CrowdER: Crowdsourcing Entity Resolution

A HYBRID HUMAN-MACHINE SYSTEM FOR ENTITY RESOLUTION

Entity Resolution

- The task of finding different records that refer to the same entity

ID	Product Name
r1	iPad Two 16GB WiFi White
r2	iPad 2nd generation 16GB WiFi White

Machine-based techniques

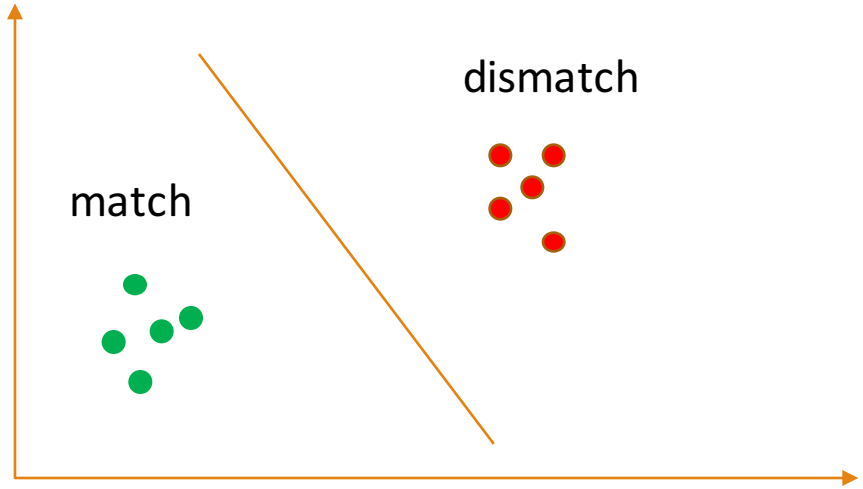
- Similarity-based
  - Idea: records look similar are more likely to refer to the same entity.
  - The similarity function takes a pair of records as input, and outputs a likelihood.

$$\mathrm{J}(r_1, r_2) = \frac{|\{\texttt{iPad}, \texttt{16GB}, \texttt{WiFi}, \texttt{White}\}|}{|\{\texttt{iPad}, \texttt{16GB}, \texttt{WiFi}, \texttt{White}, \texttt{Two}, \texttt{2nd}, \texttt{generation}\}|} = 0.57$$

ID	Product Name
r1	iPad Two 16GB WiFi White
r2	iPad 2nd generation 16GB WiFi White

Machine-based techniques

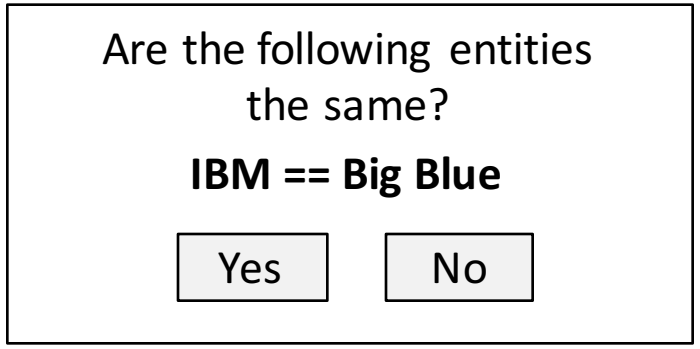
- Learning-based
  - Transfer the entity resolution problem into classification problem.



Human based techniques

- **CrowdDB**

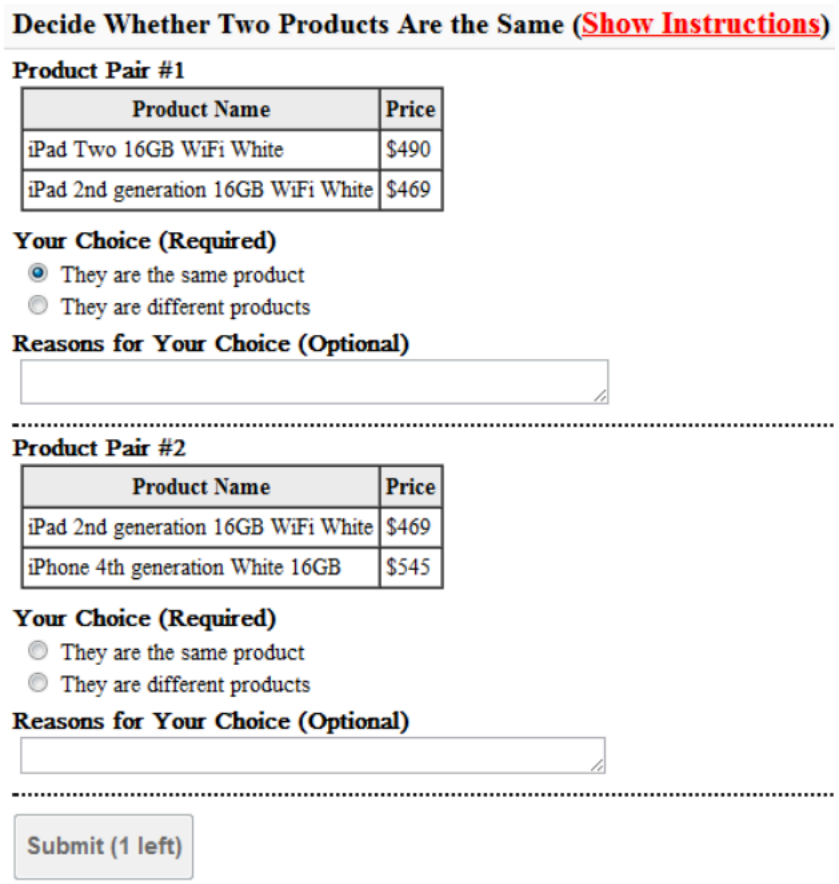
SELECT p.id, q.id FROM product p, product q
WHERE p.product_name ~= q.product_name;



