Query-Driven Approach to Entity Resolution

Amir Ilkhechi, Stavros Sintos

Paper is by Hotham Altwaijry, Dmitri V. Kalashnikov, Sharad Mehrotra

Used the video presentation by Kalashnikov

Take away points

- Query driven entity resolution
  - Significantly reduce the cleaning overhead by resolving records that influence the query's answer.
- Notion of vestigiality.
  - Use vestigiality to reduce computation.
- Different levels of clustering representation
  - Exact, Distinct, Representative
- Using blocking with QDA have a large improvement in the number of resolves.

Overview:

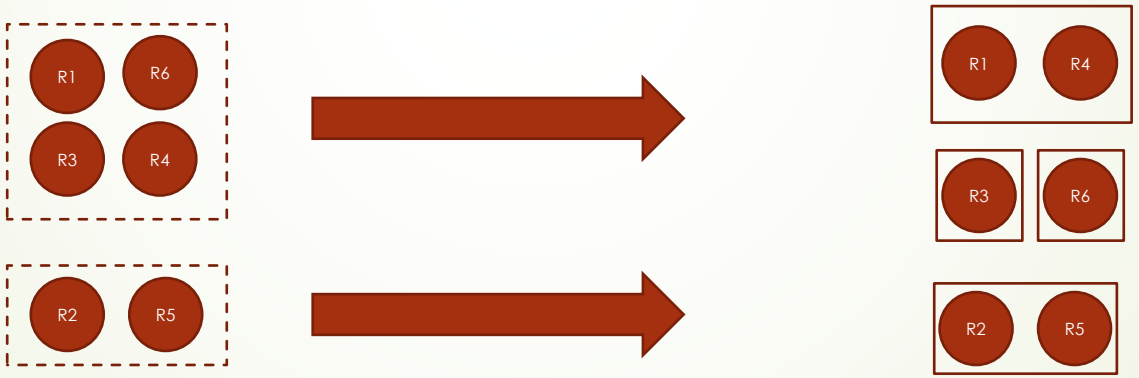
- Motivation
- High-level introduction to the problem and solutions
- Formal Problem Definition
- Algorithms (detailed explanation of the solutions)
- Experiments

Motivation:

- More than 80% of data mining researchers spend >40% of their project time on cleaning and preparation of data.
- Analysis on bad data can lead to incorrect results:
  - Fix errors before analysis (common approach)
  - Account for them during the analysis

Introduction

- ER phases:
  - Blocking
  - Similarity Computation
  - Clustering



Query Driven Approach

- Traditional ETL Process and ER:
  - Extract data from static data resources
  - Transform (and clean)
  - Load
  - Save as Data Warehouse
  - Queries are on the warehouse
- Query-driven ER:
  - Query on the raw data
  - Extract data (based on the query)
  - Do the necessary cleaning

Quality of data determines quality of decisions

Useful especially in online big data

Why query-driven

- Big data
- Queries on online data
- Know what to clean only at the query time
- Small organizations with large data sets and limited computational resources
  - Suppose that only a small portion of the data needs to be analyzed

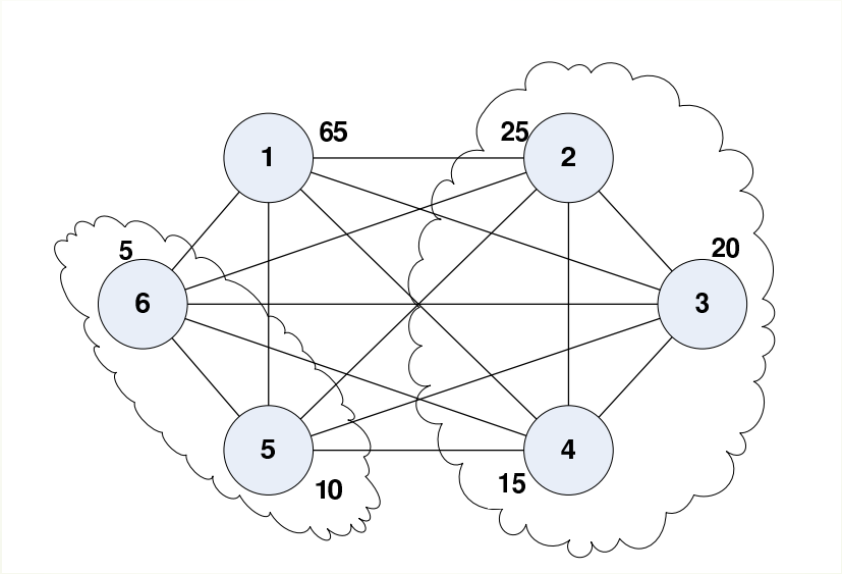
Running Example

p_id	p_title	cited	venue	authors	year
p_1	Towards efficient entity resolution	65	Very Large Data Bases	Alon Halevy	2000
p_7	Towards efficient ER	45	VLDB	Alon Halevy	2000
p_2	Entity Resolution on dynamic data	25	ACM SIGMOD	Alon Halevy, Jane Doe	2005
p_3	ER on dynamic data	20	Proc of ACM SIGMOD Conf	A. Y. Halevy, J. Doe	2005
p_4	Entity-Resolution for dynamic data	15	SIGMOD Conf	A. Halevy, Jane D.	2005
p_5	Entity-Resolution for census data	10	ICDE Conf	Alon Halevy	2002
p_6	ER on census data	5	Proc of ICDE Conf	Alon Y. Halevy	2002

Dirty relation(R)

Cluster	Pid
A	1⊕7
B	2⊕3⊕4
C	5⊕6

Ground Truth



Record with id = 7 not included

Returned Answer Semantics:

- Answer returned by first cleaning and then querying:
  - Let's call it Q*
- Exact Semantics:
  - It matches both in terms of clusters and their representations with Q*
  - E.g., {p1⊕p7,p2⊕p3⊕p4}
- Distinct Semantics
  - It matches in terms of clusters with Q*, but representations might be different
  - E.g., {p1⊕p7,p2⊕p3}
- Representative Semantics
  - It might include duplicates but it matches Q* in terms of clusters.
  - E.g., {p1,p7, p2⊕p3}

