Graph algorithms for very large graphs and their applications to bioinformatics and social network analysis

Pasquale Foggia

University of Salerno, Italy

pfoggia@unisa.it

MIVIA Lab
Intelligent Machines
for Video, Image
and Audio Analysis

MIVIA Lab

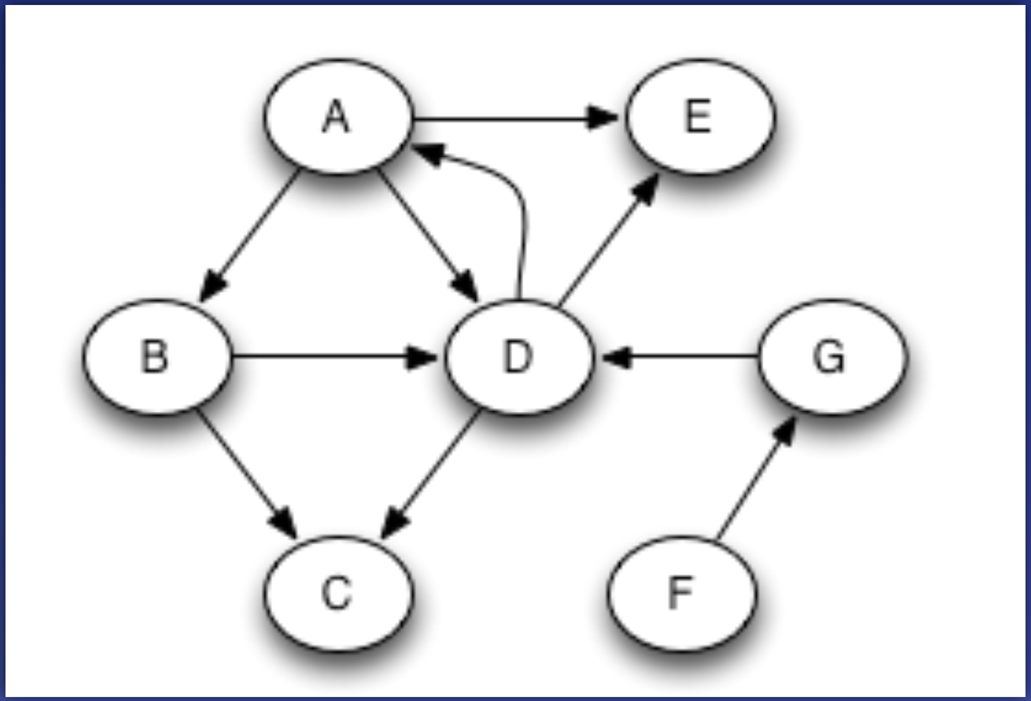
- 3 full professors
  - Mario Vento, Francesco Tortorella, Pasquale Foggia
- 2 associate professors
  - Gennaro Percannella, Pierluigi Ritrovato
- 3 researchers
  - Alessia Saggese, Luca Greco, Vincenzo Carletti
- 1 post-doc
- 3 PhD students

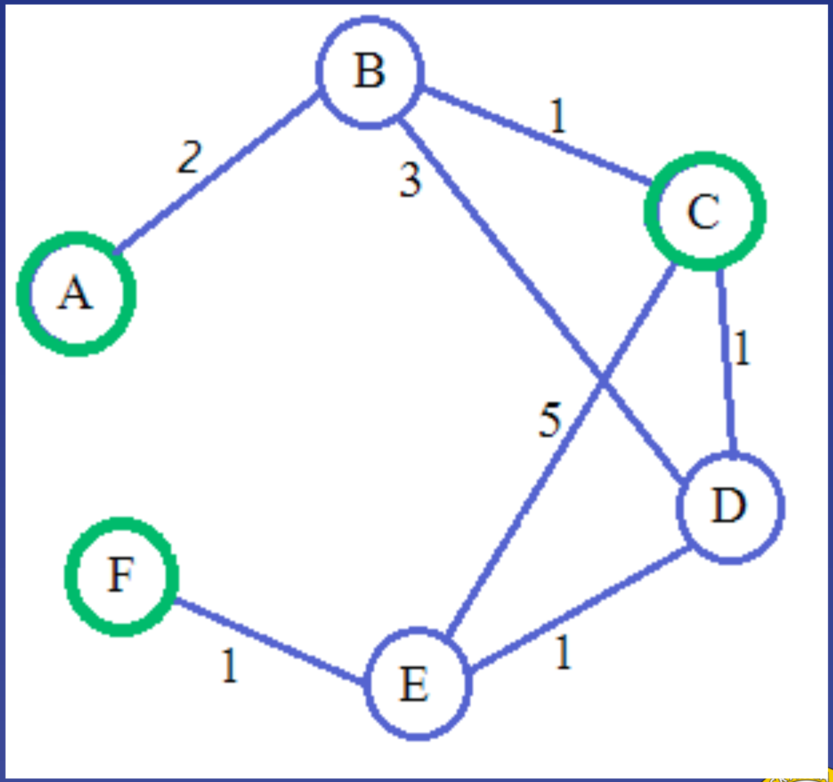
MIVIA Lab

- Artificial vision
- Cognitive robotics
- Medical image analysis
- Structural Pattern Recognition and Graph-based Algorithms
  - Graph-based algorithms applied to genomics

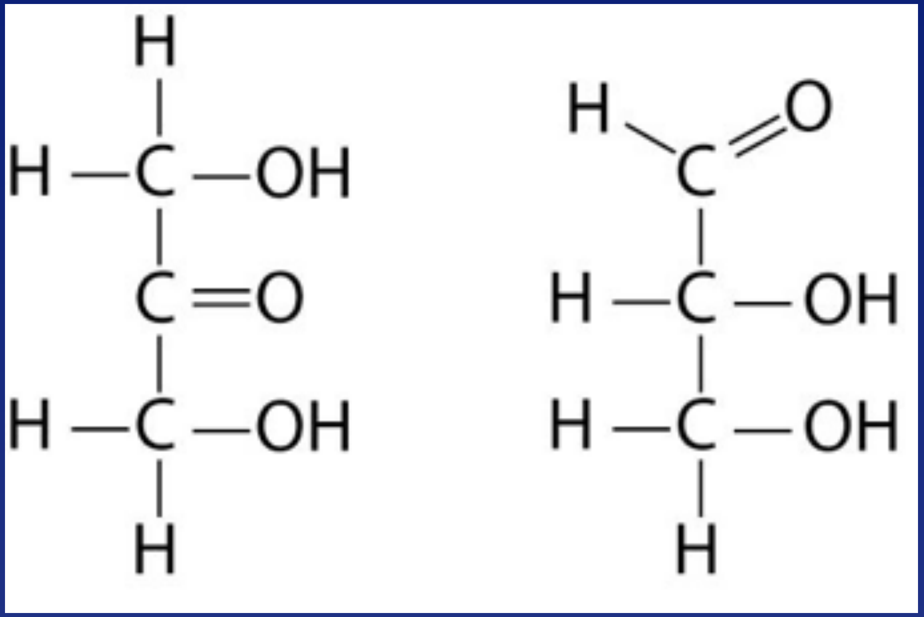
(Attributed) Graphs

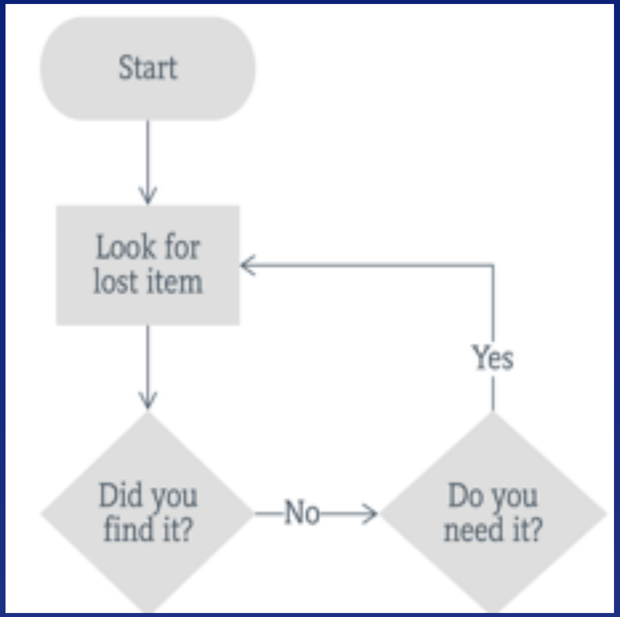
- Representation for *structured* information
  - Nodes (aka vertices) representing atomic entities
  - Edges representing relations between entities
  - Nodes and edges can have *attributes* (aka *labels*) carrying additional information about entities and relations