- **For n records** $O(n^2)$

HIT Generation

- Pair-based (Naïve Batching)
- $O(n^2/k)$



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)

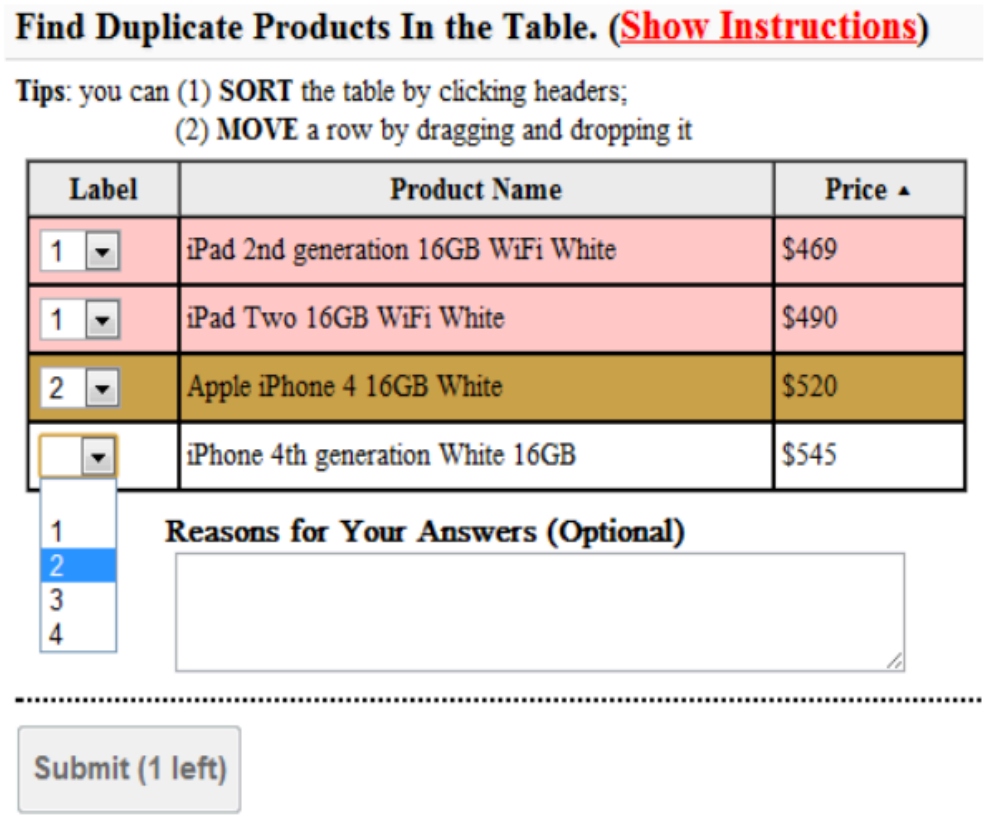
- ( ) They are the same product
- ( ) They are different products

Reasons for Your Choice (Optional)

Submit (1 left)

HIT Generation

- Cluster-based(Smart Batching)
- $O(n^2/k^2)$
  - **n = 10000**
  - **k = 20**
  - **HITs : 250000**
  - **Cost : $2500**



