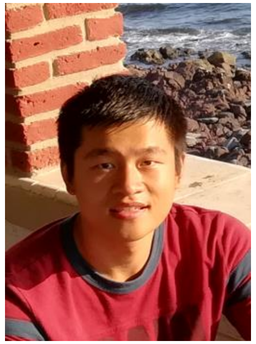
FIRST: Fast Interactive Attributed Subgraph Matching

Presented by: Boxin Du

Arizona State University



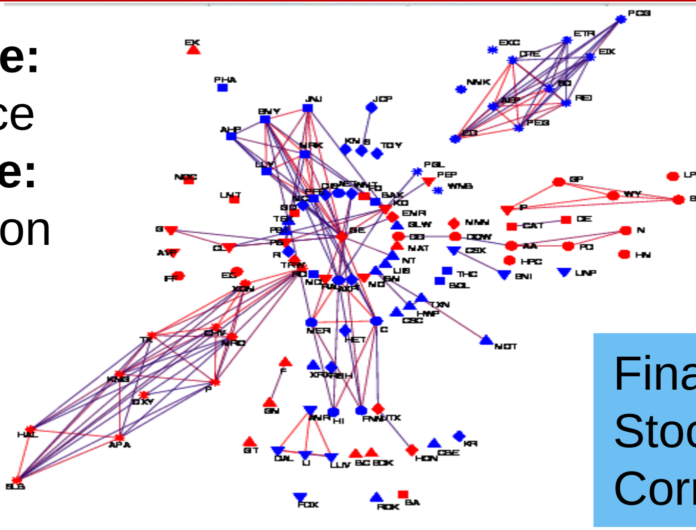
Boxin Du Si Zhang Nan Cao Hanghang Tong

Attributed Networks Are Everywhere

Node Attribute:
Age/gender/
interests…
Edge Attribute:
Friendship/like…

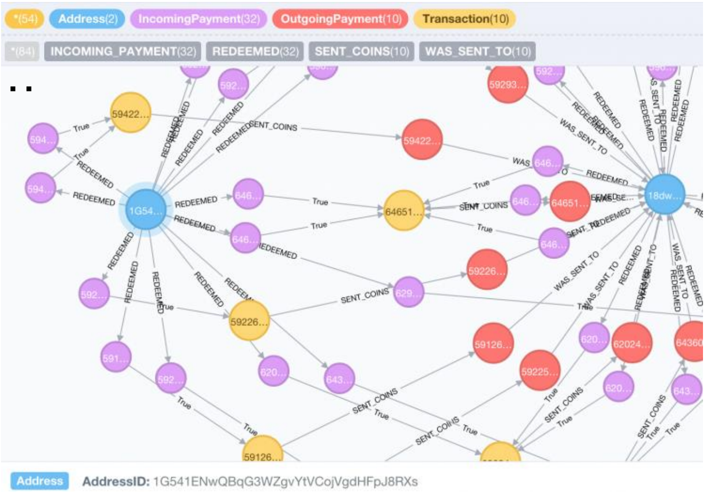
Node Attribute:
Education/skills/
occupation…
Edge Attribute:
Follow/inmail…

Node Attribute:
Stock type/price
Edge Attribute:
Stock correlation



Financial:
Stock
Correlation

Node Attribute:
Address/payments…
Edge Attribute:
Transaction types

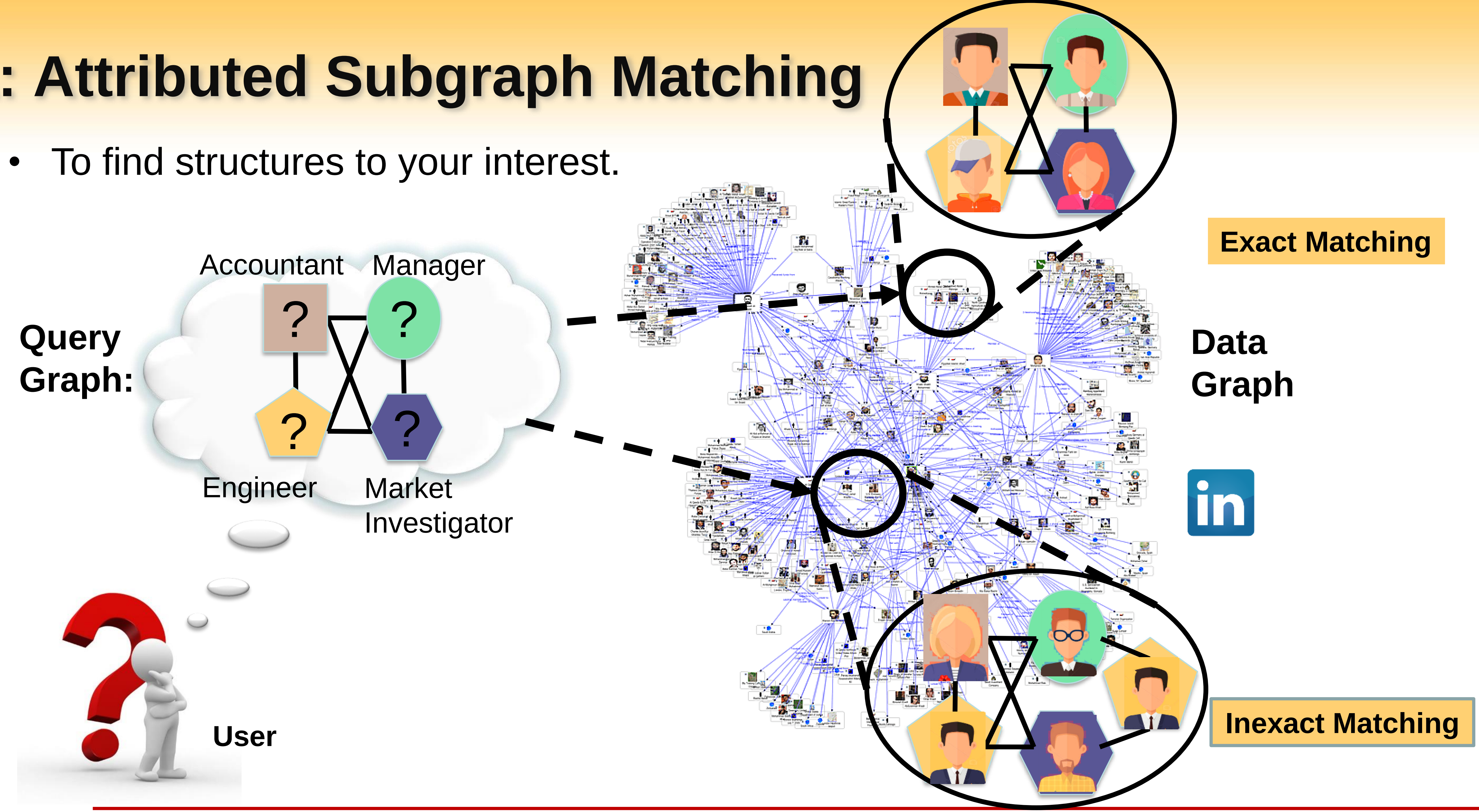


Bitcoin
Transaction

- **Q:** How to explore attributed networks?

A: Attributed Subgraph Matching

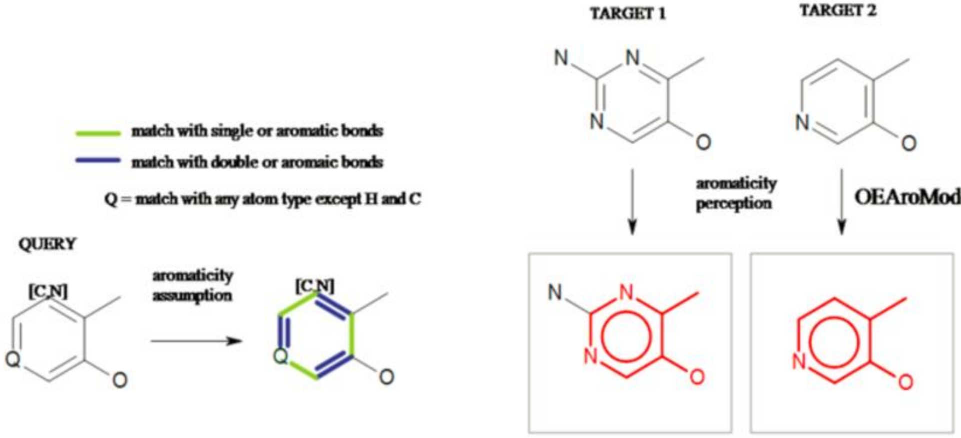
- To find structures to your interest.