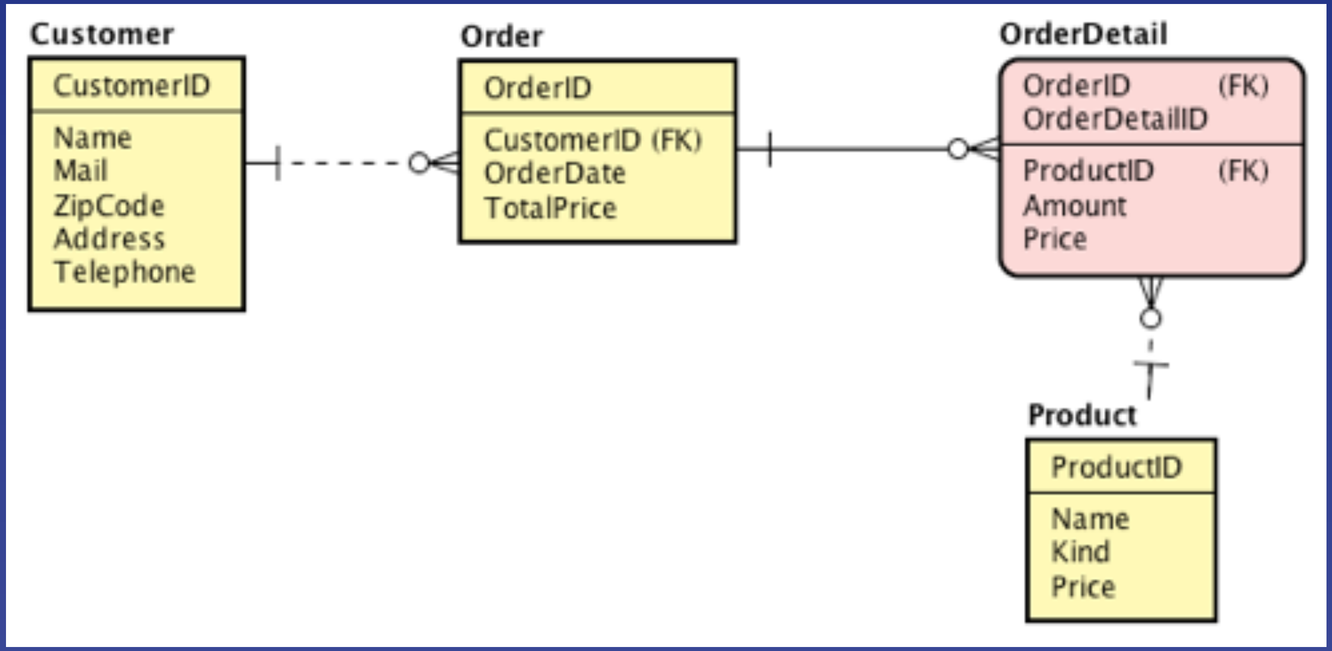


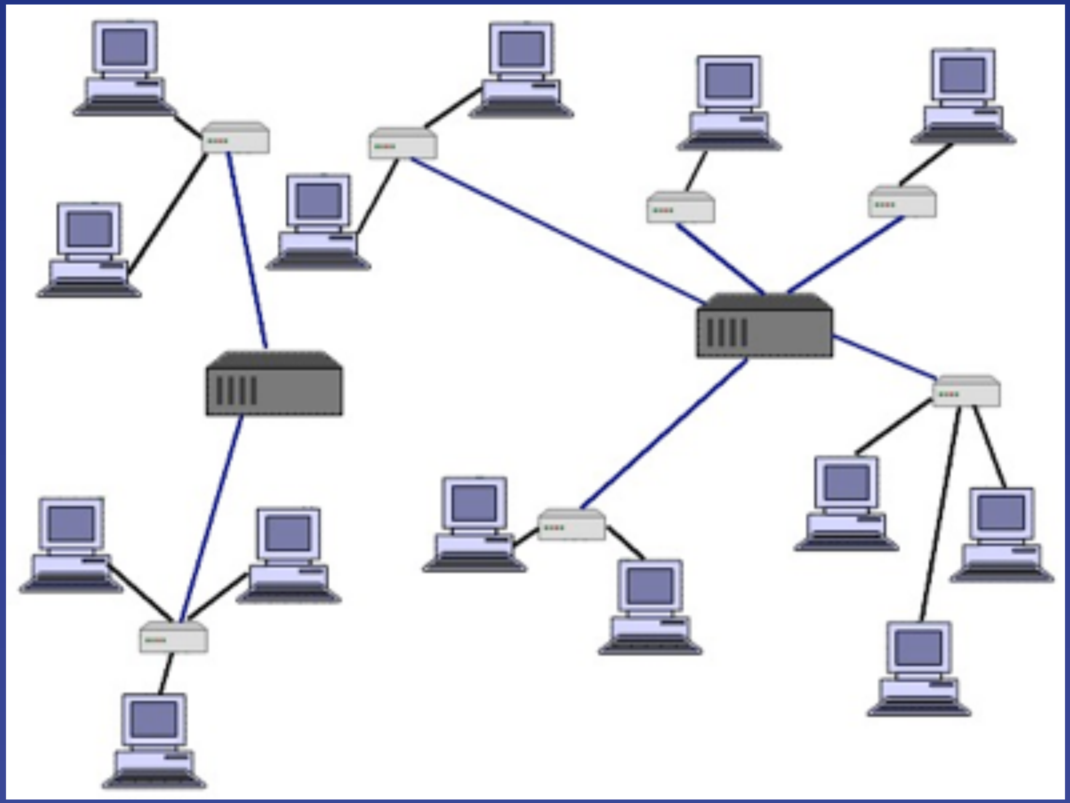


Obvious examples



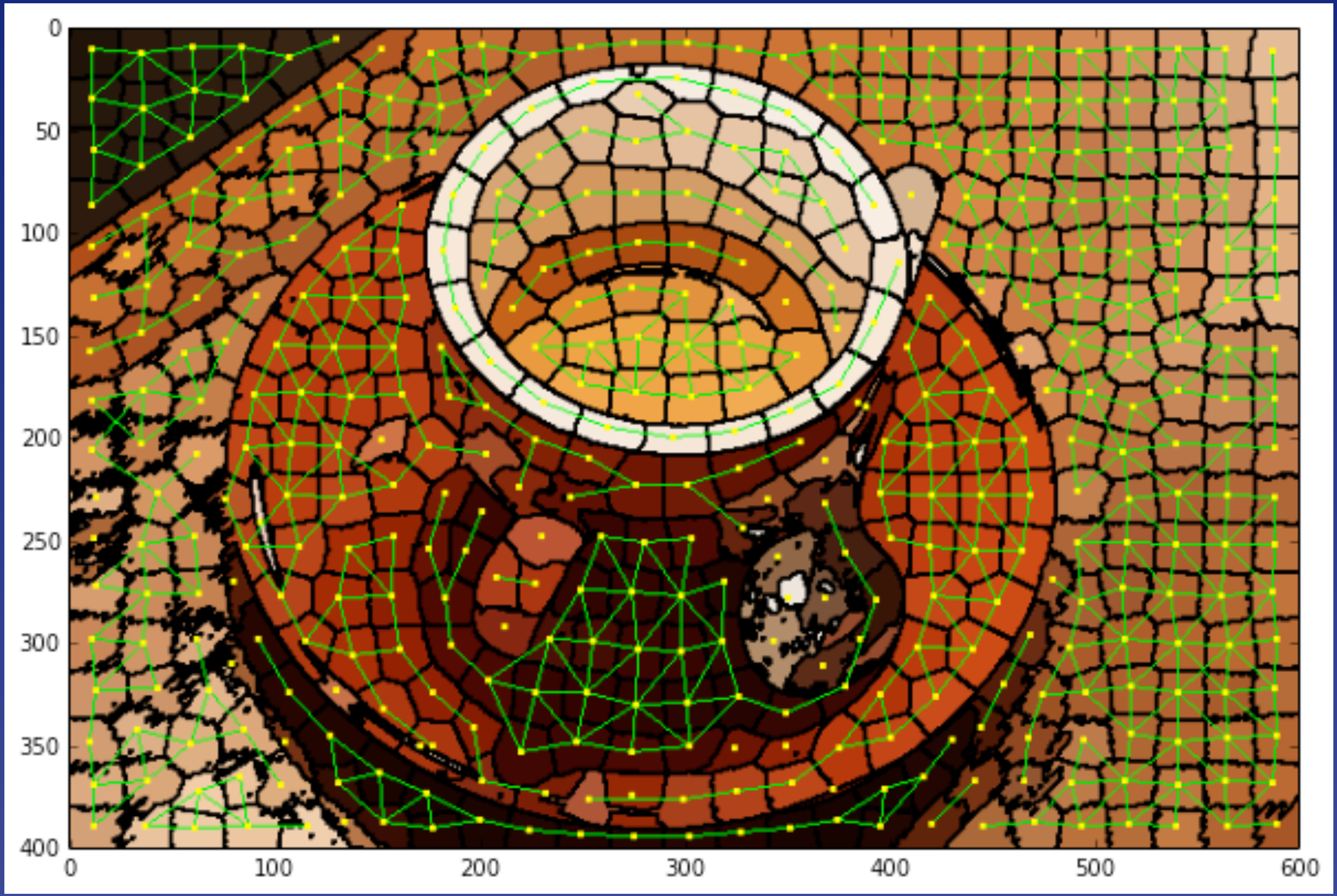






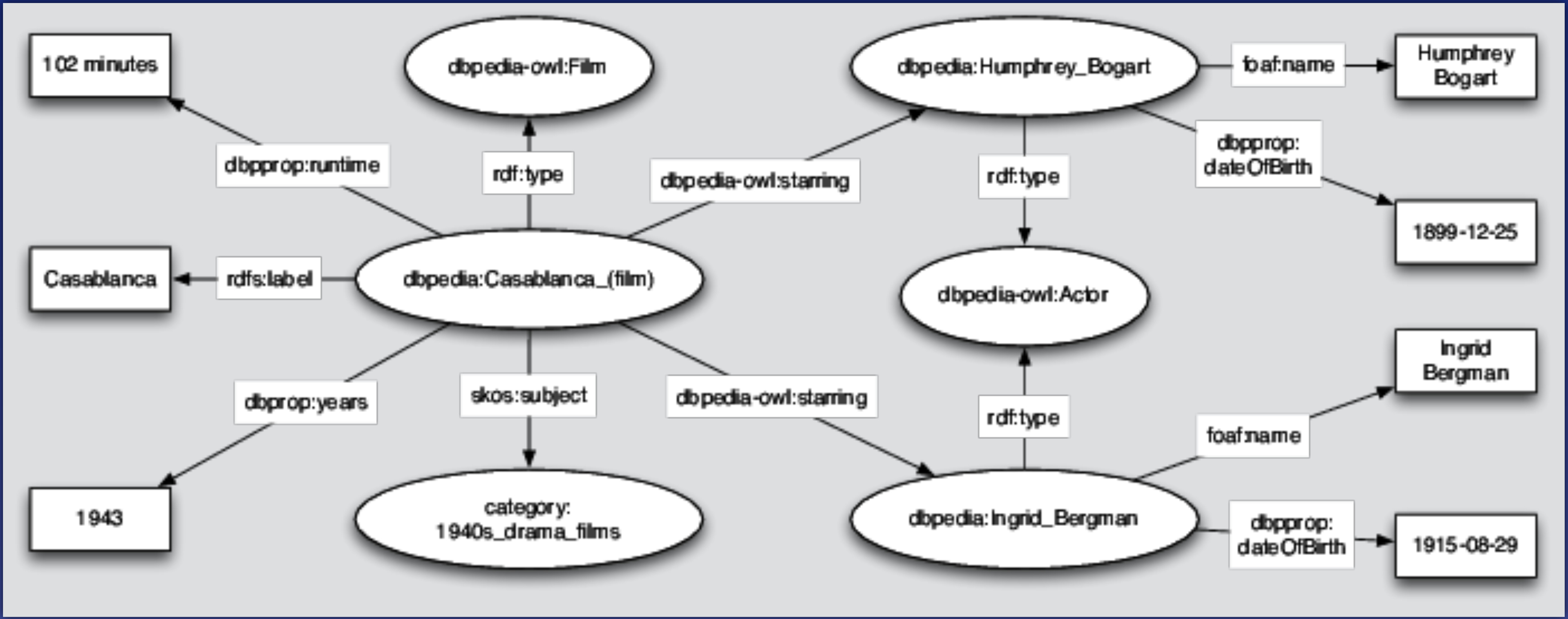
Less obvious examples

- Image segmentation



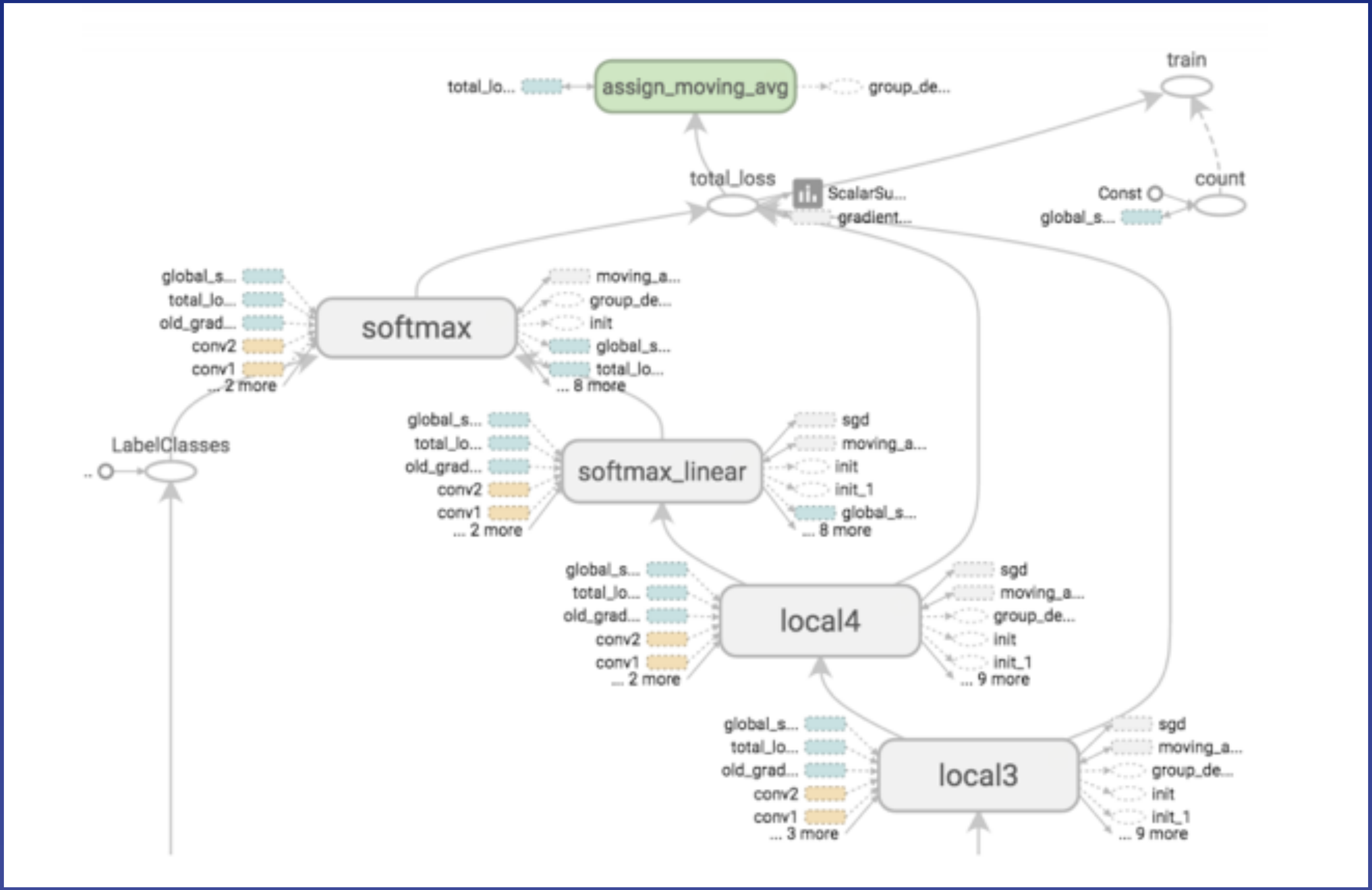
Less obvious examples

- Semantic data
  - E.g. DBpedia stores information from Wikipedia as RDF graphs



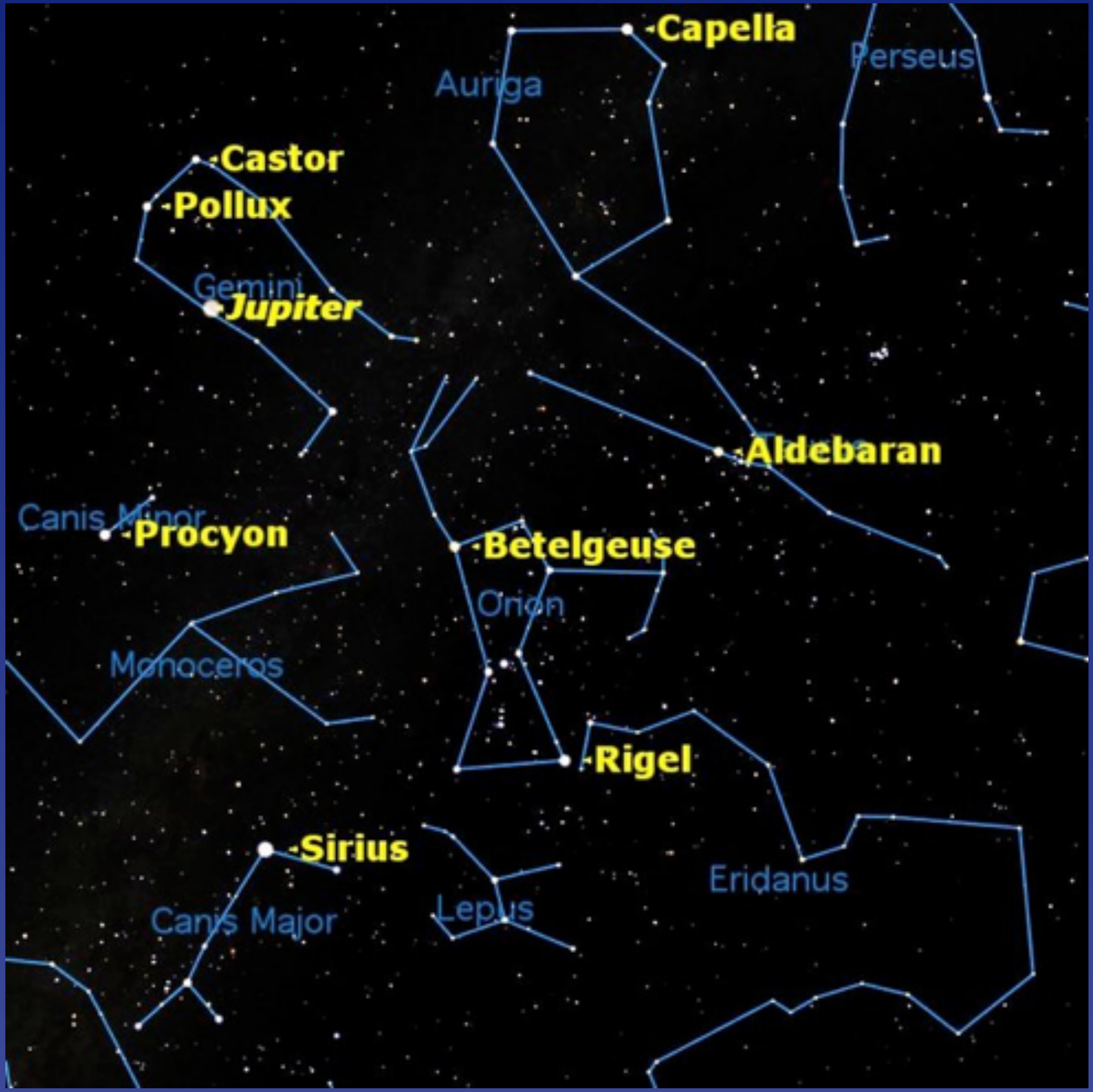
Less obvious examples

- Deep Learning optimization
  - **Tensorflow** represents the DNN operations as a dataflow graph, then uses this graph to code the computation for the available CPUs/GPUs



Less obvious examples

- … we even tend to imagine graphs when we look at the night sky!

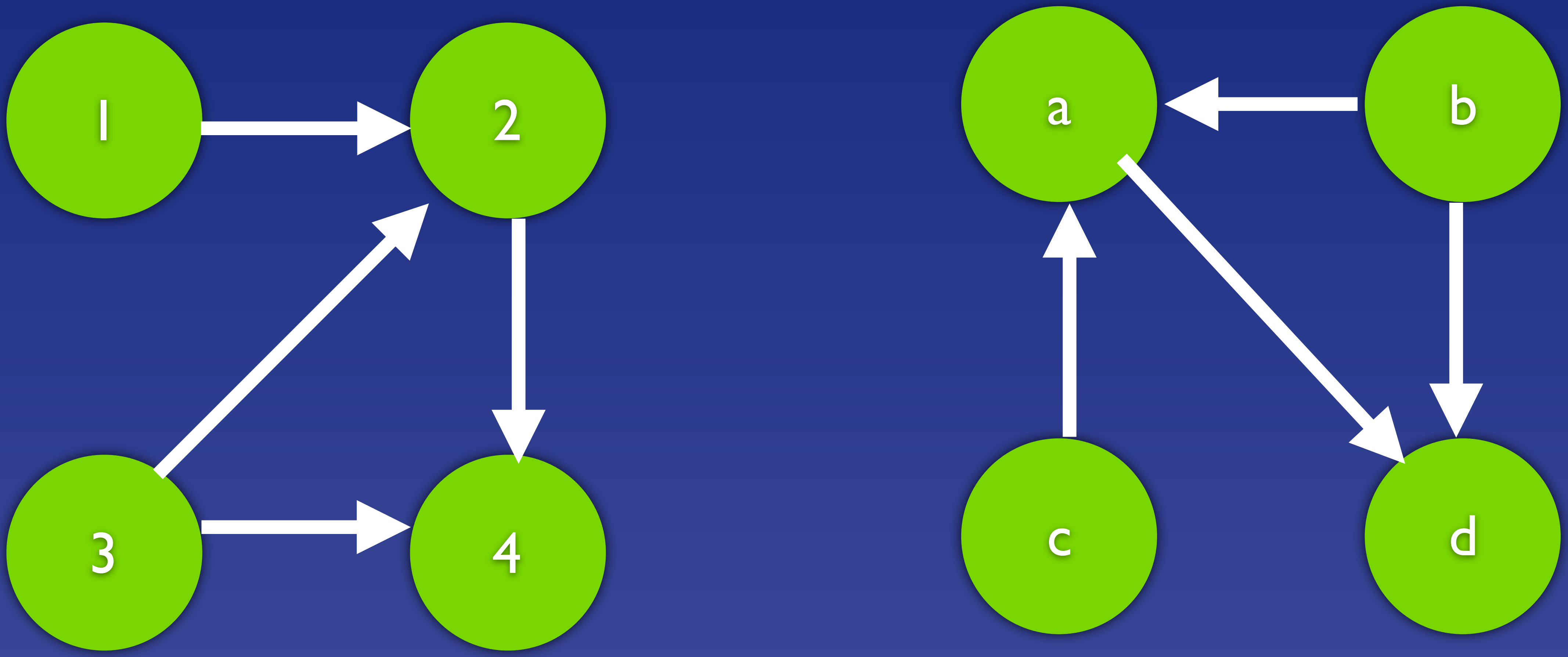


Graph matching

- A common operation on graphs is *matching*:
  - Find a correspondence between the nodes of the two graphs that preserves some interesting properties
  - Different kinds of matching depending on the properties one wants to preserve