Hybrid human-machine: CrowdER

Machine-based

Quality

Time

Money

Hybrid
Human and Machine

Quality

Time

Money

Human-based

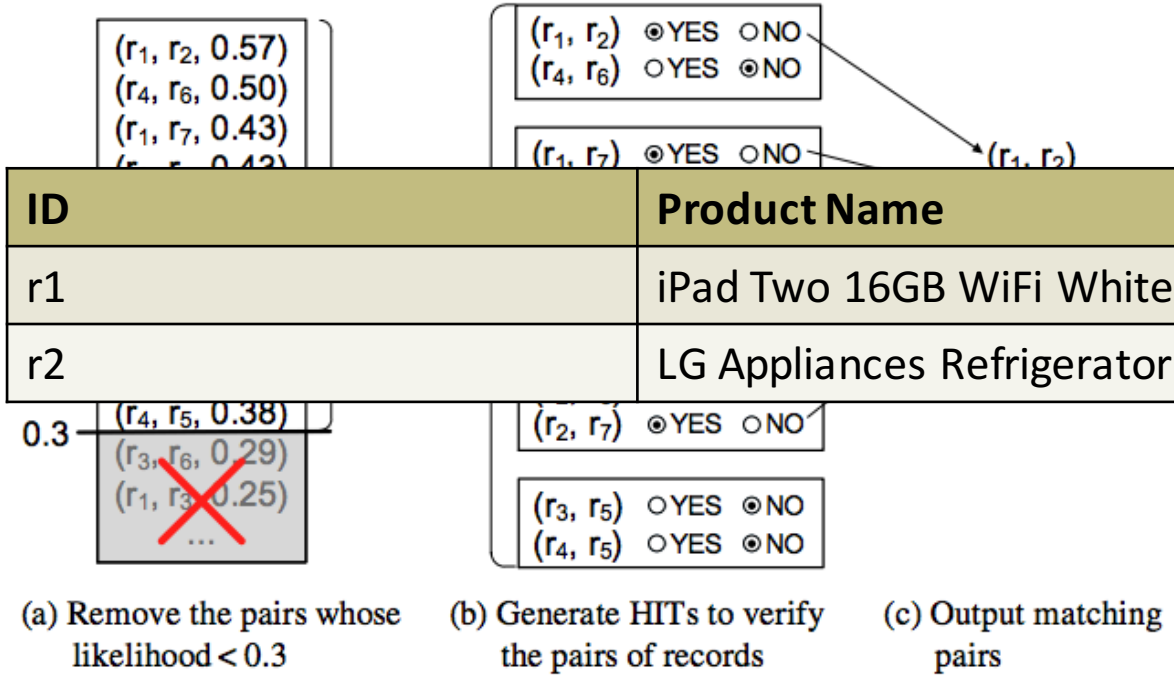
Quality

Time

Money

CrowdER

- **Core idea**
  - A large amount of records in database look very dissimilar.



ID	Product Name
r1	iPad Two 16GB WiFi White
r2	LG Appliances Refrigerator

How to design clustered-based ost?

CrowdER

- HIT Generation
  - Cluster-based HIT Generation
  - Approximation Algorithm
- Two-Tiered Approach
  - Overview
  - LCC Partitioning (Top Tier)
  - SCC Packing (Bottom Tier)

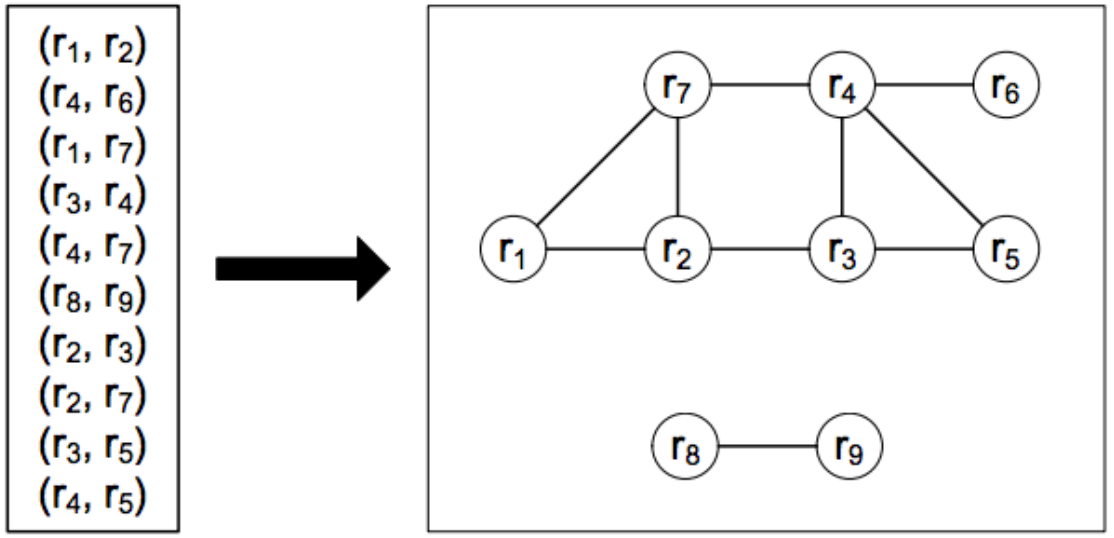
Cluster-based HIT Generation

- Requirements
  - Threshold k of numbers of records in one HIT.
  - Each pair of records should be in a HIT.
  - Smallest number of HITs.

- NP-Hard
  - Reduction from k-clique covering problem.

Traditional Approximation

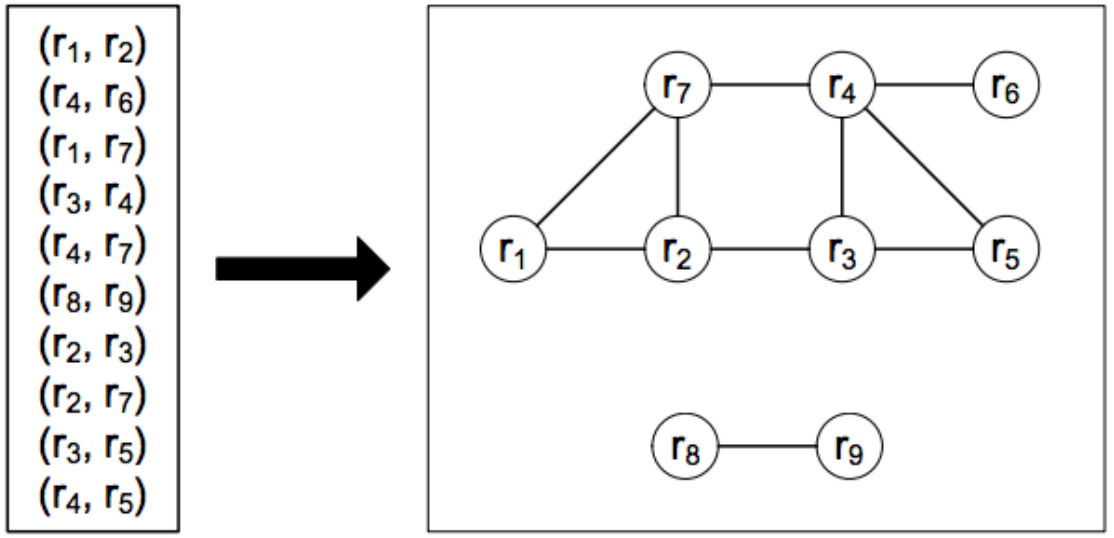
- Reduce to k-clique problem.
- |SEQ| = number of edges and vertices.
- $\lceil \frac{|SEQ|}{k-1} \rceil$



Traditional Approximation

- Performance
  - $\lceil \frac{|SEQ|}{k-1} \rceil = 7$
  - Optimal number is 3. (r_1, r_2, r_3, r_7), (r_3, r_4, r_5, r_6), (r_4, r_7, r_8, r_9)

Not good enough!



