CPT-S 415

Big Data

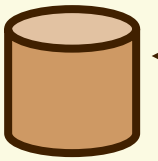
Yinghui Wu

EME B45

Why approximate query processing

Information Systems

"Big Data"



Query

Exact Answer

Long Response Times!

- ✓ Exact answers **NOT** always required
  - *Aggregate queries:* precision to "last decimal" not needed
    - e.g., "What percentage of the US sales are in NJ?" (display as bar graph)
  - *Preview* answers while waiting. *Trial* queries
  - Base data can be *remote or unavailable:* approximate processing using locally-cached *data synopses*

CPT-S 415
Big Data

Approximate query processing

- ✓ Overview
- ✓ Query-driven approximation
  - case study: graph pattern matching
- ✓ Data-driven approximation (next lecture)
  - Histogram, Sampling, Wavelet, Sketches
  - View-based evaluation (next next lecture)

Approximate query processing: overview

Case 1: Approximate Aggregation

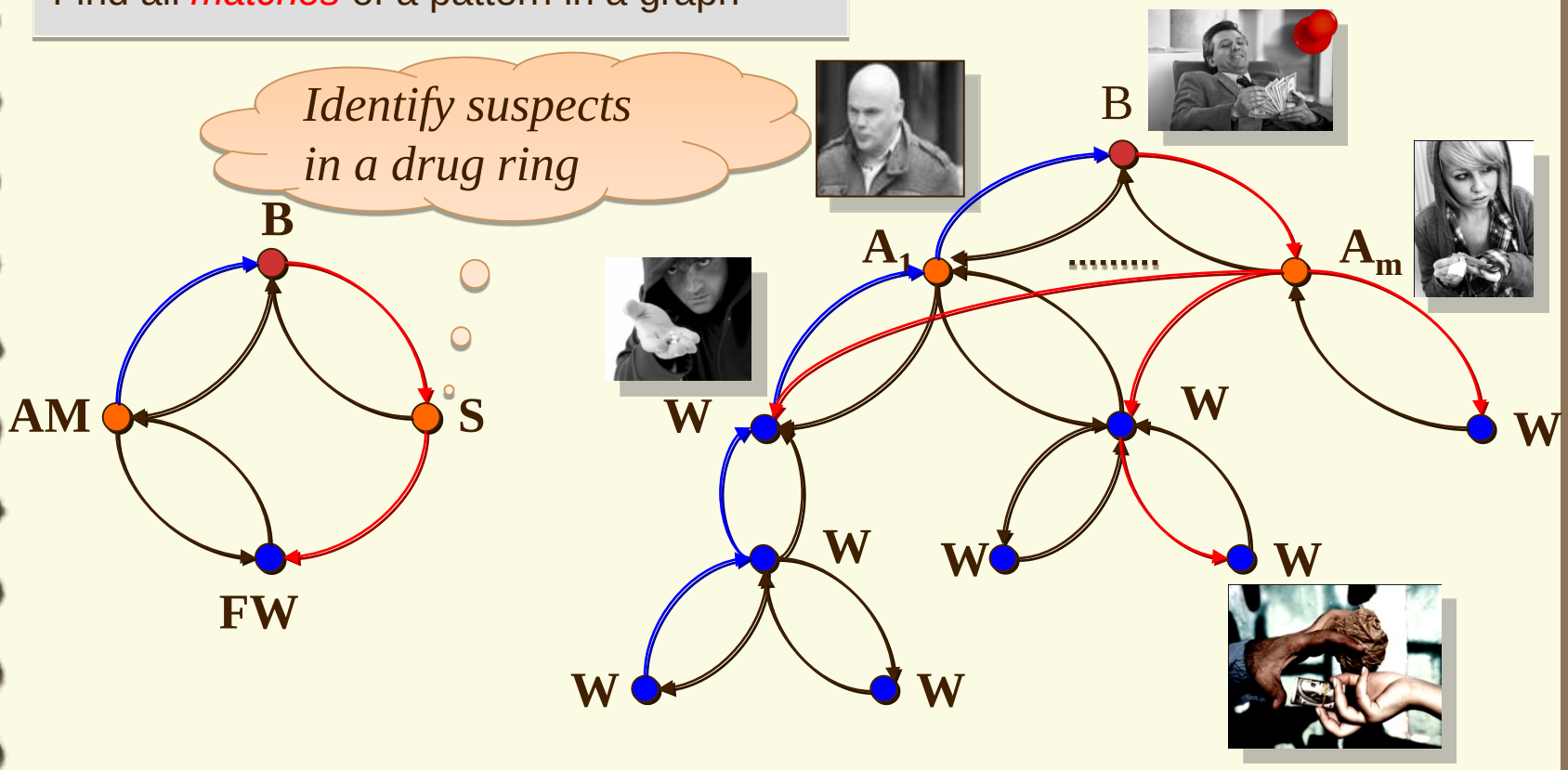
- ✓ *Aggregate Queries*
- ✓ Goal is to quickly report the leading digits of answers
  - In seconds instead of minutes or hours
  - Most useful if can provide error guarantees

> E.g., Average salary
>
> $59,000 +/- $500 (with 95% confidence) in 10 seconds
>
> vs. $59,152.25 in 10 minutes

- ✓ Achieved by answering the query based on samples or other synopses of the data
- ✓ Speed-up obtained because synopses are orders of magnitude smaller than the original data

Case 2: Pattern Matching in Social Graphs

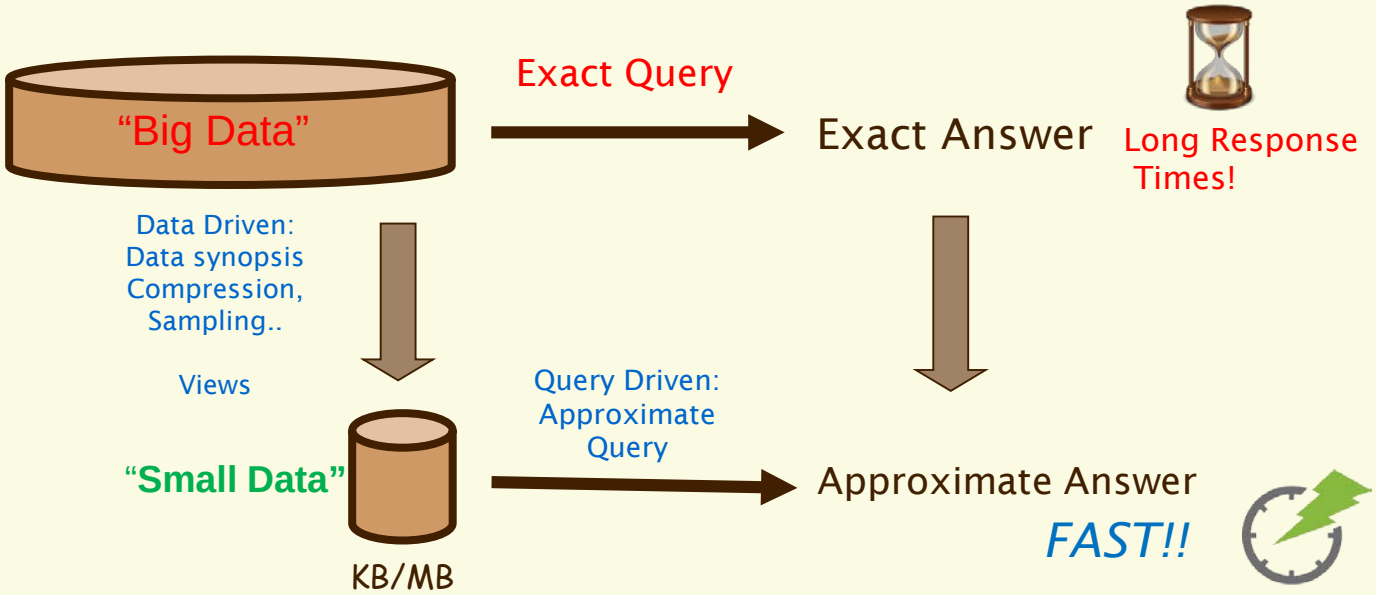
Find all *matches* of a pattern in a graph



"Understanding the structure of drug trafficking organizations"

Make case for computationally efficient queries

- ✓ Approximate query answering
  - Query-driven approximation
    - Rewrite queries to computationally efficient query classes
  - Data-driven approximation
    - Compact Data Synopses, materialized views, compression, summaries, sketches, spanners..
    - resource-bounded query answering



Query-driven approximation:

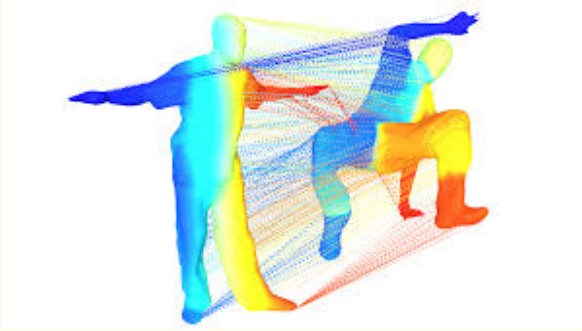
graph pattern matching

Graph Pattern Matching

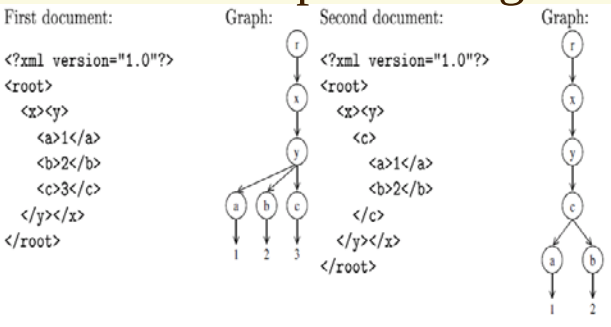
- ✓ Definition
  - Input: a pattern graph P, a data graph G, matching semantics
  - Output: correspondence from P to G
    - matching relation/function
    - matched noes/edges/subgraphs
- ✓ A “special case” of general graph matching. Difference? semantic of P (as a graph or a pattern)
- ✓ Variants of graph (pattern) matching
  - small pattern vs. large graph matching
  - single data graph vs. multiple graphs
  - rich semantics vs. simple label equality
  - flexible matching semantics vs. strict matching functions
  - approximate matching vs. exact alignment

