Neural Trees for Learning on Graphs

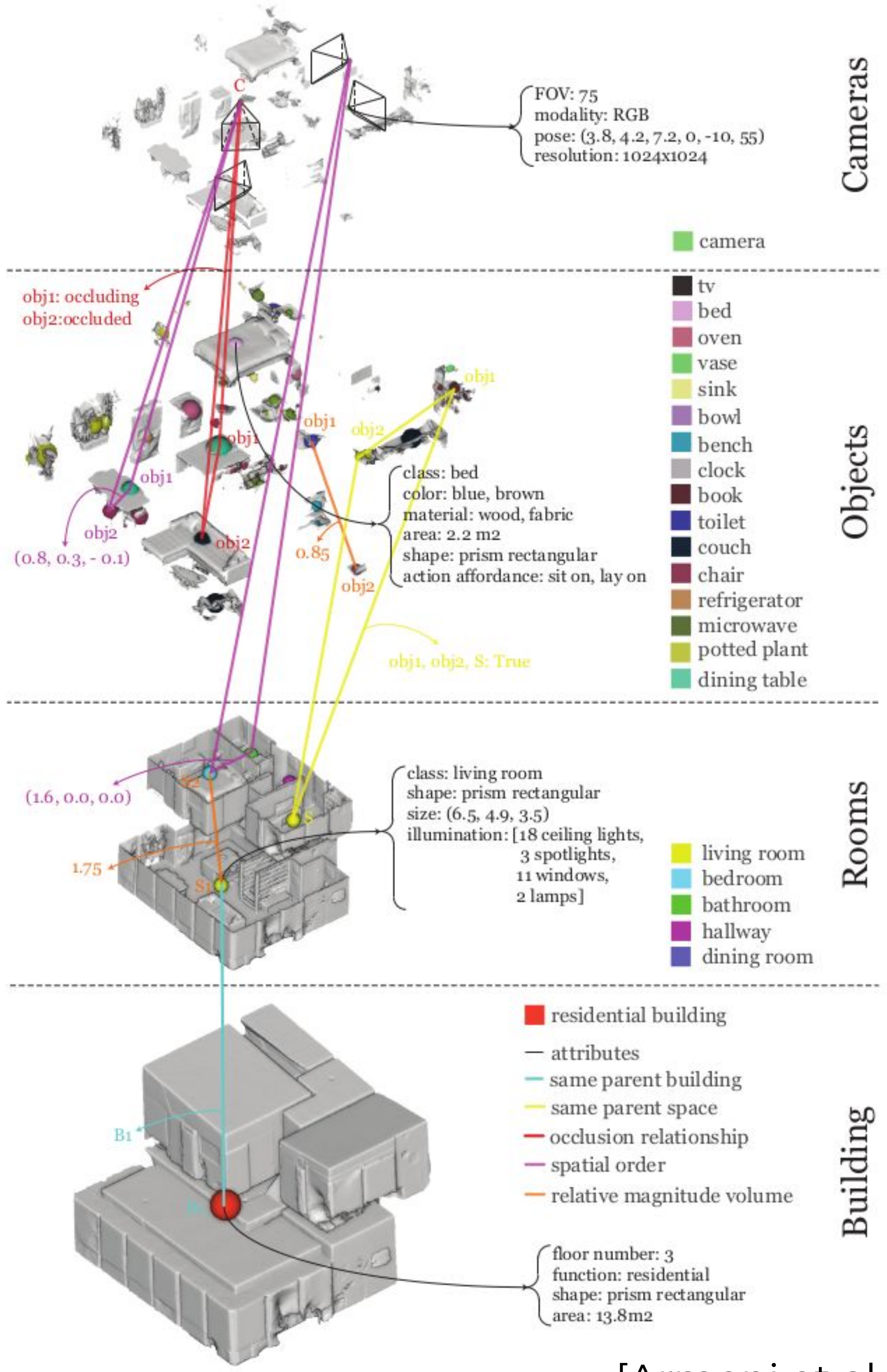
Rajat Talak, Siyi Hu, Lisa Peng, and Luca Carlone
Massachusetts Institute of Technology