CrowdER

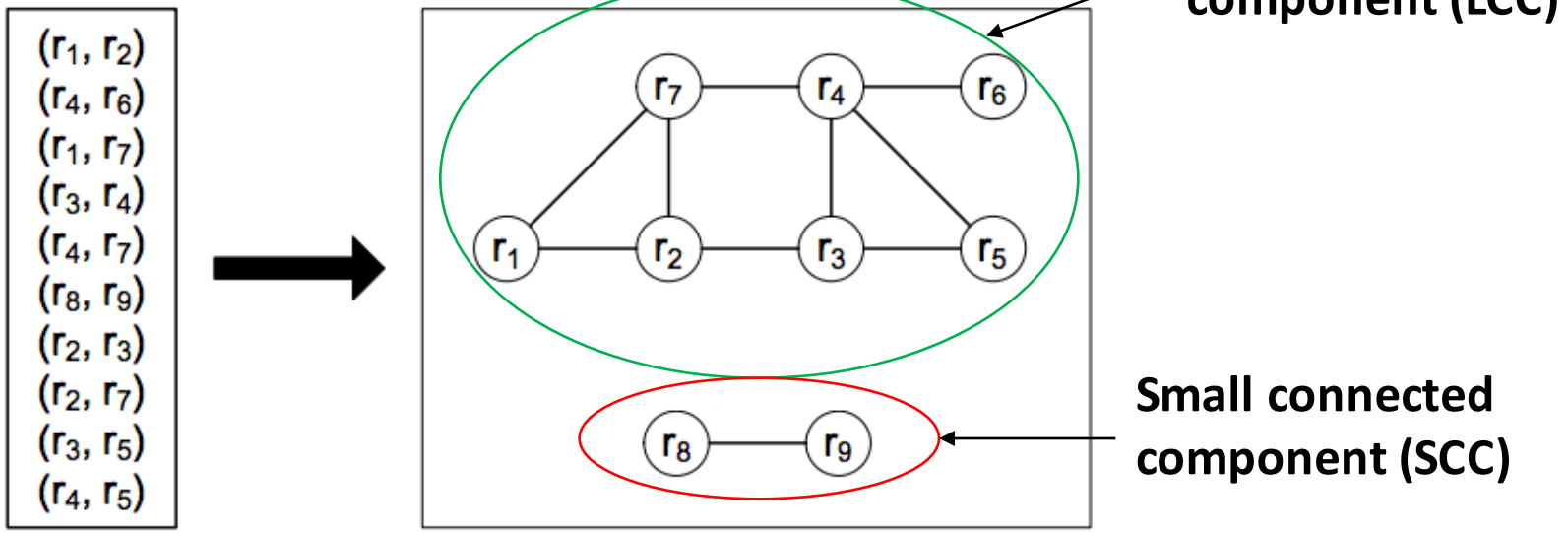
- HIT Generation
  - Cluster-based HIT Generation
  - Approximation Algorithm
- Two-Tiered Approach
  - Overview
  - LCC Partitioning (Top Tier)
  - SCC Packing (Bottom Tier)

Two-Tiered Approach Overview

k = 4

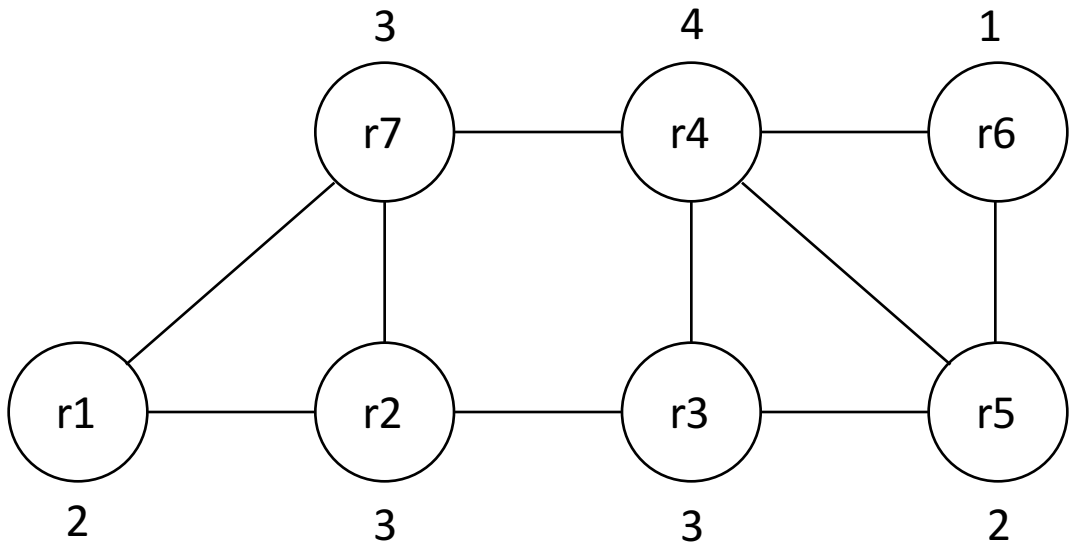
Tier1 : Decompose LCCs

Tier2 : Packing SCCs



LCC Partitioning (Top Tier)