Graph Pattern Matching: Manifold application areas

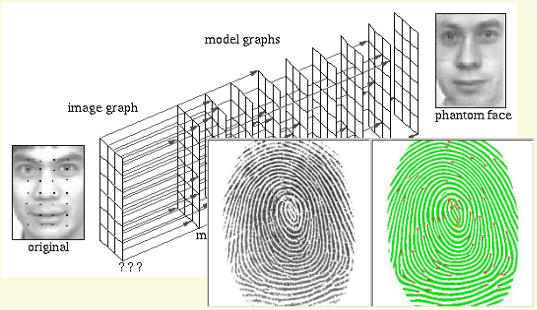
2D and 3D image analysis



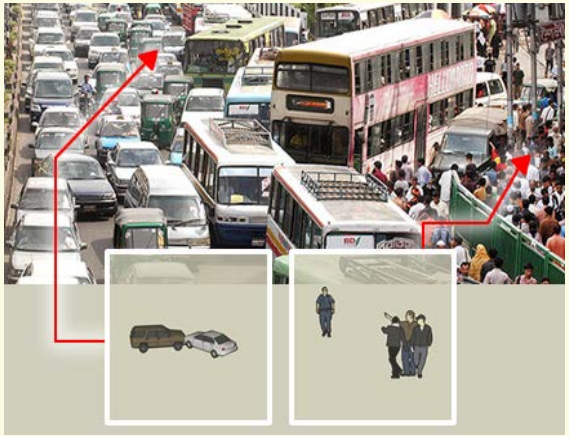
Document processing



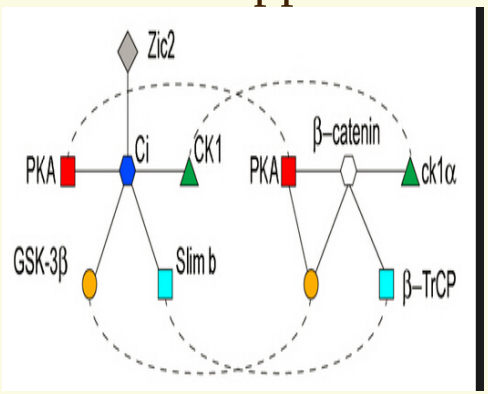
Biometric identification



Video analysis



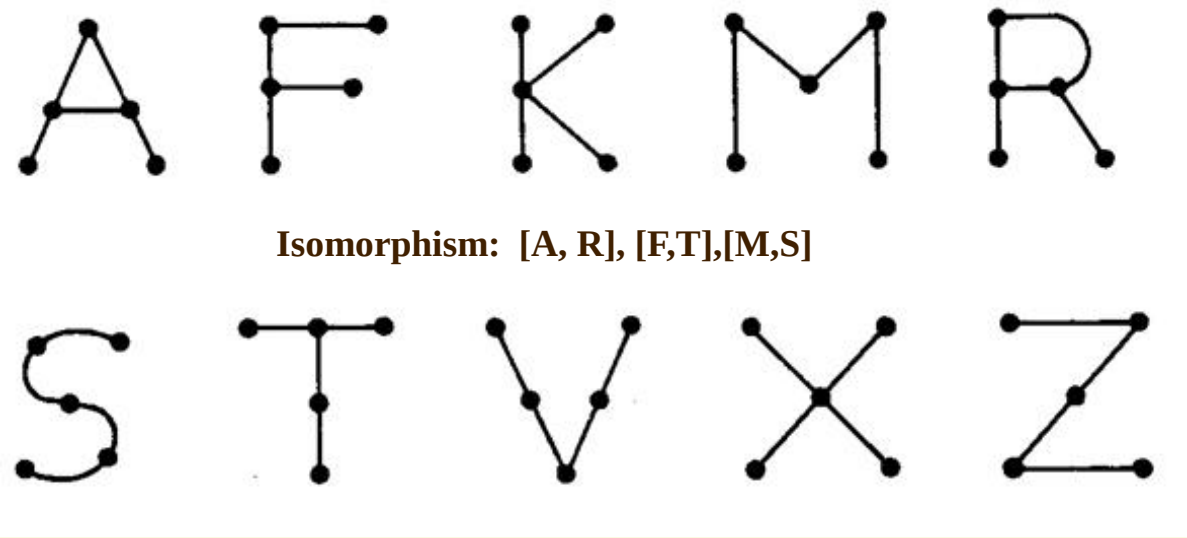
Biological and biomedical applications



Subgraph isomorphism

A function f from the nodes of Q to the nodes of G:

- ✓ For each node u in Q, u and f(u) have the same label;
- ✓ There exists an edge (u, u') in Q if and only if there exists an edge (f(u), f(u')) in G



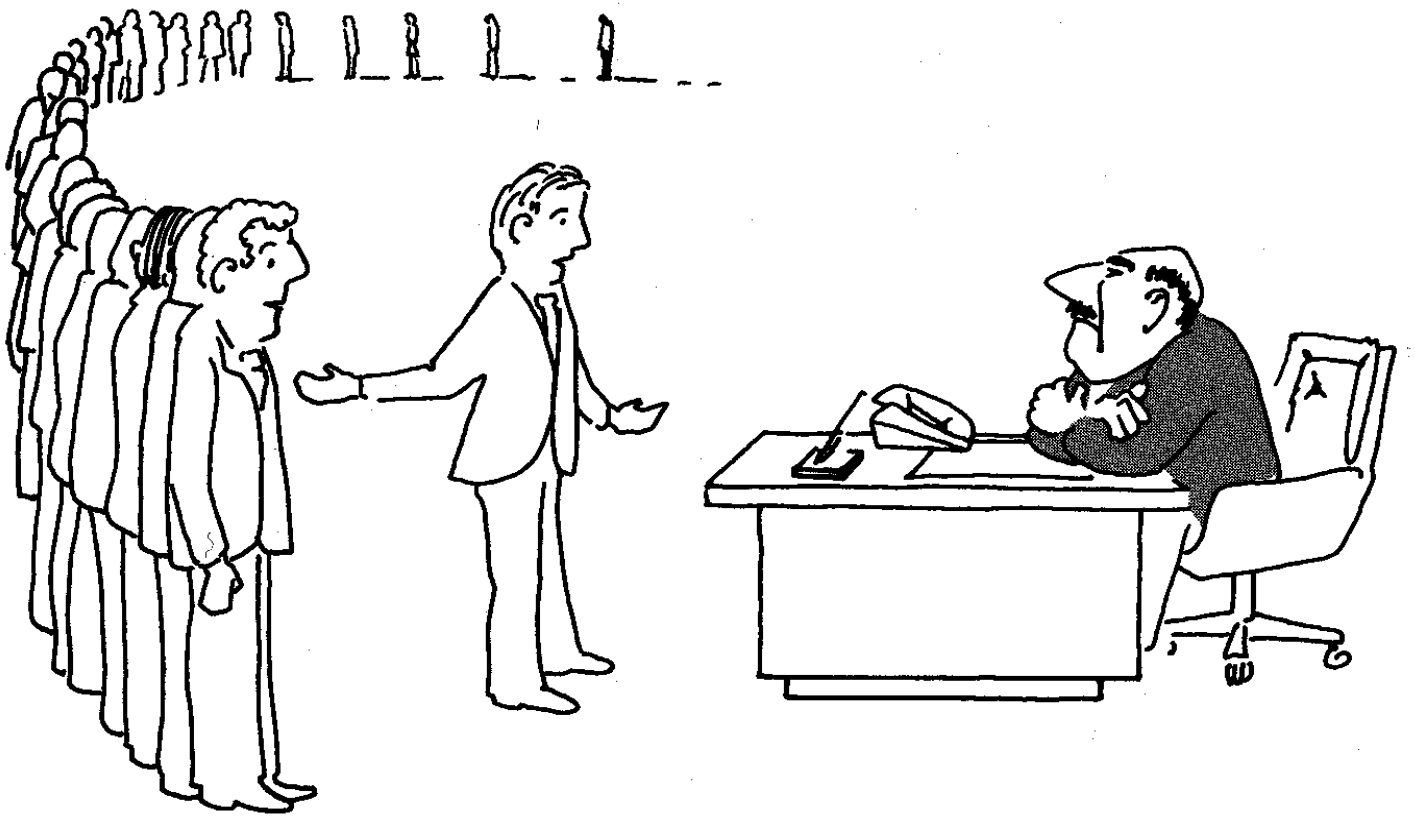
A bijection: identical label matching, edge-to-edge relations

Matching by subgraph isomorphism

- ✓ Input: A directed graph G, and a graph pattern Q
- ✓ Output: all subgraphs of G that are isomorphic to Q
- ✓ Complexity — NP-complete — *Exponentially* many matches
  - Remains NP-hard even when
    - Q is a forest and G is a tree
    - Q is a tree and G is acyclic
- ✓ PTIME if Q is a tree and G is a forest

intractable

NP-completeness



"I can't find an efficient algorithm, but neither can all these famous people."

Algorithms for computing subgraph isomorphism

Input: pattern Q and graph G

Output: all isomorphic mappings P from Q to G

P: partial mappings, initially empty

Match(P)

- if P covers all nodes in Q then output P;
- else compute the set S(P) of all candidate pairs for inclusion in P
- for each pair p = (u, v) in S(P)
  - if p passes feasibility check
  - then P' ← P ∪ {p}; call Match(P');
- restore data structures

nodes that are directly connected to those already in P, with the same labels

for each pair p = (u, v) in S(P):

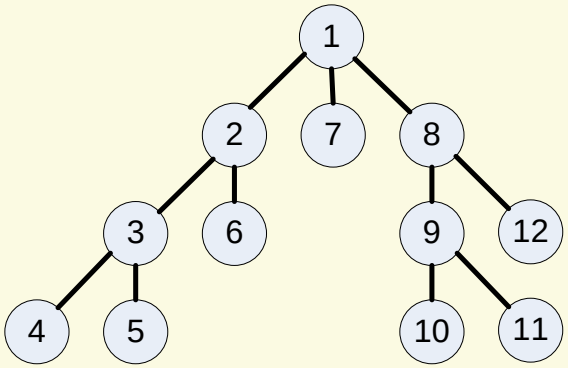
Guarantee correctness

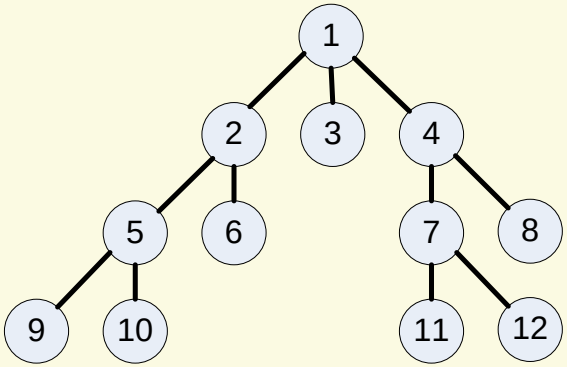
- ✓ enumerate all possible extensions, for refinement
- ✓ if the feasibility test is not successful, drop it and try the next

Recursion, refinement

Backtracking for isomorphism test

- **The backtracking algorithm**
  - depth-first search(DFS): expanding the first child node of the search tree; going deeper until a goal node is found, or until it hits a node that has no children.
  - Branch and bound(B&B): BFS(breadth-first search)-like search for the optimal solution. Branch is that a set of solution candidates is splitted into two or more smaller sets; bound is that a procedure upper and lower bounds.





Ullman's algorithm

Use adjacency matrices of G and Q, their transposes, and a form of permutation matrices

Backtrack(P)

- if P covers all nodes in Q then output P and return;
- for each node u in Q that is not yet in P
  - find a node v in G; p ' ← (u, v); P' ← P ∪ {p};
  - if P' makes a partial mapping (injective function, preserving edges)
  - then call Backtrack(P');

Expanding permutation matrices representing P

for each candidate pair p = (u, v):

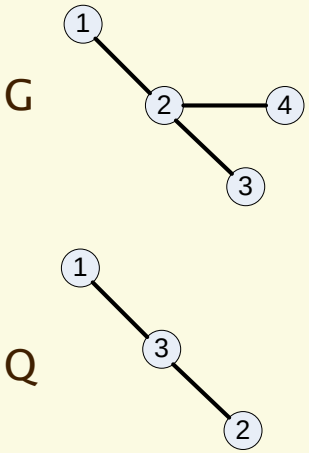
- ✓ enumerate all possible extensions, for refinement
- ✓ Backtracking: no matter whether the test is successful or not, go back to the previous level and try another p

J. R. Ullman. An Algorithm for Subgraph Isomorphism. JACM 1976

An algorithm that is still being used

Ullman's algorithm

- Input: graphs Q and G with their adjacency matrix A_Q and A_G
  - A refinement procedure based on matrix of possible future matched node pairs to prune unfruitful matches
  - The enumeration algorithm for the isomorphisms between a graph Q and a subgraph of another graph G with the adjacency matrices A_Q and A_G



$$A_G = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$A_Q = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

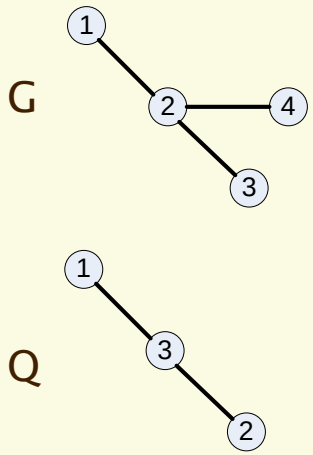
a permutation matrix

$$M^0 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$m_{i,j}^{(0)} = \begin{cases} 1 & \text{if } \deg(V_{Gj}) >= \deg(V_{Qj}) \\ 0 & \text{otherwise} \end{cases}, m_{i,j} \in \{0,1\}$$

Ullman's algorithm: Matrix M'

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$



1) M'[i][j] = 1 means that the j-th vertex in G corresponds to i-th vertex in query Q;

2) Each row in M' contains exactly one 1;

3) No column contains more than one 1.