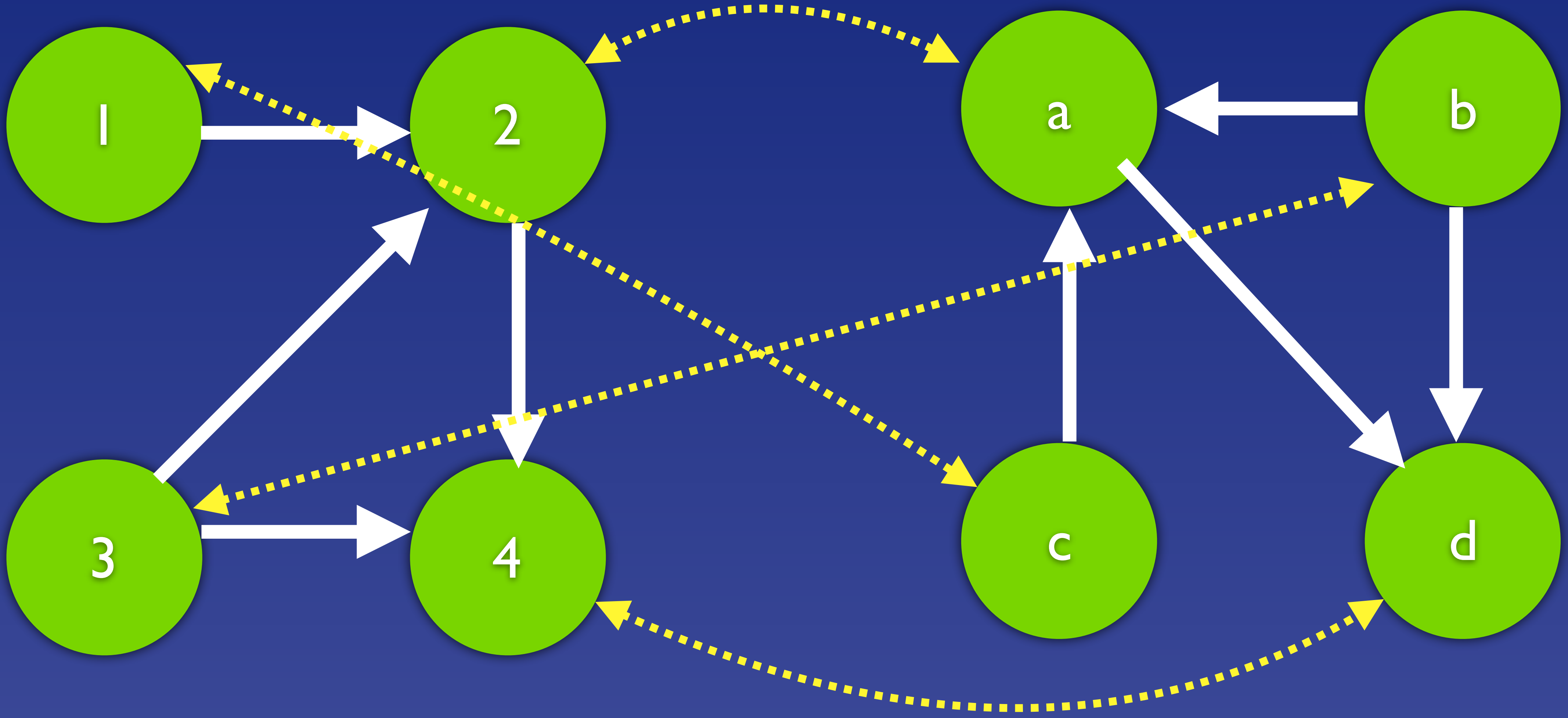
Isomorphism

- Find if two graphs have the same structure



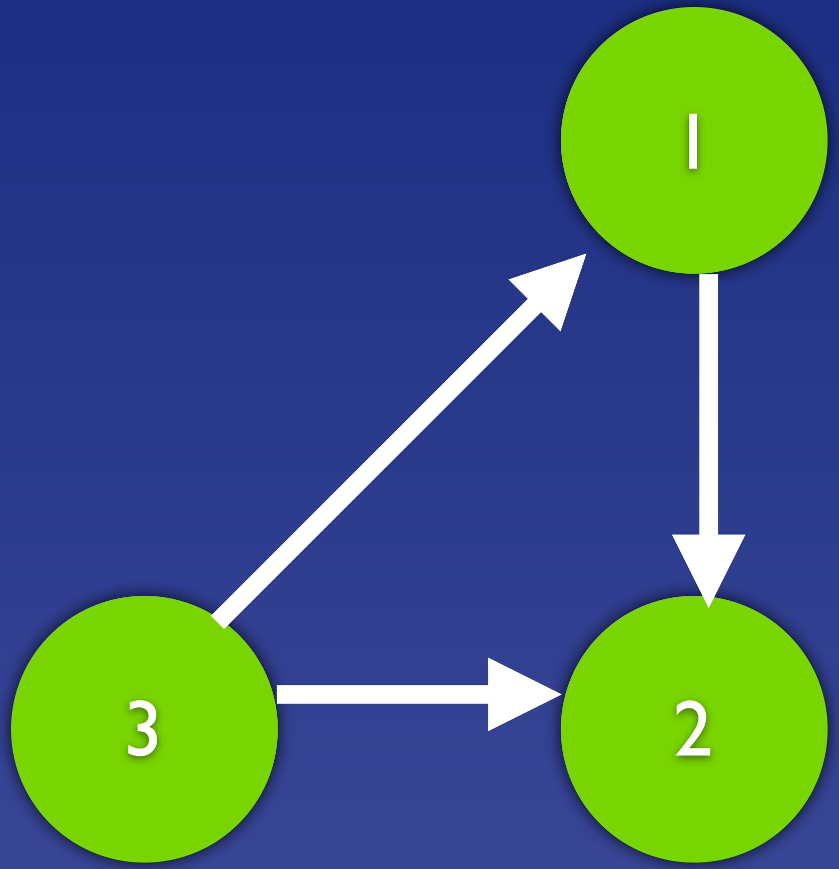
Isomorphism

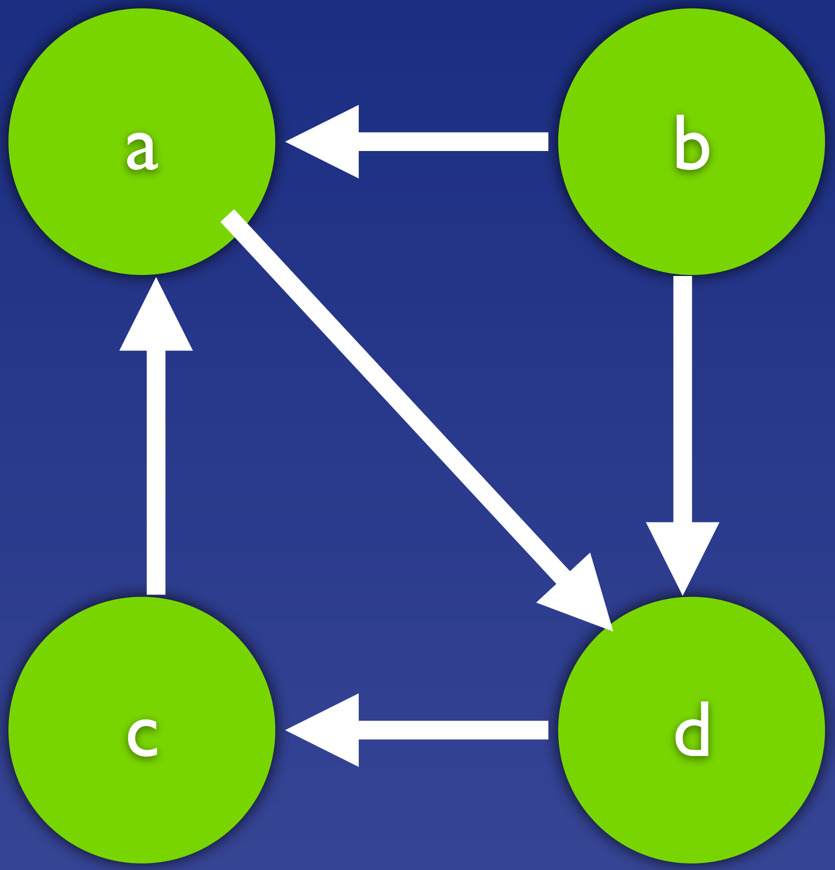
- Find if two graphs have the same structure



Subgraph isomorphism

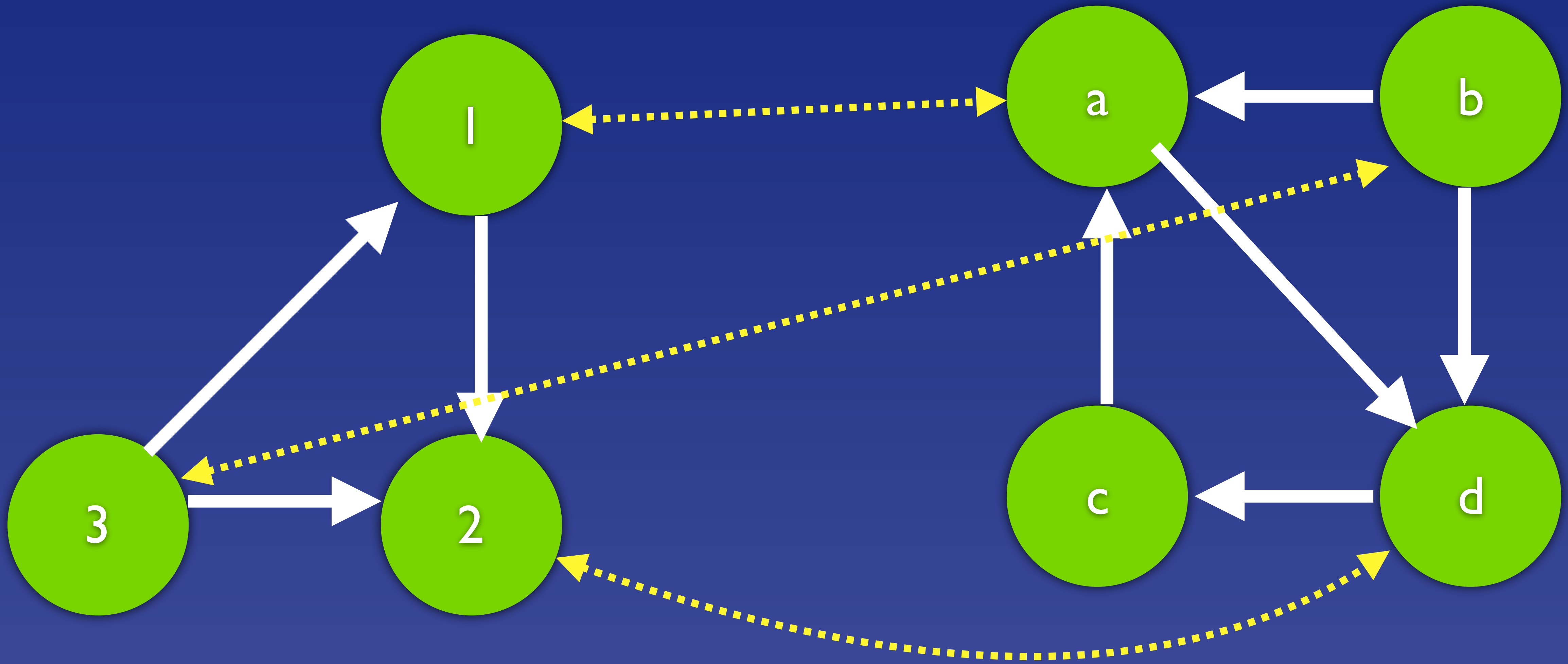
- Find if a graph contains the other as a subgraph





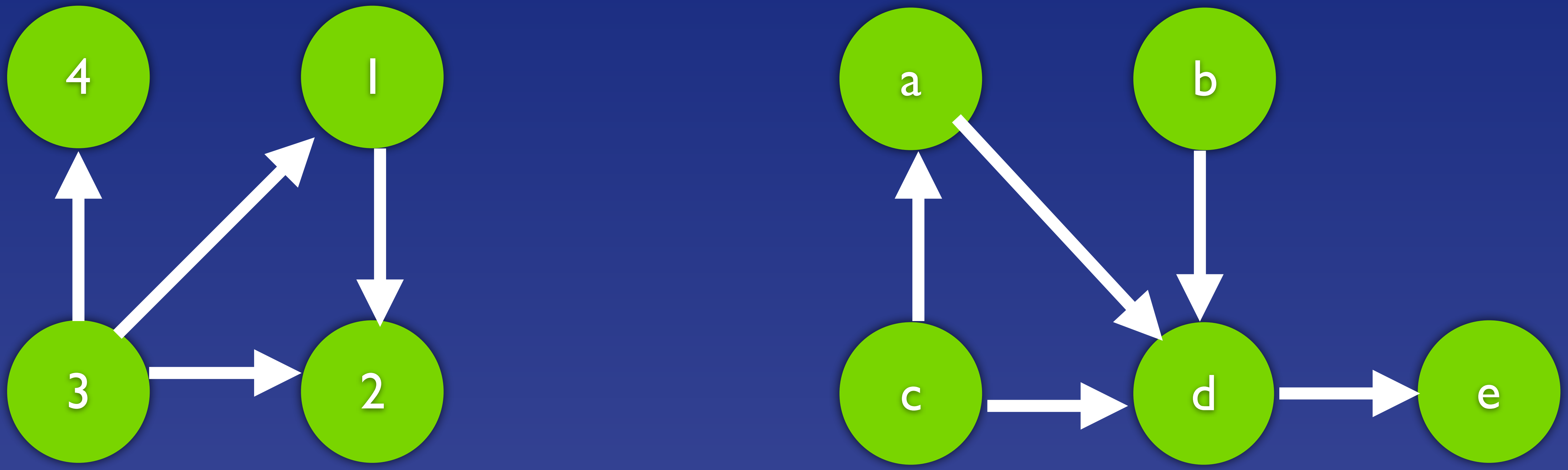
Subgraph isomorphism

- Find if a graph contains the other as a subgraph



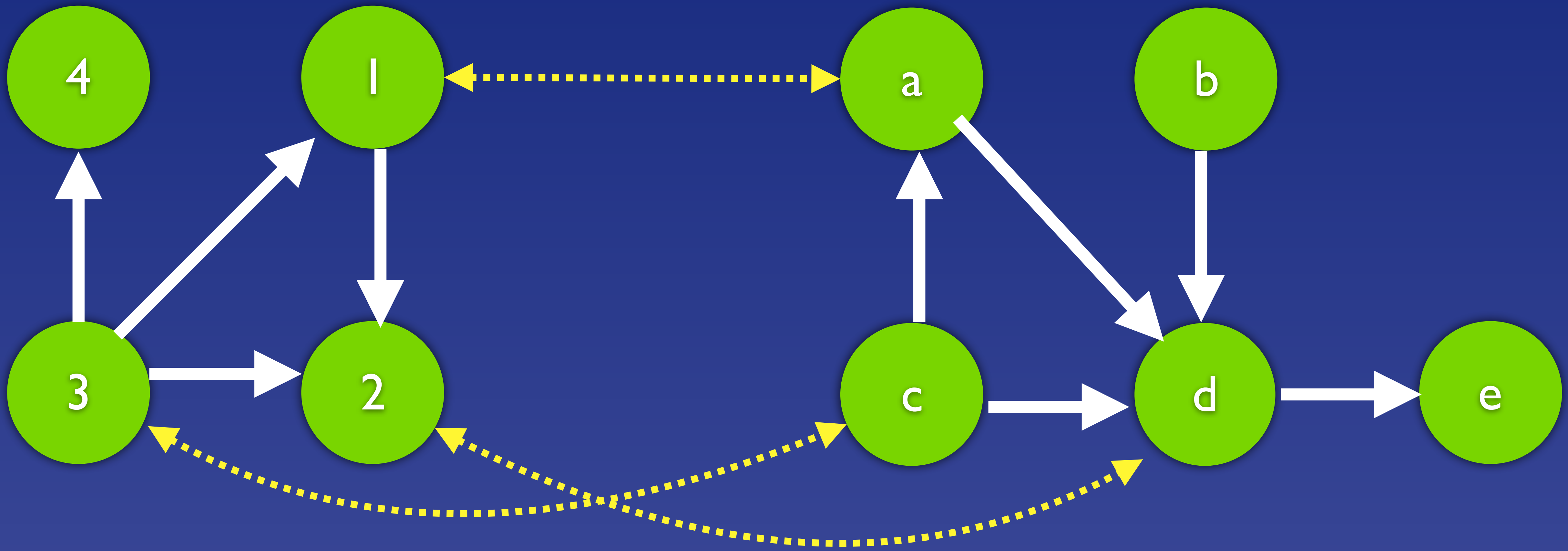
Maximum common subgraph

- Find the largest subgraph common to the two graphs



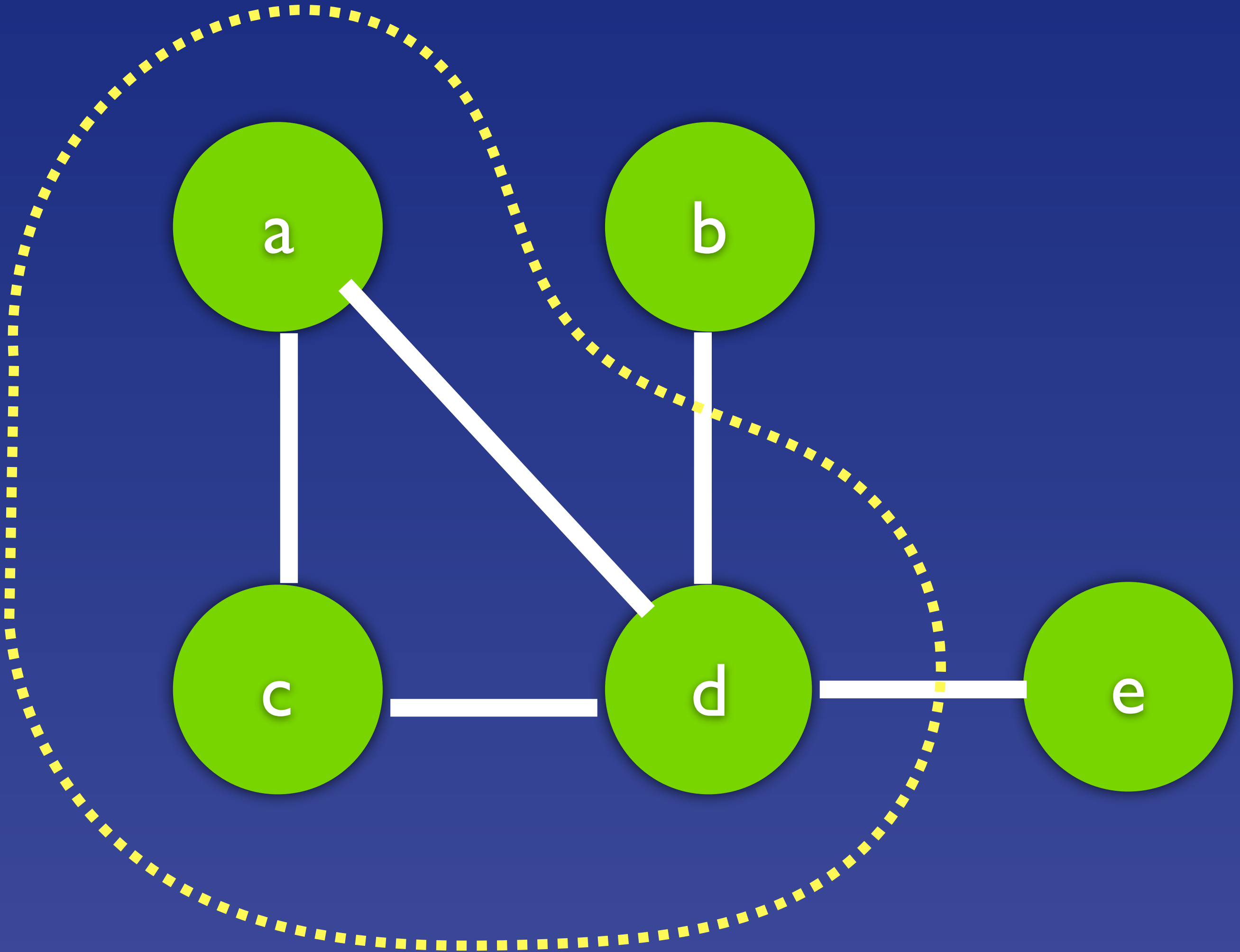
Maximum common subgraph

- Find the largest subgraph common to the two graphs



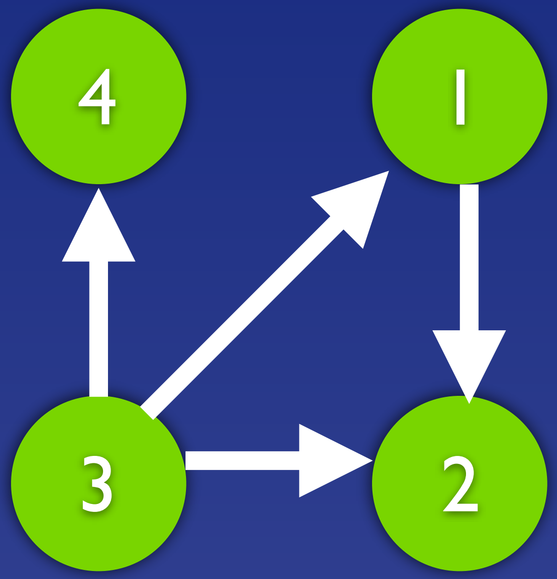
Maximum common subgraph

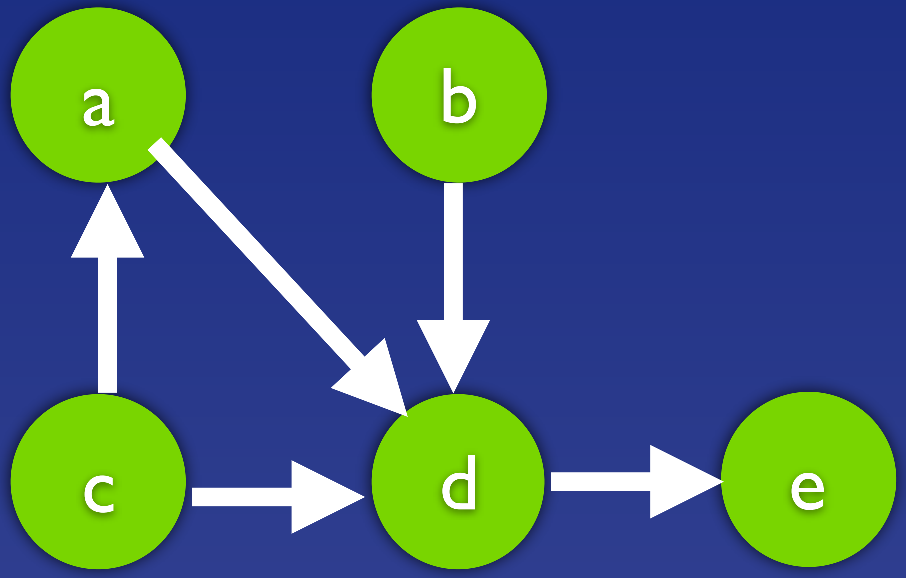
- MCS is related to the Maximum Clique problem: find the largest subgraph that is *fully connected*