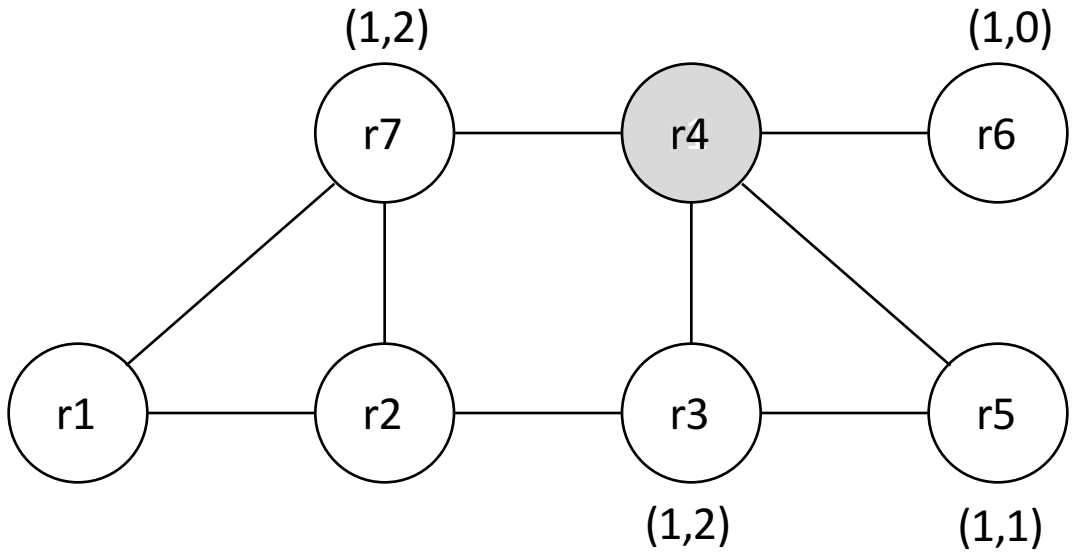
- Greedy Algorithm
  - Keep adding the most relevant vertex



Initial SCC = {r4}

LCC Partitioning (Top Tier)

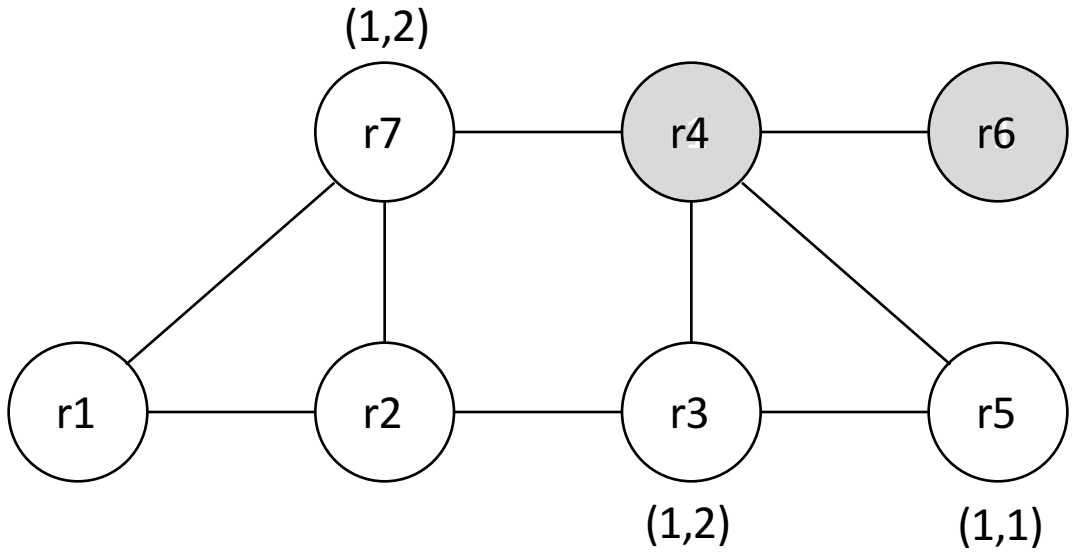
- Greedy Algorithm
  - Keep adding the most relevant vertex



SCC = {r4, r6}

LCC Partitioning (Top Tier)