M' specifies an subgraph isomorphism from Q to G.

Given two graphs Q and G, their corresponding matrixes are $MA_{n\times n}=[a_{ij}]$ and $MB_{m\times m} = [b_{ij}]$.

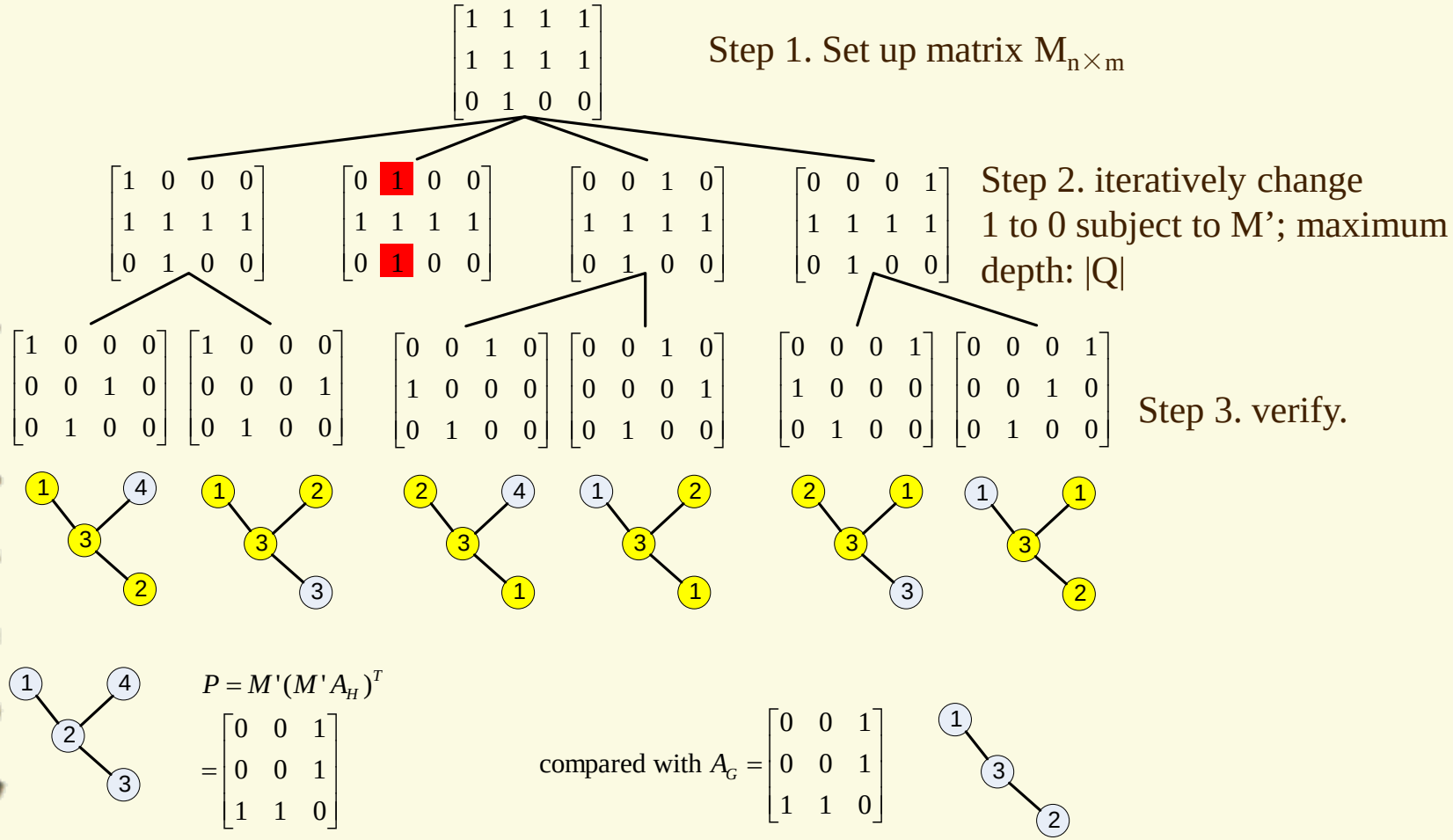
Goal: 1) Find matrix $M'_{n\times m}$ such that

$$MC = M'(M' \bullet MB)^T \qquad \forall i \forall j : (MA[i][j] = 1) \Rightarrow (MC[i][j] = 1)$$

2) or report no such marix M'.

find M' s.t.

Ullmann's algorithm



VF2

Match(P)

- if P covers all nodes in Q then output P;
- else compute the set S(P) of all candidate pairs for inclusion in P
- for each pair p = (u, v) in S(P)
  - if p passes feasibility check
  - then P' ← P ∪ {p}; call Match(P'),
- restore data structures

Five k-look-ahead rules, to make sure that P is a partial isomorphic mapping

Feasibility rules: for each pair (u, v) in P

Guarantee correctness and reduce backtracking

- ✓ their predecessors are already mapped and included in P
- ✓ their successor
- ✓ Certain conditi
  successors to e

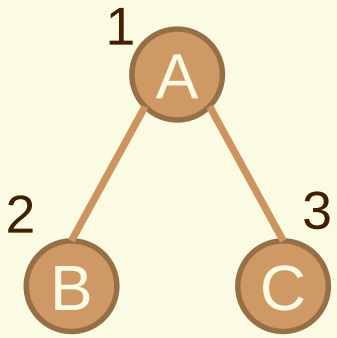
L. P. Cordella, P. Foggia, C. Sansone, M. Vento. A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs, IEEE Trans. Pattern Anal. Mach. Intell. 26, 2004

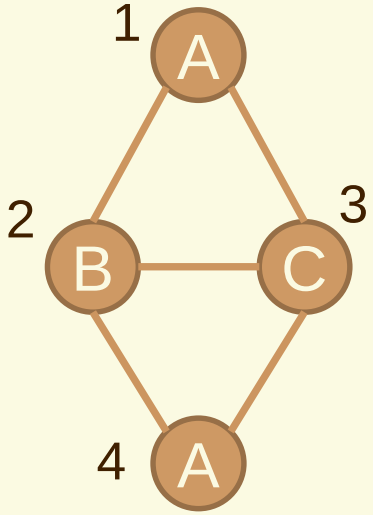
VF2: a popular algorithm for subgraph isomorphism

VF2

Considering two graphs Q and G, the (sub)graph isomorphism from Q to G is expressed as the set of pairs (n,m) (with n ∈Q, m ∈G)

- ✓ Idea: finding the (sub)graph isomorphism between Q and G is a sequence of state transition.
- ✓ an intermediate state s denotes a partial mapping from Q to G





(1, 1) (1, 4)
(2, 2) (2, 2)
(3, 3) (3, 3)

	Intermediate States
s1	(2,2)
s2	(2,2) (1,1)
s3	(2,2)(1,1)(3,3)

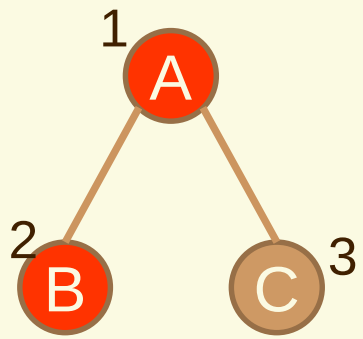
Some Structural Feasibility Rules

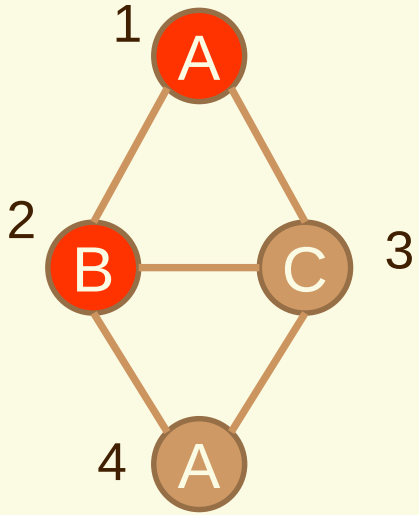
$$F(s,n,m) = F_{structure}(s,n,m) \wedge F_{label}(s,n,m)$$

- ✓Neighbor Connection

$$F(s,n,m) \Leftrightarrow (\forall n' \in (V_1(s) \cap N_1(n,Q)))$$
$$\exists m' \in (V_2(s) \cap N_2(m,G))$$

$$F(s,n,m) \Leftrightarrow |N_1(n,Q) \cap T_1(s,Q)|$$
$$\leq |N_2(m,G) \cap T_2(s,G)|$$





$$F_{structure}(s,n,m)$$

$$s:$$

$$1 \leftrightarrow 1$$

$$2 \leftrightarrow 2$$

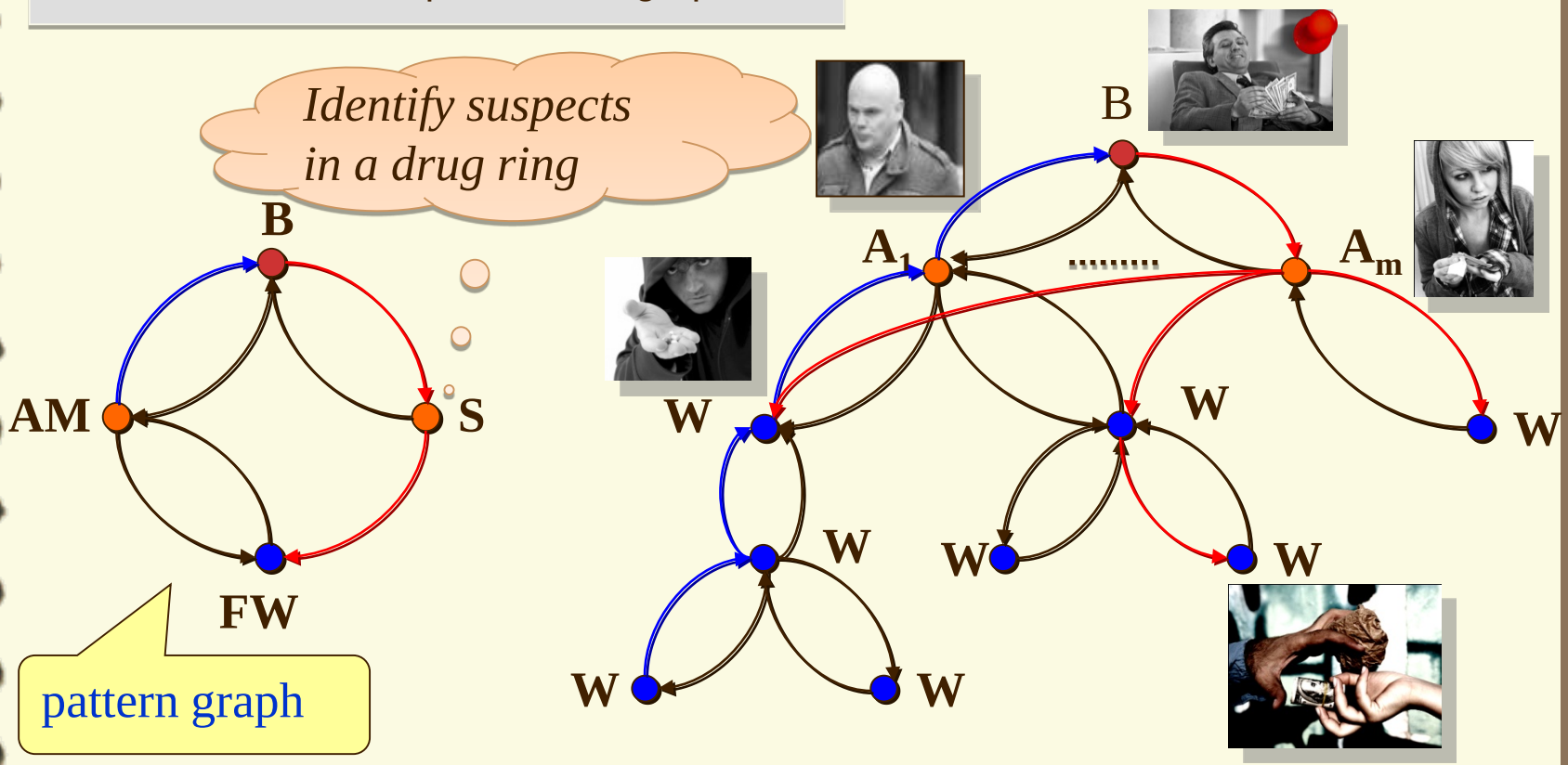
$$n = 3; m = 3$$

Approximate graph pattern matching:

from intractability to PTIME

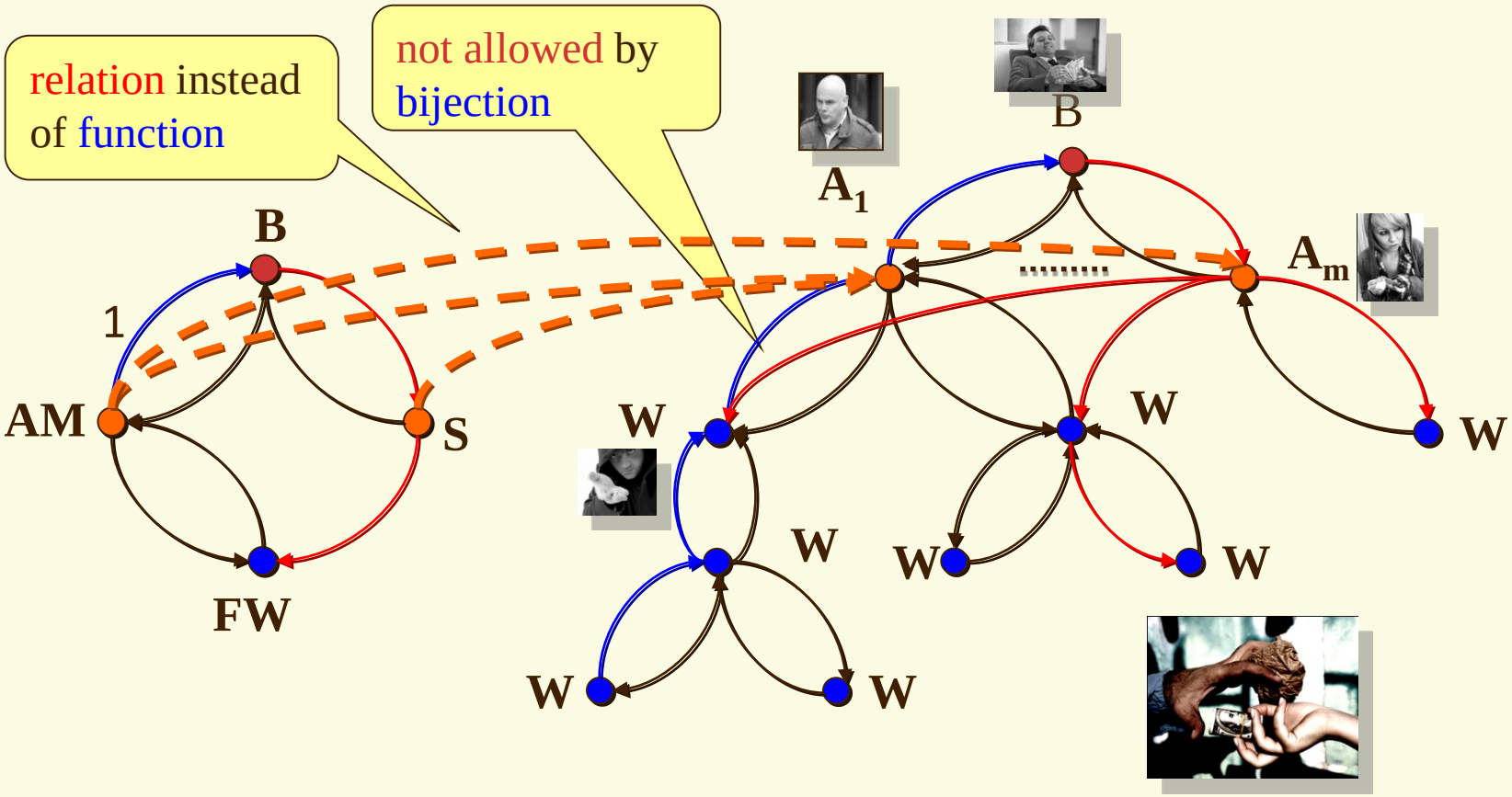
Pattern Matching in Social Graphs

Find all *matches* of a pattern in a graph



"Understanding the structure of drug trafficking organizations"

Pattern Matching in Social Graphs



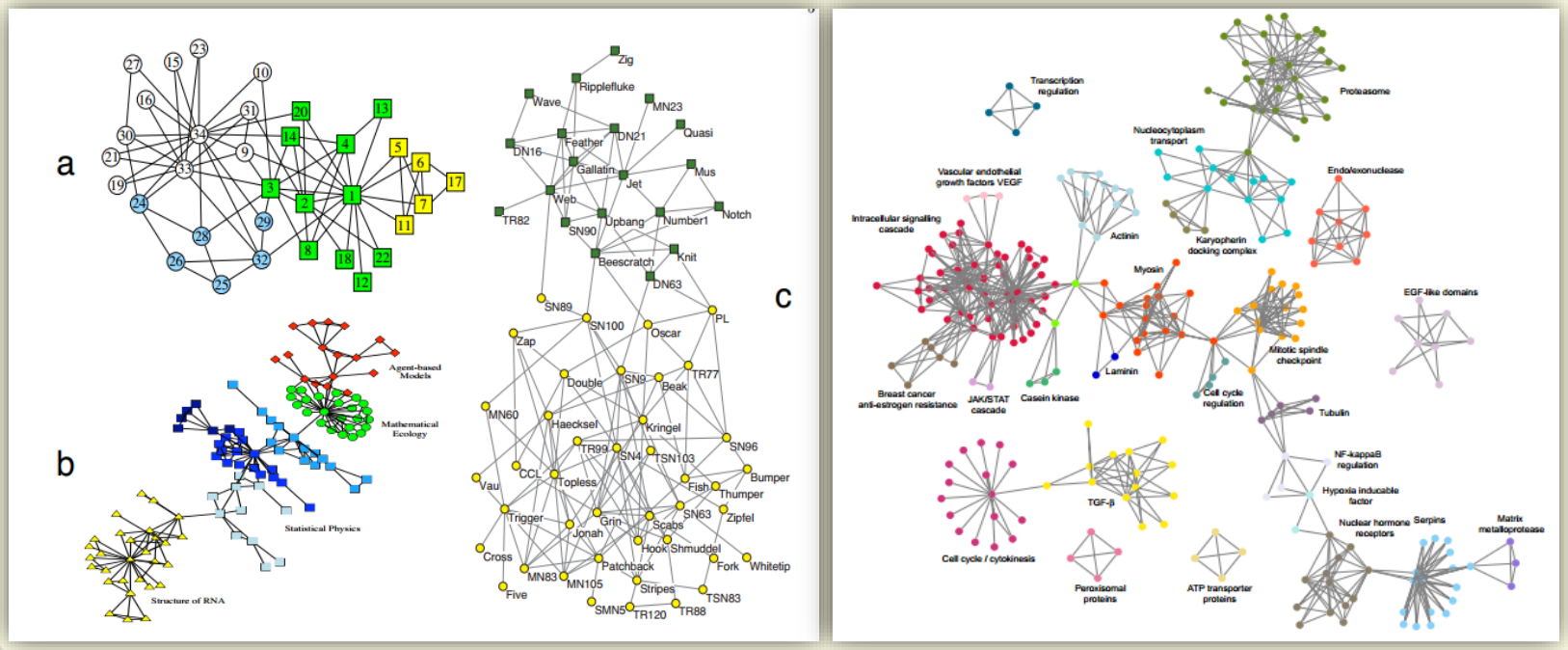
The quest for a new form of graph pattern matching

Community and social role detection

Positions: actors that show similar structure of relationships

Roles: pattern of relationships of members of same or different positions

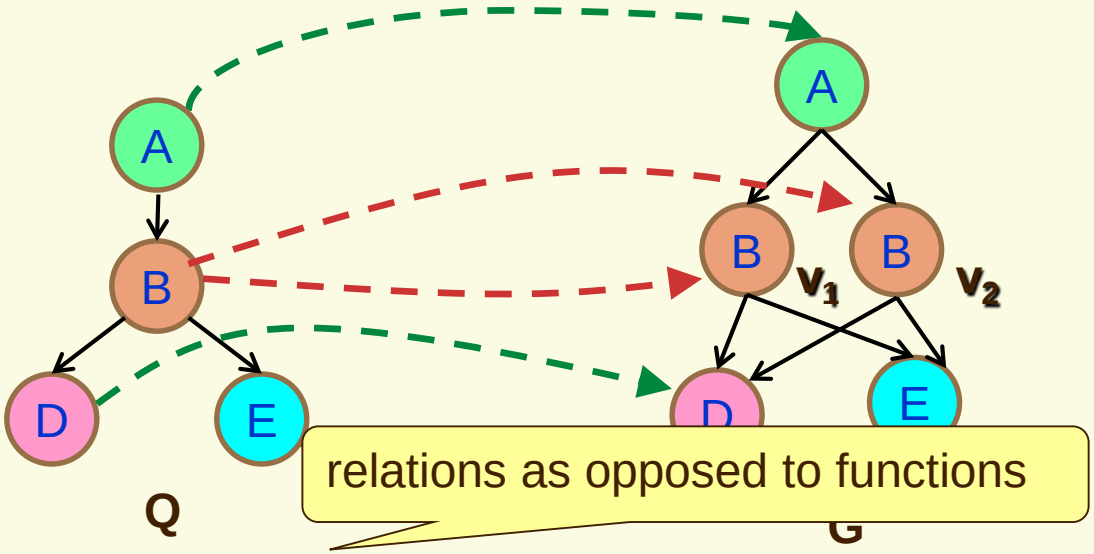
“Role equivalence”: two positions are considered as similar, if every important aspect of the observed structure applies (or does not apply) for both positions



Graph Simulation

A binary relation R on the nodes of Q and the nodes of G:

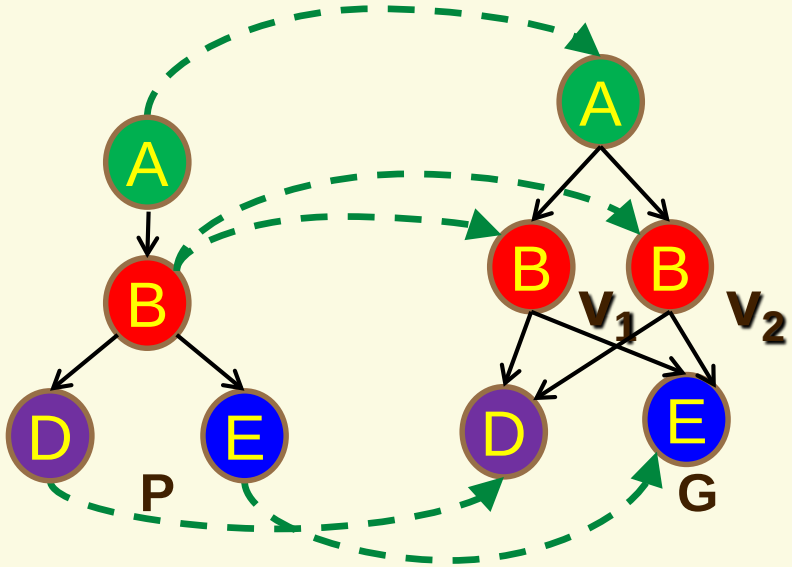
- ✓ For each node u in Q, there exists a node v in G such that (u, v) is in R, and u and v have the same label;
- ✓ If there exists an edge (u, u') in Q and each pair (u, v) is in R, then there exists an edge (v, v') in G such that (u', v') is in R