Graph edit distance

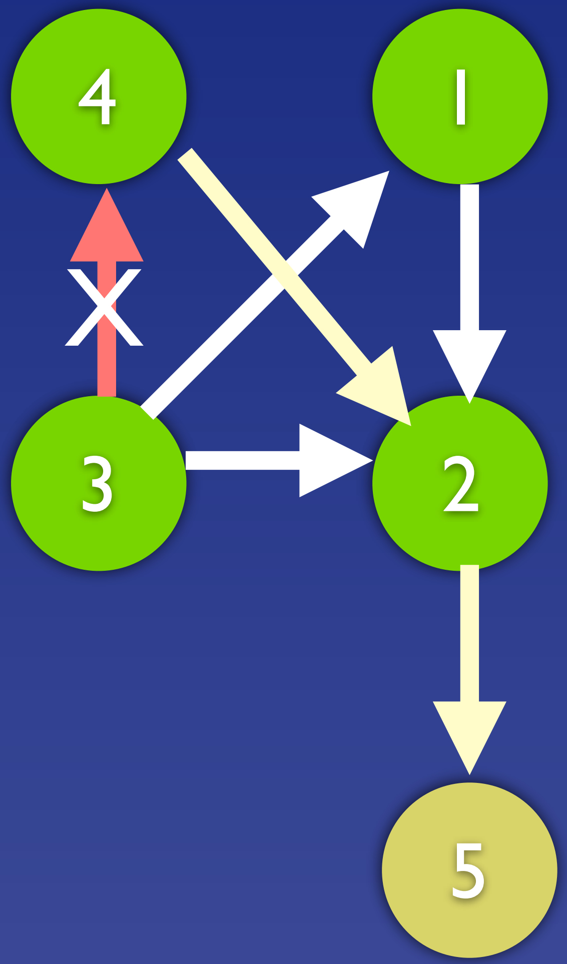
- Find the "minimum cost" set of edit operations that makes the first graph isomorphic to the second



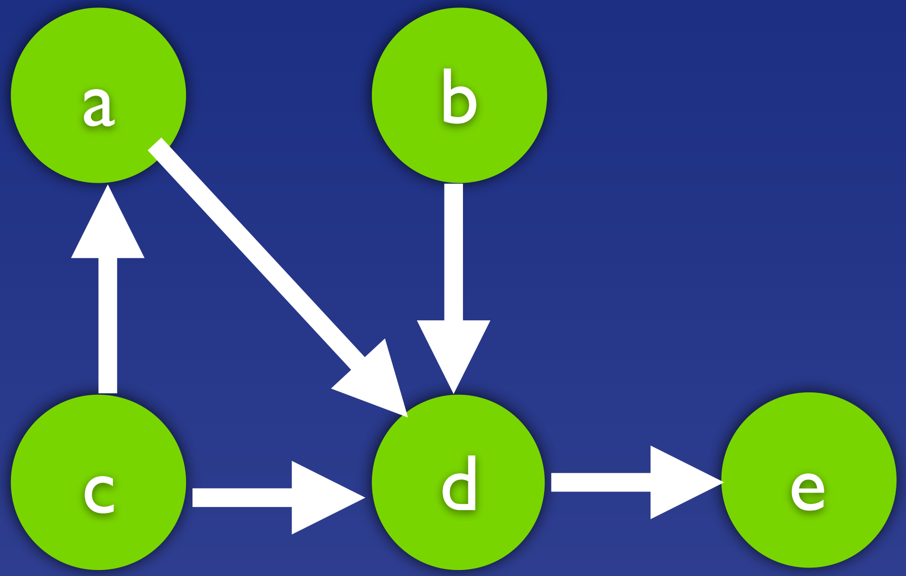


Graph edit distance

- Find the "minimum cost" set of edit operations that makes the first graph isomorphic to the second



Remove edge (3,4)
Add edge (4,2)
Add node 5
Add edge (2,5)



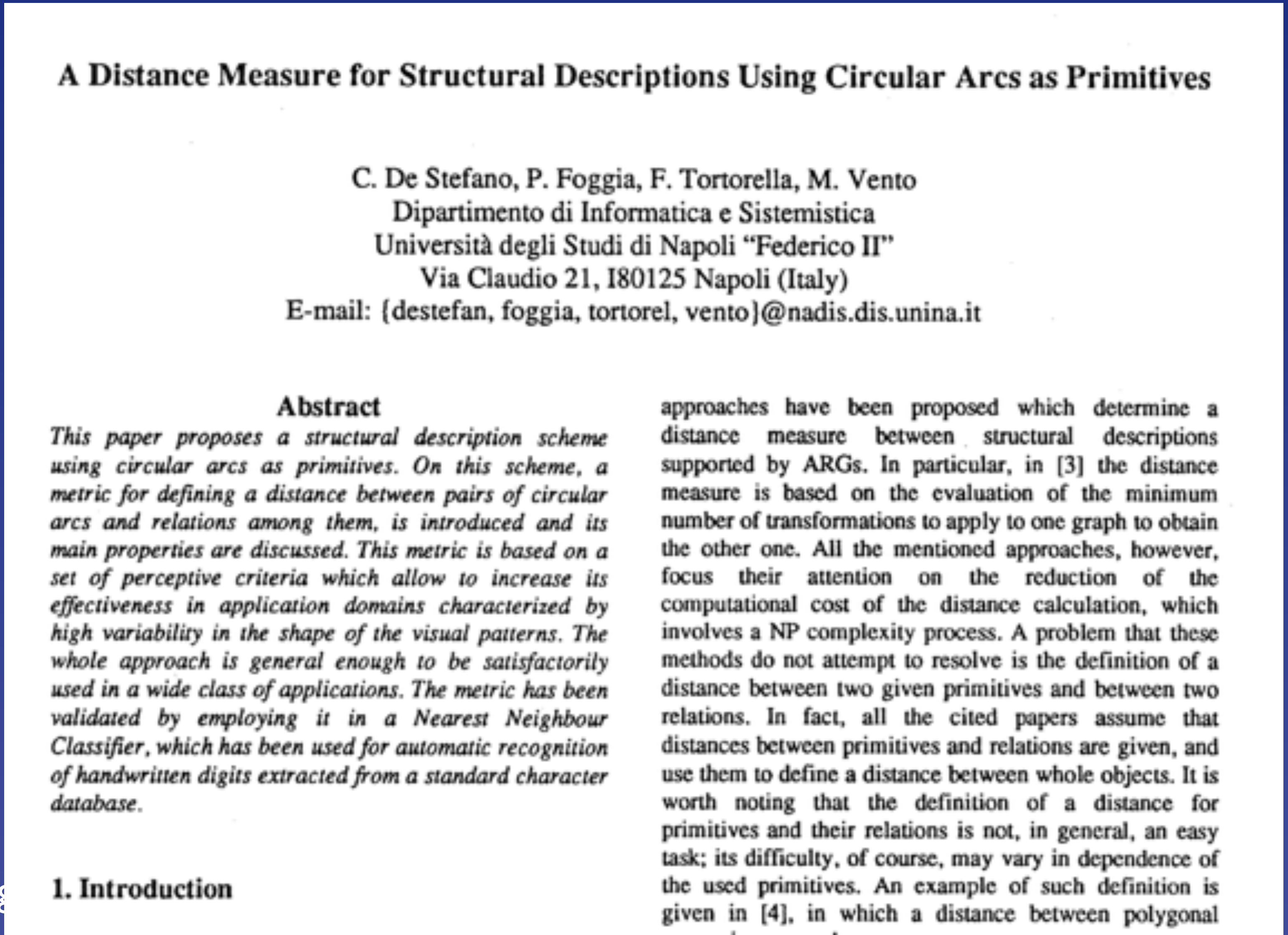
Time complexity

- All the above problems except isomorphism have been proved to be NP-Complete
- For isomorphism there is a demonstration (pending verification) that the problem is QP (Quasi-Polynomial), with complexity:

$$T(n) = 2^{O((\log n)^3)}$$

Large graphs

- Early applications of graph matching only worked with very small graphs
  - e.g. my (modest) very first scientific paper!



A Distance Measure for Structural Descriptions Using Circular Arcs as Primitives

C. De Stefano, P. Foggia, F. Tortorella, M. Vento
Dipartimento di Informatica e Sistemistica
Università degli Studi di Napoli "Federico II"
Via Claudio 21, I80125 Napoli (Italy)
E-mail: {destefan, foggia, tortorel, vento}@nadis.dis.unina.it

Abstract

This paper proposes a structural description scheme using circular arcs as primitives. On this scheme, a metric for defining a distance between pairs of circular arcs and relations among them, is introduced and its main properties are discussed. This metric is based on a set of perceptive criteria which allow to increase its effectiveness in application domains characterized by high variability in the shape of the visual patterns. The whole approach is general enough to be satisfactorily used in a wide class of applications. The metric has been validated by employing it in a Nearest Neighbour Classifier, which has been used for automatic recognition of handwritten digits extracted from a standard character database.

1. Introduction

approaches have been proposed which determine a distance measure between structural descriptions supported by ARGs. In particular, in [3] the distance measure is based on the evaluation of the minimum number of transformations to apply to one graph to obtain the other one. All the mentioned approaches, however, focus their attention on the reduction of the computational cost of the distance calculation, which involves a NP complexity process. A problem that these methods do not attempt to resolve is the definition of a distance between two given primitives and between two relations. In fact, all the cited papers assume that distances between primitives and relations are given, and use them to define a distance between whole objects. It is worth noting that the definition of a distance for primitives and their relations is not, in general, an easy task; its difficulty, of course, may vary in dependence of the used primitives. An example of such definition is given in [4], in which a distance between polygonal

Large graphs

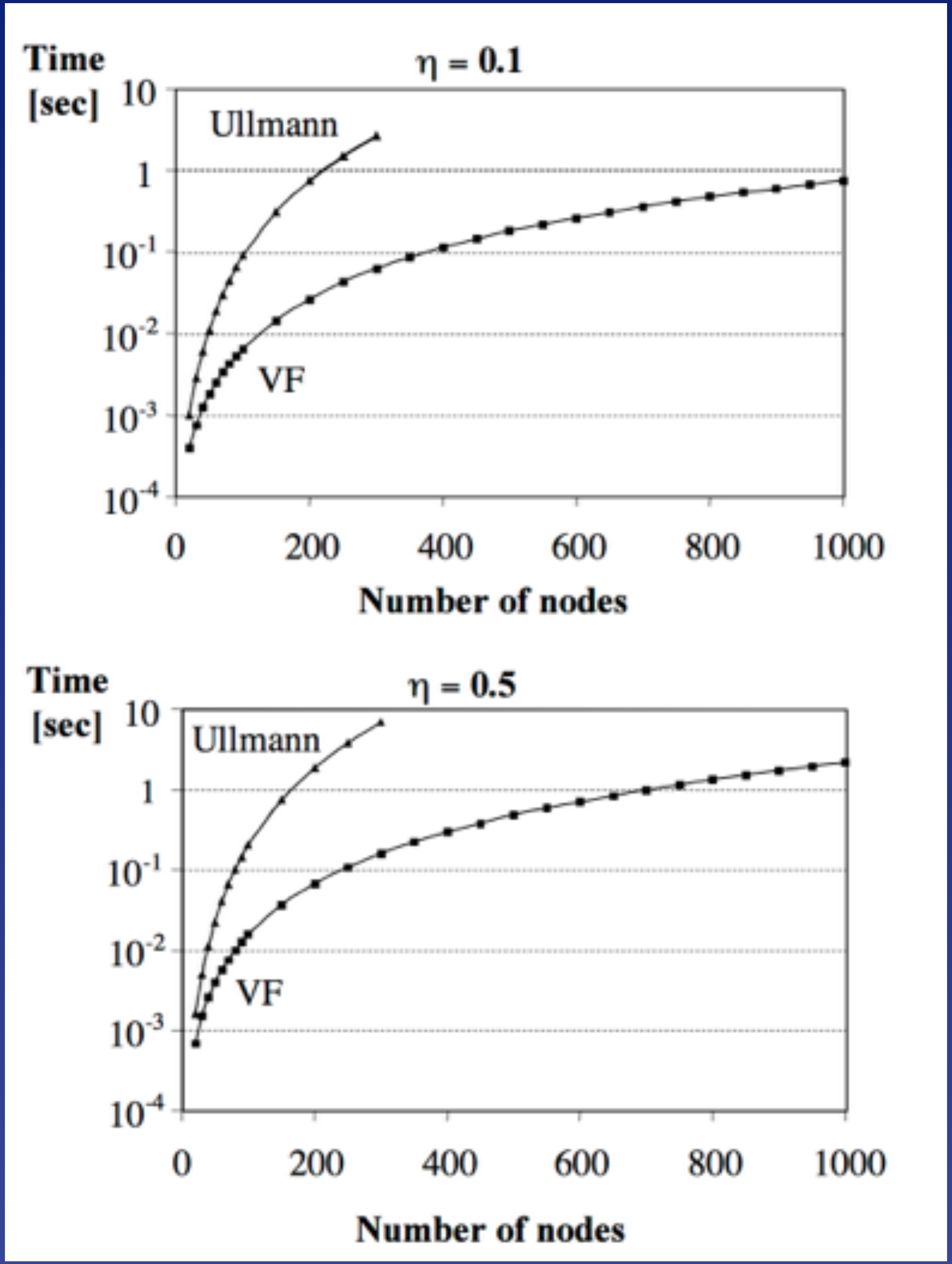
- Initial generation of algorithms able to work with graphs ranging from tens of nodes (graph edit distance) to a hundred nodes (subgraph isomorphism)
- In the late '90s and early 2000s several works addressed the problem of large graphs

Subgraph isomorphism

- ✦ The most popular subgraph isomorphism algorithm was Ullmann's (1976)
- ✦ Ullmann required a memory space that is cubic ($O(n^3)$) wrt to the size of the graphs
  - Only suitable up to a few hundred nodes

Subgraph isomorphism

- In 1999 we proposed the VF algorithm, that requires $O(n^2)$ space
  - Cordella, L.P., Foggia, P., Sansone, C., Vento, M. Performance evaluation of the VF graph matching algorithm (1999) Proc. ICIAP 1999, pp. 1172-1177



Subgraph isomorphism

- In 2004 we proposed the VF2 algorithm, that requires O(n) space
  - Cordella, L.P., Foggia, P., Sansone, C., Vento, M. A (sub)graph isomorphism algorithm for matching large graphs (2004) IEEE Trans PAMI, 26 (10), pp. 1367-1372.

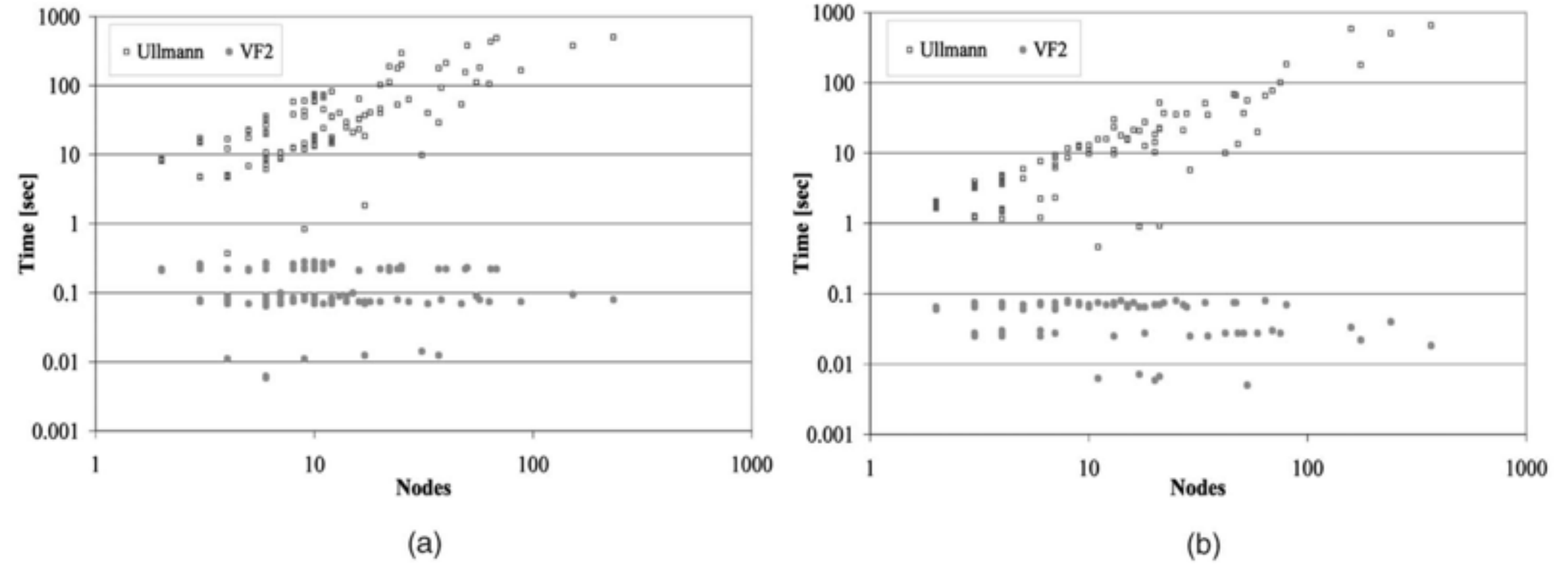


Fig. 3. Matching times for the subgraphs of the MAP-1 image (a) and of the ENGINE-2 image (b).