Example Query

- Select * From R Where cited >= 45

p_id	p_title	cited	venue	authors	year
p_1	Towards efficient entity resolution	65	Very Large Data Bases	Alon Halevy	2000
p_7	Towards efficient ER	45	VLDB	Alon Halevy	2000
p_2	Entity Resolution on dynamic data	25	ACM SIGMOD	Alon Halevy, Jane Doe	2005
p_3	ER on dynamic data	20	Proc of ACM SIGMOD Conf	A. Y. Halevy, J. Doe	2005
p_4	Entity-Resolution for dynamic data	15	SIGMOD Conf	A. Halevy, Jane D.	2005
p_5	Entity-Resolution for census data	10	ICDE Conf	Alon Halevy	2002
p_6	ER on census data	5	Proc of ICDE Conf	Alon Y. Halevy	2002

Dirty relation(R)

Standard solution:

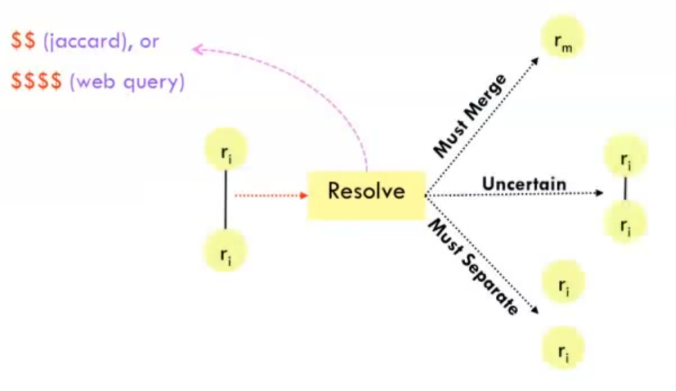
- Step 1: De-duplicate the dirty relation thoroughly
  - Perhaps by calling the resolve function on all pairs of records

cluster	p_id	p_title	cited	venue	authors	year
C_1	$p_1 \oplus p_7$	Towards efficient entity resolution	110	Very Large Data Bases	Alon Halevy	2000
C_2	$p_2 \oplus p_3 \oplus p_4$	Entity-Resolution on dynamic data	60	Proc of ACM SIGMOD Conf	Alon Halevy, Jane Doe	2005
C_3	$p_5 \oplus p_6$	Entity-Resolution for census data	15	Proc of ICDE Conf	Alon Halevy	2002

- Step 2: Compute the query result over the acquired clean relation
- Issues:
  - Large number of calls to resolve function
  - Resolve itself is generally expensive

Resolve Function

- Resolve is a pairwise function R(r_i,r_j)



Merge Function

- It consolidates two duplicate records to produce a new record.
  - Combine (⊕) function is used to combine each attribute

p_2	Entity Resolution on dynamic data	25	ACM SIGMOD	Alon Halevy, Jane Doe	2005
p_3	ER on dynamic data	20	Proc of ACM SIGMOD Conf	A. Y. Halevy, J. Doe	2005

45	**ADD semantics: v_i ⊕ v_j = v_i + v_j**
25	MAX semantics: v_i ⊕ v_j = max(v_i, v_j)
20	MIN semantics: v_i ⊕ v_j = min(v_i, v_j)

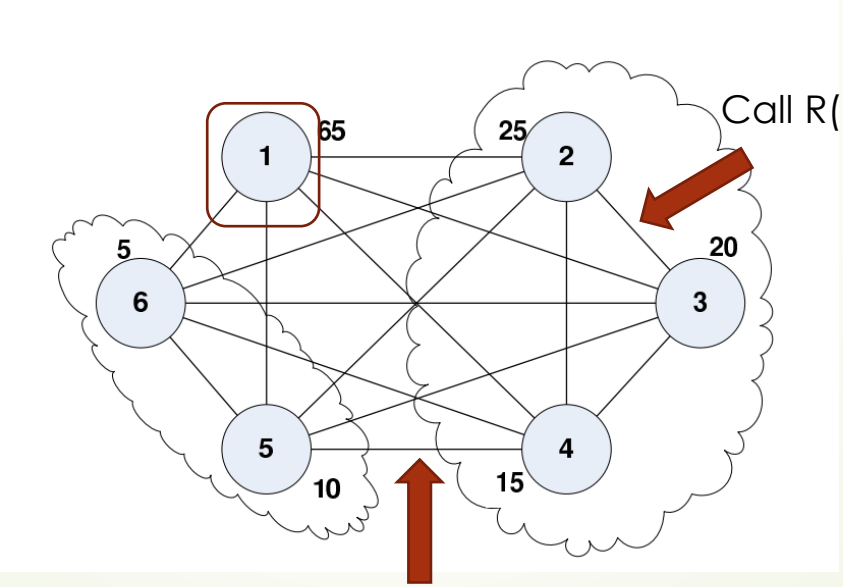
{Alon Halevy, A. Y.Halevy}	**UNION semantics: v_i ⊕ v_j = v_i ∪ v_j**
{Alon Halevy}	EXEMPLAR semantics: v_i ⊕ v_j = choose v_i or v_j

Exploiting query semantics

- Query: Select * From R Where cited >= 30
  - Assume ADD semantics
- We can prune the resolves

answer 1 = {1}

answer 2 = {1, 2⊕3}



Gains compared to other approaches:

Algorithm	# of resolve calls
Query-driven algorithm	2
Standard Transitivity Closure Algorithm	11
Correlation Clustering Algorithm	15

Formal Problem Definition

- Notations:
  - The following represents a relation in the database $R = \{r_1, r_2, \ldots, r_{|R|}\}$
  - The set of attributes are given as $\langle a_1, a_2, \ldots, a_n \rangle$
  - The k'th record is represented as $r_k = \langle \nu_{k1}, \nu_{k2}, \ldots, \nu_{kn} \rangle$
- The goal of traditional ER:
  - Partition records in R into a set of non-overlapping clusters
  - 2 records from the same cluster correspond to the same entity
  - 2 records from distinct clusters correspond to different entities
- Queries:
  - $$\texttt{SELECT } [\texttt{DISTINCT}|\texttt{EXACT}] \ * \ \texttt{FROM } R \ \texttt{WHERE } a_\ell \ op \ t$$

    $$op \text{ is } \begin{cases} <, \leq, >, \geq, \text{ or } = & \text{if } a_\ell \text{ is a numeric attribute;} \\ = & \text{if } a_\ell \text{ is a categorical attribute.} \end{cases}$$

Problem: Optimization

- Given: Query Q
- Minimize: # of calls to resolve function
- Subject to:
  - Query satisfaction:
    - Each cluster returned by QDA must satisfy Q.
  - User-defined equivalence:
    - The answer generated by QDA must be (exactly, distinctly, or representatively) equivalent to that of TC

Vestigiality

Vestigiality

- Categorize triples $(p, \oplus, a_l)$
- Deal with multi-predicate selection queries
- Construction of the labeled graph
- Use relevant clique, minimal clique to test for vestigiality.