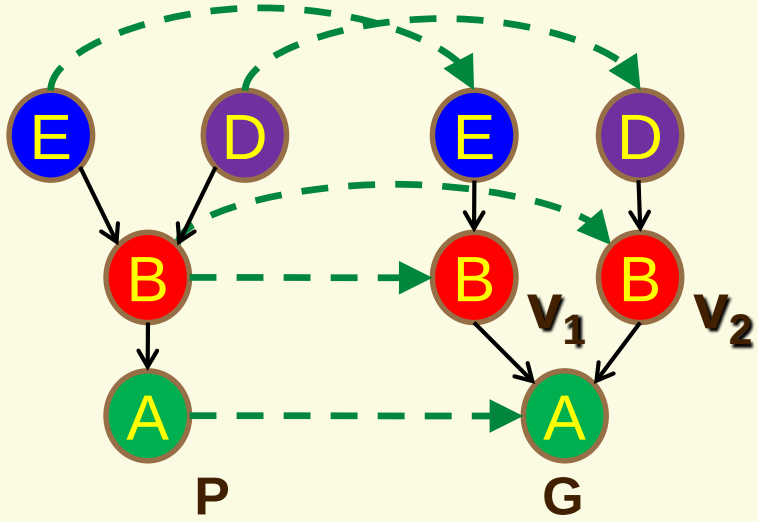


A relation: label matching, edge-to-edge mapping

Subgraph isomorphism vs Graph Simulation

- ✓ Node label equivalence vs node search constraints
- ✓ Bijective function vs. many-many relation





Identical label matching, edge-to-edge function/relations

Matching by graph simulation

- ✓ Input: A directed graph G, and a graph pattern Q
- ✓ Output: the maximum simulation relation R

- ✓ Maximum simulation relation: always exists and is unique
  - If a match relation exists, then there exists a maximum one
  - Otherwise, it is the empty set – still maximum

- ✓ Complexity: $O((|V| + |V_Q|)(|E| + |E_Q|)$

- ✓ The output is a unique relation, possibly of size $|Q||V|$

Use relations instead of functions

Quadratic time

Algorithm for computing graph simulation

Input: pattern Q and graph G

Output: for each u in Q, sim(u): the matches w in G

Similarity(P)

- for all nodes u in Q do
  - sim(u) ← the set of candidate matches w in G;
- while there exist (u, v) in Q and w in sim(u) (in G) that violate the simulation condition
  - sim(u) ← sim(u) – {w};
- output sim(u) for all u in Q

with the same label; moreover, if u has an outgoing edge, so does w

successor(w) ∩ sim(v) = ∅

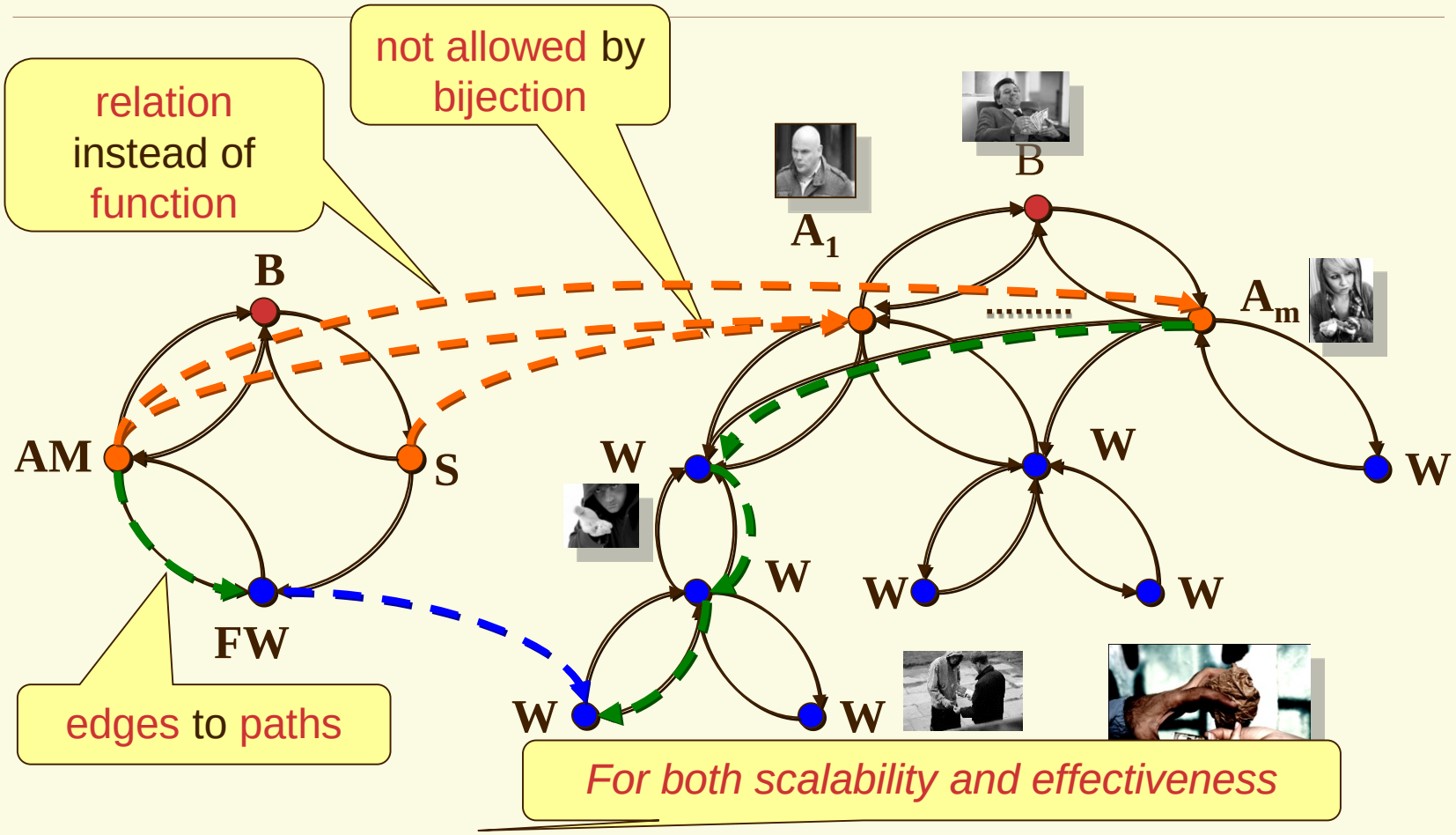
refinement

successor(w) ∩ sim(v) = ∅

- There exists an edge from u to v in Q, but the candidate w of u has no corresponding edge to a node w' that matches v

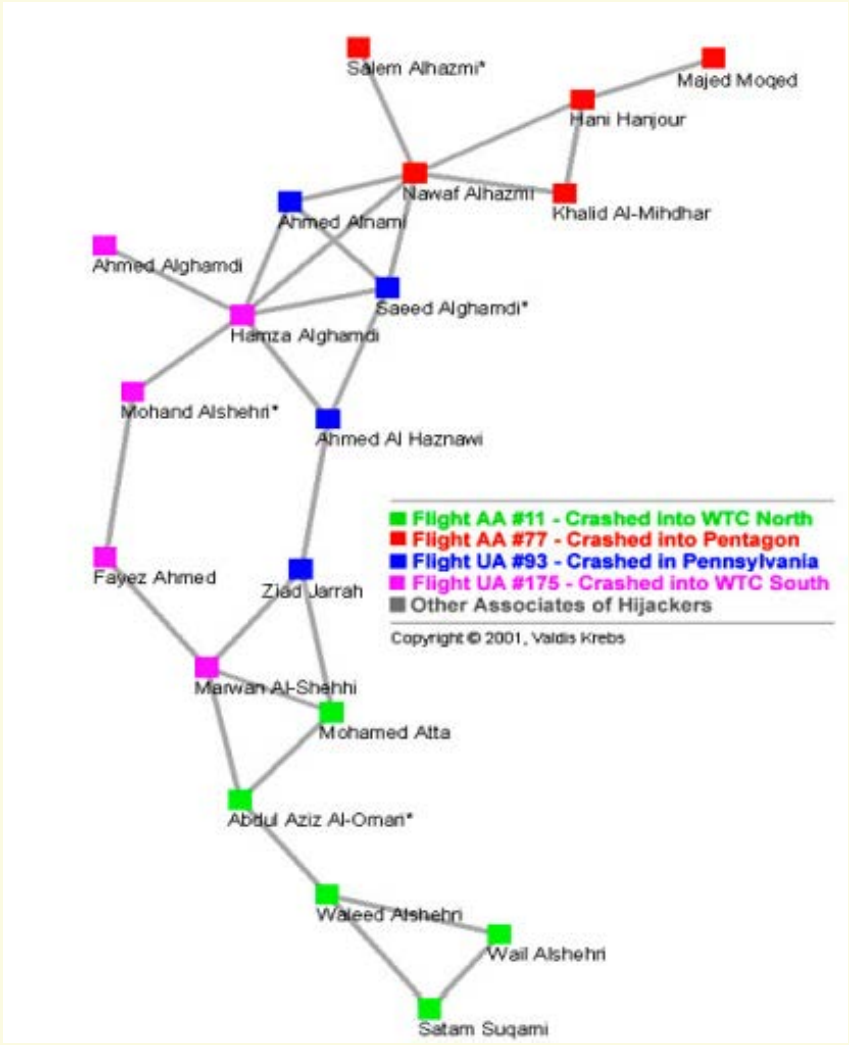
Approximate graph pattern matching for social network analysis

Pattern matching in social graphs



Neither subgraph isomorphism nor graph simulation works

Trusted Contacts in 911 Terrorist network



"Those who were trained to fly didn't know the others. One group of people did not know the other group"

- Bin Laden, 2001

Bounded patterns

Pattern graph: $Q = (V_Q, E_Q, f_v, f_e)$

Search condition

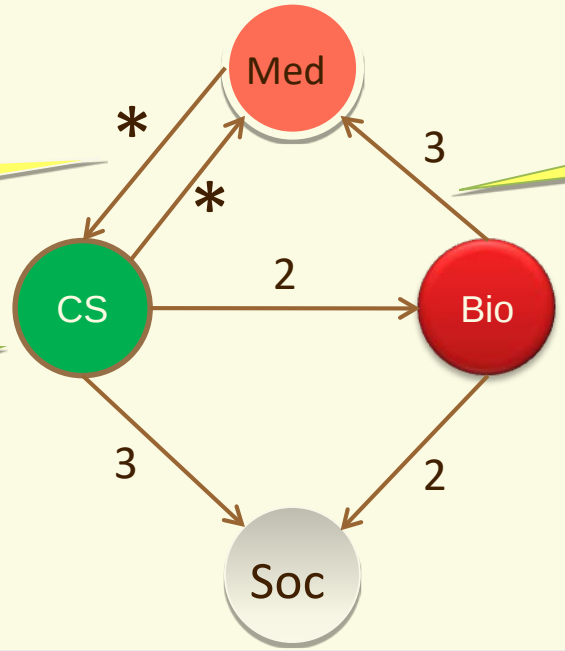
- ✓ $f_v(u)$: a conjunction of $A\ op\ a$, op in $<, \leq, =, \neq, >, \geq$
- ✓ $f_e(u,u')$: a constant k or a symbol $*$, bound

within k hops

Unbounded

Bounded

$f_v()$: 'dept'=CS



Incorporating search conditions and bounds on the number of hops

Bounded Simulation

G = … *h that*

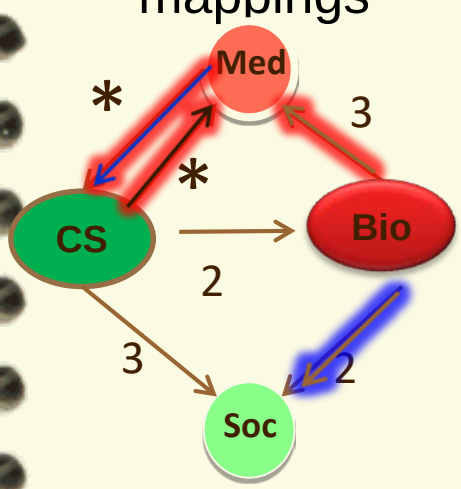
for each $(u,v) \in S$,

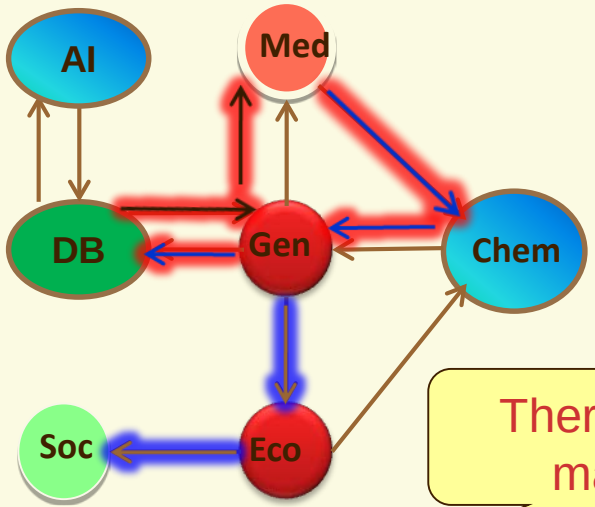
- ✓ attributes $f_A(v)$ *satisfies* predicate $f_v(u)$
- ✓ each (u,u') in E_Q is mapped to a path from v to v' of length $f_e(u,u')$ in G, $(u',v') \in S$