Graph edit distance

- In 2007, Riesen, Neuhaus and Bunke proposed a method for *approximating* the GED using the Linear Assignment Problem, that can be solved in $O(n^3)$ time
  - Riesen K., Neuhaus M., Bunke H. (2007) Bipartite Graph Matching for Computing the Edit Distance of Graphs. In: Escolano F., Vento M. (eds) Graph-Based Representations in Pattern Recognition. GbRPR 2007. Lecture Notes in Computer Science, vol 4538.

Graph edit distance

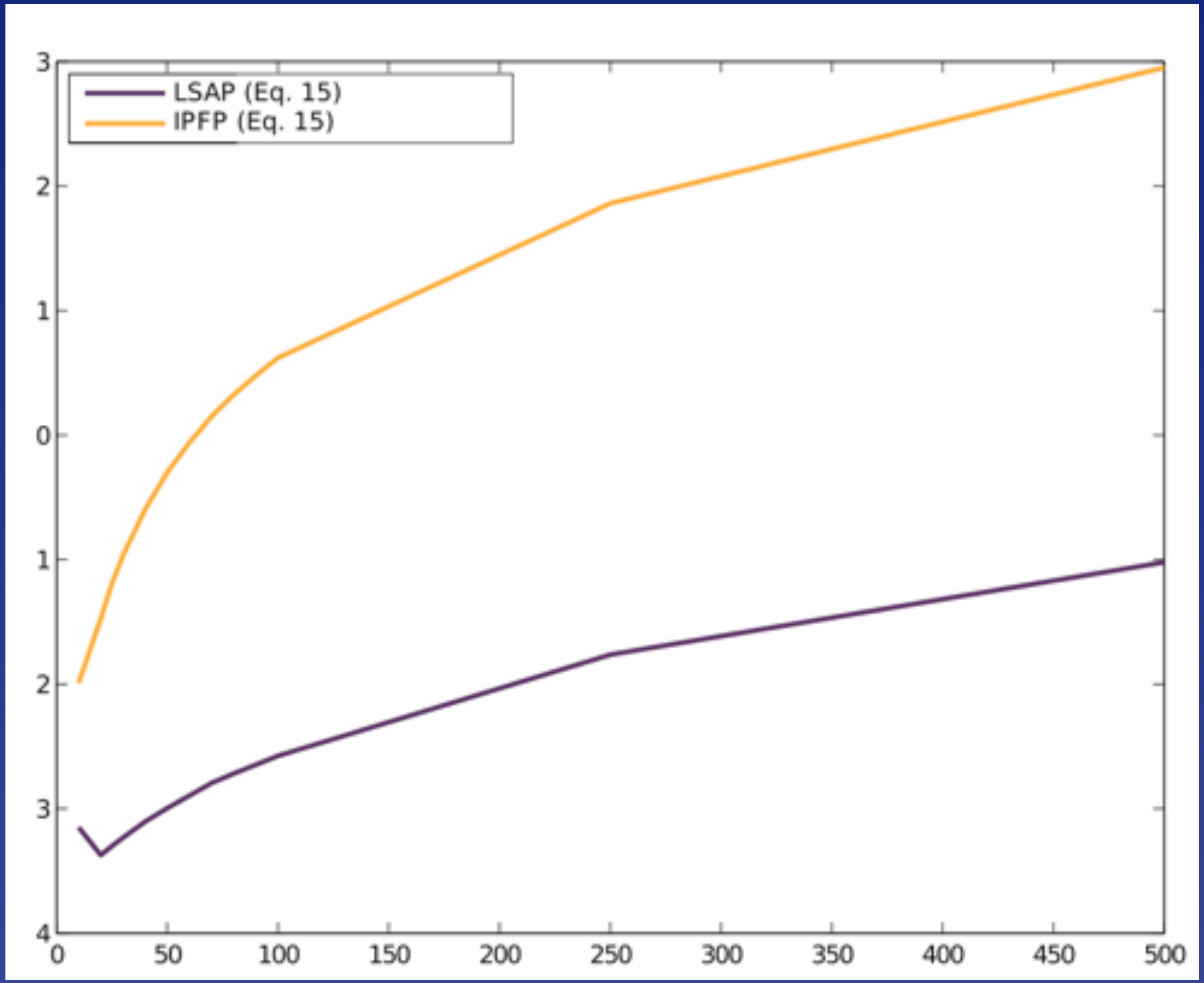
- In 2012, we used approximate GED for analyzing the dynamics of socio-economic networks
  - V Carletti, D De Stefano, M Fattore, P Foggia, R Grassi, Dynamical analysis of interlocking directorates using graph edit distance, International Workshop on Network Models in Statistics (2012)
  - Applied to the network induced by mutual participation in the board of directors of italian firms in the industrial, financial and publishing sector from 2007 to 2011, to detect structural changes

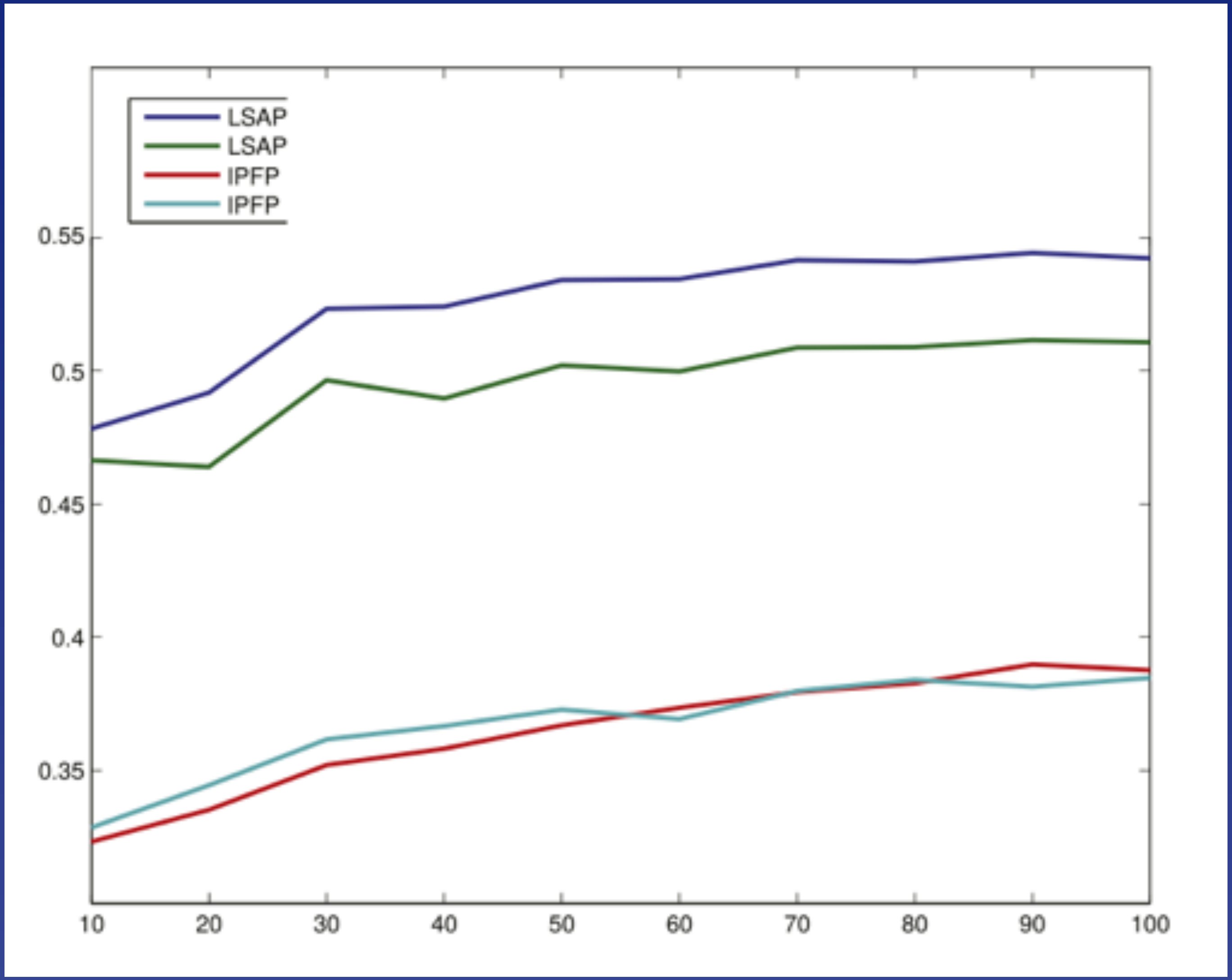
Graph edit distance

- In 2017, we proposed a better but slower approximation using the Quadratic Assignment Problem
  - Bougleux, S., Brun, L., Carletti, V., Foggia, P., Gaüzère, B., Vento, M. Graph edit distance as a quadratic assignment problem (2017) Pattern Recognition Letters, 87, pp. 38-46.

Graph edit distance

- LAP vs QAP





Time

Error

Large enough?

- Although the above mentioned algorithms made possible to use larger graphs than before, new applications raised the bar regarding what “large” means

Examples: bioinformatics

- Protein Data Bank: world wide collection of information on proteins
  - Protein chemical structures: 500 to 10000 nodes, very sparse (max degree 4)
  - Protein contact maps: 150 to 800 nodes, but denser (average degree 20)
  - Protein interaction networks:
    - Yeast: ~3000 nodes, ~12000 edges
    - Human: ~4600 nodes, ~86000 edges

Examples: social networks

- Social circles in Facebook (ego-Facebook, 2012)
  - ~4000 nodes representing (anonymized) Fb users, with ~90000 undirected edges representing friendship
- Wikipedia Admin elections (wiki-Vote, 2010)
  - ~7000 nodes representing (anonymized) Wikipedia users, with ~100000 directed edges representing votes for admin role

Sequential approach

- Several algorithms have been proposed in the 2010s, that try to improve over 2nd generation matching algorithms using better heuristics (making assumption on the graph properties)

RI

- Bonnici and Giugno proposed RI, a subgraph isomorphism algorithm suited to large and sparse graphs in bioinformatics
  - Bonnici, V., Giugno, R.: On the variable ordering in subgraph isomorphism algorithms. IEEE/ACM Trans. Comput. Biol. Bioinform. (2016)

VF3

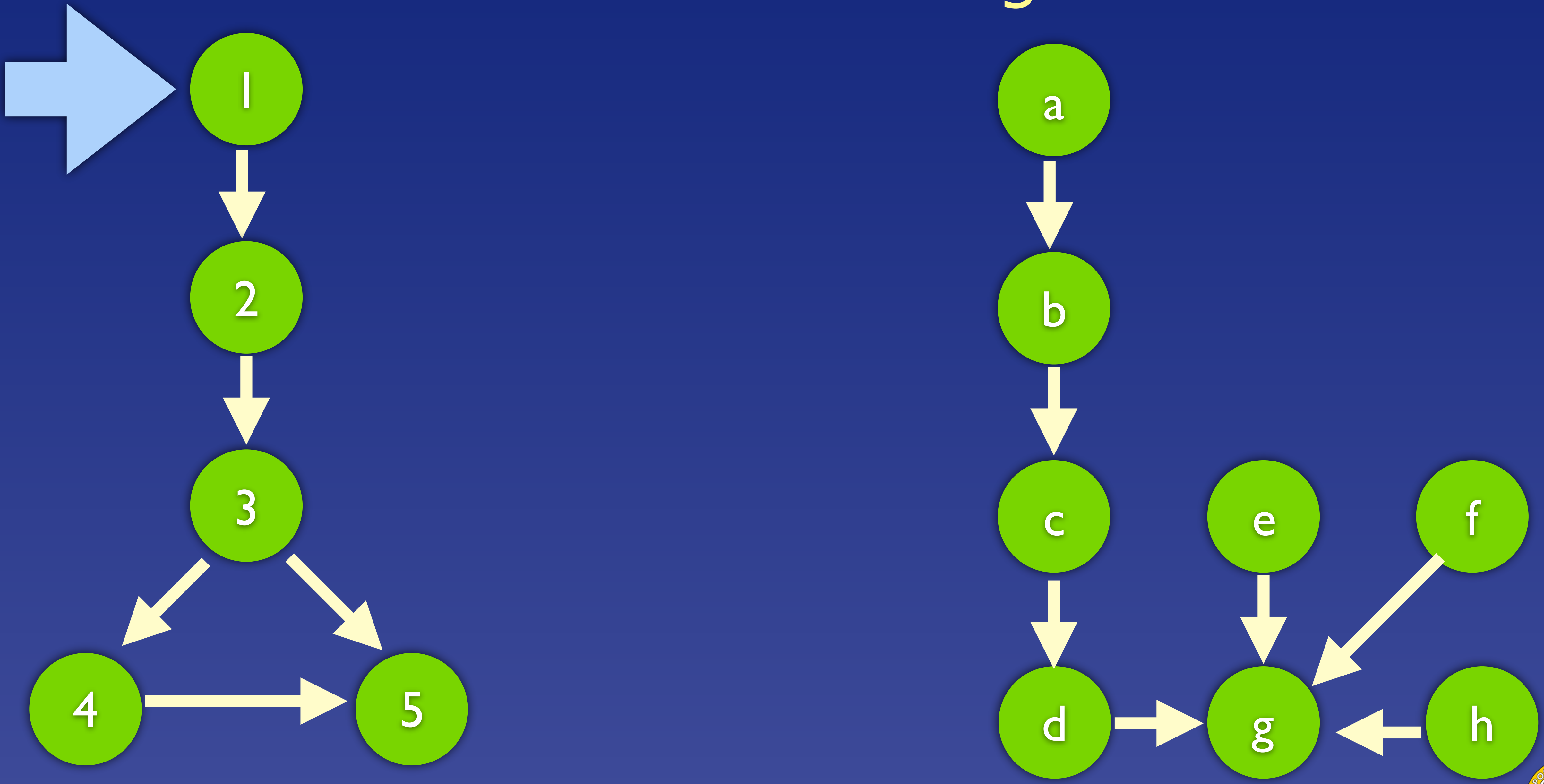
- We proposed VF3, an evolution of VF2 better suited to large and dense graphs
  - V. Carletti, P. Foggia, A. Saggese and M. Vento, Challenging the Time Complexity of Exact Subgraph Isomorphism for Huge and Dense Graphs with VF3, in IEEE Trans. on PAMI, vol. 40, no. 4, pp. 804-818, 2018

Node reordering

- Both RI and VF3 use the idea of reordering the nodes before starting the matching so as to detect as early as possible if a tentative matching is going to fail
  - The ordering criteria are different
  - VF3 also has additional heuristics for dense graphs

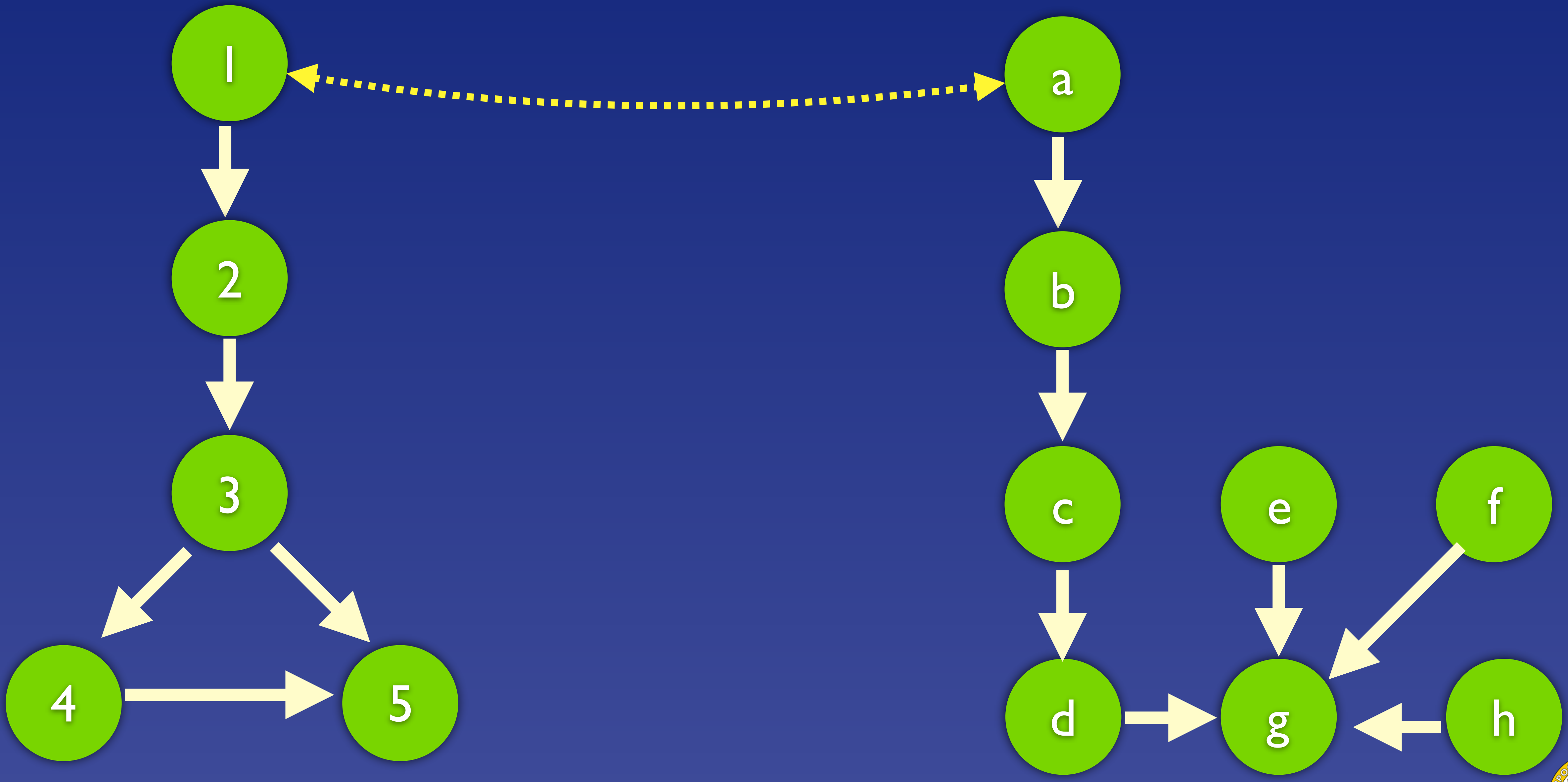
Node reordering

- Starting here will take several steps before discovering that the small graph is not contained in the large one



