al., WWW 2001
- **[Franklin-SIGMOD11]** *CrowdDB: answering queries with crowdsourcing*, Michael J. Franklin et al, SIGMOD 2011
- **[Franklin-ICDE13]** *Crowdsourced enumeration queries*, Michael J. Franklin et al, ICDE 2013
- **[Gokhale-SIGMOD14]** *Corleone: Hands-Off Crowdsourcing for Entity Matching*, Chaitanya Gokhale et al., SIGMOD 2014
- **[Guo-SIGMOD12]** *So who won?: dynamic max discovery with the crowd*, Stephen Guo et al., SIGMOD 2012
- **[Howe-08]** *Crowdsourcing*, Jeff Howe, 2008

Reference

- **[LawAhn-11]** *Human Computation*, Edith Law and Luis von Ahn, 2011
- **[Li-HotDB12]** *Crowdsourcing: Challenges and Opportunities*, Guoliang Li, HotDB 2012
- **[Liu-VLDB12]** *CDAS: A Crowdsourcing Data Analytics System***,** Xuan Liu et al., VLDB 2012
- **[Marcus-VLDB11]** *Human-powered Sorts and Joins*, Adam Marcus et al., VLDB 2011
- **[Marcus-VLDB12]** *Counting with the Crowd*, Adam Marcus et al., VLDB 2012
- **[Miller-13]** *Crowd Computing and Human Computation Algorithms*, Rob Miller, 2013
- **[Parameswaran-SIGMOD12]** *CrowdScreen: Algorithms for Filtering Data with Humans*, Aditya Parameswaran et al., SIGMOD 2012
- **[Polychronopoulous-WebDB13]** *Human-Powered Top-k Lists*, Vassilis Polychronopoulous et al., WebDB 2013
- **[Sarma-ICDE14]** Crowd-Powered Find Algorithms, Anish Das Sarma et al., ICDE 2014
- **[Shirky-08]** *Here Comes Everybody*, Clay Shirky, 2008

Reference

- **[Surowiecki-04]** *The Wisdom of Crowds*, James Surowiecki, 2004
- **[Venetis-WWW12]** *Max Algorithms in Crowdsourcing Environments*, Petros Venetis et al., WWW 2012
- **[Wang-VLDB12]** *CrowdER: Crowdsourcing Entity Resolution*, Jiannan Wang et al., VLDB 2012
- **[Wang-SIGMOD13]** *Leveraging Transitive Relations for Crowdsourced Joins*, Jiannan Wang et al., SIGMOD 2013
- **[Whang-VLDB13]** *Question Selection for Crowd Entity Resolution*, Steven Whang et al., VLDB 2013
- **[Yan-MobiSys10]** *CrowdSearch: exploiting crowds for accurate real-time image search on mobile phones*, T. Yan et al., MobiSys 2010