DATA Lab

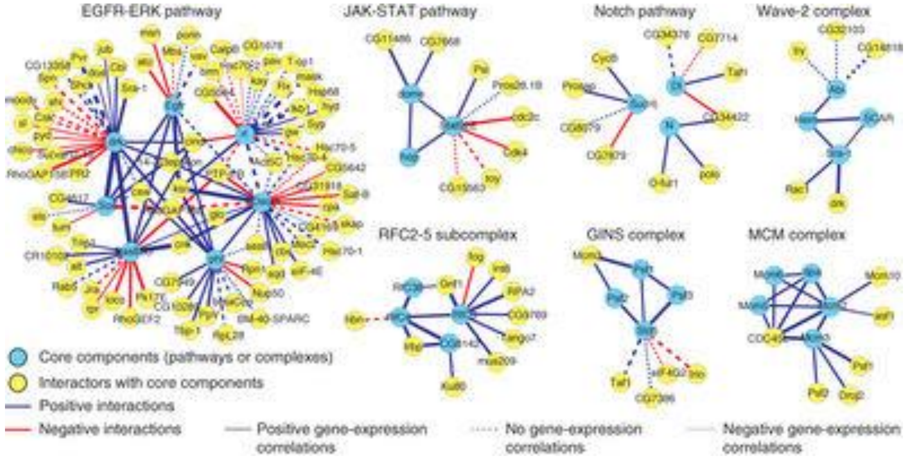
Attributed Subgraph Matching Applications

Finding Chemical Substructures/ Similarities [OEChem TK library]



Substructure -> chemical property

Protein-to-protein Interaction Network [A Vinayagam' 14]

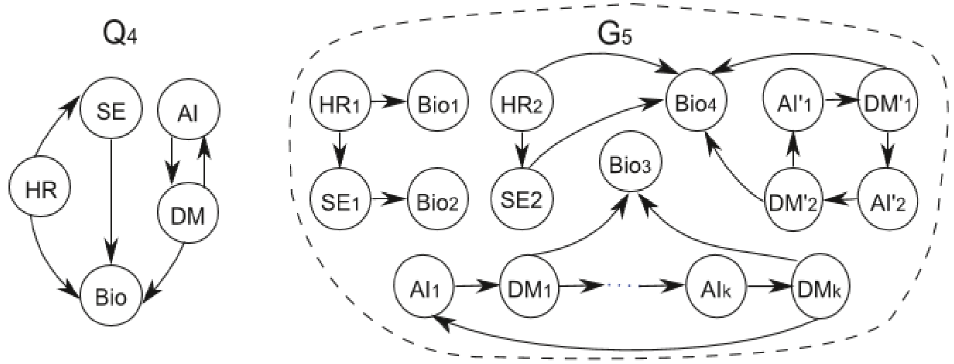


Pathways in PPI -> Protein function

Financial Fraud Pattern Detection [Neo4j]

Subgraph Pattern -> Fraud Pattern

Social Media Analysis [W Fan' 12]



Graph patterns -> social patterns

and so on…

Existing Methods for Attributed Subgraph Matching (MANY!)

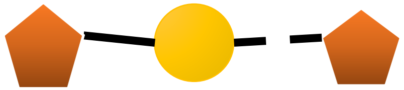
Algorithm	Author & Conference	Exact Matching	Inexact Matching	Node Attribute	Edge Attribute	Require no Index	Accurate query
R-WAG/I-WAG/S-WAG	S Roy et al. TKDE' 15	✓	✓	✓	✗	✓	✗
MAGE	R. Pienta et al. IEEE BigData' 14	✓	✓	✓	✓	✓	✗
NeMa	A. Khan et al. VLDB' 13	✓	✓	✓	✗	✓	✗
IncMatch	W. Fan et al. SIGMOD'11	✓	✗	✓	✗	✓	✗
SIGMA	M.Mongiovi et al. CSB' 09	✗	✓	✓	✗	✗	✗
TALE	Y. Tian et al. ICDE' 08	✓	✓	✓	✓	✗	✗
G-Ray	H. Tong et al. KDD' 07	✓	✓	✓	✗	✓	✗
…	…	…	…	…	…	…	…

- **Obs:** The User Needs to Provide the **Accurate** Query Graph.
- **Q:** What if the user does not know exactly what s/he is looking for?

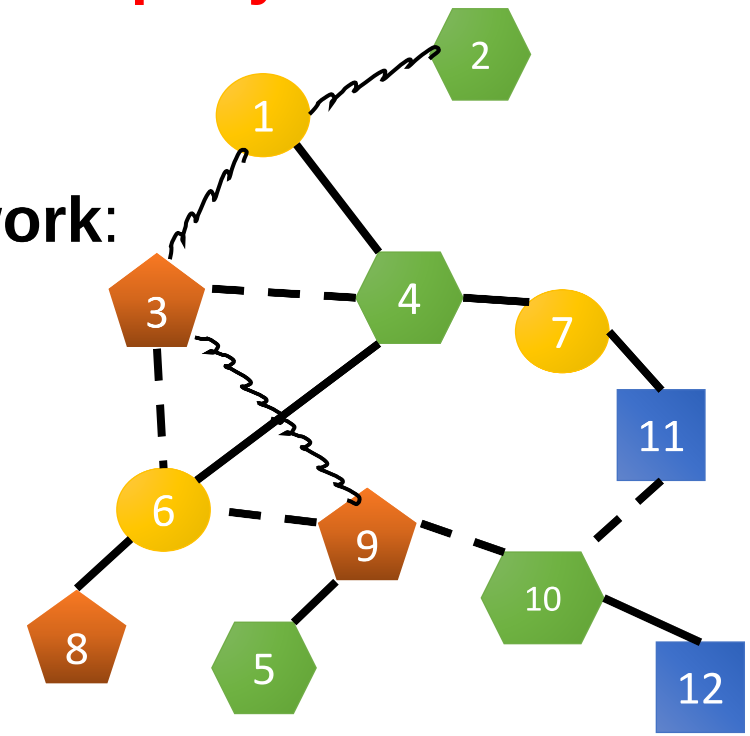
Interactive Attributed Subgraph Matching

- An illustrative example:
  - **Given:**
    - a social network with node attributes and edge attribute;
    - an initial query with attributes;
  - **Find:** the best matching subgraph(s) with **query revision on-the-fly**.

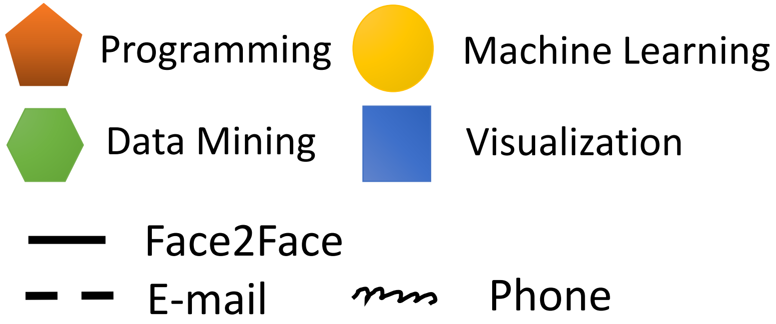
Initial Query:



Data Network:

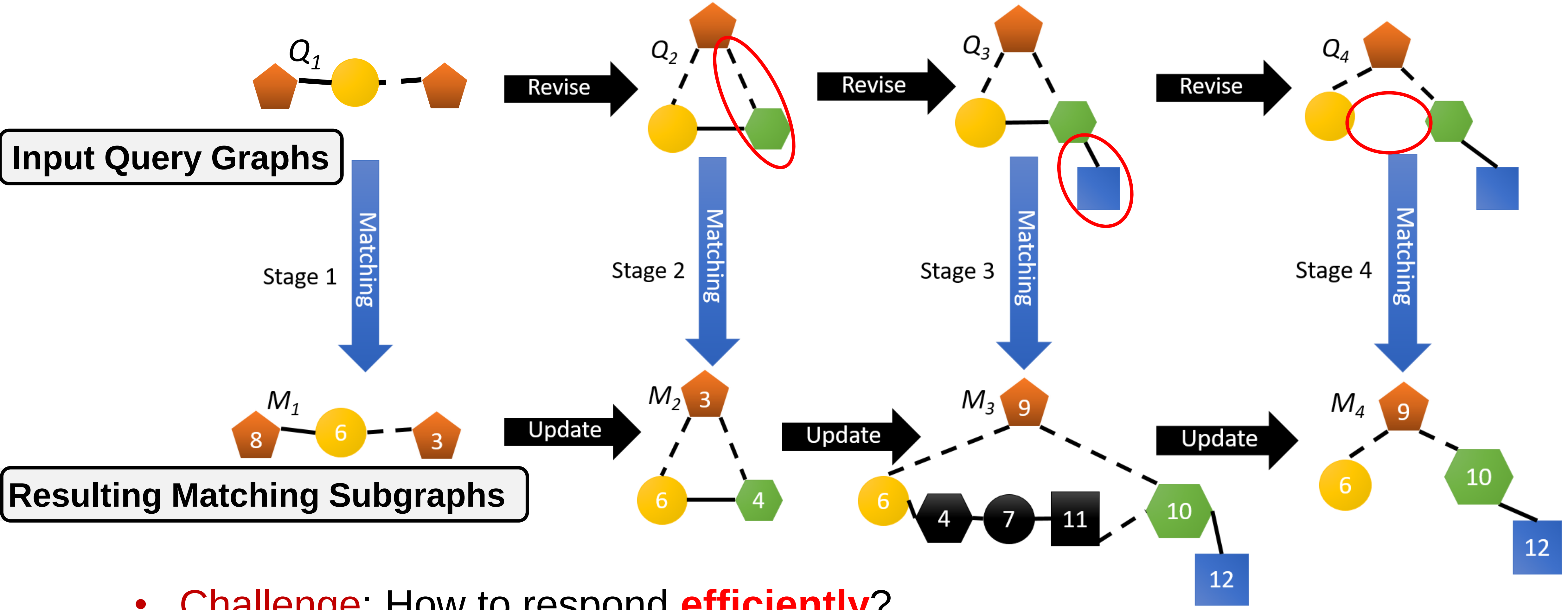


Attributes:



Interactive Attributed Subgraph Matching (cont'd)

- An illustrative example (cont'd): Revising and matching process.