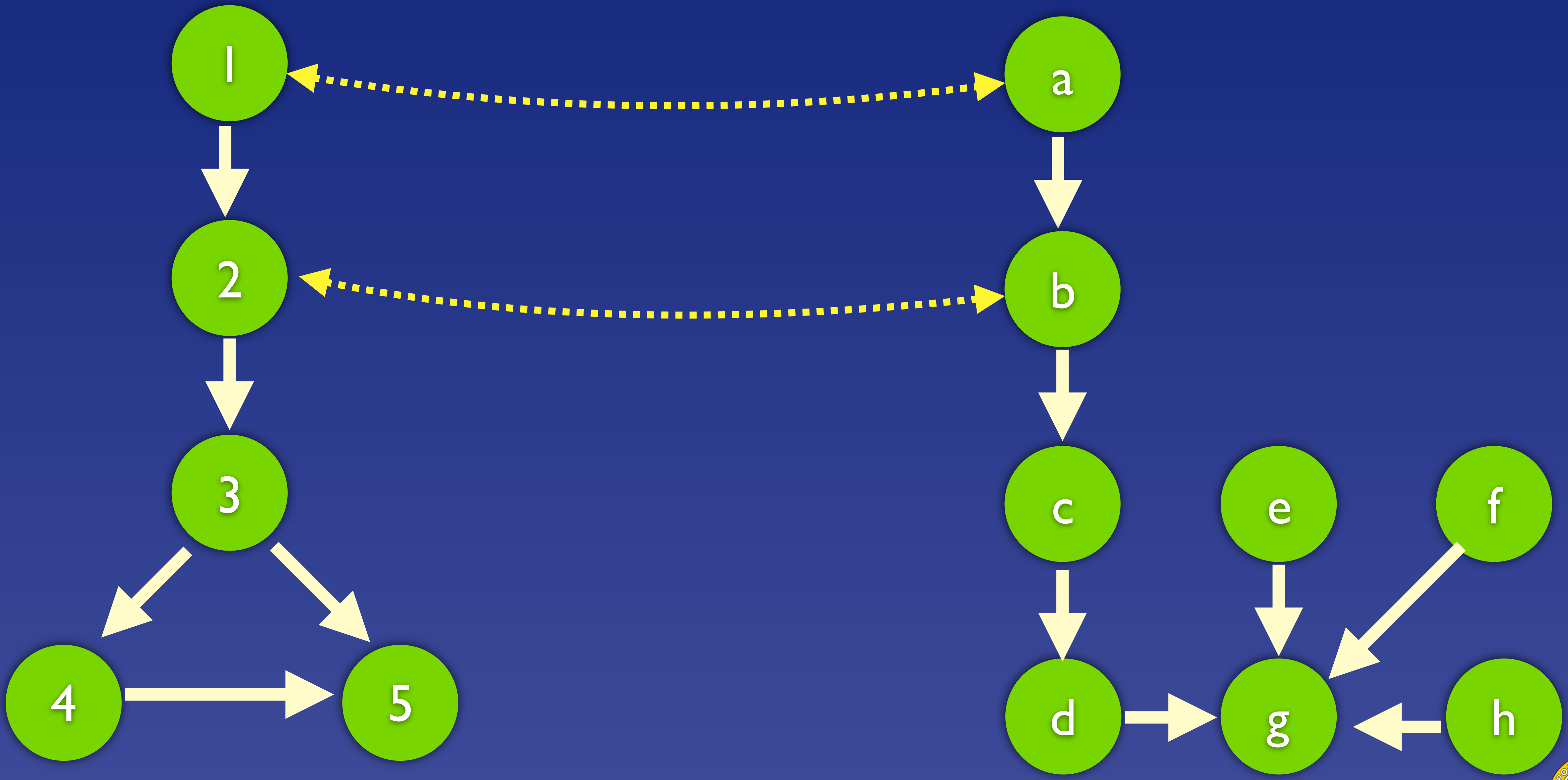
Node reordering

- Starting here will take several steps before discovering that the small graph is not contained in the large one



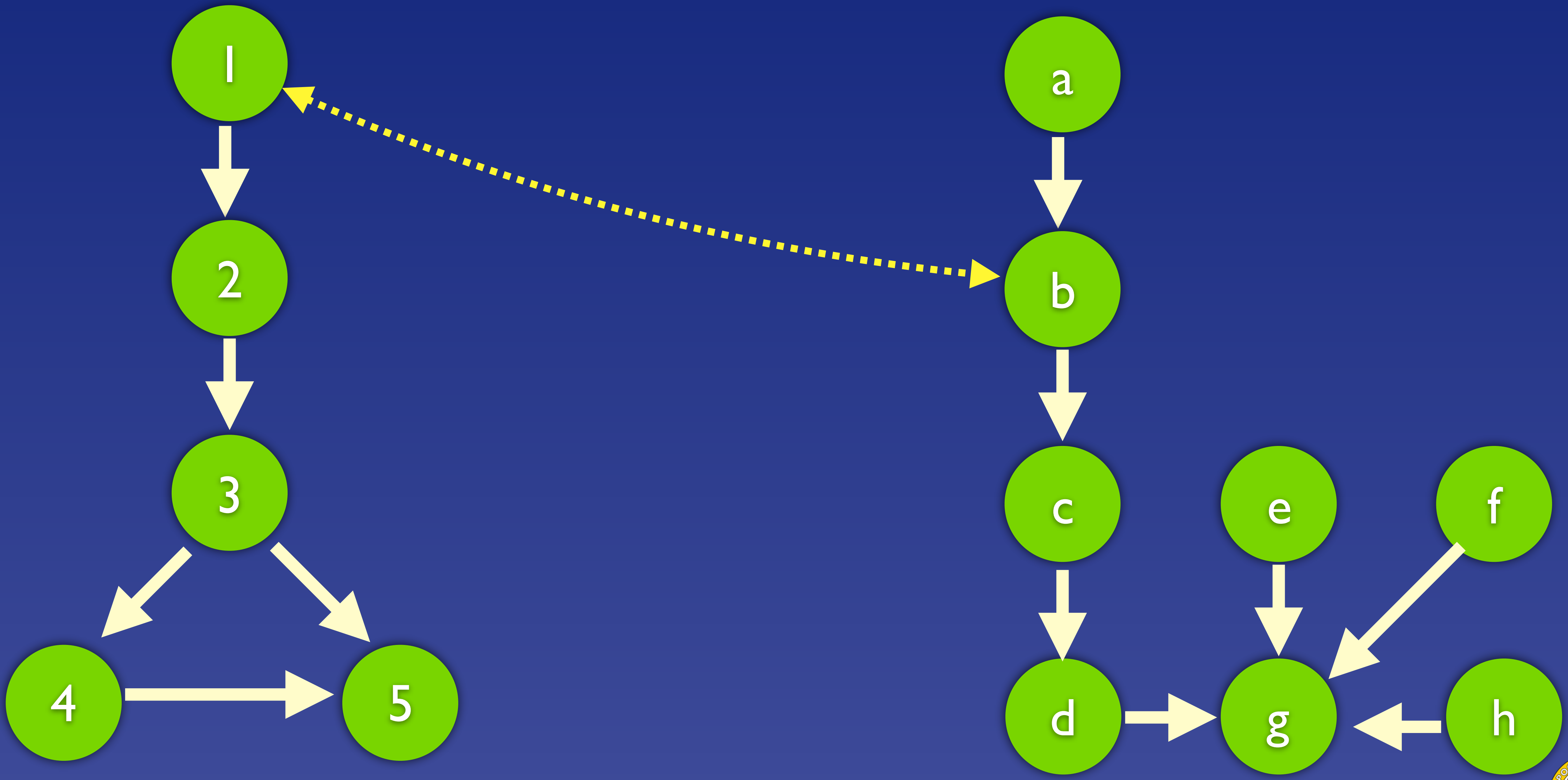
Node reordering

- Starting here will take several steps before discovering that the small graph is not contained in the large one



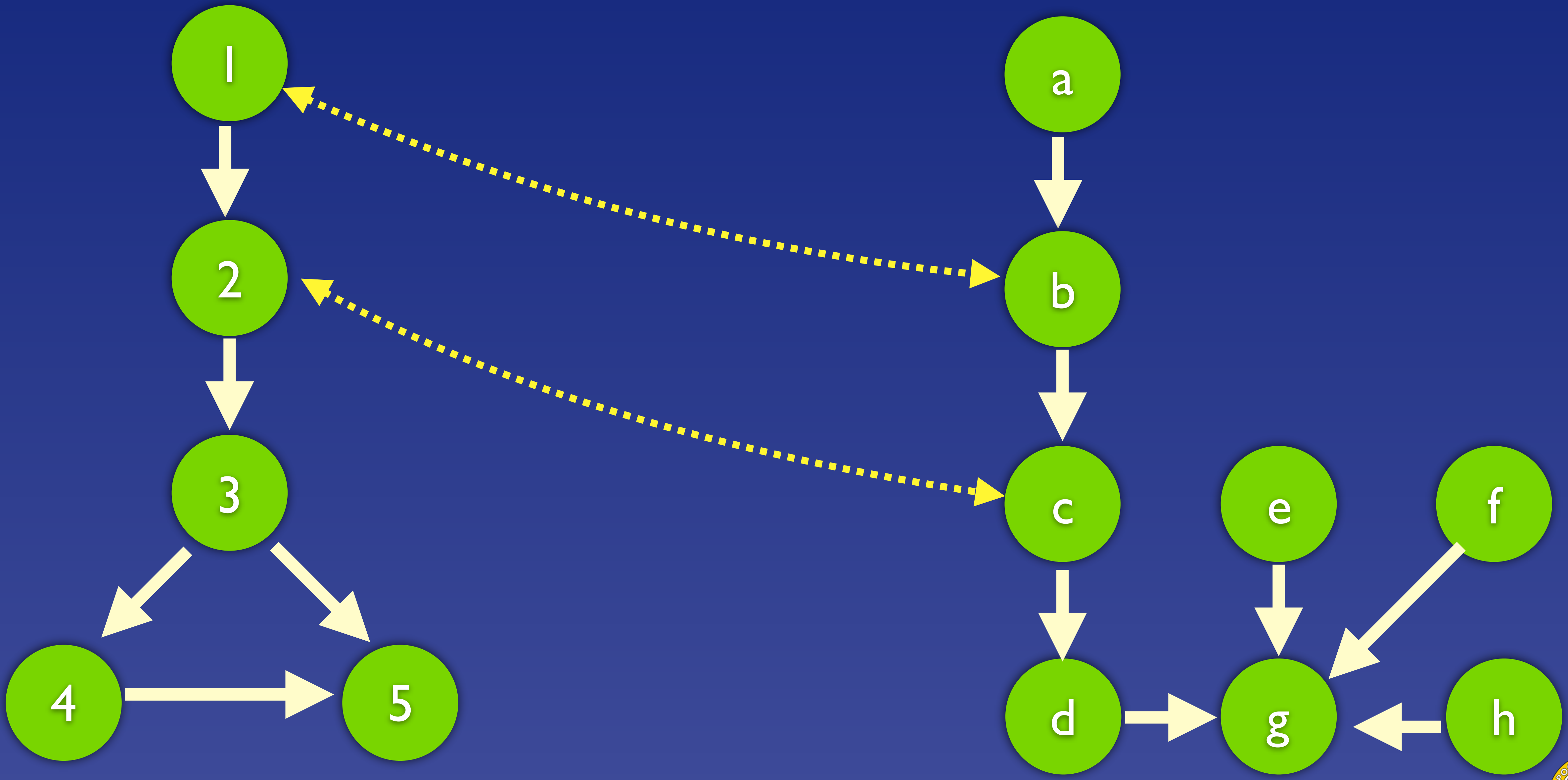
Node reordering

- Starting here will take several steps before discovering that the small graph is not contained in the large one



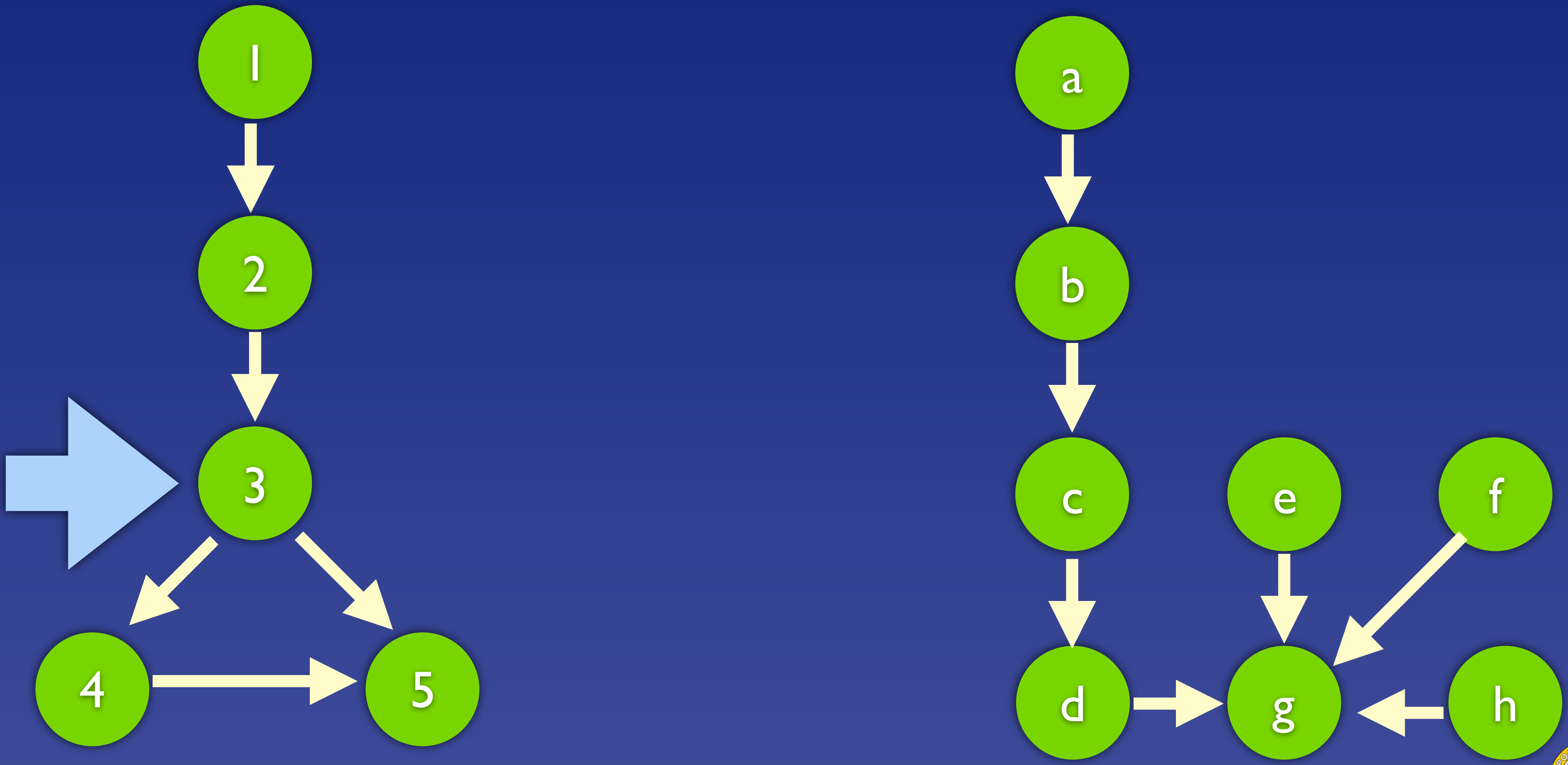
Node reordering

- Starting here will take several steps before discovering that the small graph is not contained in the large one