NeurIPS 2021

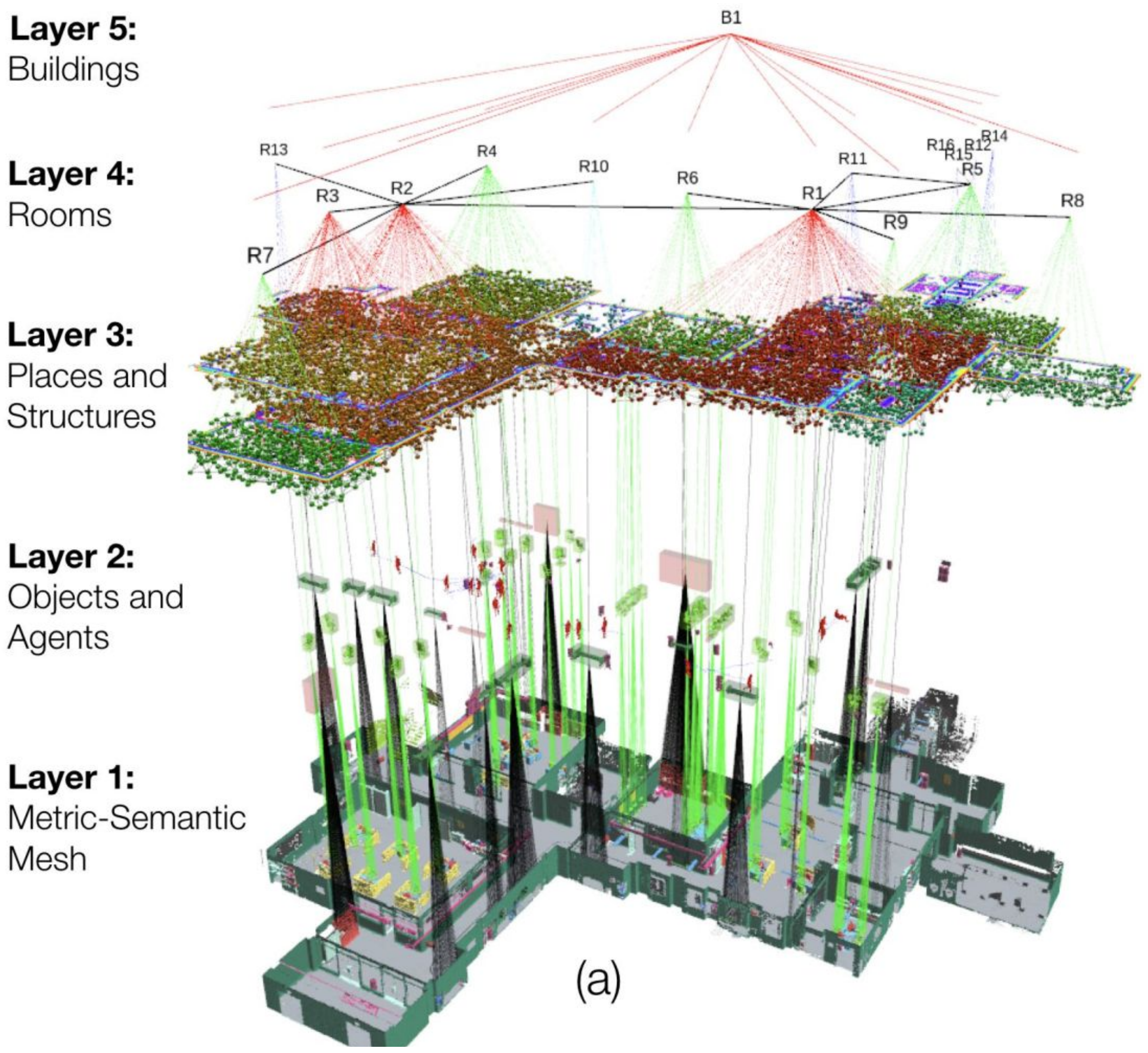
MIT AEROASTRO LIDS

Motivation

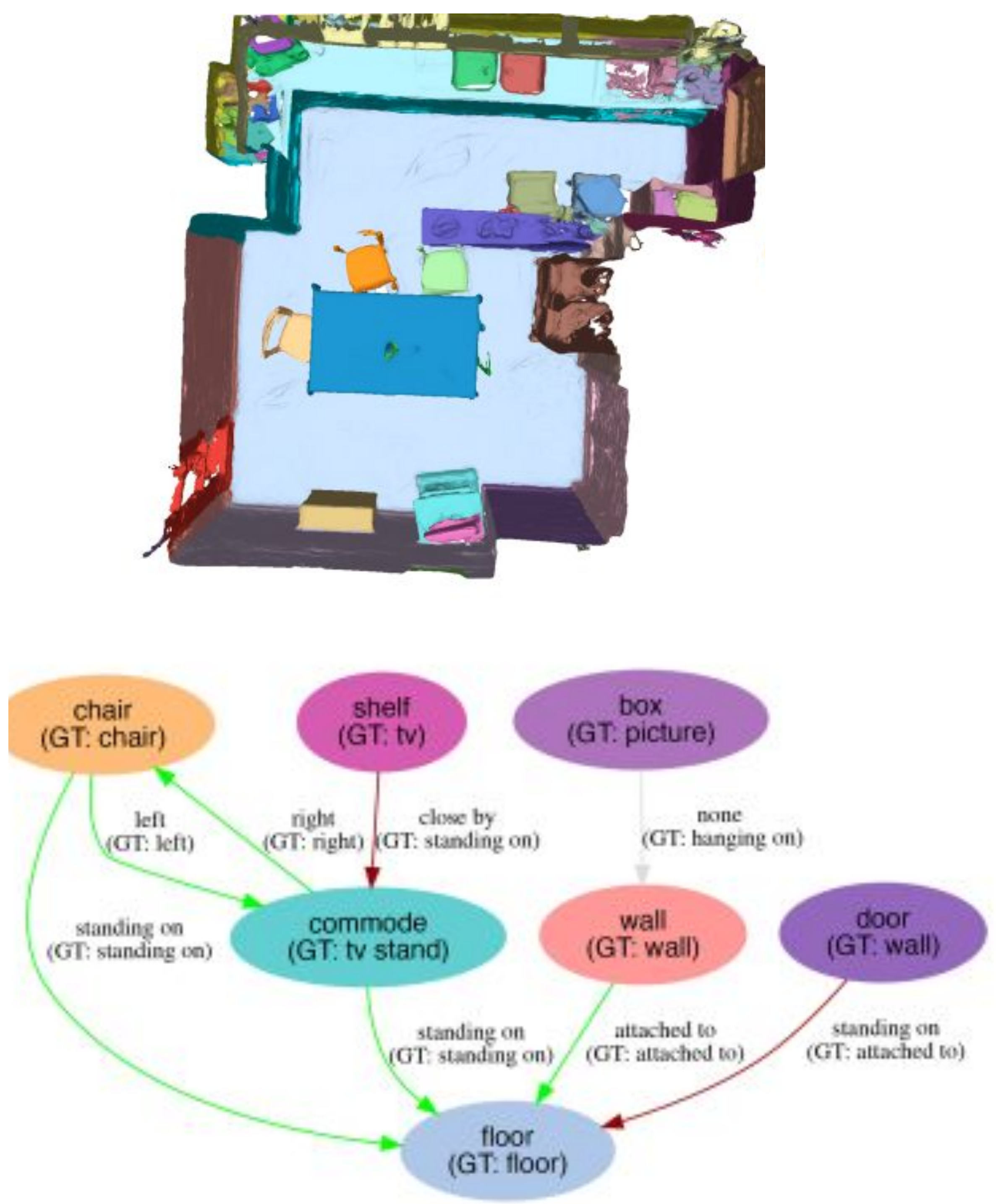
Automated generation of 3D scene graphs is an important problem



[Armeni et al 2019]



[Rosinol et al 2020]