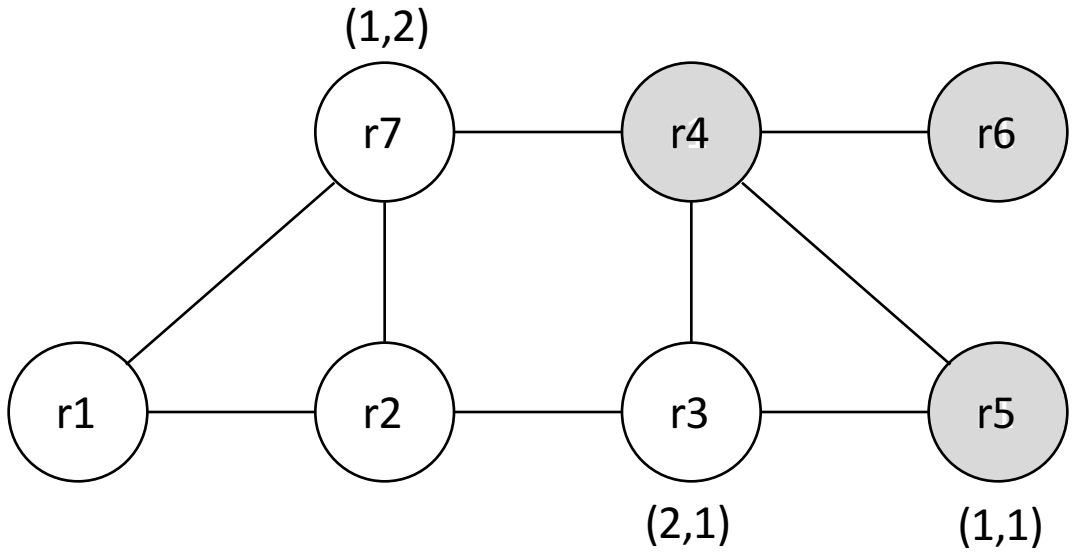
- Greedy Algorithm
  - Keep adding the most relevant vertex



SCC = {r4, r6, r5}

LCC Partitioning (Top Tier)

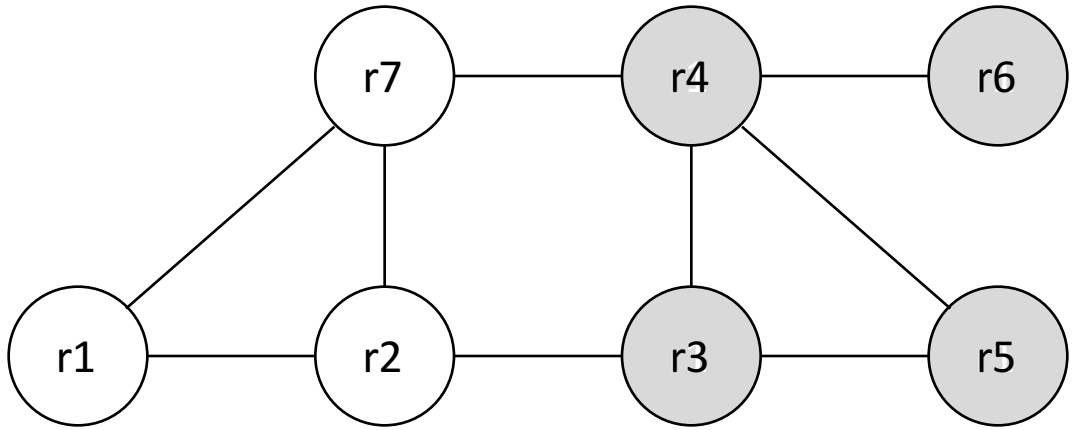
- Greedy Algorithm
  - Keep adding the most relevant vertex



SCC = {r4, r6, r5, r3}

LCC Partitioning (Top Tier)

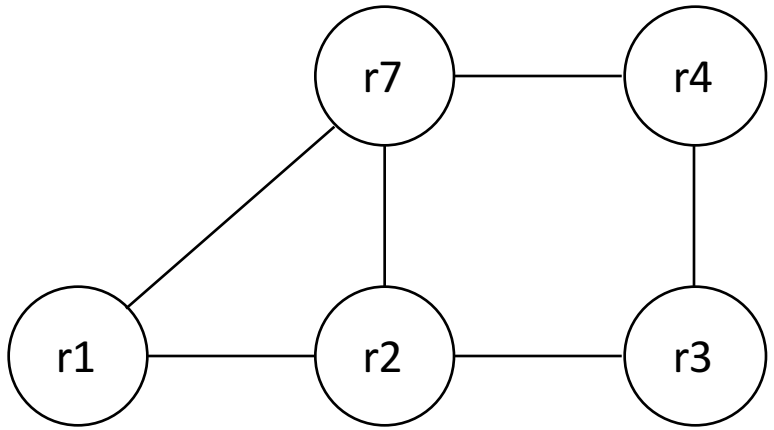
- Greedy Algorithm
  - Keep adding the most relevant vertex

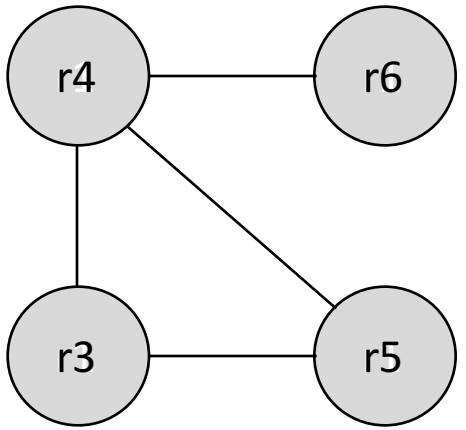


SCC = {r4, r6, r5, r3}

LCC Partitioning (Top Tier)

- Greedy Algorithm
  - Keep adding the most relevant vertex





Output SCC
Continue

SCC Packing (Bottom Tier)