- Challenge: How to respond **efficiently**?
  - w/o re-running algorithm or re-building Graph indexes

Roadmap

- Motivation
- Problem Definition
- Proposed Solution: FIRST family
- Experiments
- Conclusions

Problem Definition

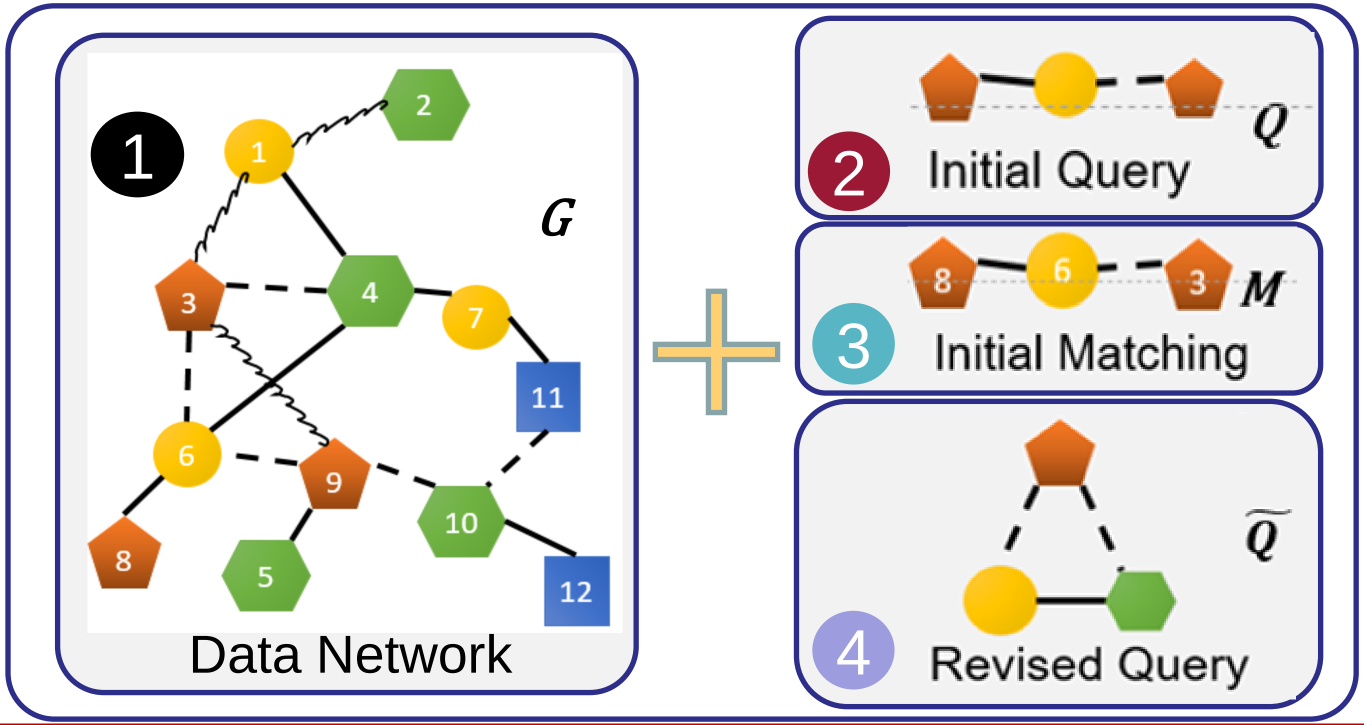
Given:

1. an undirected attributed network $G = (A,\ N_A, E_A)$, $[A\text{: } n\ \times n]$
2. an undirected initial query graph $Q = \left(A_q, N_q, E_q\right)$, $[A_q\text{: } k\ \times k]$
3. the initial matching graph M
4. the revised query graph $\tilde{Q}$;

Output:
the updated matching subgraph $\widetilde{M}$.

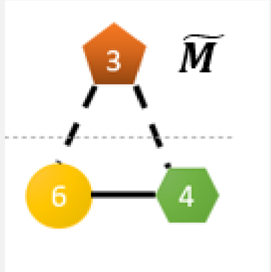
Input:

n=12



k=3

Output:



Roadmap:

- Motivation
- Problem Definition
- Proposed Solution: FIRST family
  - Key ideas
  - Details
- Experiments
- Conclusions

Key ideas

- Key Idea #1: Matching as cross-network node similarity
- Potential Benefit:
  - Encodes both topology and attribute during matching
  - Major computation: Sylvester equation

- Key Idea #2: Explore the smoothness of query graphs
- Potential Benefit:
  - View the revised query as a perturbation of previous query
  - Incrementally solve Sylvester equation for fast response

Key Idea #1: Matching as cross-network node similarity

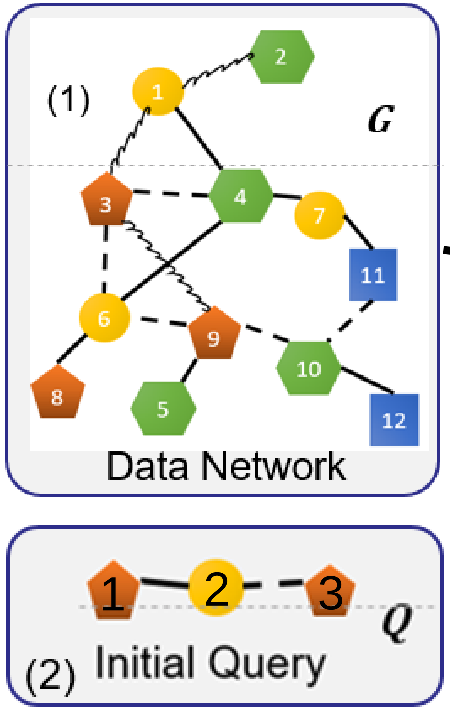
- Step 1: Find Similarity Matrix (**S**): *FIRST-Q/N/E*
  - **Intuition:** cross-network node similarity [Zhang, et al KDD'16]
  - **Major Computation:** to solve the Sylvester Equation

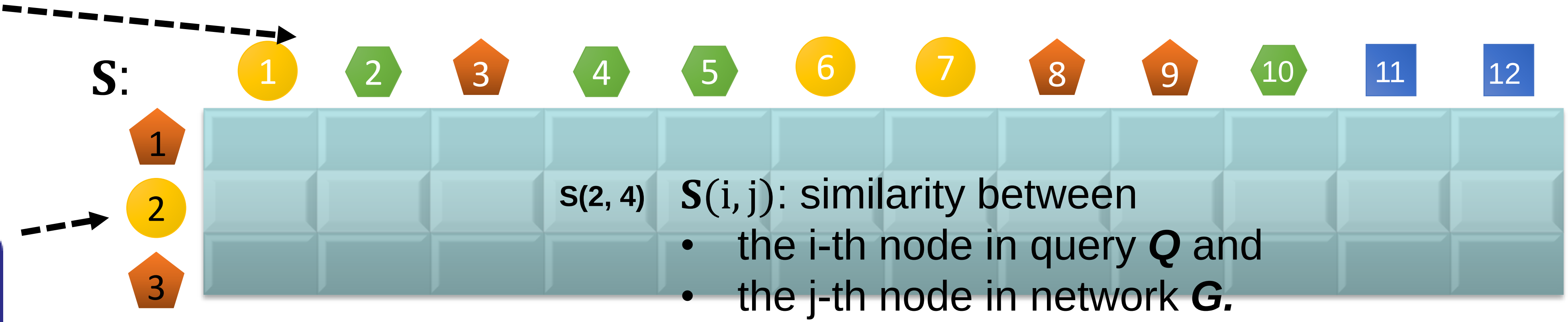
$$\mathbf{s} = \alpha \mathbf{W} \mathbf{s} + (1 - \alpha)\mathbf{h}$$

$\mathbf{W}$: Kronecker graph of ***G*** and ***Q*** (filtered by node/edge attribute);

$\mathbf{s}$: Vectorized similarity matrix $\boldsymbol{S}$;

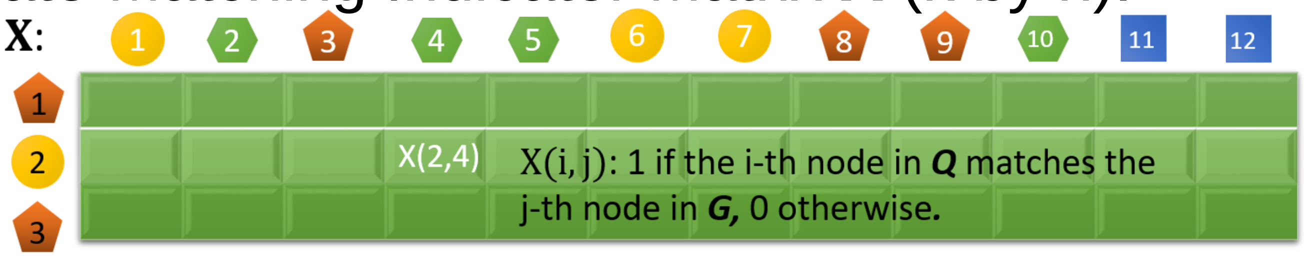
$\mathbf{h}$: Vectorized preference matrix $\mathbf{H}$.