Triple $(p, \oplus, a_l)$ Categorization

- p: Query predicate
- $\oplus$: Combine function
- a_l: Attribute

- Help to significantly reduce the cleaning overhead by resolving only those edges that may influence the answer of Q.

Triple $(p, \oplus, a_l)$ Categorization

Definition 5. Triple $(p, \oplus, a_\ell)$ is *in-preserving*, if for all possible values $\nu_{i\ell}, \nu_{j\ell} \in a_\ell$, if p is **true** for $\nu_{i\ell}$, then p is also **true** for all $\nu_{i\ell} \oplus \nu_{j\ell}$.

- (cited>=45, ADD, cited): in-preserving
- (cited<=45, ADD, cited): not in-preserving

Triple $(p, \oplus, a_l)$ Categorization

Definition 6. Triple $(p, \oplus, a_\ell)$ is *out-preserving*, if for all possible values $\nu_{i\ell}, \nu_{j\ell} \in a_\ell$, if p is `false` for $\nu_{i\ell}$, then it is also `false` for all $\nu_{i\ell} \oplus \nu_{j\ell}$.

- (cited<=45, ADD, cited): out-preserving

Multi-Predicate Selection Queries

- More complex selection queries (AND, OR, NOT).

$\tau_i = (p_i, \oplus_i, a_i)$	$\tau_j = (p_j, \oplus_j, a_j)$	$\tau_i \wedge \tau_j$	$\tau_i \vee \tau_j$
in-preserving	in-preserving	in-preserving	in-preserving
in-preserving	out-preserving	neither	neither
out-preserving	out-preserving	out-preserving	out-preserving
in-preserving	neither	neither	neither
out-preserving	neither	neither	neither
neither	neither	neither	neither

Multi-Predicate Selection Queries

Query 2. SELECT $*$ FROM R WHERE $cited \geq 45$ AND $cited \leq 65$

- p_1: $cited \geq 45$
- p_2: $cited \leq 65$
- $\tau_1 = (cited \geq 45, ADD, cited)$: in-preserving
- $\tau_2 = (cited \leq 65, ADD, cited)$: out-preserving
- $\tau_1 \cap \tau_2$: neither
- Why?
  - $r_i \rightarrow cited = 10$
  - $r_j \rightarrow cited = 40$
  - $r_k \rightarrow cited = 20$
  - $r_i \oplus r_j \rightarrow cited = 50$ (IN the answer)
  - $r_i \oplus r_j \oplus r_k \rightarrow cited = 70$ (OUT the answer)

Creating and Labeling the Graph

- Build and label the Graph.
- Avoid creating as many nodes and edges as possible.
- Blocking to reduce the edges in the graph.
- Remove from consideration nodes and edges that will not influence further processing of Q.

```
CREATE-GRAPH(R, Q, A_cur, V_out, V_maybe, ⊕)
 1  for each r_k ∈ R do
 2      v_k ← CREATE-NODE(r_k)
 3      if IS-IN-PRESERVING(p, ⊕, a_ℓ) and SATISFY-QRY(v_k, Q) then
 4          A_cur ← A_cur ∪ {v_k}
 5      else if IS-OUT-PRESERVING(p, ⊕, a_ℓ)
        and not SATISFY-QRY(v_k, Q) then
 6          V_out ← V_out ∪ {v_k}
 7      else V_maybe ← V_maybe ∪ {v_k}
 8  V ← {A_cur, V_out, V_maybe}
 9  E ← CREATE-EDGES-WITH-BLOCKING(V_maybe, V_maybe)
10  E ← E ∪ CREATE-EDGES-WITH-BLOCKING(V_out, V_maybe)
11  return G(V, E)
```

Create and label the nodes

```
1   for each r_k ∈ R do
2       v_k ← CREATE-NODE(r_k)
3       if IS-IN-PRESERVING(p, ⊕, a_ℓ) and SATISFY-QRY(v_k, Q) then
4           A_cur ← A_cur ∪ {v_k}
5       else if IS-OUT-PRESERVING(p, ⊕, a_ℓ)
        and not SATISFY-QRY(v_k, Q) then
6           V_out ← V_out ∪ {v_k}
7       else V_maybe ← V_maybe ∪ {v_k}
```

1. $\ell[v_k] =$ `in` when triple $(p, \oplus, a_\ell)$ is in-preserving and v_k satisfies Q. Node v_k is added to A_{cur} as it is guaranteed to be in the final answer.
2. $\ell[v_k] =$ `out` when triple $(p, \oplus, a_\ell)$ is out-preserving and v_k does not satisfy Q. Node v_k is added to V_{out}.
3. $\ell[v_k] =$ `maybe`, otherwise. Node v_k is added to V_{maybe}.

Create and label the edges

9 $E \leftarrow$ CREATE-EDGES-WITH-BLOCKING(V_{maybe}, V_{maybe})
10 $E \leftarrow E \cup$ CREATE-EDGES-WITH-BLOCKING(V_{out}, V_{maybe})

- Add the edge $e_{ij} = (v_i, v_j)$
  - Nodes v_i, v_j belong in the same block.
  - $v_i, v_j \in V_{maybe}$
  - $(v_i \in V_{out}$ AND $v_j \in V_{maybe})$ OR $(v_j \in V_{out}$ AND $v_i \in V_{maybe})$

Create and label the edges

1. $\ell[e_{ij}] =$ **yes**, when $\Re(r_i, r_j)$ has already been called and returned **MustMerge**,
2. $\ell[e_{ij}] =$ **no**, when $\Re(r_i, r_j)$ has already been called and returned **MustSeparate**,
3. $\ell[e_{ij}] =$ **maybe**, when $\Re(r_i, r_j)$ has already been called and returned **Uncertain**,
4. $\ell[e_{ij}] =$ **vestigial**, when, Definition 8 holds. Note that as QDA proceeds forward, some edges that were not vestigial previously may become vestigial. But once they become vestigial, they remain so.
5. $\ell[e_{ij}] =$ **unresolved**, otherwise.