✓ …

✓ …

mappings





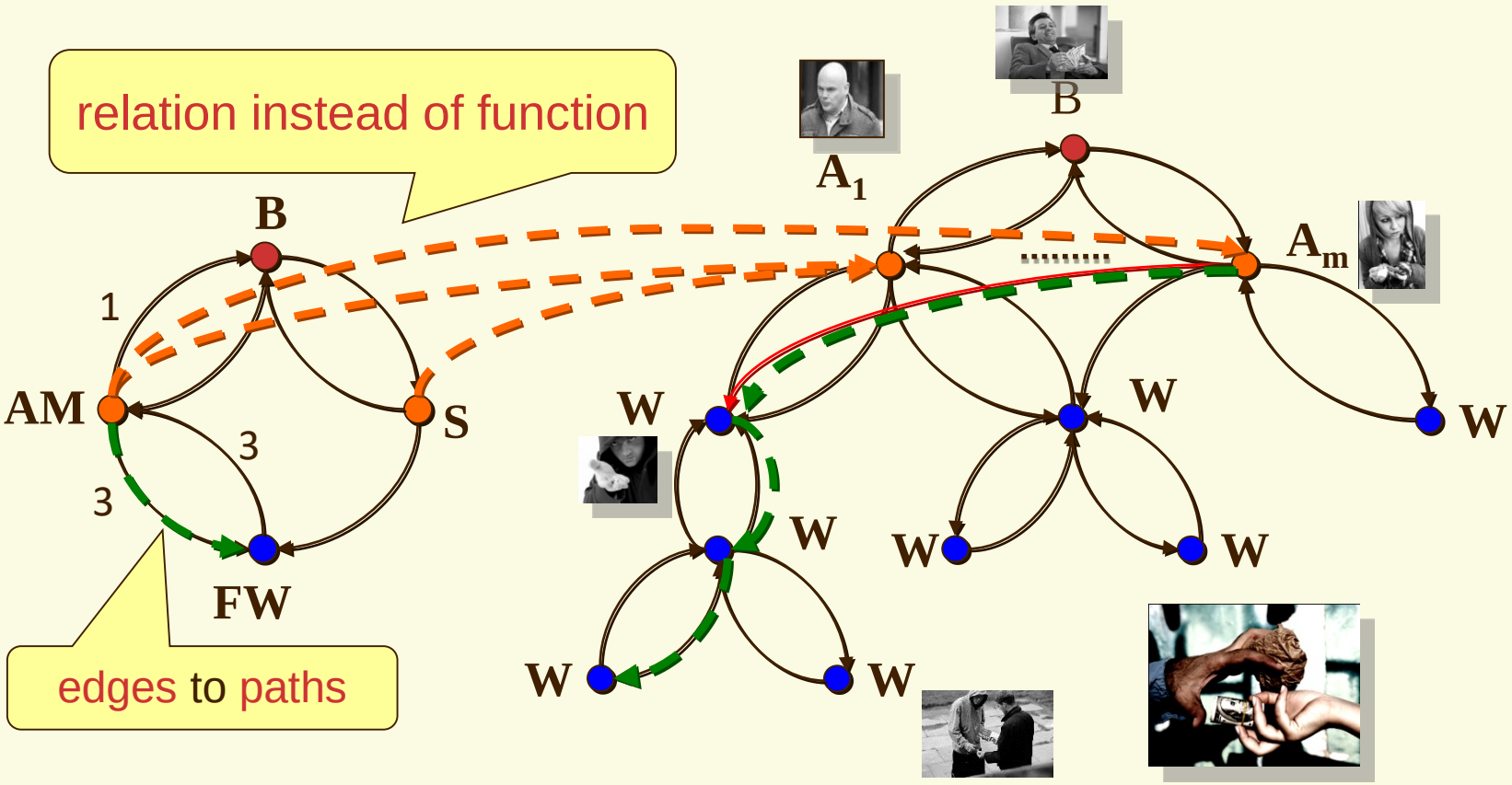
match

(CS, DB)
(Bio, Gen)
(Med, Med)
(Soc, Soc)

There exists a unique maximum match

Mapping edges to bounded paths

Bounded simulation in social graphs



The set of all suspects involved in a drug ring

Complexity

- ✓ Input: Pattern Q and data graph G
- ✓ Output: Q(G), the unique maximum match relation

$$O(|V||E| + |E_Q||V|^2 + |V_Q||V|)$$

- ✓ Subgraph isomorphism: intractable
- ✓ Graph simulation: $O((|V| + |V_Q|)(|E| + |E_Q|)$

Query driven approximation: use bounded simulation instead of subgraph isomorphism. Criteria:

- ✓ Lower complexity
- ✓ Effectiveness: the query answers are sensible

Algorithm? The reading list

To identify sensible matches and be computable in low PTIME

Bounded simulation vs. graph simulation

Graph simulation: a special case of bounded simulation

- ✓ The same bound 1 on all pattern edges (edge-to-edge mapping)
- ✓ Unique attributes vs. search conditions: label equality
- ✓ $O((|V_G| + |V_Q|)(|E_G| + |E_Q|)$

vs.

$O(|V_G||E_G| + |E_Q||V_G|^2 + |V_Q||V_G|)$

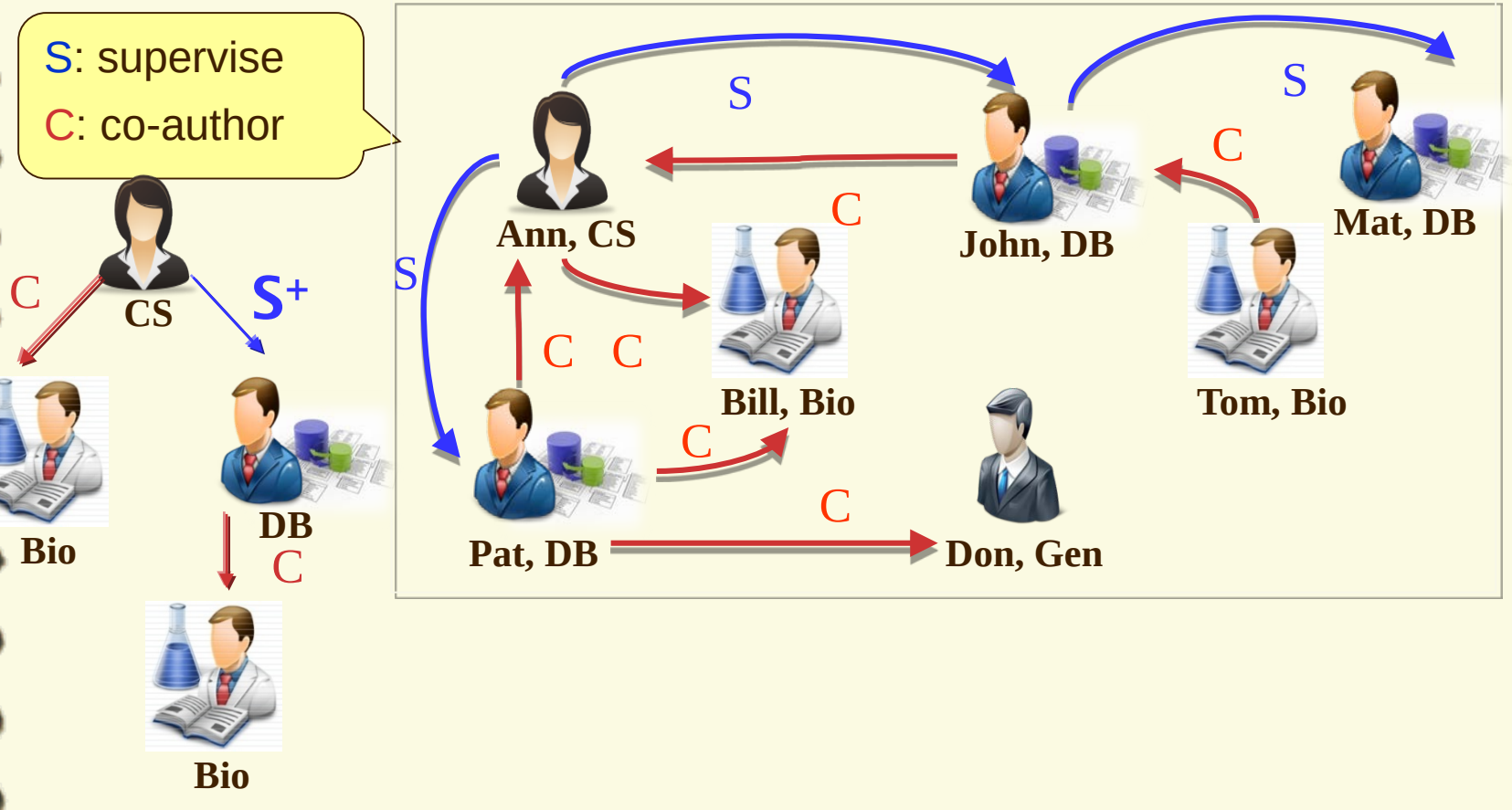
Process calculus
Web site classification
Social position detection, …

Capture more sensible matches in social graphs (by 80%)

Approximate graph pattern matching:

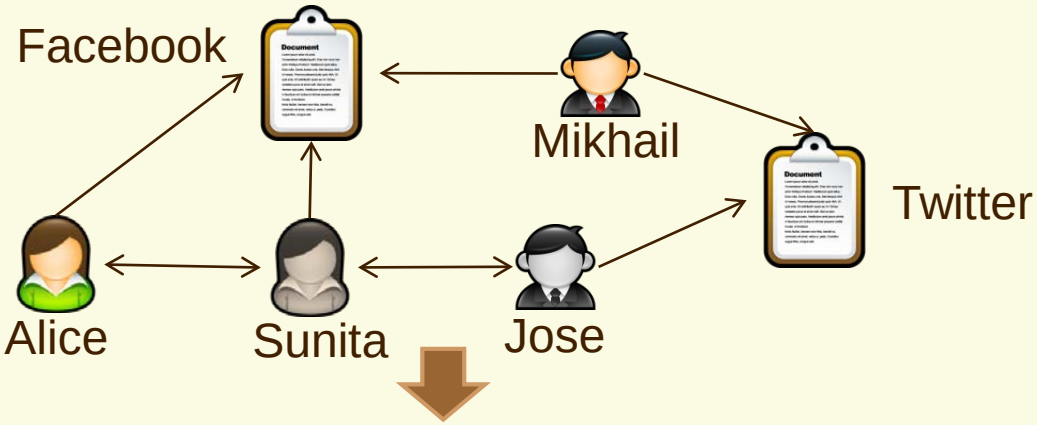
- *Incorporating edge relationships*

Edge relationships



What is this pattern to find?