Key Idea #1 (cont'd):

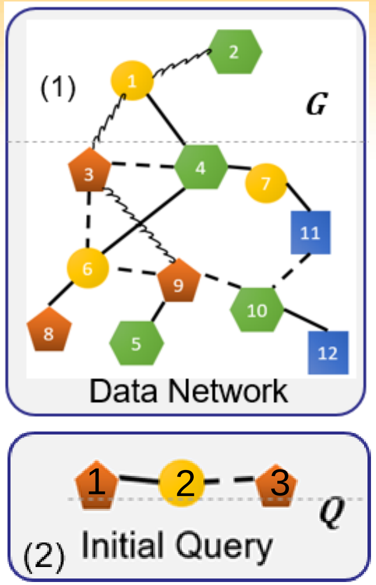
- Step 2: Find Matching Subgraph: *Sim2Sub*
  - **Intuition:**
    1. From similarity matrix to permutation matrix **X**;
    2. From permutation matrix to subgraph.
  - **Major Computation:**
    - ➢ Calculate Matching Indicator Matrix **X** (k by n).



    - ➢ **How to**: Use 'goodness' function $g(\mathbf{X})$:

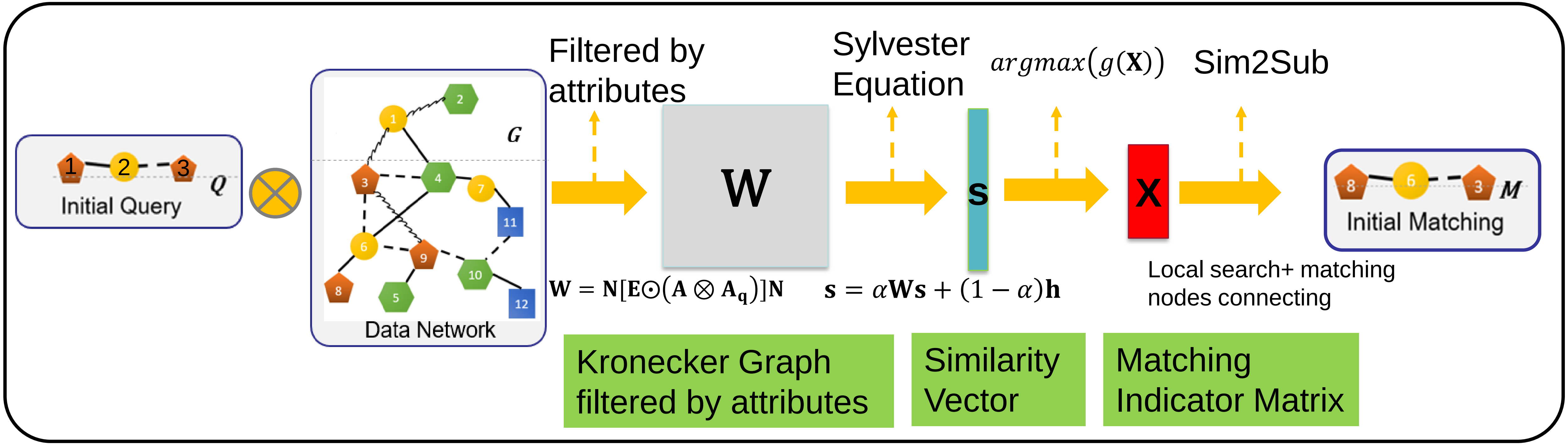
$$\mathbf{X}^* = argmax\big(g(\mathbf{X})\big)$$

$$= argmax[-\underbrace{\left|\left|\mathbf{XAX}' - \mathbf{A_q}\right|\right|_F^2}_{\text{Matching subgraph Connectivity}} + a * \underbrace{trace(\mathbf{SX}')}_{\text{Quality of individual matching nodes}} - b * \underbrace{\left|\left|\mathbf{XX}' - \mathbf{I}\right|\right|_F^2}_{\text{Permutation matrix}}]$$



Summary of steps in Key Idea #1

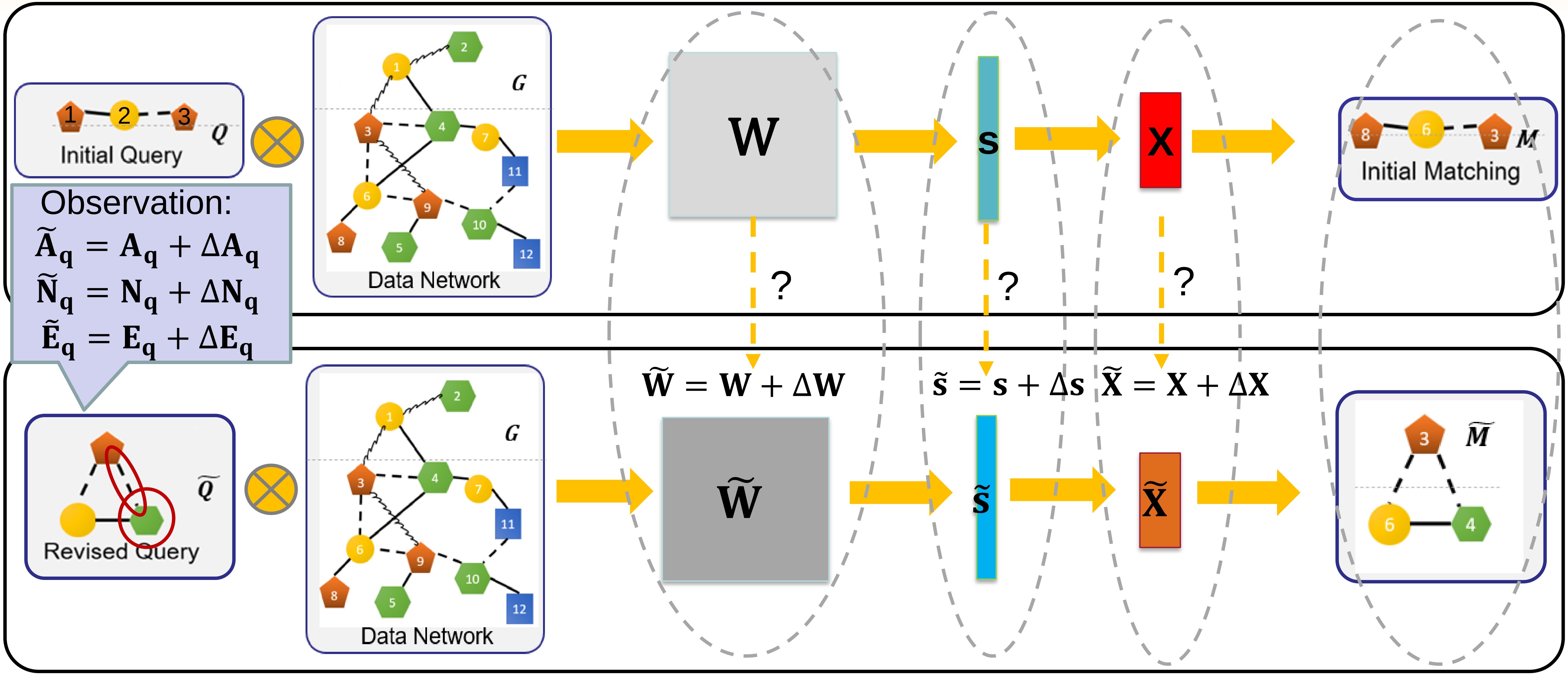
- Procedure: Input graphs ➡ cross-network similarity vector ➡ matching subgraph



Step 1: FIRST-Q/N/E

Step 2: Sim2Sub

Key Idea #2: Smoothness of query graphs



Details:

Scenarios	W
Topology only	$\mathbf{W} = \mathbf{A} \otimes \mathbf{A_q}$
Topology + node attribute	$\mathbf{W} = \mathbf{N}(\mathbf{A} \otimes \mathbf{A_q})\mathbf{N}$
Topology + node attribute + edge attribute	$\mathbf{W} = \mathbf{N}[\mathbf{E} \odot (\mathbf{A} \otimes \mathbf{A_q})]\mathbf{N}$

N: the node attribute matrix of input networks ($\boldsymbol{G} \,\&\, \boldsymbol{Q}$):

$\mathbf{N} = \sum_{p=1}^{K} \boldsymbol{N}_A^p \otimes \boldsymbol{N}_q^p$, K: the number of distinct node labels.

E: the edge attribute matrix of input networks ($\boldsymbol{G} \,\&\, \boldsymbol{Q}$):

$\mathbf{E} = \sum_{l=1}^{L} \boldsymbol{E}_A^l \otimes \boldsymbol{E}_q^l$, L: the number of distinct edge labels.

FIRST-Q: Handle Topology Revision

<u>**Scenario 1**</u>: During query revision, only graph topology is changed.

- **Goal**:

  Fast computation of similarity vector after topology revision.