- For efficiency:
  - Label no edge -> remove the edge from the graph.
  - Label yes edge (v_i, v_j) -> Merge nodes v_i, v_j.

Create and label the edges

1. $\ell[e_{ij}] =$ **yes**, when $\Re(r_i, r_j)$ has already been called and returned **MustMerge**,
2. $\ell[e_{ij}] =$ **no**, when $\Re(r_i, r_j)$ has already been called and returned **MustSeparate**,
3. $\ell[e_{ij}] =$ **maybe**, when $\Re(r_i, r_j)$ has already been called and returned **Uncertain**,
4. $\ell[e_{ij}] =$ **vestigial**, when, Definition 8 holds. Note that as QDA proceeds forward, some edges that were not vestigial previously may become vestigial. But once they become vestigial, they remain so.
5. $\ell[e_{ij}] =$ **unresolved**, otherwise.

- For efficiency:
  - Label no edge -> remove the edge from the graph.
  - Label yes edge (v_i, v_j) -> Merge nodes v_i, v_j.

A_{cur} is the answer result assuming all vestigial and unresolved edges are NO edges.

Vestigial Edges

Definition 8. Let $\mathcal{A}$ be the original entity resolution algorithm. An edge $e_{ij} \in E$ is `vestigial` when, regardless of what the ground truth for e_{ij} might be, QDA can guarantee that by treating e_{ij} as a `no` edge, it can still compute an equivalent answer to that of $\mathcal{A}$.

- Identify Vestigial Edges?

Vestigiality Testing using Cliques.

- Clique:
  - Given $G(V, E)$ a set $S \subseteq V$ is a clique if for every $v_i, v_j \in S$, $(v_i, v_j) \in E$.

Lemma 1. Nodes (records) co-refer only if they form a clique consisting of only **yes** edges in the ground truth.

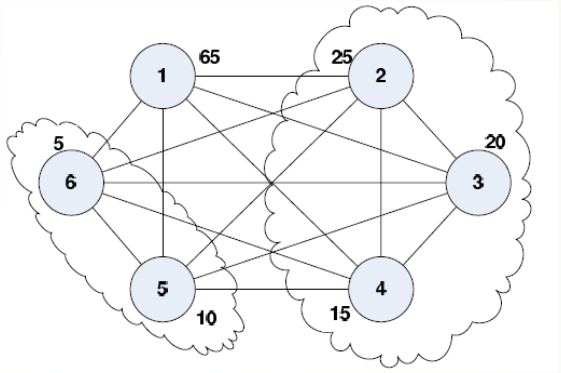
- If a group of nodes is not a clique, that group corresponds to at least two distinct entities.

Relevant Clique

Definition 9. A clique S is called *relevant* to Q, if we can assign labels to its edges such that this labeling might change $\mathcal{C}_{cur}$, by either adding (at least one) new cluster to $\mathcal{C}_{cur}$, or removing (at least one) cluster from $\mathcal{C}_{cur}$.

Theorem 1. Given the current labeled graph G, a selection query Q with predicate p on attribute a_ℓ, if no relevant clique exists that includes e_{ij}, then e_{ij} is vestigial. However, the reverse does not hold: a vestigial edge could be part of a relevant clique. Proof is covered in [1].

Relevant Clique



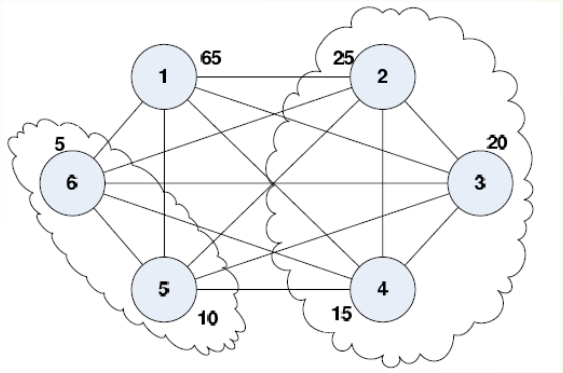
- Resolve e_{23}
- $p = (cited \geq 45)$
- $A_{cur} = \{p_1, p_2 \oplus p_3\}$
- All edges incident to $p_1, p_2 \oplus p_3$ are vestigial.
- Nodes 4, 5, 6 form a clique S.
- Sum up of the cited attribute of S -> $5 + 10 + 15 = 30 \not> 45$.
- Merge nodes in S cannot change A_{cur}
- Edges e_{45}, e_{46}, e_{56} are not part of any relevant clique, so vestigial edges!

Minimal Clique

Definition 10. A relevant clique S is called a *minimal clique*, if no subset of nodes in S can form a relevant clique.

Theorem 2. Given a graph G and an in-preserving $(p, \oplus, a_\ell)$, an unresolved edge e_{ij} is vestigial if and only if no minimal clique exists that includes e_{ij}. Proof is covered in [1].

Minimal Clique



- Resolve e_{45}
- Triple $(cited \geq 45, ADD, cited)$ is in-preserving
- $S = \{p_2, p_3, p_4\}$ is a relevant clique $(25 + 20 + 15 = 60 \geq 45)$
- $S_m = \{p_2, p_3\} \subset S$ forms a minimal clique$(25 + 20 = 45 \geq 45)$
- Edges e_{24}, e_{34} are vestigial since both e_{24}, e_{34} do not belong to any minimal clique.

IS_VESTIGIAL()

```
IS-VESTIGIAL(e_ij, G, Q, ⊕)
1  if IS-IN-PRESERVING(p, ⊕, a_ℓ) then
2      return not IS-IN-A-MINIMAL-CLIQUE(e_ij, G, Q)
3  else return not IS-IN-A-RELEVANT-CLIQUE(e_ij, G, Q)
```

- NP-hard to test for Vestigiality.
  - Straightforward Reduction from the k-Clique problem.
- Worse than $O(n^2)$ calls of the resolve function.

- Challenge: Design QDA that performs vestigiality testing efficiently.