Edge relation



(Alice, Facebook)

(Alice, Sunita)

(Jose, Twitter)

(Jose, Sunita)

(Mikhail, Facebook)

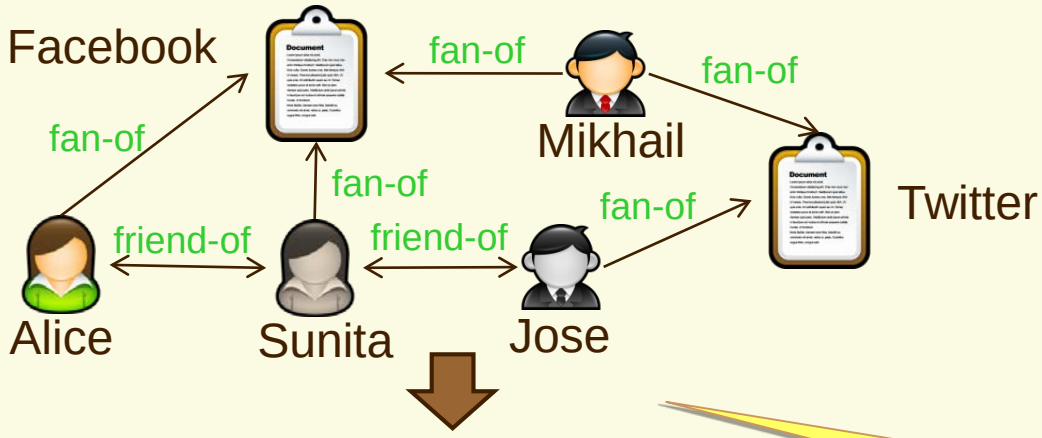
(Mikhail, Twitter)

(Sunita, Facebook)

(Sunita, Alice)

(Sunita, Jose)

Graph encodings: Adding edge types



(Alice, fan-of, Facebook)

(Alice, friend-of, Sunita)

(Jose, fan-of, Twitter)

(Jose, friend-of, Sunita)

(Mikhail, fan-of, Facebook)

(Mikhail, fan-of, Twitter)

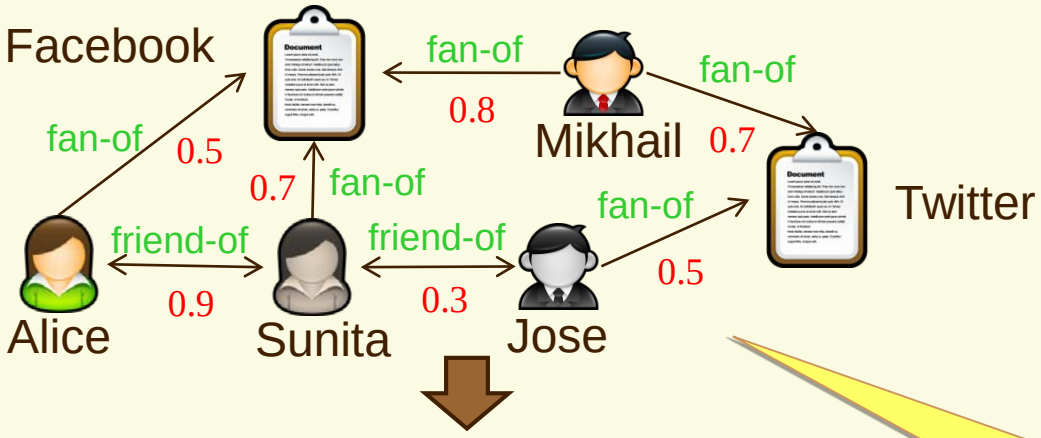
(Sunita, fan-of, Facebook)

(Sunita, friend-of, Alice)

(Sunita, friend-of, Jose)

Adding edge labels

Graph encodings: Adding weights



(Alice, fan-of, 0.5, Facebook)

(Alice, friend-of, 0.9, Sunita)

(Jose, fan-of, 0.5, Twitter)

(Jose, friend-of, 0.3, Sunita)

(Mikhail, fan-of, 0.8, Facebook)

(Mikhail, fan-of, 0.7, Twitter)

(Sunita, fan-of, 0.7, Facebook)

(Sunita, friend-of, 0.9, Alice)

(Sunita, friend-of, 0.3, Jose)

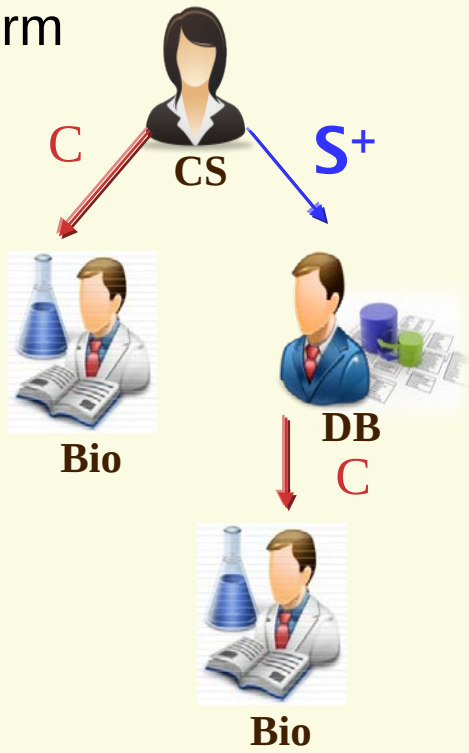
Even further, you can add weights and others

Regular patterns

Pattern: $Q = (V_Q, E_Q, f_v, f_e)$

- ✓ $f_v(u)$: a conjunction of $A\ op\ a$, op in $<, \leq, =, \neq, >, \geq$
- ✓ $f_e(u,u')$: a regular expression of the form

$$F ::= c \mid c^{\leq k} \mid c^{+} \mid FF$$



Simple regular expressions:

✓ fairly common

✓low complexity in matching

Mapping edges to paths satisfying associated regular expressions

Complexity

- Input: Pattern Q and data graph G
- Output: Q(G)

m: the number of distinct colors in Q

$$O(|V||E| + m|E_Q||V|^2 + |V_Q||V|)$$

- bounded simulation: a special case
  - single color c (hence m = 1)
  - $f_e(u,u') = c$

general regular expressions?

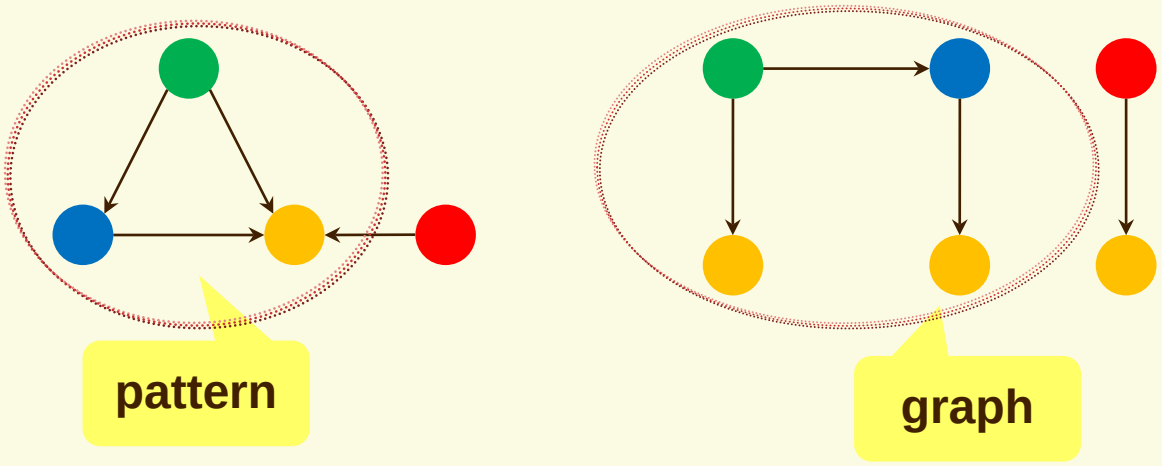
Adding edge colors does not incur extra complexity

Graph pattern matching:

- *Capturing graph topology*

46

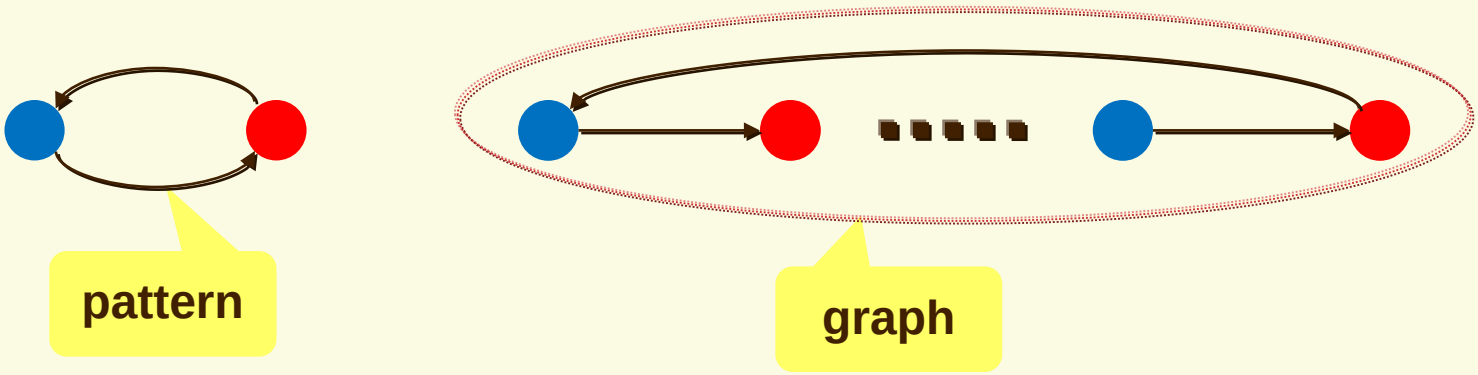
Limitations of graph simulation



- ✓ A disconnected graph matches a connected pattern
- ✓ The yellow node in the pattern has 3 “parents”, in contrast to 1 in the data graph
- ✓ An undirected cycle matches a tree

Simulation does not preserve the topology well in matching

Limitations of graph simulation



- ✓ A cycle with two nodes matches a cycle of unbounded length
- ✓ The match relation may be excessively large

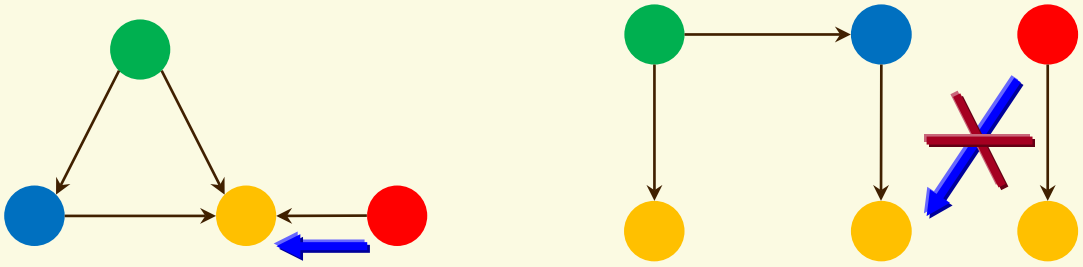
When social distances increase, the closeness of relationships decrease

The need for revising simulation to enforce locality

Dual simulation

$G = (V, E, f_A)$ *matches* $Q = (V_Q, E_Q, f_v, f_e)$ via *dual simulation*, if there exists a binary relation $S \subseteq V_Q \times V$ *such that* S