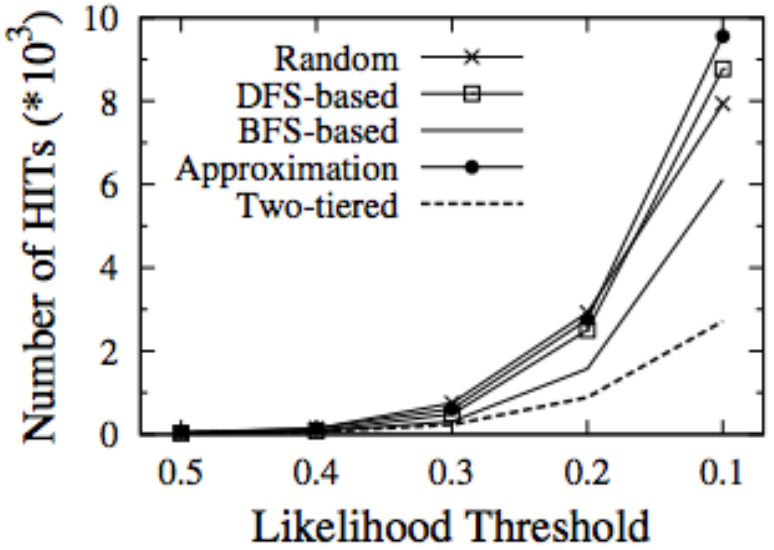
- How to pack them into the minimum number of HITs
  - NP-Hard (cutting-stock problem and knapsack problem)

- Solution
  - Transmit to integer linear program
  - Solve it by using column generation and branch-and-bound

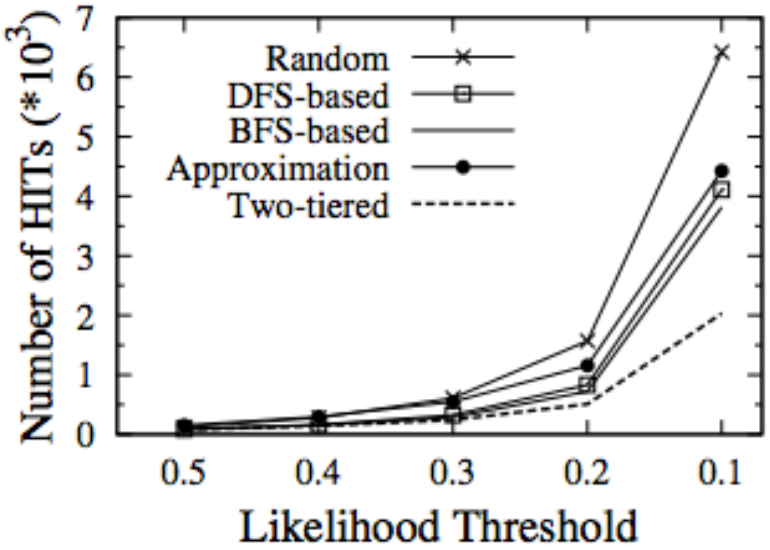
Experimental Results

- **Databases**
  - Restaurants: 858 records, 367653 pairs.
  - Products: 1081 records, 1180253 pairs.