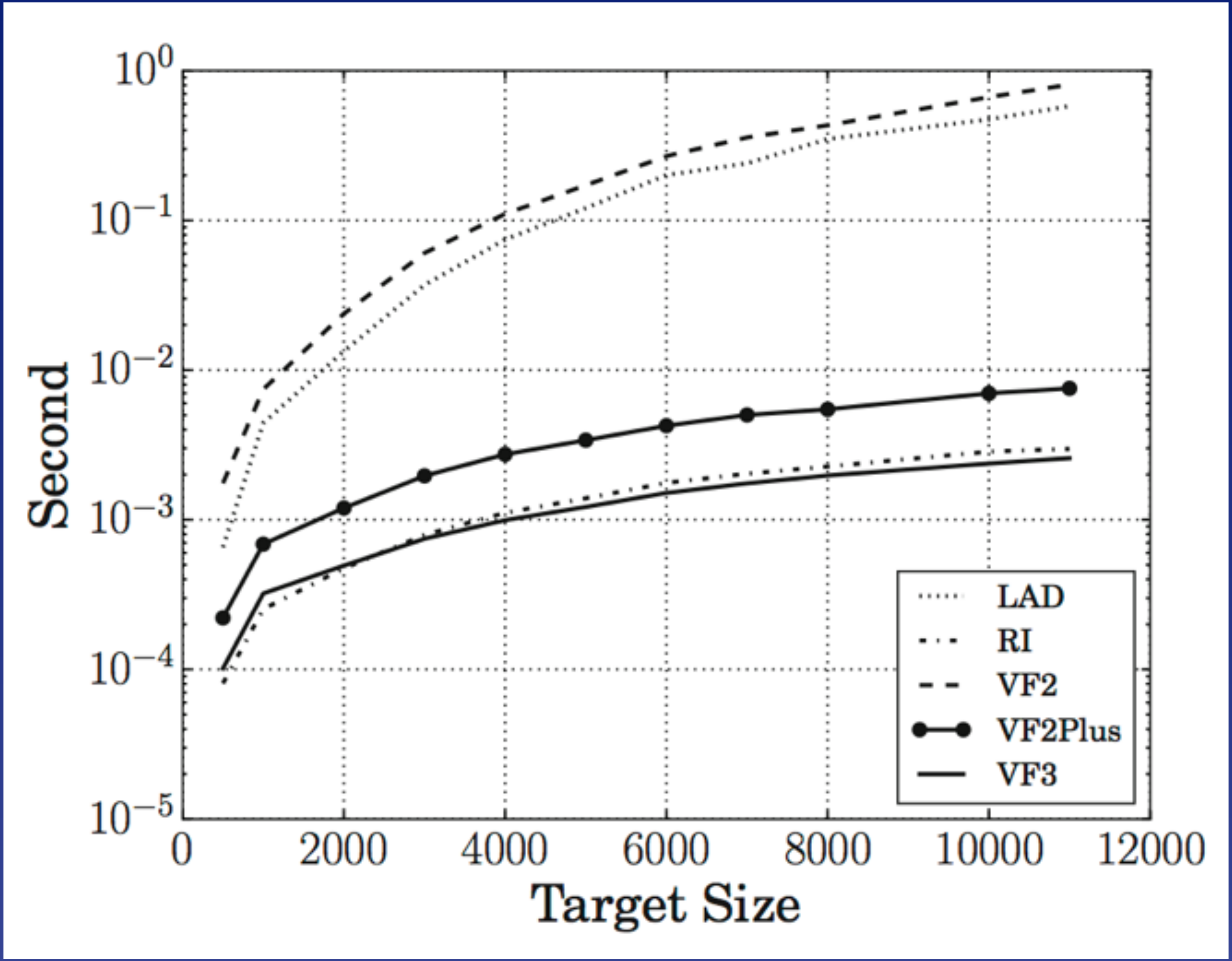


Node reordering

- Instead, starting here will immediately discover that no node can be associated to "3"

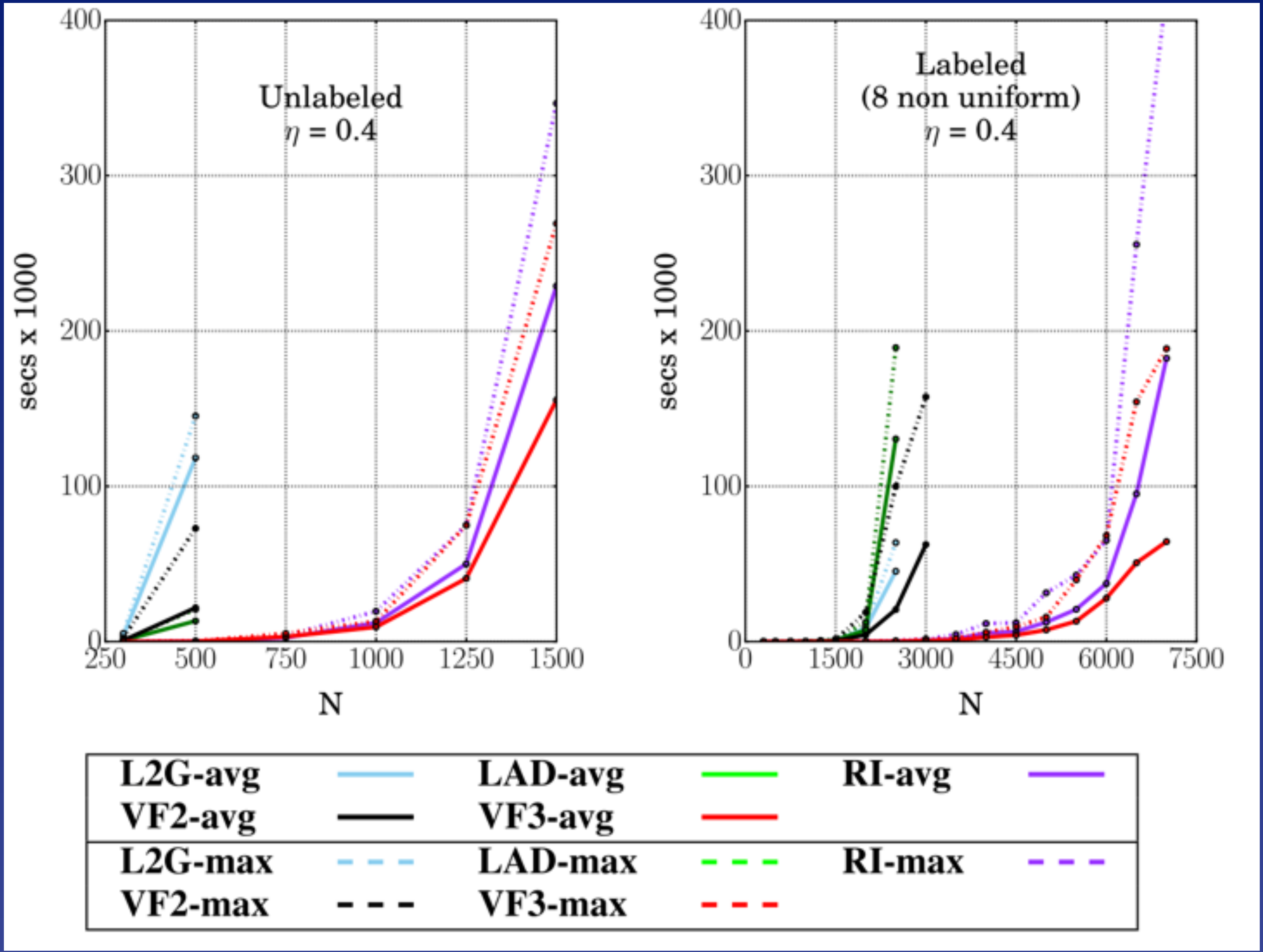


RI and VF3 performance



Protein structures

RI and VF3 performance



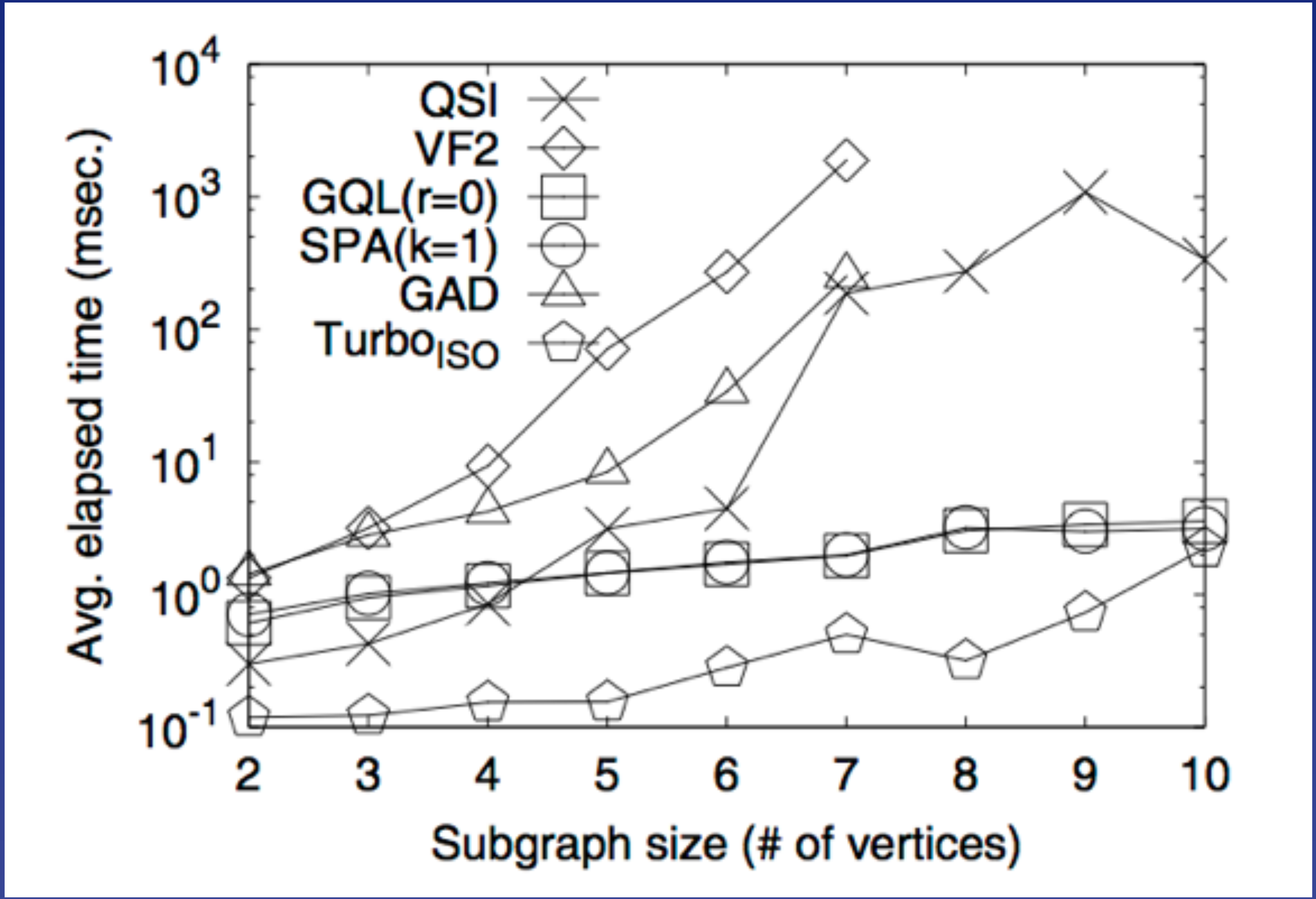
Dense random graphs

TurboISO

- Avoids exploring equivalent permutations of nodes by using auxiliary data structures
  - Wook-Shin Han, Jinsoo Lee, and Jeong-Hoon Lee, Turboiso: towards ultrafast and robust subgraph isomorphism search in large graph databases. In Proc. SIGMOD ’13, pp. 337-348, 2013.

TurboISO

- Can be very efficient in matching very small subgraphs with very large graphs



Human Protein Interaction Network (~4600 nodes)

Parallel approach

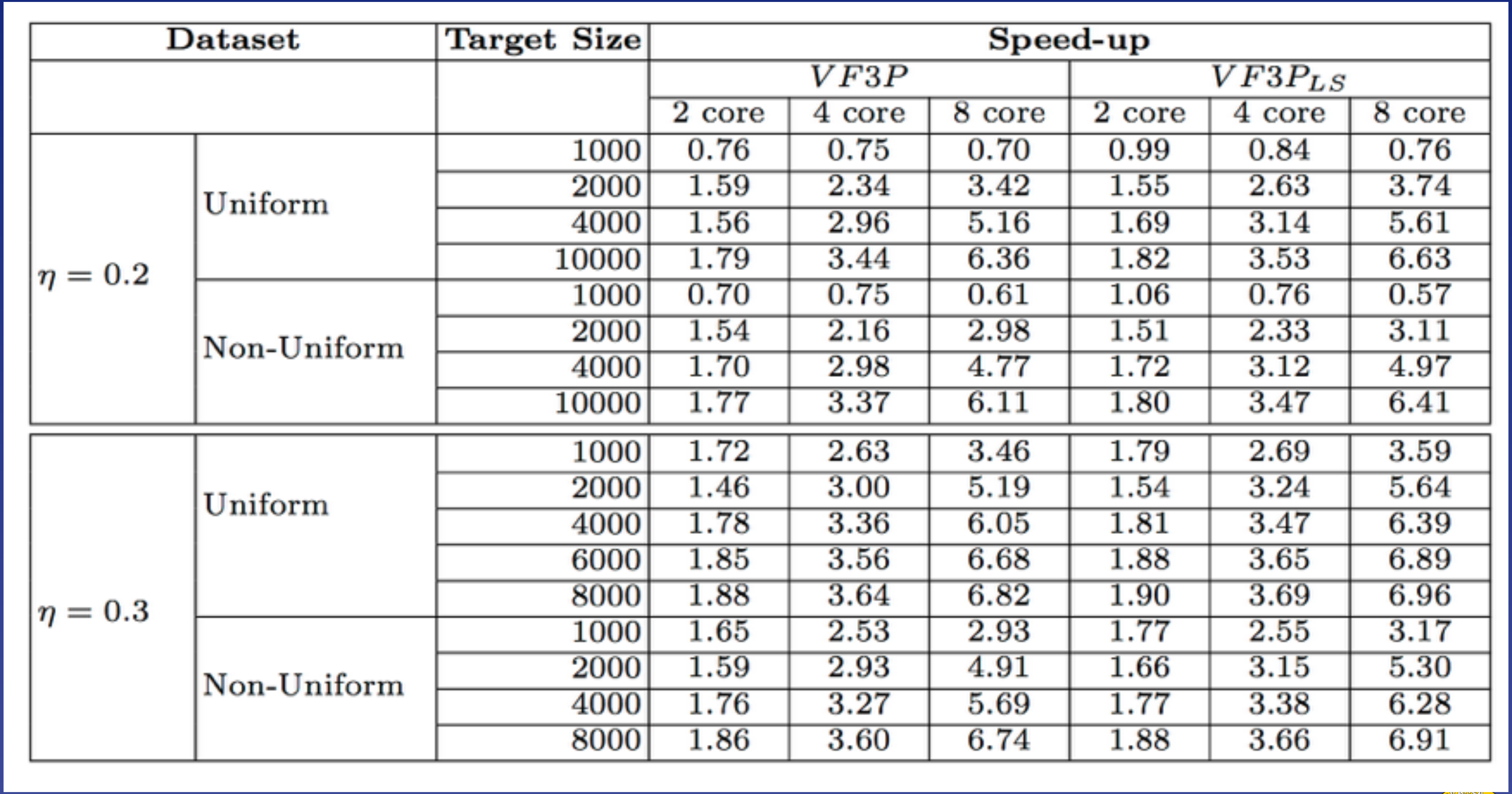
- Given the wide availability of multiprocessor systems, several works have proposed parallelization as a mean for reducing the matching time
  - Usually using a shared memory architecture

VF3P

- We have developed a parallel version of VF3, called VF3P (presented at GbR2019)
  - We cannot divide the graph among the processors (data parallelism)
  - So we divide the parts of the solution space (using a State-Space-Representation) to be explored (task parallelism)
    - Finding the right granularity
    - Reducing the communication overhead

VF3P

- Our preliminary results show a speedup that is close to the number of processors when the graphs get large



Dataset		Target Size	Speed-up					
			VF3P			$VF3P_{LS}$		
			2 core	4 core	8 core	2 core	4 core	8 core
$\eta = 0.2$	Uniform	1000	0.76	0.75	0.70	0.99	0.84	0.76
		2000	1.59	2.34	3.42	1.55	2.63	3.74
		4000	1.56	2.96	5.16	1.69	3.14	5.61
		10000	1.79	3.44	6.36	1.82	3.53	6.63
	Non-Uniform	1000	0.70	0.75	0.61	1.06	0.76	0.57
		2000	1.54	2.16	2.98	1.51	2.33	3.11
		4000	1.70	2.98	4.77	1.72	3.12	4.97
		10000	1.77	3.37	6.11	1.80	3.47	6.41
$\eta = 0.3$	Uniform	1000	1.72	2.63	3.46	1.79	2.69	3.59
		2000	1.46	3.00	5.19	1.54	3.24	5.64
		4000	1.78	3.36	6.05	1.81	3.47	6.39
		6000	1.85	3.56	6.68	1.88	3.65	6.89
		8000	1.88	3.64	6.82	1.90	3.69	6.96
	Non-Uniform	1000	1.65	2.53	2.93	1.77	2.55	3.17
		2000	1.59	2.93	4.91	1.66	3.15	5.30
		4000	1.76	3.27	5.69	1.77	3.38	6.28
		8000	1.86	3.60	6.74	1.88	3.66	6.91

Future challenges

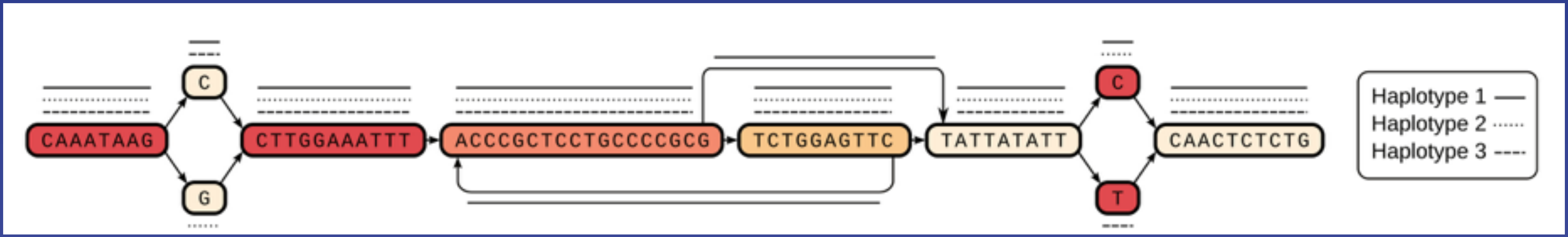
- The algorithms we have seen allow us to work with graphs with tens of thousands nodes
- Still, we have new applications yielding much larger graphs