Query-Driven Solution

Overview

- The answer of QDA equivalent to first applying TC on the whole dataset and then querying the cleaned dataset.
- TC Method
  - Choose a pair of nodes to resolve
  - Apply the resolve function
  - Merge nodes if resolve function returns positive answer
- QDA similar to TC
  - Uses its own pair-picking strategy.
  - Instead of calling the resolve function, is it a vestigial edge?

QDA Approach

- Create and label the graph
- Choose an edge to resolve
  - Not the main focus of the paper
  - Quickly add some cluster-representatives or break many relevant cliques.
  - Pick edges according to a weight $w_{ij} = v_{il} \oplus v_{jl}$
- Lazy edge removal
  - Implemented many optimizations.
  - Check if an edge exists.
  - After merging two nodes only common edges remain
  - $O(|R|)$ in the worst case!
  - Do not remove edges at the time of merge – Check if a node has been merged with another node. $O(1)$ time!
- Vestigiality testing (next)
- Stopping condition
  - If there exists an edge that is neither resolved nor vestigial.
- Compute the answer (later)

Vestigiality Testing

- Given an edge e_{ij} decide if it is vestigial.
- Recall, NP-hard
- Use an inexact but fast check if e_{ij} is potentially part of any relevant clique.

```
VESTIGIALITY-TESTING(e_ij, G, Q, ⊕)
 1  if IS-IN-PRESERVING(p, ⊕, a_ℓ)
    and MIGHT-CHANGE-ANSWER(∅, v_i ⊕ v_j, Q) then
 2      res ← ℜ(v_i, v_j)
 3      if res = MustMerge then
 4          A_cur ← A_cur ∪ {v_i ⊕ v_j}
 5          V_maybe ← V_maybe − {v_i, v_j}
 6      else if res = MustSeparate then
 7          E ← E − {e_ij}
 8      else ℓ[e_ij] = maybe
 9  else if CHECK-POTENTIAL-CLIQUE(e_ij, G, Q) then
10      res ← ℜ(v_i, v_j)
11      if res = MustMerge then
12          v_i ← v_i ⊕ v_j
13          𝒩[v_i] = 𝒩[v_i] ∩ 𝒩[v_j]
14          V_maybe ← V_maybe − {v_j}
15      else if res = MustSeparate then
16          E ← E − {e_ij}
17      else ℓ[e_ij] = maybe
18  else E ← E − {e_ij}      // this edge is vestigial
```

Edge Miniclique Check Optimization

- An Optimization
- If $(p, \oplus, a_l)$ is in-preserving and e_{ij} can change the current answer to Q
  - e_{ij} is not vestigial and the algorithm calls the resolve function.

```
1  if IS-IN-PRESERVING(p, ⊕, a_ℓ)
   and MIGHT-CHANGE-ANSWER(∅, v_i ⊕ v_j, Q) then
2     res ← ℜ(v_i, v_j)
3     if res = MustMerge then
4        A_cur ← A_cur ∪ {v_i ⊕ v_j}
5        V_maybe ← V_maybe − {v_i, v_j}
6     else if res = MustSeparate then
7        E ← E − {e_ij}
8     else ℓ[e_ij] = maybe
```

Check for Potential Clique

- CHECK-POTENTIAL-CLIQUE(): Test if e_{ij} can potentially be part of any relevant clique.
- If yes, call the resolve function, otherwise it marks as vestigial.

```
 9   else if CHECK-POTENTIAL-CLIQUE(e_ij, G, Q) then
10      res ← ℜ(v_i, v_j)
11      if res = MustMerge then
12          v_i ← v_i ⊕ v_j
13          𝒩[v_i] = 𝒩[v_i] ∩ 𝒩[v_j]
14          V_maybe ← V_maybe − {v_j}
15      else if res = MustSeparate then
16          E ← E − {e_ij}
17      else ℓ[e_ij] = maybe
18  else E ← E − {e_ij}      // this edge is vestigial
```

CHECK-POTENTIAL-CLIQUE()

- Quickly check if e_{ij} is involved in any relevant/minimal clique.
- Safe approximation function:
  - False only when e_{ij} is not part of any relevant/minimal clique
  - True when e_{ij} might be part of some relevant clique.

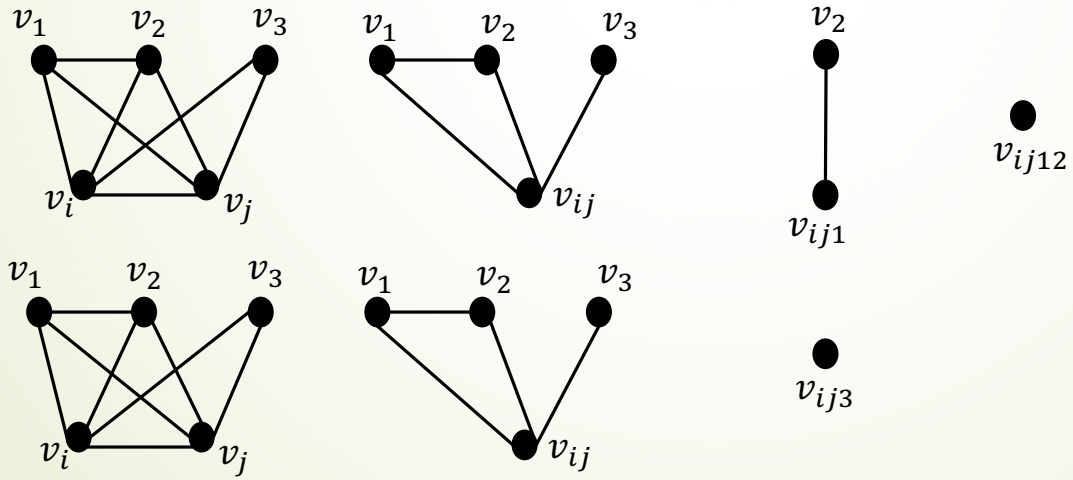
CHECK-POTENTIAL-CLIQUE()