- **Observation**:

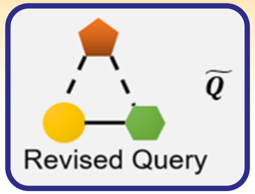
  1. We already have: $\mathbf{s} = \alpha \mathbf{W}\mathbf{s} + (1-\alpha)\mathbf{h}$, $\tilde{\mathbf{W}} = \mathbf{W} + \Delta\mathbf{W}$,
  2. The approximated similarity matrix:

$$\hat{\mathbf{s}} = (1-\alpha)(\mathbf{I} - \alpha\widehat{\mathbf{W}})^{-1}\mathbf{h}$$

- **Solution**:

  1. Calculate $\mathbf{W}$ in pre-computing stage, only $\Delta\mathbf{W}$ in interactive stage.
  2. Low-rank approx. & matrix inverse lemma for fast computation.

FIRST-Q: Handle Topology Revision

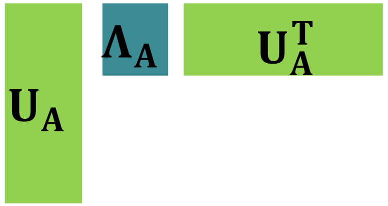


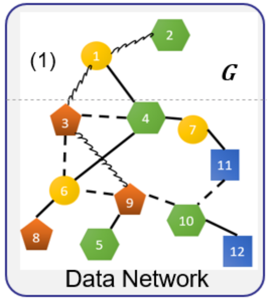
- **How to (Details):**

Pre-computing Stage:

1. Low-rank approximation for data network:

$$\mathbf{A} \approx \mathbf{U_A \Lambda_A U_A^T}$$

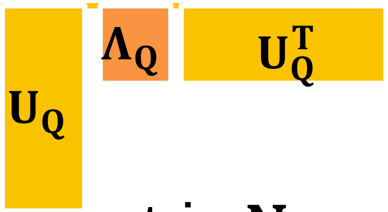




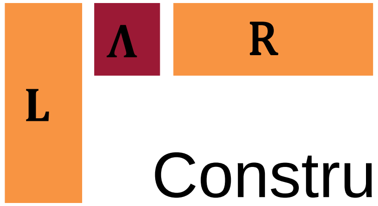
2. Store $\mathbf{U_A}$, $\mathbf{\Lambda_A}$.

Interactive Stage:

1. $\widetilde{\mathbf{A}}_\mathbf{q} \approx \mathbf{U_Q \Lambda_Q U_Q^T}$



2. Compute: node attribute matrix $\mathbf{N}$, diagonal degree matrix $\mathbf{D}$.
3. $\widehat{\mathbf{W}}$ =



Construct $\mathbf{L}$,$\mathbf{\Lambda}$,$\mathbf{R}$ from the approximation of $\mathbf{A}$ & $\widehat{\boldsymbol{A}}_{\boldsymbol{q}}$.

4. $$\hat{\mathbf{s}} = (1-\alpha)\mathbf{P}^{-1}\left[\mathbf{D_1}^{-1} + \alpha\mathbf{D_1}^{-1}\mathbf{L}\left(\mathbf{\Lambda}^{-1} - \alpha\mathbf{R}\mathbf{D_1}^{-1}\mathbf{L}\right)^{-1}\right]\mathbf{P}^{-1}\mathbf{h}$$

By Sherman-Morrison Lemma

- Result:
  - Time complexity: $O(r^2t^2kn + rtkn + K^2kn)$
  - Space complexity: $O(k^2rn + m_1)$

FIRST-N: Handle Node Attribute Revision

Scenario 2: During query revising, only node attribute is changed.

- **Goal**:

  Fast computation of similarity vector after node attribute revision.
- **Observation**:
  - We already have: $\hat{\mathbf{s}} = (1-\alpha)(\mathbf{I}-\alpha\widehat{\mathbf{W}})^{-1}\mathbf{h}, \widetilde{\mathbf{N}_q} = \mathbf{N}_q + \Delta\mathbf{N}$,
  - The topology keeps unchanged.
  - Low-rank approx. of both $\mathbf{A}$ and $\mathbf{A}_q$ (pre-compute).
- **Solution**:
  - Calculate $\mathbf{W}$ in pre-computing stage, only $\Delta\mathbf{W}$ in interactive stage.
  - Low-rank approx. & matrix inverse lemma for fast computation.

FIRST-N: Handle Node Attribute Revision

- **How to (Details):**

This can further speed up this algorithm compared to FIRST-Q.

<u>Pre-computing Stage:</u>

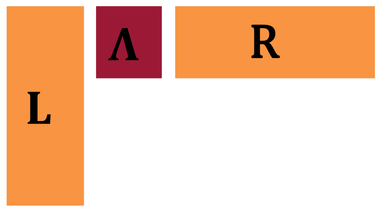
1. Low-rank approximation:

$$\mathbf{A} \approx \mathbf{U_A \Lambda_A U_A^T}$$

$$\mathbf{A}_q \approx \mathbf{U_q \Lambda_q U_q^T}$$

2. Construct approximated $\mathbf{W}$:

$\widehat{\mathbf{W}}$ =



3. Store $\mathbf{L}, \mathbf{R}, \mathbf{\Lambda}$.

<u>Interactive Stage:</u>

1. Major Computation: Compute $\mathbf{N}$, diagonal degree matrix $\mathbf{D}$ with $\widetilde{\mathbf{N}_\mathbf{q}}$.

2. Intermediate matrix:

$$\mathbf{P} = \mathbf{D}^{-1/2}\widetilde{\mathbf{N}_\mathbf{q}}, \mathbf{D_1} = \mathbf{P}^{-1}\mathbf{P}^{-1}.$$

3. Calculate similarity vector:

$$\hat{\mathbf{s}} = (1-\alpha)\mathbf{P}^{-1}\,[\mathbf{D_1}^{-1} + \alpha \mathbf{D_1}^{-1}\mathbf{L}\,(\mathbf{\Lambda}^{-1} - \alpha \mathbf{R}\mathbf{D_1}^{-1}\mathbf{L})^{-1}]\mathbf{P}^{-1}\mathbf{h}$$

By Sherman-Morrison Lemma

- Result:
  - Time complexity: $O(r^2t^2kn + rtkn + K^2kn)$
  - Space complexity: $O(k^2rn + m_1)$

FIRST-E: Handle Edge Attribute Revision

<u>**Scenario 3**</u>: During query revising, only edge attributes are changed.

- **Goal**:

  Fast computation of similarity vector after edge attribute revision.
- **Observation**:
  1. Pre-compute: The low-rank approximation of the <u>edge attributed</u> adjacency matrix $(\mathbf{E}_{\mathbf{A}}^{\mathbf{l}} \odot \mathbf{A})$;
  2. Interactive: Only approx. of the revised <u>edge attributed</u> adjacency matrix.
- **Solution keys:**
  1. Low-rank approximation;
  2. matrix inverse lemma;
  3. Block matrix property.

FIRST-E: Handle Edge Attribute Revision (cont'd)

- **How to (Details):**

Pre-computing Stage:

1. The edge attributed networks:

$\mathbf{E}_{\mathrm{A}}^{\mathrm{l}} \odot \mathbf{A} \approx$

This can further speed up computation.

2. Store $\mathbf{U}_{\mathrm{A}}^{l}, \mathbf{\Lambda}_{\mathrm{A}}^{l}$;

3. If the index **l'** of changed edge attribute is available:

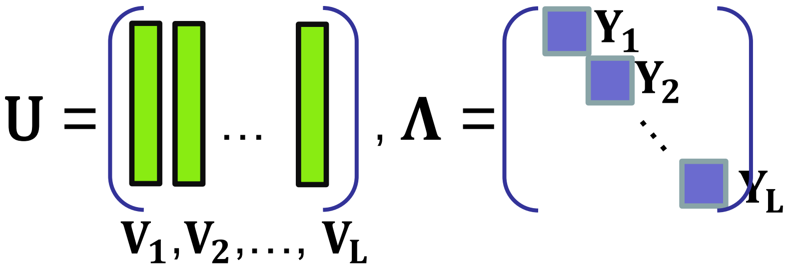
$\mathbf{E}_q^k \odot \mathbf{A}_q \approx \mathbf{U}_{\mathrm{q}}^{k} \mathbf{\Lambda}_{\mathrm{q}}^{k} (\mathbf{U}_{\mathrm{q}}^{k})^T$ (all k)

4. Store $\mathbf{U}_{\mathrm{q}}^{k}, \mathbf{\Lambda}_{\mathrm{q}}^{k}$;

Interactive Stage:

1. Compute: **N**, **D**. $\mathbf{E}_q^k \odot \mathbf{A}_q \approx \mathbf{U}_{\mathrm{q}}^{k} \mathbf{\Lambda}_{\mathrm{q}}^{k} (\mathbf{U}_{\mathrm{q}}^{k})^T$ ($k \in$ **l'**)

2. Construct block matrix **U**, **Λ**;



$\mathbf{V}_l = \mathbf{U}_{\mathrm{A}}^{l} \otimes \mathbf{U}_{\mathrm{q}}^{k}$ $\quad \mathbf{Y}_{\mathrm{j}} = \mathbf{\Lambda}_{\mathrm{A}}^{l} \otimes \mathbf{\Lambda}_{\mathrm{q}}^{k}$

3. $\hat{\mathbf{s}} = (1-\alpha)\left[\mathbf{I} + \alpha \mathbf{L}\left(\mathbf{\Lambda}^{-1} - \alpha \mathbf{R} \mathbf{L}\right)^{-1} \mathbf{R}\right]\mathbf{h}$;

$(\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{N}\mathbf{U}, \mathbf{R} = \mathbf{U}^{\mathrm{T}}\mathbf{N}\mathbf{D}^{-1/2})$.