Future challenges

- The Stanford Large Networks dataset collections (SNAP, http://snap.stanford.edu) includes very large graphs from social networks
  - LiveJournal members: almost 5 millions nodes, 69 millions edges
  - Friendster on line gaming: ~65 millions nodes, 1.8 billions edges

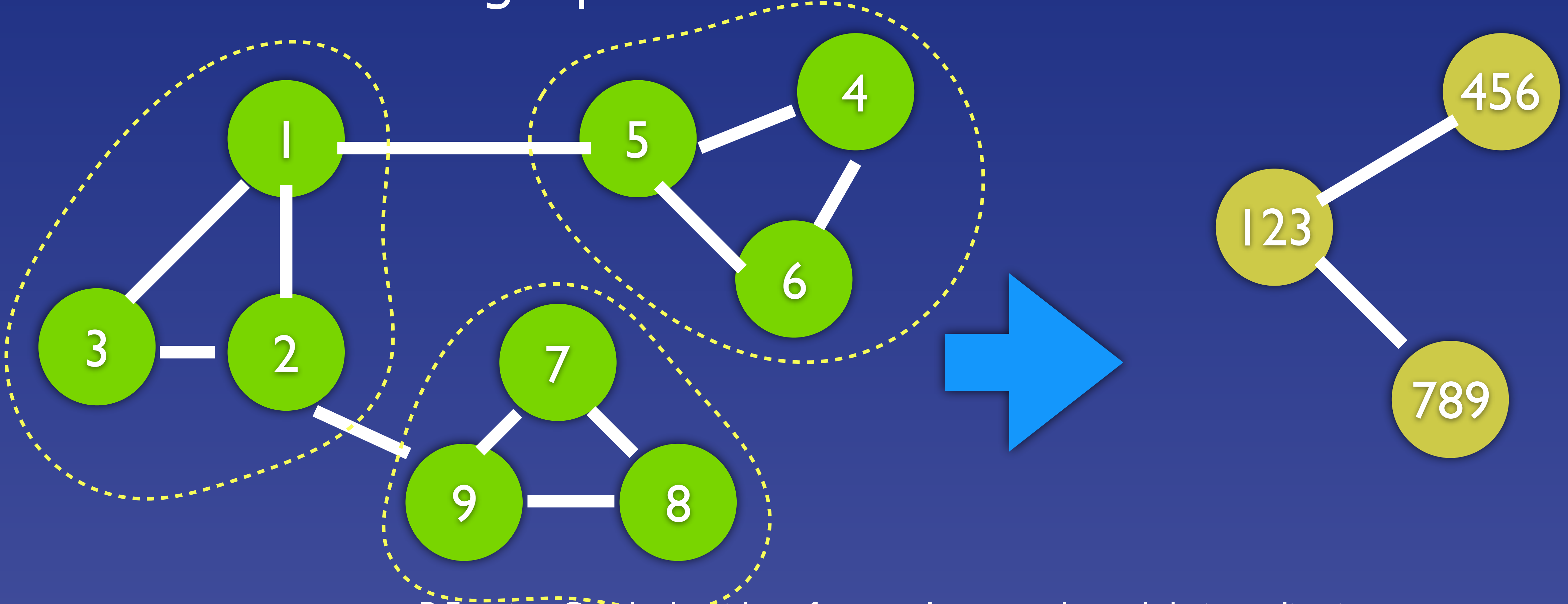
Future challenges

- A new trend in bioinformatics is the use of graph representations for genomic data
  - Genome graph: represents the genetic information of a population, describing the common parts and the individual variations as paths within the graph
  - A genome graph for 1000 human subjects: about 100 millions nodes, 3-400 millions edges



Graph summarization

- One approach for working with such large graphs is Graph Summarization
  - Convert groups of nodes into nodes for obtaining a smaller graph with the same overall structure



Graph summarization

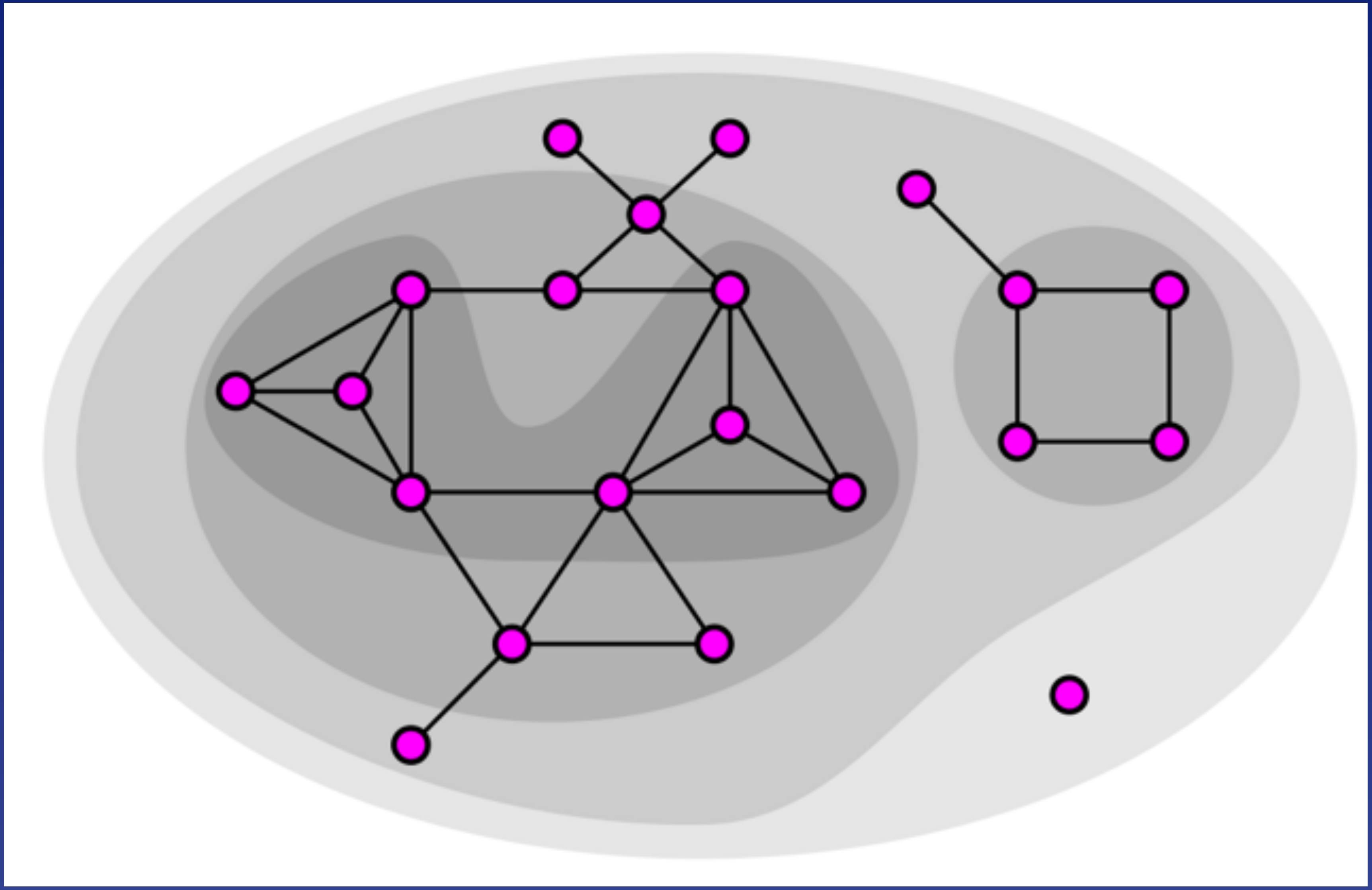
- A recent paper proposes a nice theoretical result about summarization
  - M. Pelillo, I. Elezi, M. Fiorucci, Revealing structure in large graphs: Szemerédi's regularity lemma and its use in pattern recognition, Pattern Recognition Letters, Volume 87, 2017, pp. 4-11
  - Basically, the RL says that every graph large enough and dense enough is partitionable into a bounded number of regular bipartite graphs plus a small number of extra nodes and edges
  - Does it work on practical applications? Can it be used to implement some form of hierarchical matching?

Subquadratic algorithms

- Another way to tackle these huge graphs is to replace information that is expensive to compute with approximations that can be computed in less than O(n^2) time
- Example: in Social Network Analysis, cliques can be replaced by *k-cores* as a way of finding groups of nodes with strong internal connections

Subquadratic algorithms

- k-Cores can be computed in a O(k*n) time



- can we do something similar for (some form of) graph matching?

Distributed computing

- Shared-memory parallelism has inherent limits on the scalability it can achieve
- Big Data applications often use other architectures, with distributed computing
  - Example: the Map/Reduce programming model made popular by Google

Spark/GraphX

- The Apache distributed computing framework Spark includes an extension named GraphX offering APIs for implementing massively parallel algorithms on graph
  - the programming model can be seen as an adaptation to graphs of Map/Reduce
  - Facebook uses a similar approach for running some graph-based algorithms (e.g. for page ranking)
  - Can graph matching (or a suitable surrogate of it) be adapted to such a paradigm?

GPU algorithms

- GPU accels make available tens to hundreds of teraflops at an affordable price
  - we don’t have yet graph matching algorithms able to fully exploit this huge computational power (GPUs are best suited for data parallelism, while most parallel matching algorithms use task parallelism)
  - Can graph matching be formulated in a data-parallel way?

Neural networks

- Deep neural network can approximate very complicated functions (e.g. object recognition). And what's best is that they *learn* how to do it!
  - with tensor GPU accelerators they can be quite fast
  - Can we use deep learning to learn how to match graphs (even approximately)?

Neural networks

- Can we learn a graph similarity measure?
  - Given two graphs, the network should output an approximation of their edit distance
  - Most graph NN process one graph at a time

Neural networks

- Can we learn how to search a pattern subgraph inside a larger target graph?
  - YOLO detects and classifies smaller objects inside a large image…

Neural networks

- Can we learn a useful compact representation for graphs?
  - a sort of graph autoencoder
  - this representation can be used as the starting point for other operations (e.g. graph similarity)
  - Several graph NN have already been proposed for *node* embedding

Conclusions

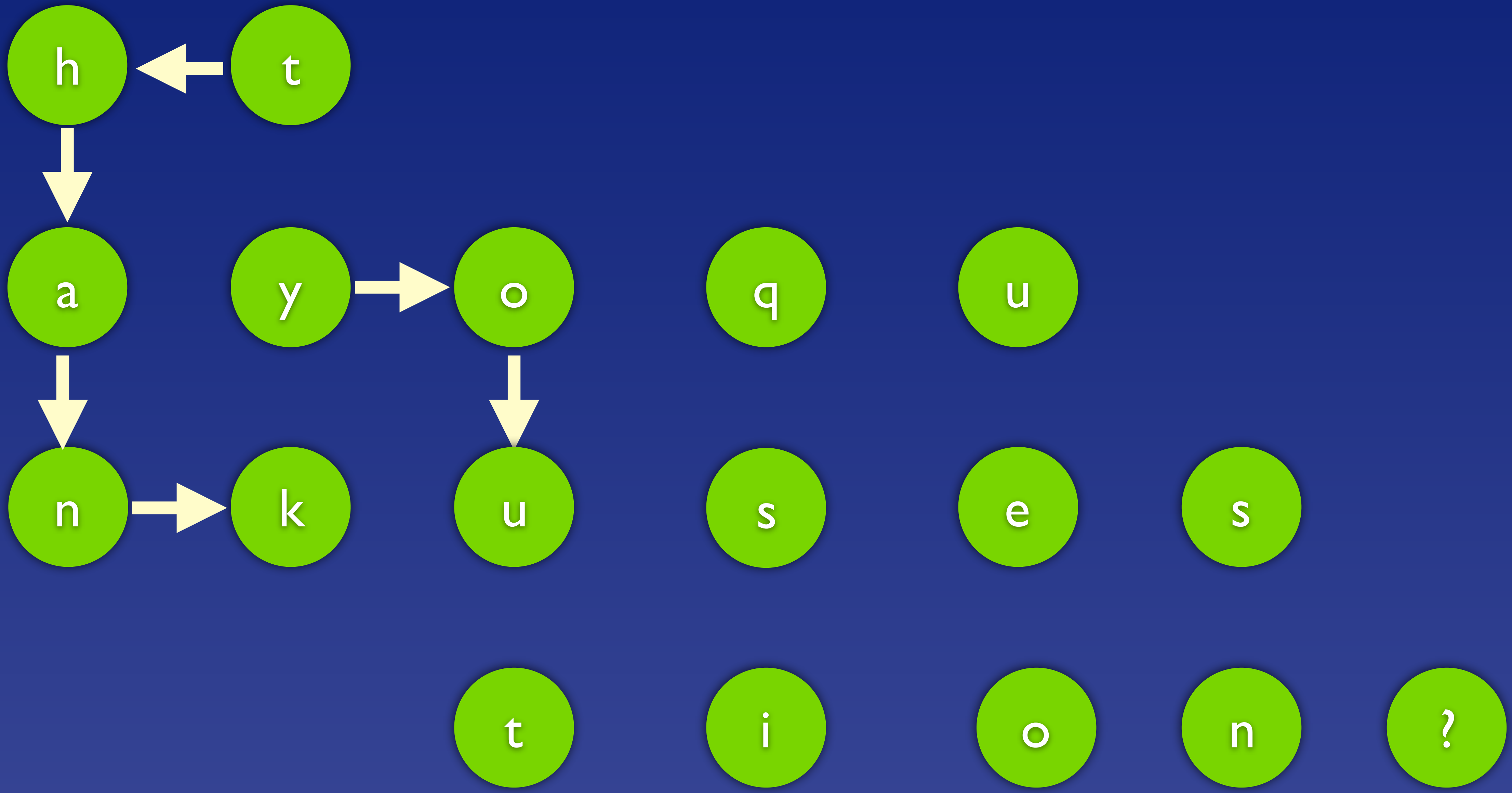
- In the last three decades we have increased the size of the graphs in our applications from a few nodes to tens of thousands of nodes and more
- We are now at a point where we cannot just evolve our algorithms, if we want to face the challenge coming from the huge graphs of Bioinformatics and Social Network Analysis

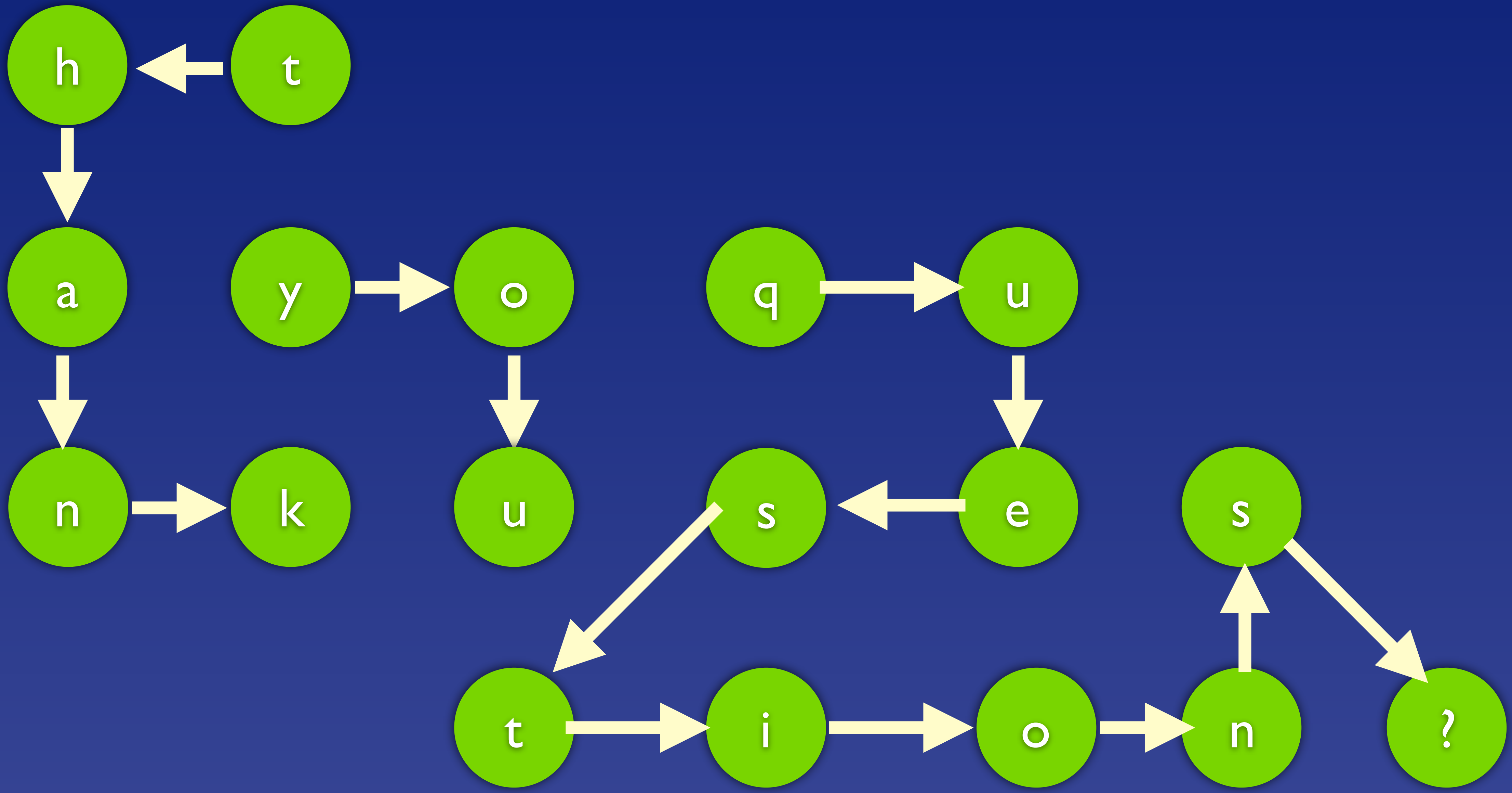
h t

a y o q u

n k u s e s

t i o n ?


h
t
a
y
o
q
u
n
k
u
s
e
s
t
i
o
n
?


h
t
a
y
o
q
u
n
k
u
s
e
s
t
i
o
n
?