- Merge nodes v_i, v_j and check if their merge change Q's answer.
- If not, find all common neighbors of v_i, v_j and tries to find the smallest potential clique which might change Q's answer.
- Keep expanding the size of such clique until no common neighbors left.

```
CHECK-POTENTIAL-CLIQUE(e_ij, G, Q)
 1  v_new ← v_i ⊕ v_j
 2  if MIGHT-CHANGE-ANSWER(∅, v_new, Q) then
 3      return true
 4  V_intersect ← N[v_i] ∩ N[v_j]
 5  for each v_k ∈ V_intersect do
 6      v_old ← v_new
 7      v_new ← v_old ⊕ v_k
 8      if MIGHT-CHANGE-ANSWER(v_old, v_new, Q) then
 9          return true
10  return false
```

CHECK-POTENTIAL-CLIQUE()

```
CHECK-POTENTIAL-CLIQUE(e_ij, G, Q)
 1  v_new ← v_i ⊕ v_j
 2  if MIGHT-CHANGE-ANSWER(∅, v_new, Q) then
 3      return true
 4  V_intersect ← N[v_i] ∩ N[v_j]
 5  for each v_k ∈ V_intersect do
 6      v_old ← v_new
 7      v_new ← v_old ⊕ v_k
 8      if MIGHT-CHANGE-ANSWER(v_old, v_new, Q) then
 9          return true
10  return false
```



Computing Answer of Given Semantics

- Compute the final answer A_{cur} to query Q based on the answer semantics the user requested.

```
COMPUTE-ANSWER(A_cur, V_maybe, Q, S)
 1  for each v_i ∈ V_maybe do
 2      if SATISFY-QRY(v_i, Q) then
 3          A_cur ← A_cur ∪ v_i
 4  if S = Distinct then
 5      for each v_i ∈ A_cur do
 6          for each v_j ≠ v_i ∈ A_cur do
 7              if ℜ(v_i, v_j) = MustMerge then
 8                  v_i ← v_i ⊕ v_j
 9                  A_cur ← A_cur − {v_j}
10  else if S = Exact then
11      for each v_i ∈ A_cur do
12          for each v_j ≠ v_i ∈ A_cur ∪ V_maybe do
13              if ℜ(v_i, v_j) = MustMerge then
14                  v_i ← v_i ⊕ v_j
15                  if v_j ∈ A_cur then
16                      A_cur ← A_cur − {v_j}
```

Representative answer semantics

- Add nodes from V_{maybe} which satisfy Q to A_{cur}.
- At this point, A_{cur} satisfies representative answer semantics.

```
1   for each v_i ∈ V_maybe do
2       if SATISFY-QRY(v_i, Q) then
3           A_cur ← A_cur ∪ v_i
```

Distinct answer semantics

- Clean the representative answers in the current A_{cur} using the original TC algorithm.
  - Remove duplicates by resolving all pairs of nodes in A_{cur}.
- Additional cost: $O(|A_{cur}|^2)$

```
4  if S = Distinct then
5     for each v_i ∈ A_cur do
6        for each v_j ≠ v_i ∈ A_cur do
7           if ℜ(v_i, v_j) = MustMerge then
8              v_i ← v_i ⊕ v_j
9              A_cur ← A_cur − {v_j}
```

Exact answer semantics

- Compare all nodes in A_{cur} with all nodes in $A_{cur} \cup V_{maybe}$
- Extra cost: $O(|A_{cur}| \cdot |R|)$

```
10  else if S = Exact then
11      for each v_i ∈ A_cur do
12          for each v_j ≠ v_i ∈ A_cur ∪ V_maybe do
13              if ℜ(v_i, v_j) = MustMerge then
14                  v_i ← v_i ⊕ v_j
15                  if v_j ∈ A_cur then
16                      A_cur ← A_cur − {v_j}
```

Exact answer semantics

- Compare all nodes in A_{cur} with all nodes in $A_{cur} \cup V_{maybe}$
- Extra cost: $O(|A_{cur}| \cdot |R|)$

```
10  else if S = Exact then
11      for each v_i ∈ A_cur do
12          for each v_j ≠ v_i ∈ A_cur ∪ V_maybe do
13              if ℜ(v_i, v_j) = MustMerge then
14                  v_i ← v_i ⊕ v_j
15                  if v_j ∈ A_cur then
16                      A_cur ← A_cur − {v_j}
```

To produce distinct or exact answer, vestigial edges are considered unresolved.

Answer Correctness

Lemma 2. If the resolve function is always accurate, then QDA will compute answers that are: *representationally*, *distinctly*, or *exactly* equivalent to those in $\mathcal{C}_{gt}$.

- Naturally, resolve functions are not always accurate -> no ER technique can guarantee correctness.
- They do not assume that resolve is always accurate.