- Time complexity: $O(r^2t^2kn + Lrtkn + K^2Lkn)$
- Space complexity: $O(Lrtkn + m_1)$

Roadmap:

- Motivation
- Problem Definition
- Proposed Solution: FIRST family
  - Key ideas
  - Three scenarios
- Experiments
- Conclusions

Experimental Setup

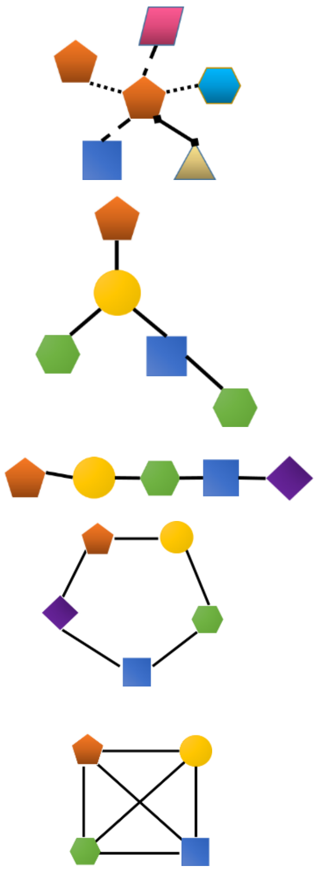
- Datasets Summary

Name	# of Nodes	# of Edges	Node/Edge Attribute
DBLP	9,143	16,338	Node attribute only
Flickr	12,974	16,149	Node attribute only
LastFm	136,421	1,685,524	Node attribute only
ArnetMiner	1,274,360	4,756,194	Node & Edge attribute
LinkedIn	6,726,290	19,360,690	Node attribute only

- Baseline methods:
  - G-ray [Tong et al. KDD' 07]
  - MAGE [Pienta et al. IEEE BigData' 14]
  - FINAL (and its variants) [Zhang et al. KDD'16]

Experiment Result - Effectiveness (Nodes)

Query samples:



% Exact Matching Nodes (Higher is Better)

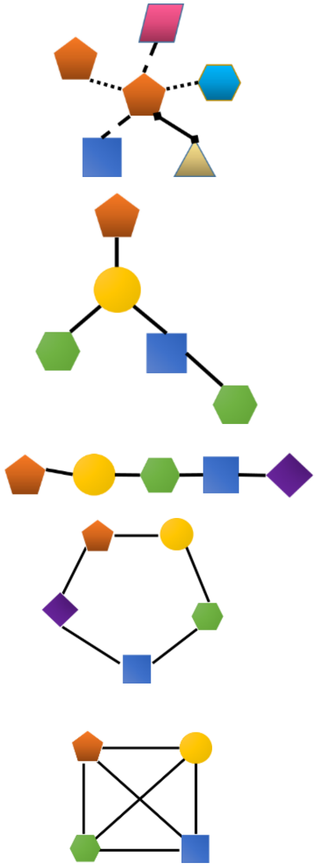
Algorithms	G-Ray	MAGE	FIRST
Star(N)	37.5	*	**75.0**
E-Star(N)	**83.3**	*	71.4
Line(N)	50.0	*	**83.3**
Loop(N)	27.3	*	**71.4**
Clique(N)	25.0	*	**57.1**
Star(NE)	*	30.0	**40.0**
E-Star(NE)	*	33.3	**41.7**
Line(NE)	*	33.3	**62.5**
Loop(NE)	*	27.3	**33.3**
Clique(NE)	*	60.0	**66.7**

* = Not applicable

Observation: Generally FIRST family generates more accurate results.

Experiment Result - Effectiveness (Nodes)

Query samples:



% of Extra Nodes (Lower is better)

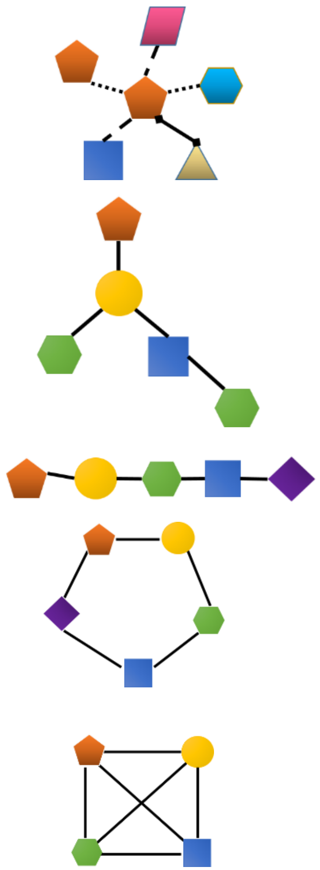
Algorithms	G-Ray	MAGE	FIRST
Star(N)	62.5	*	**0.0**
E-Star(N)	**0.0**	*	**0.0**
Line(N)	50.0	*	**0.0**
Loop(N)	**0.0**	*	**0.0**
Clique(N)	25.0	*	**0.0**
Star(NE)	*	50.0	**0.0**
E-Star(NE)	*	**0.0**	**0.0**
Line(NE)	*	33.3	**0.0**
Loop(NE)	*	**27.3**	44.4
Clique(NE)	*	40.0	**0.0**

* = Not applicable

Observation: Generally FIRST family generates more accurate results.

Experiment Result - Effectiveness (Edges)

Query samples:



% Exact Matching Edges (Higher is better)

Algorithm	G-Ray	MAGE	FIRST
Star(N)	33.3	*	**57.1**
E-Star(N)	**60.0**	*	50.0
Line(N)	40.0	*	**60.0**
Loop(N)	8.3	*	**42.9**
Clique(N)	7.1	*	**12.5**
Star(NE)	*	0.0	**14.3**
E-Star(NE)	*	7.1	**9.0**
Line(NE)	*	0.0	**14.3**
Loop(NE)	*	0.0	**14.3**
Clique(NE)	*	0.0	**12.5**

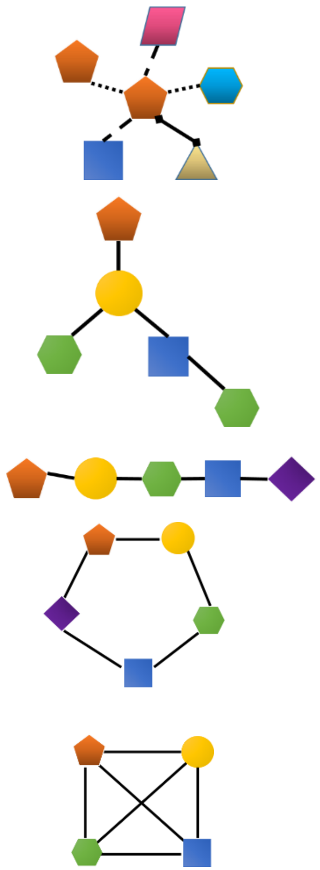
* = Not applicable

Observation: Generally FIRST family generates more accurate results.

DATA Lab

Experiment Result - Effectiveness

Query samples:



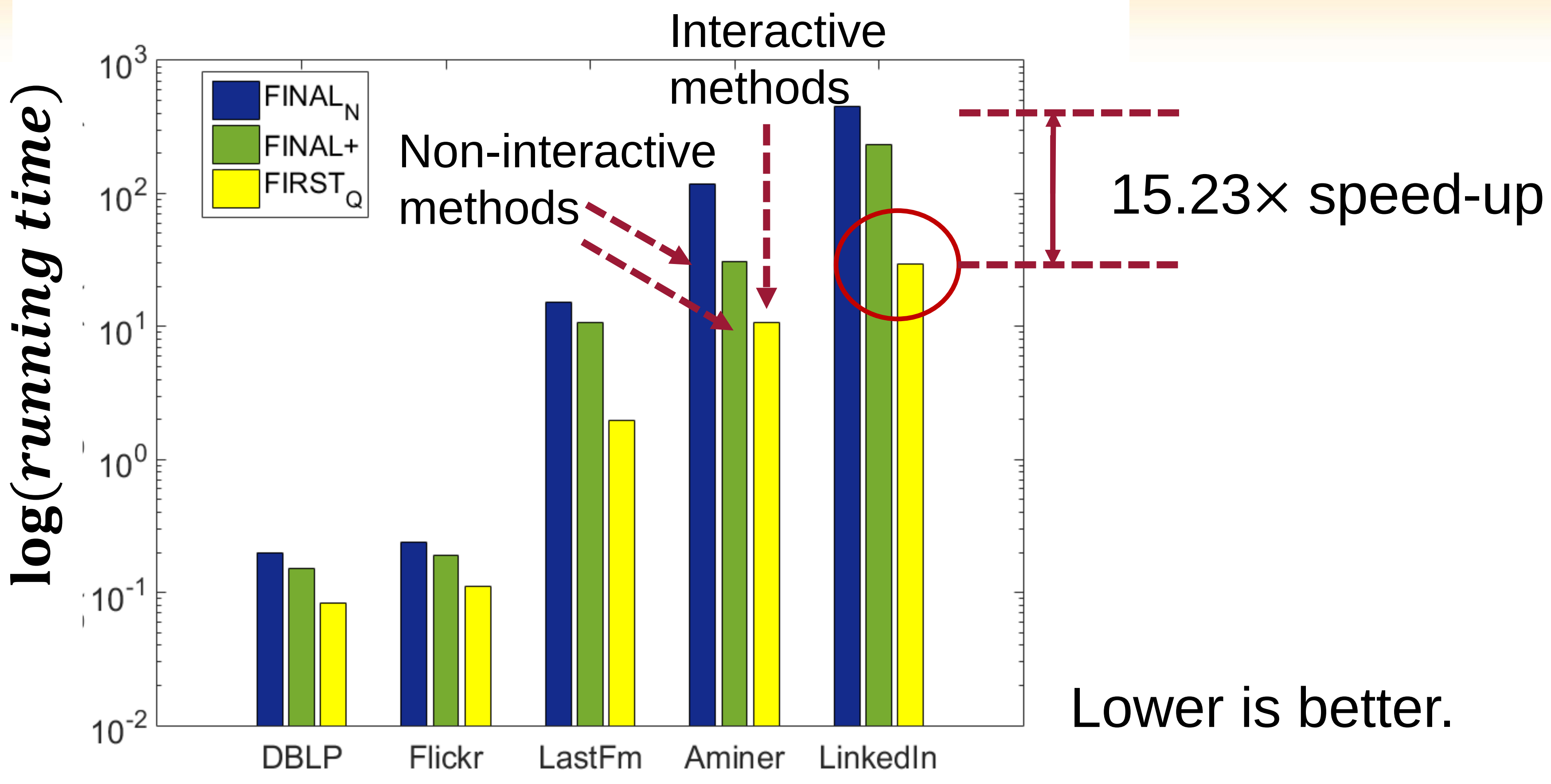
% Extra Matching Edges (Lower is better)