- ✓ is a total mapping,
- ✓ satisfies search conditions, and
- ✓ preserves both "child" and "parent" relationships

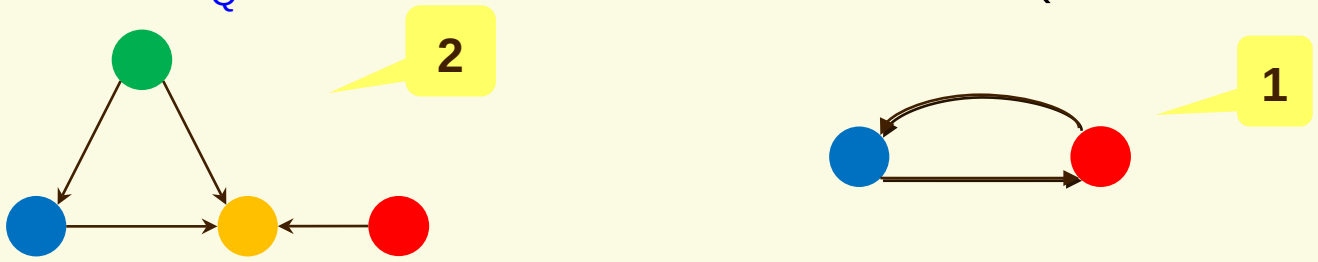


- ✓ *Q(G)* : a *unique maximum match relation*

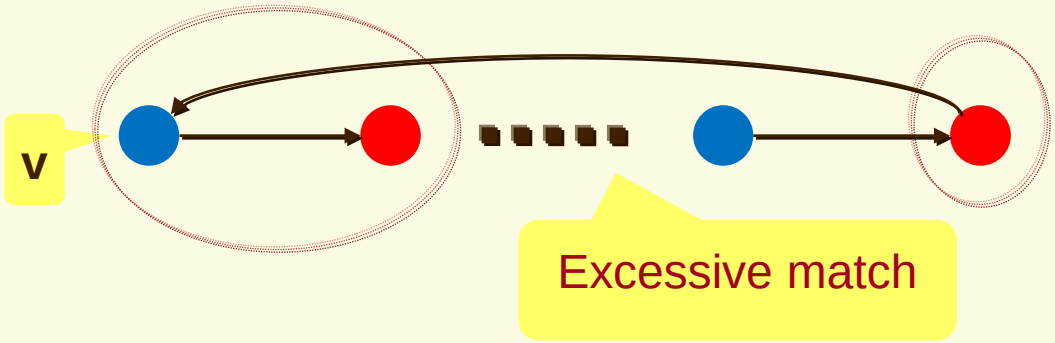
Preserve "parent" relationships and connectivity

Locality

- diameter d_Q: the maximum shortest distance (undirected paths)



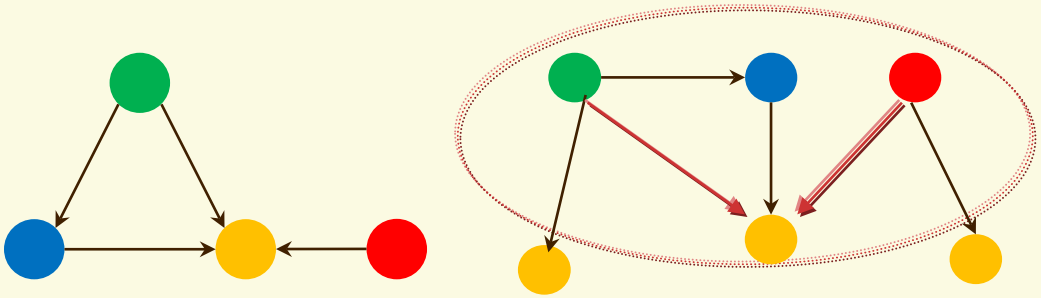
- d_Q-radius subgraph $G[v, d_Q]$: centered at v, within d_Q hops



Locality: matches contained in $G[v, d_Q]$ for some v

Strong simulation

- ✓ *G matches Q* via *strong simulation*, if there exists a node v in *G* such that $G[v, d_Q]$ matches *Q* via dual simulation
  - duality
  - local

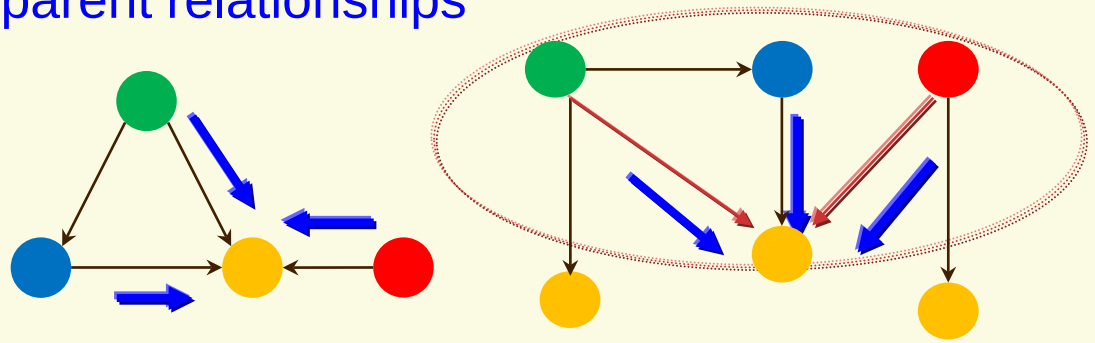


- ✓ *Match:* the subgraph G_S of $G[v, d_Q]$ representing *the maximum match S*

Matching: given Q and G, find the set Q(G) of all matches

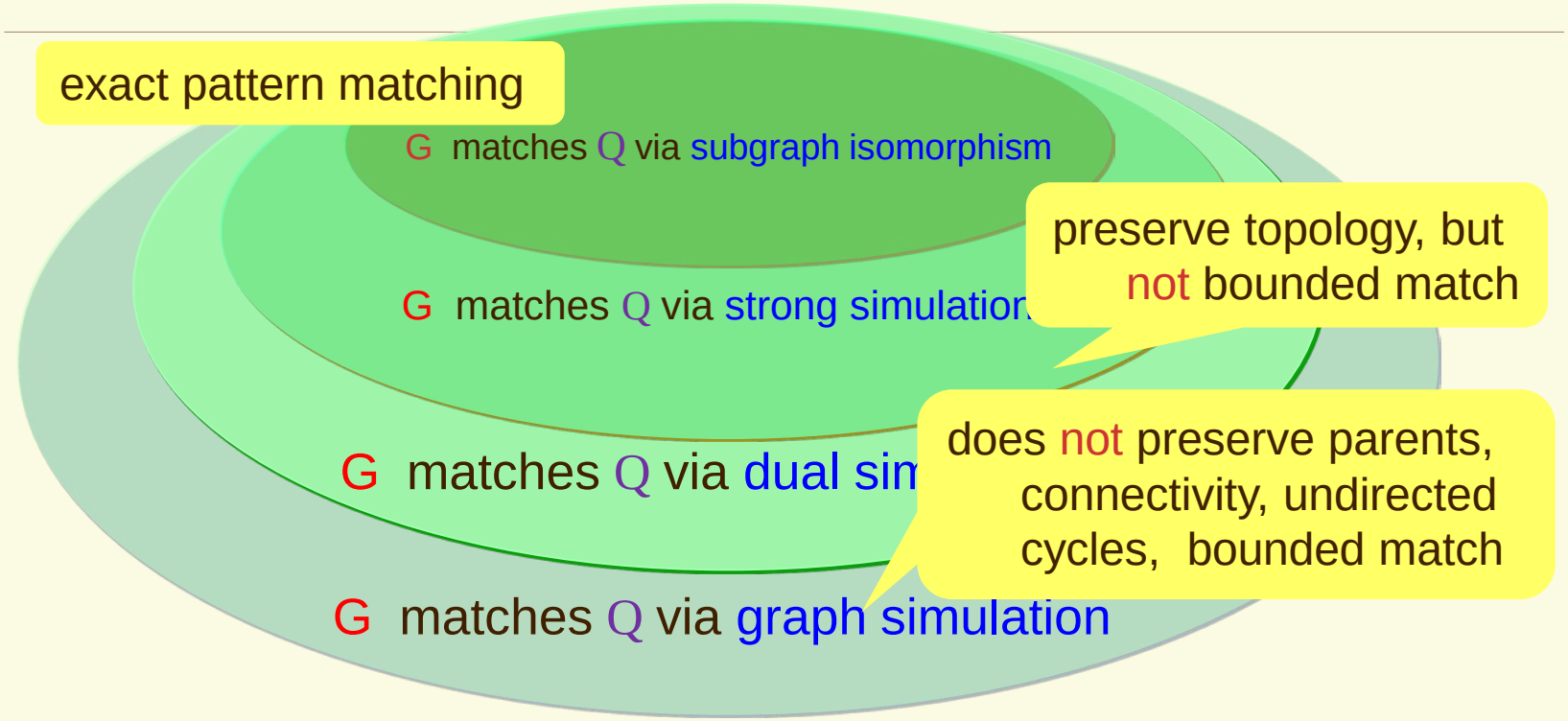
Preserving the topology of patterns

- Child and parent relationships



- connectivity: if Q is connected (via undirected path), so is G_S
- cycles: a directed (resp. undirected) cycle in Q matches a directed (resp. undirected) cycle in G_S
- bounded matches:
  - the diameter of G_S is at most $2 * d_Q$
  - $|M(Q, G)| \leq |V|$

Strong simulation vs. graph simulation



- ✓ Input: Pattern Q and data graph G
- ✓ Output: Q(G)

cubic time

$$O(|V|(|V| + (|V_Q| + |E_Q|)(|V| + |E|))$$

A balance between the complexity and the ability to preserve topology

Summing up

Various notions for graph pattern matching

matching	complexity	match size
subgraph isomorphism	NP-complete	$\|V\|^{\|V_Q\|}$
graph simulation	quadratic time	$\|V\|\,\|V_Q\|$
bounded simulation	cubic time	$\|V\|\,\|V_Q\|$
regular matching	cubic time	$\|V\|\,\|V_Q\|$
strong simulation	cubic time	$\|V\|$

Query driven approximation: from subgraph isomorphism (intractable) to strong simulation or bounded simulation (cubic-time)

Summary and review

- ✓ Query-driven approximation
- ✓ What is subgraph isomorphism? Complexity? Algorithm? Name a few applications
- ✓ What is graph simulation? Complexity? Understand its algorithm. Name a few applications
- ✓ Why do we need to revise conventional graph pattern matching for social network analysis? How should we do it? Why?
- ✓ Understand bounded simulation. Read its algorithm. Complexity?
- ✓ What is strong simulation? Complexity? Name a few applications in which strong simulation is useful.
- ✓ Find other revisions of conventional graph pattern matching that are not covered in the lecture.

Papers for you to review

- M. R. Henzinger, T. Henzinger, and P. Kopke. Computing simulations on finite and infinite graphs. FOCS, 1995. http://infoscience.epfl.ch/record/99332/files/HenzingerHK95.pdf
- L. P. Cordella, P. Foggia, C. Sansone, M. Vento. A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs, IEEE Trans. Pattern Anal. Mach. Intell. 26, 2004 (search Google scholar)
- ✓ A. Fard, M. U. Nisar, J. A. Miller, L. Ramaswamy, Distriuted and scalable graph pattern matching: models and algorithms. Int. J. Big Data. http://cobweb.cs.uga.edu/~ar/papers/IJBD_final.pdf
- W. Fan J. Li, S. Ma, and N. Tang, and Y. Wu. *Graph pattern matching: From intractable to polynomial time*, VLDB, 2010.
- W. Fan, J. Li, S. Ma, N. Tang, and Y. Wu. Adding Regular Expressions to Graph Reachability and Pattern Queries, ICDE 2011.
- S. Ma, Y. Cao, W. Fan, J. Huai, T. Wo: Strong simulation: Capturing topology in graph pattern matching. TODS 39(1): 4, 2014.