Experimental Evaluation

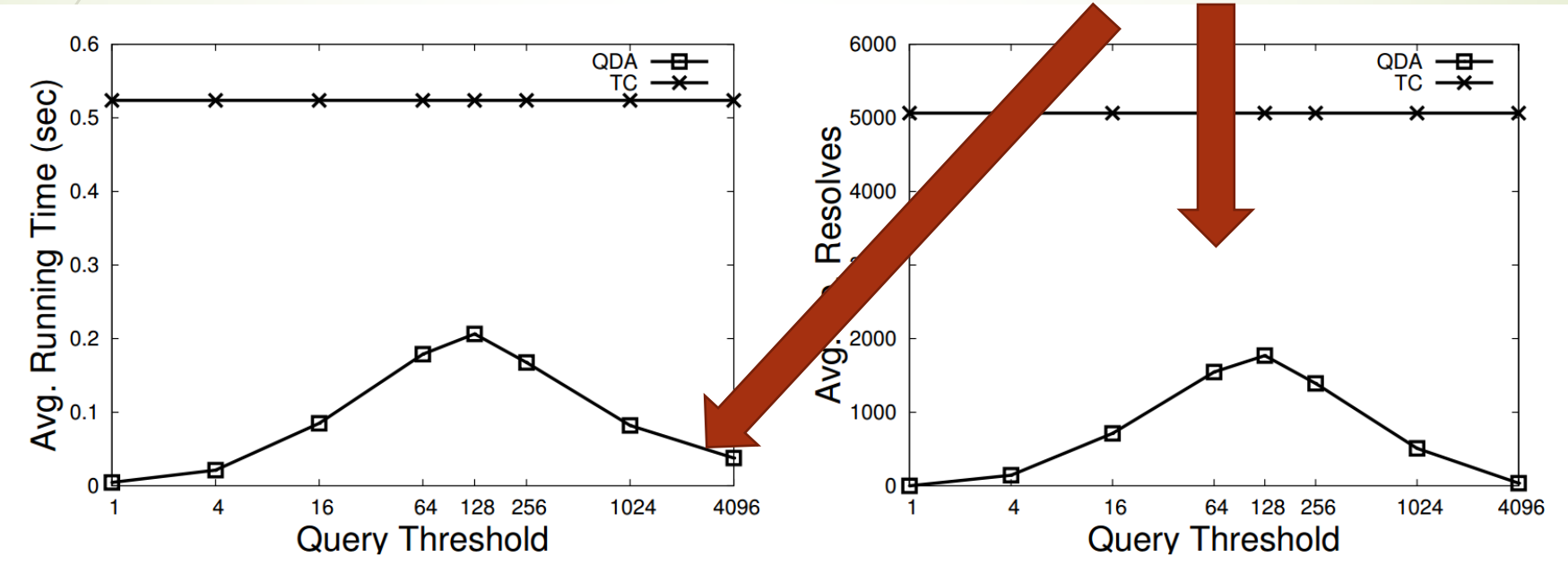
Experimental Results

- Dataset:
  - Bibliographical entries from Google Scholar
  - Contains 50 researchers
    - 16396 records, 14.3% of which are duplicates
- Resolve Function
  - Soft-TF-IDF on titles
  - Jaro-Winkler distance on author names
- Blocking Techniques:
  - Bucketize records that might be duplicates
    - 1: First two letters of titles as the hash key
    - 2: Last two letters of titles as the hash key

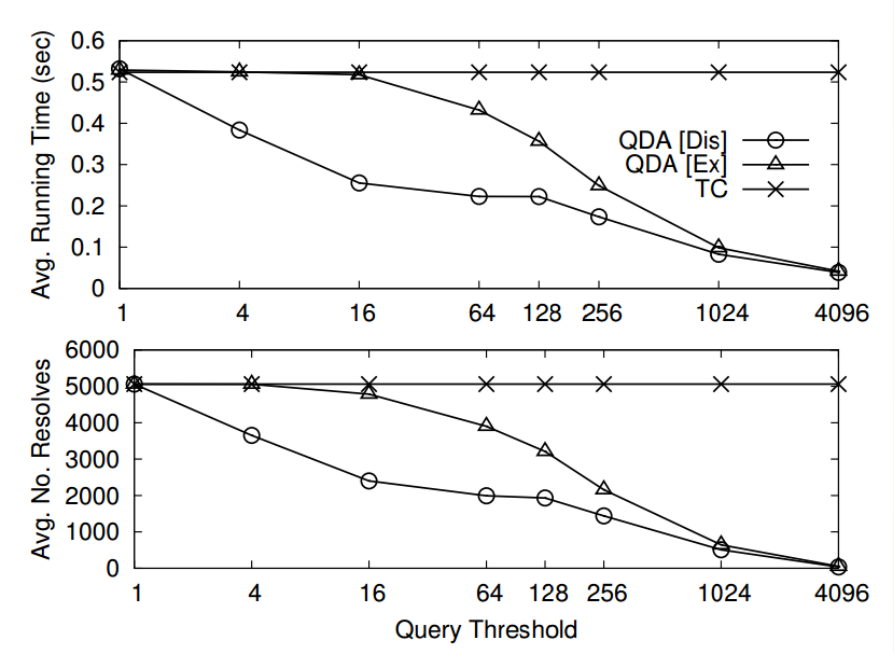
QDA vs. TC (Transitive Closure)

- Query: Select * From R Where cited >= t

Very similar: means bottleneck!

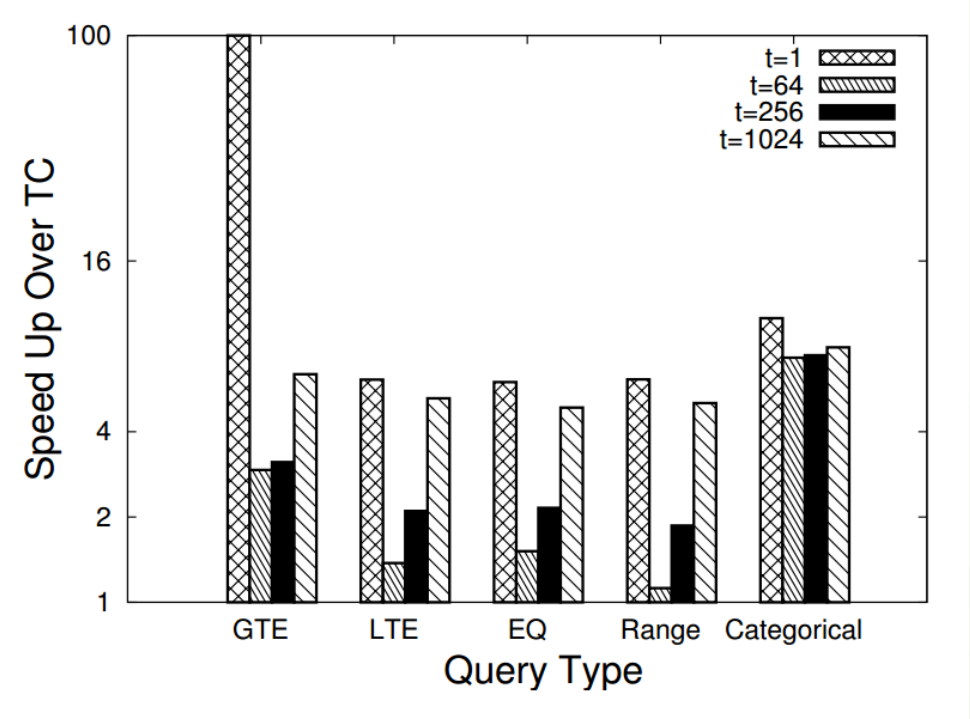


QDA vs. TC [Answer Semantics]