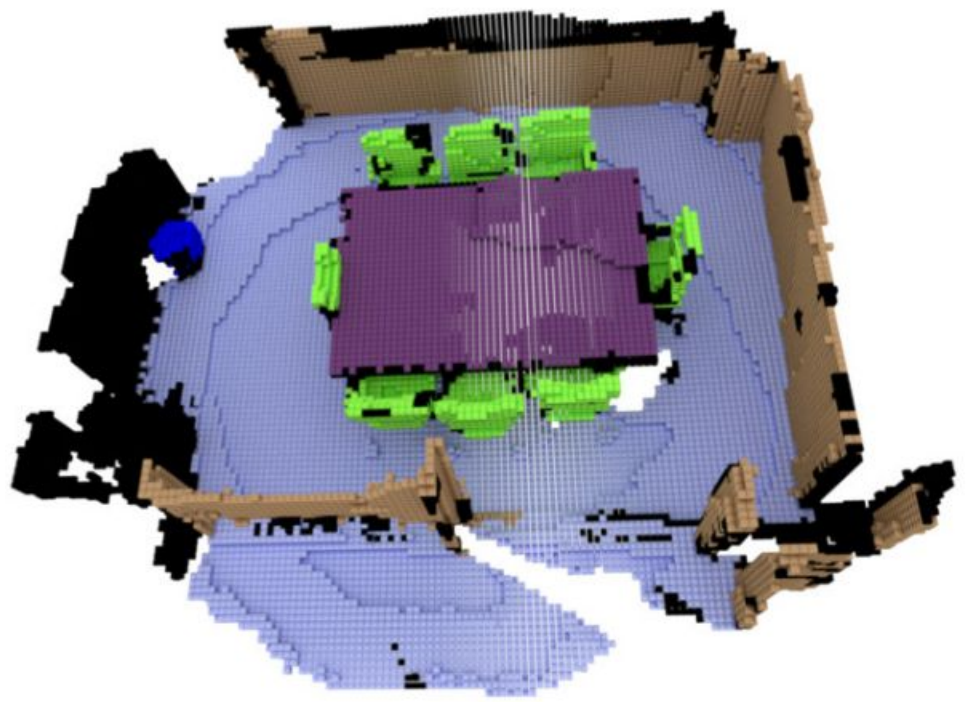


[Wald et al 2020]