Cluster-based HIT Generation

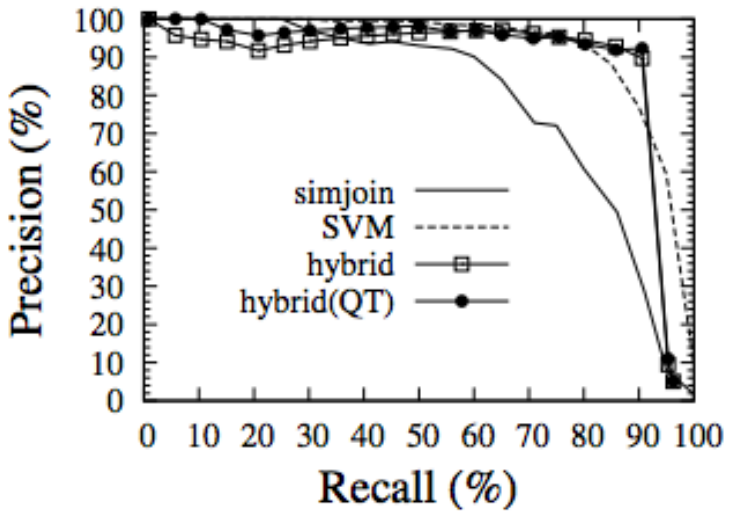


(a) *Restaurant*

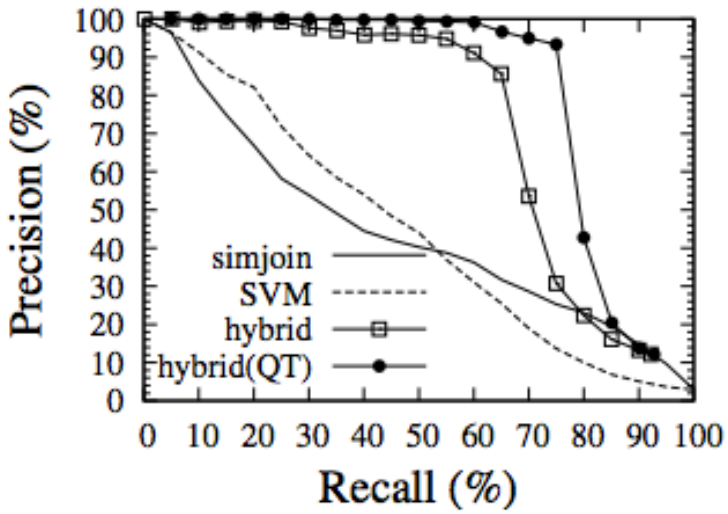


(b) *Product*

Entity-Resolution Techniques Experience

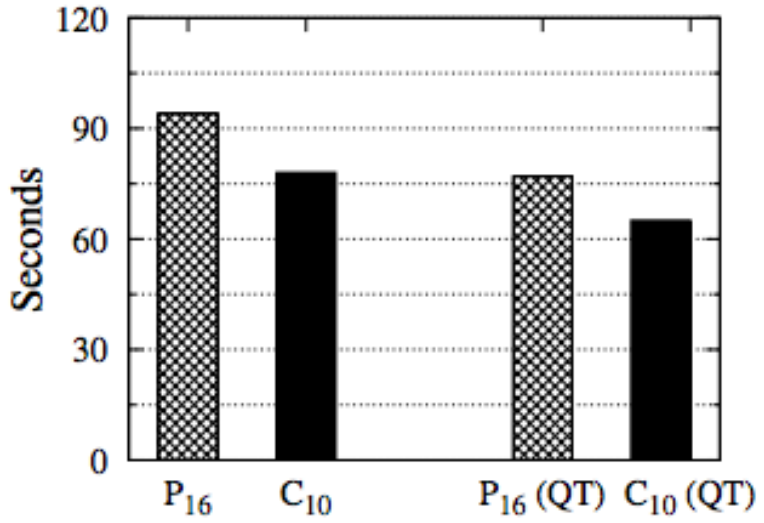


(a) *Restaurant*

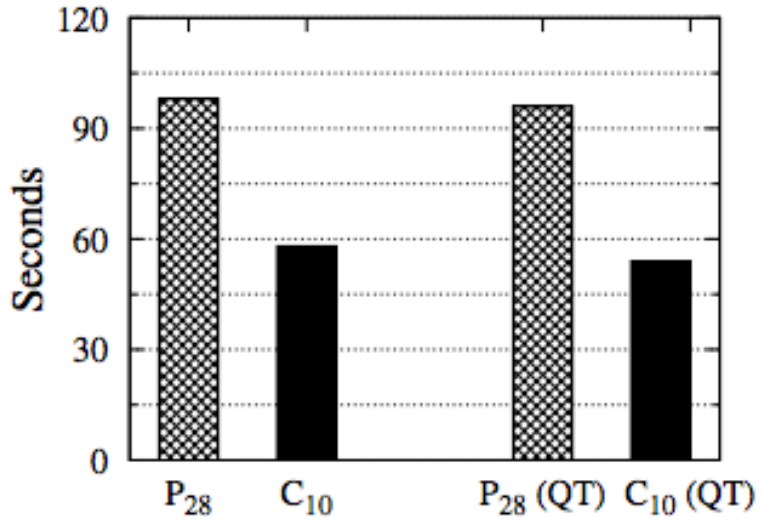


(b) *Product*

Pair-based vs. Cluster-based Experience

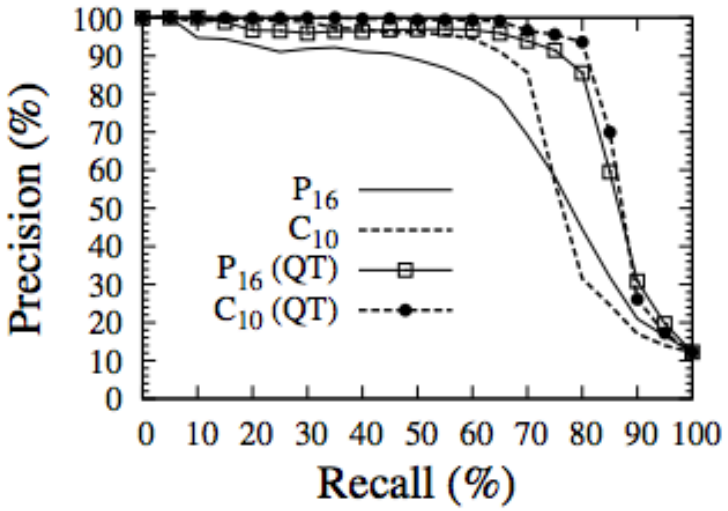


(a) *Product*

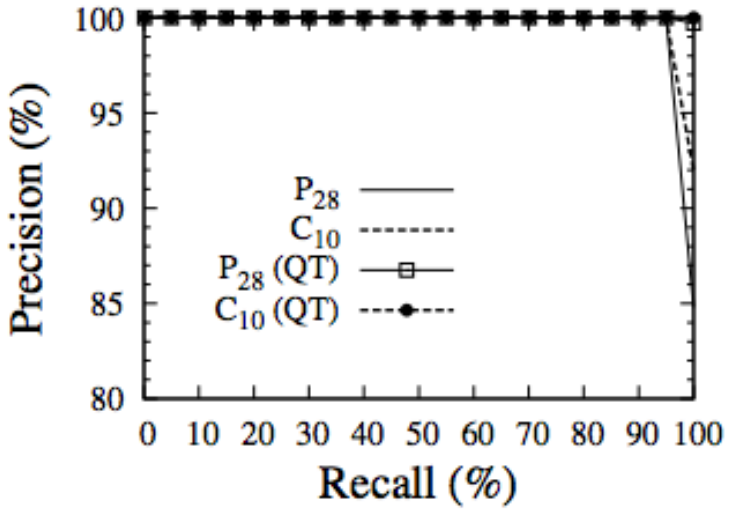


(b) *Product+Dup*

Pair-based vs. Cluster-based Experience



(a) *Product*



(b) *Product+Dup*

Conclusion

- **Discussed the problem of the existing approaches**
- **Proposed a hybrid human-machine workflow**
- **Devised a heuristic two-tiered approach**