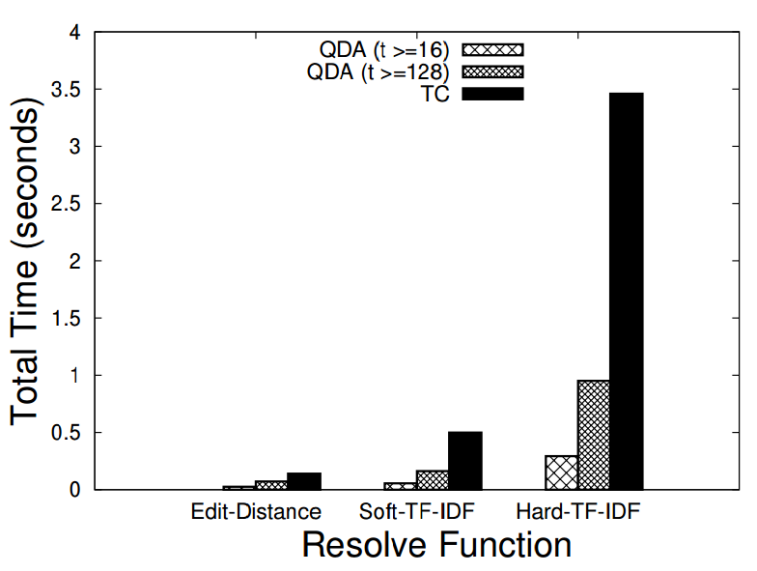


The larger the size of the answer, the higher the extra cost QDA will pay

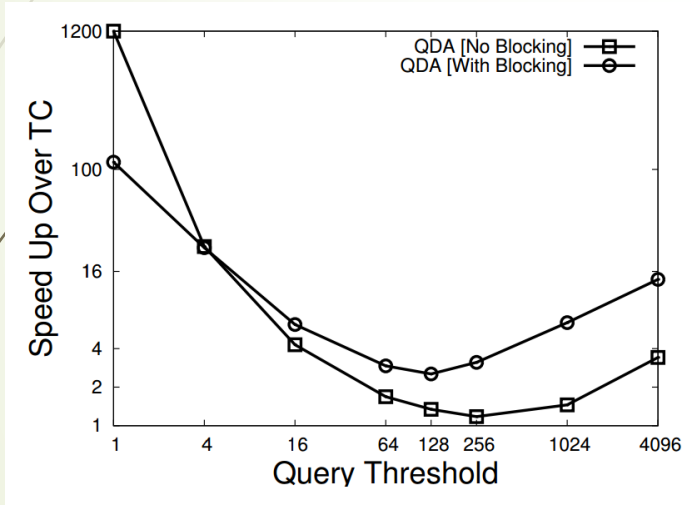
QDA speed-up for different types of queries

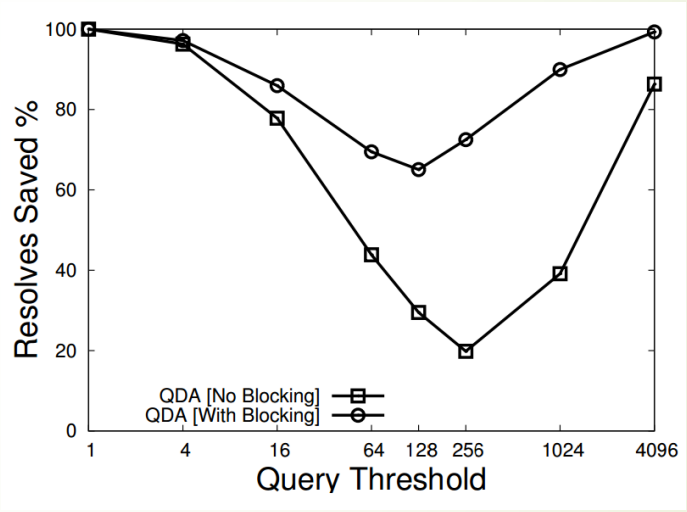


The effect of resolve function

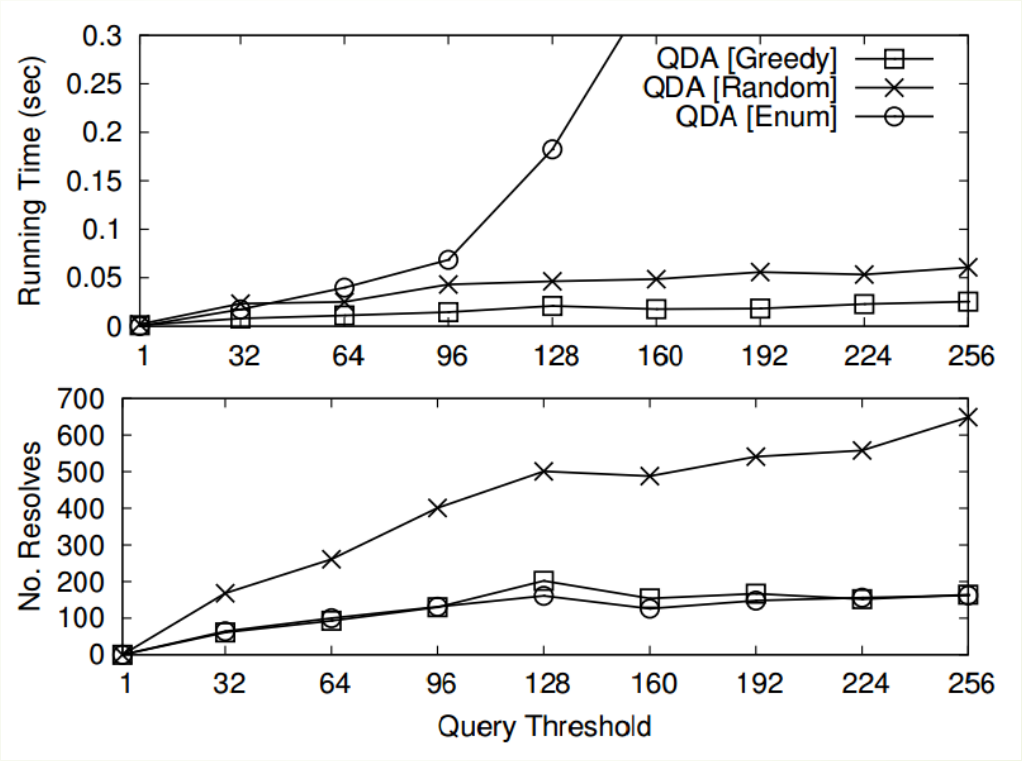


Combined with blocking…





Effect of Edge Picking



Thank you!
谢谢!
آپ کا شکریہ
Ευχαριστώ!
Teşekkürler!