Algorithm	G-Ray	MAGE	FIRST
Star(N)	66.7	*	**0.0**
E-Star(N)	**0.0**	*	**0.0**
Line(N)	60.0	*	**0.0**
Loop(N)	8.3	*	**0.0**
Clique(N)	35.7	*	**0.0**
Star(NE)	*	42.9	**0.0**
E-Star(NE)	*	**0.0**	**0.0**
Line(NE)	*	27.3	**0.0**
Loop(NE)	*	**30.0**	42.9
Clique(NE)	*	33.3	**0.0**

* = Not applicable

Observation: Generally FIRST family generates more accurate results.

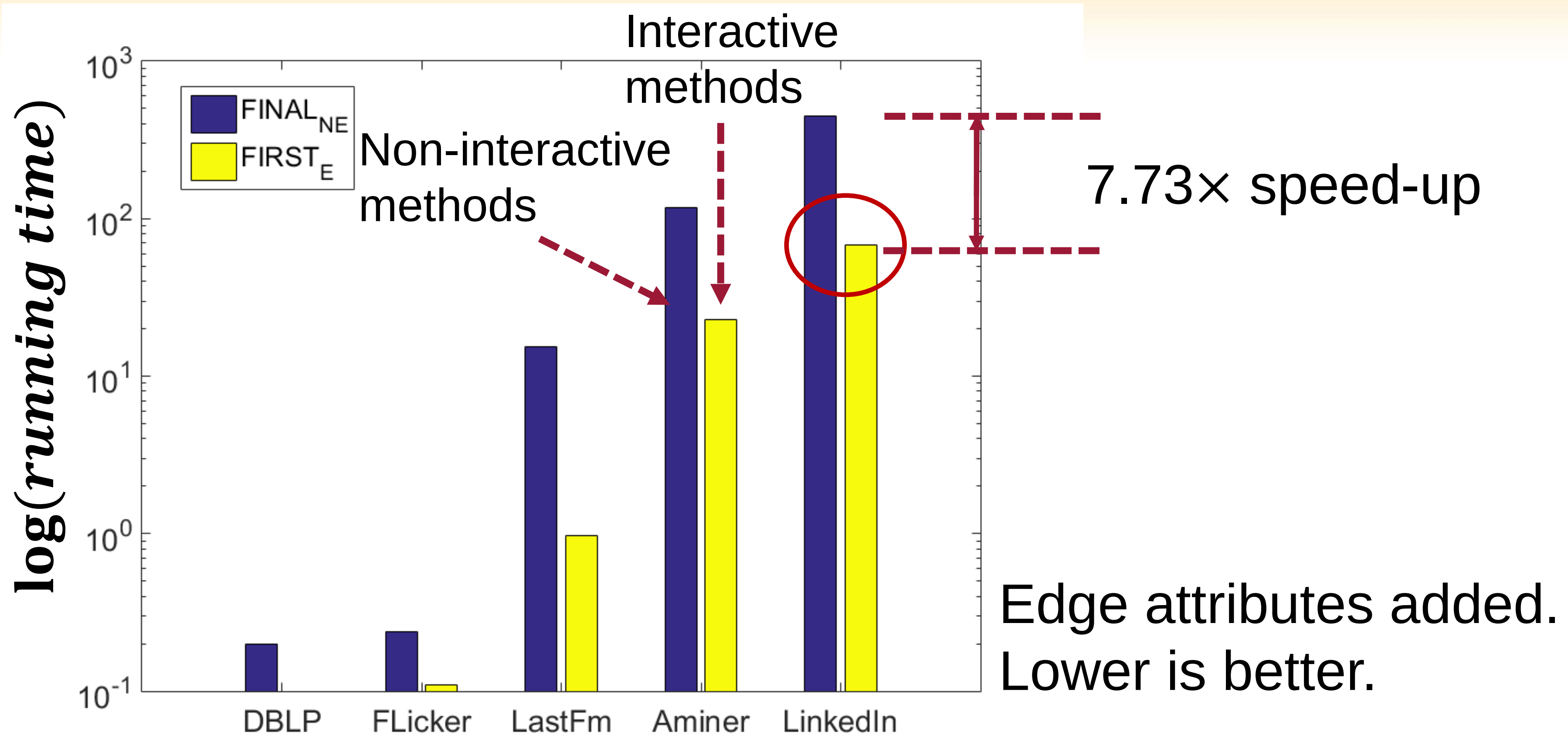
Experiment Result - Efficiency



Lower is better.

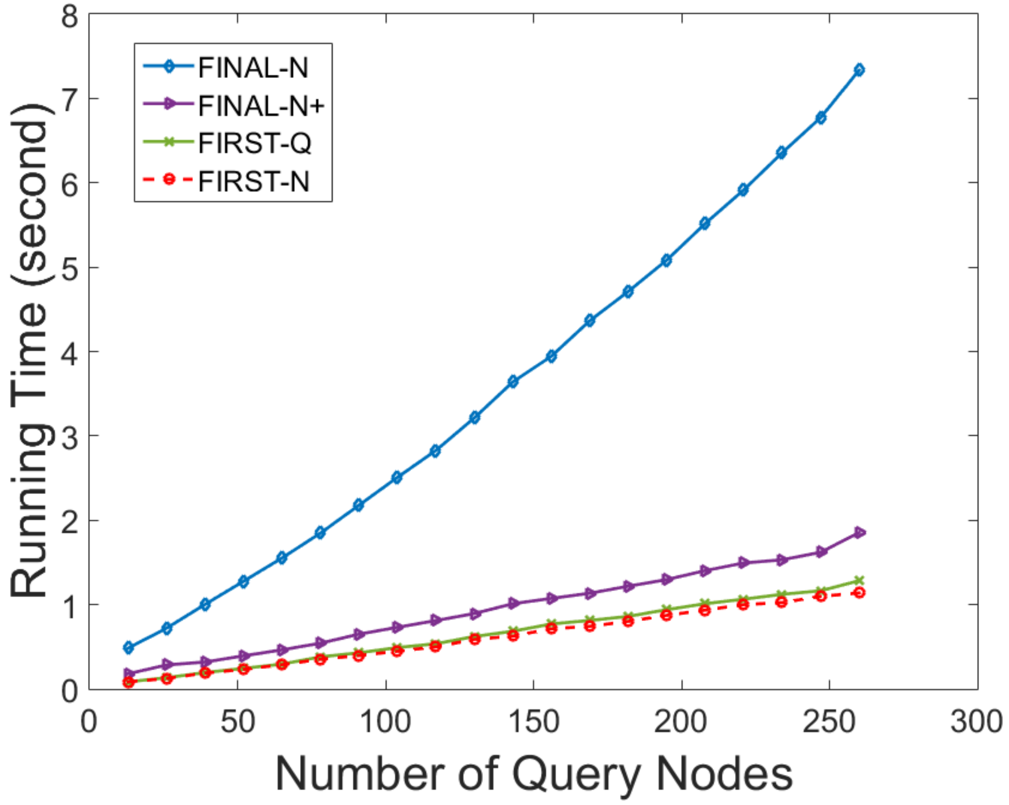
- Observation: >15× speedup with 6,726,290-node data network.