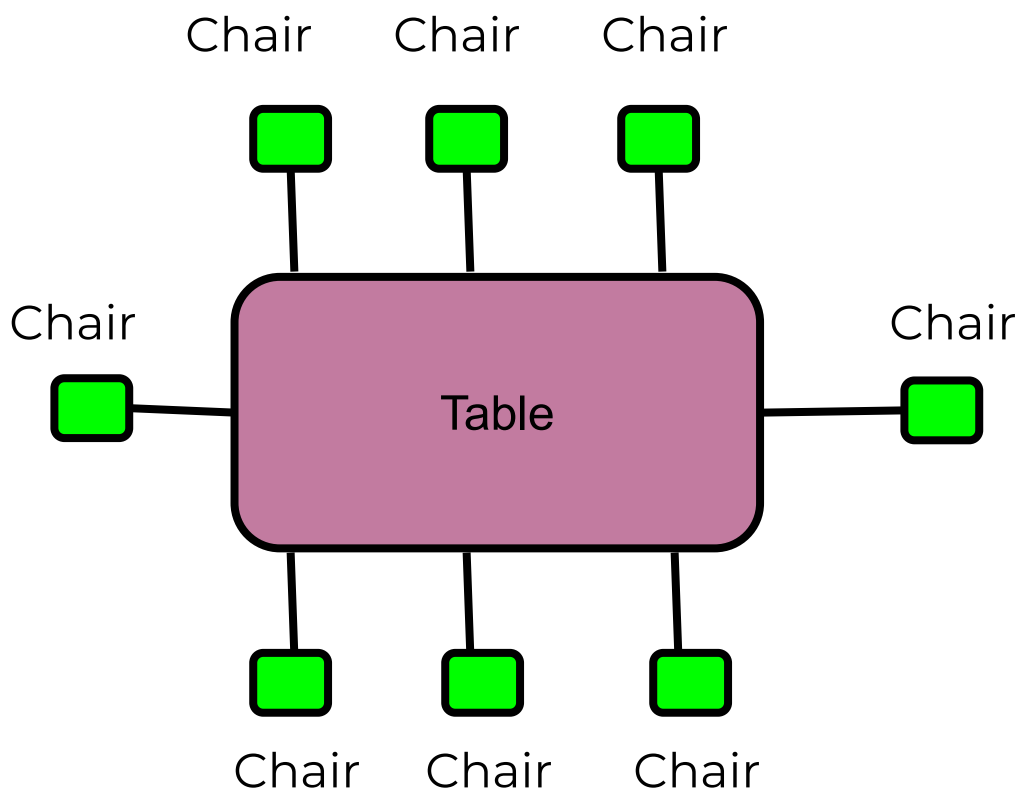
Motivation

Automated generation of 3D scene graphs is an important problem

Inputs

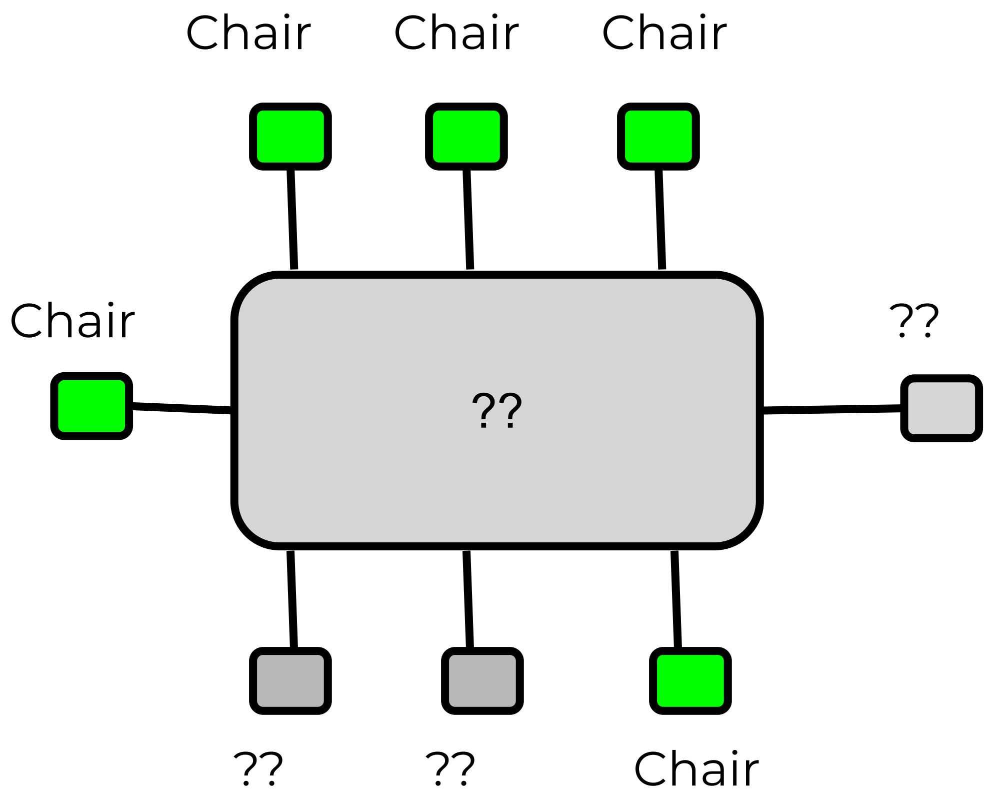


3D reconstruction
(using ScanNet for representation)