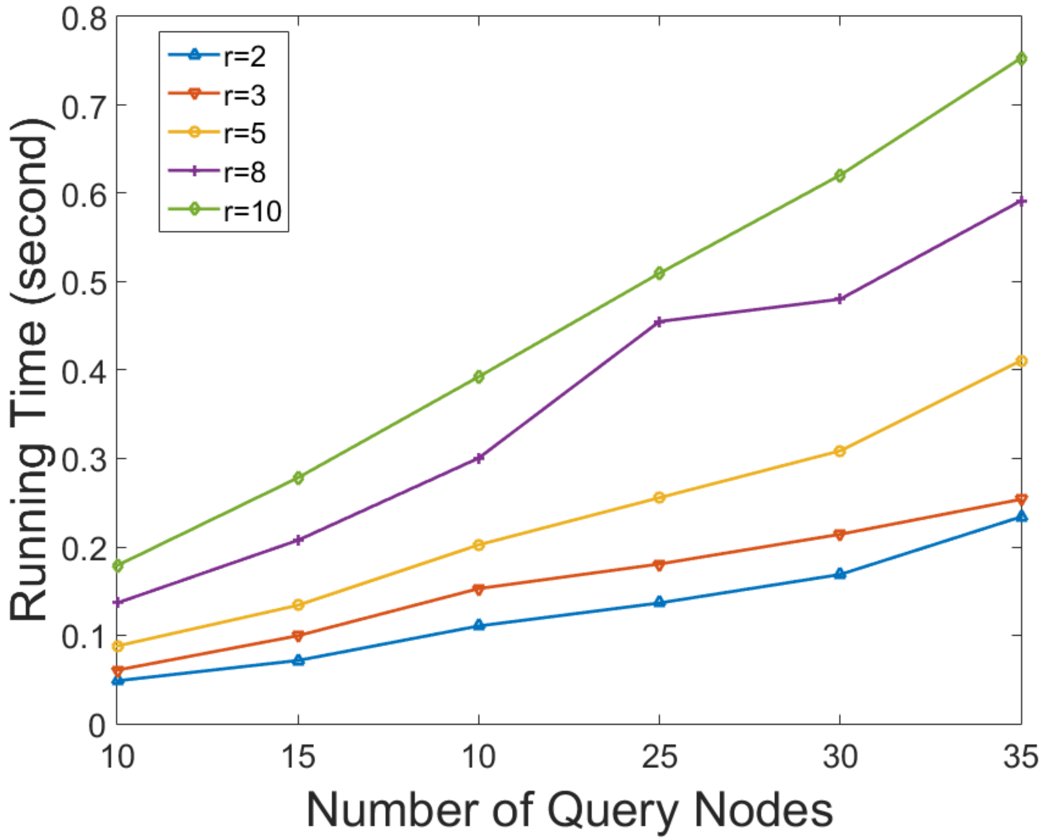
Experiment Result- Efficiency



- Observation: FIRST family is more efficient than baseline methods.

Experiment Result- Efficiency



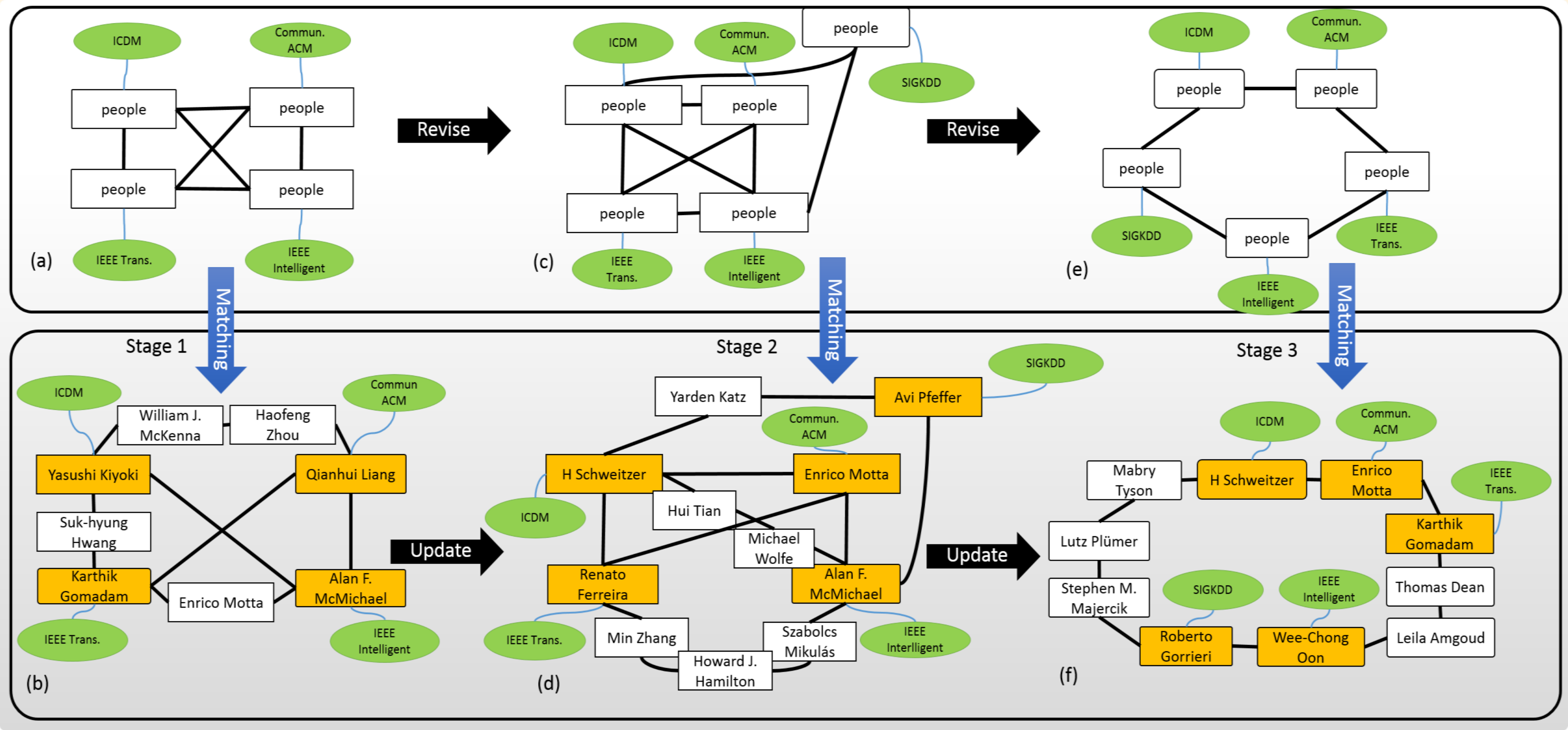


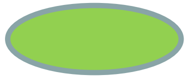
- Observation: FIRST family scales linearly with regard to size of query graph.

Experiments – Case Studies (on DBLP)

Query graphs:

Matching subgraphs:




Node attribute (conferences)

people Query node

Name Extra/intermediate node (in matching subgraph)

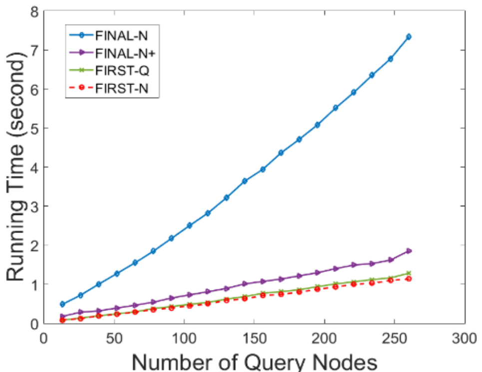
Name Matching node

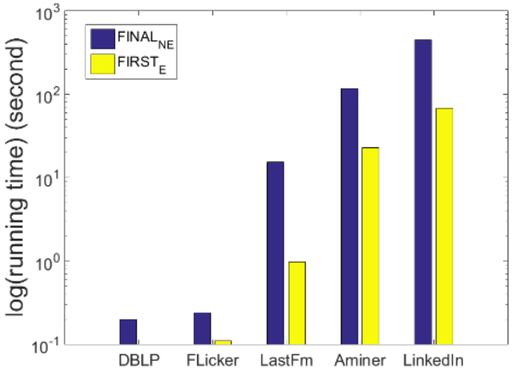
Roadmap:

- Motivation
- Problem Definition
- Proposed Solution: FIRST family
  - Key ideas
  - Three scenarios
- Experiments
  - Setup
  - Results
  - Case Study
- Conclusions

Conclusions

- **Goal**: **Efficient** Methods for **Interactive** Attributed Subgraph Matching.
- **Solution**: **FIRST family**
  - Key Idea #1: Subgraph matching as cross-network node similarity
  - Key Idea #2: Explore the smoothness of query graphs
- **Results:**
  - Linear scalability w.r.t the size of data network/query;
  - Better quality of matching subgraph against baselines.





	% Exact Matching Nodes		
Algorithm	G-Ray	MAGE	FIRST
Star(N)	37.5	*	**75.0**
E-Star(N)	**83.3**	*	71.4
Line(N)	50.0	*	**83.3**
Loop(N)	27.3	*	**71.4**
Clique(N)	25.0	*	**57.1**
Star(NE)	*	30.0	**40.0**
E-Star(NE)	*	33.3	**41.7**
Line(NE)	*	33.3	**62.5**
Loop(NE)	*	27.3	**33.3**
Clique(NE)	*	60.0	**66.7**