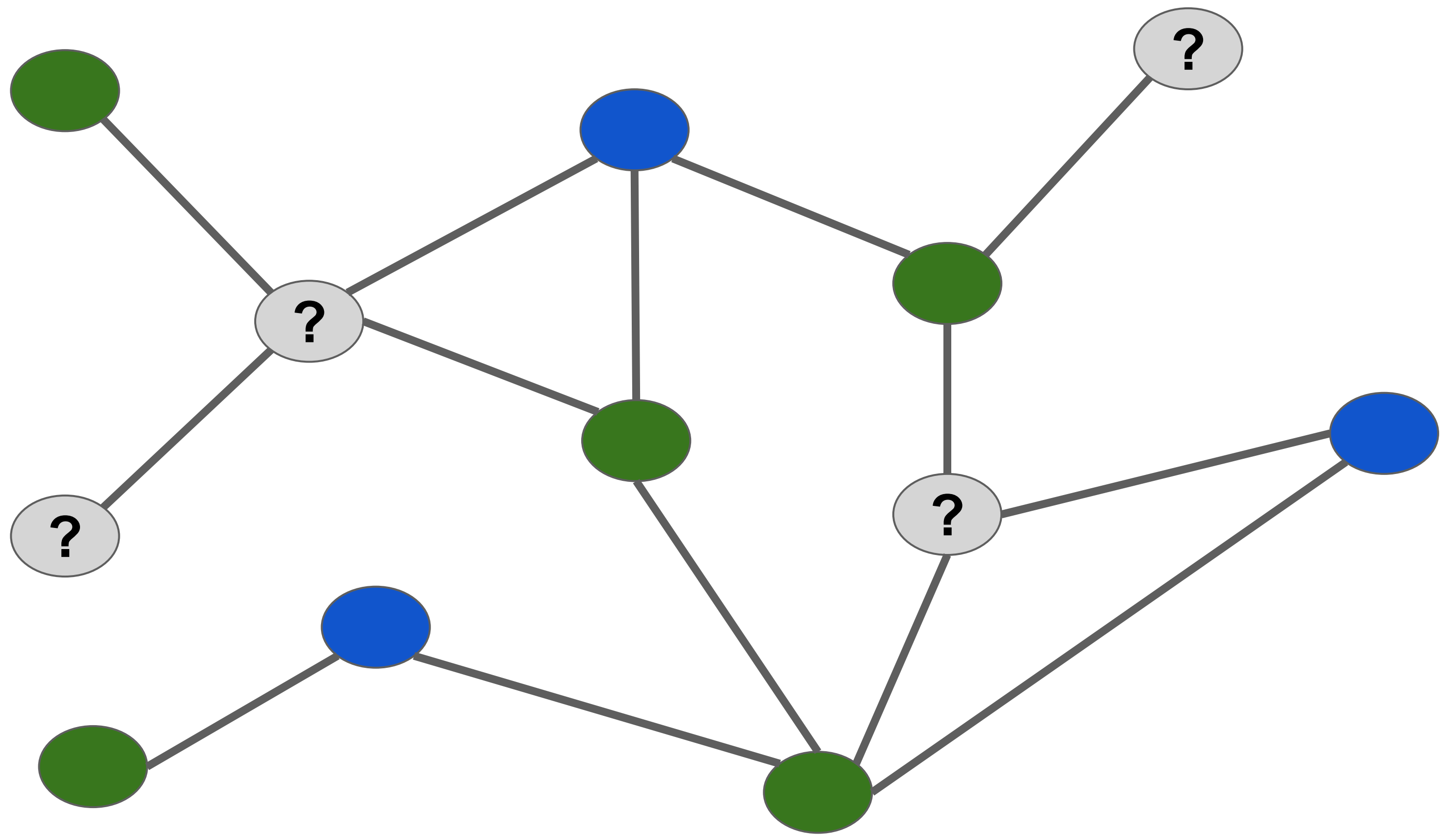
Scene Graph

A problem encountered in *automatic* 3D scene graph generation



Given a scene graph with some node labels, how to predict the missing?

Semi-supervised Node Classification



Given a graph with some nodes labeled, how do we assign labels to other nodes?

Semi-supervised Node Classification

Plantoid

Revisiting Semi-Supervised Learning with Graph Embeddings

Zhilin Yang ZHILINY@CS.CMU.EDU
William W. Cohen WCOHEN@CS.CMU.EDU
Ruslan Salakhutdinov
School of Computer Science, Carneg

Abstract

We present a semi-supervised
work based on graph embedding
between instances, we train an
each instance to jointly predict th
the neighborhood context in the

DATASET	#CLASSES	#NODES	#EDGES
CITESEER	6	3,327	4,732
CORA	7	2,708	5,429
PUBMED	3	19,717	44,338
DIEL	4	4,373,008	4,464,261
NELL	210	65,755	266,144

OPEN GRAPH BENCHMARK

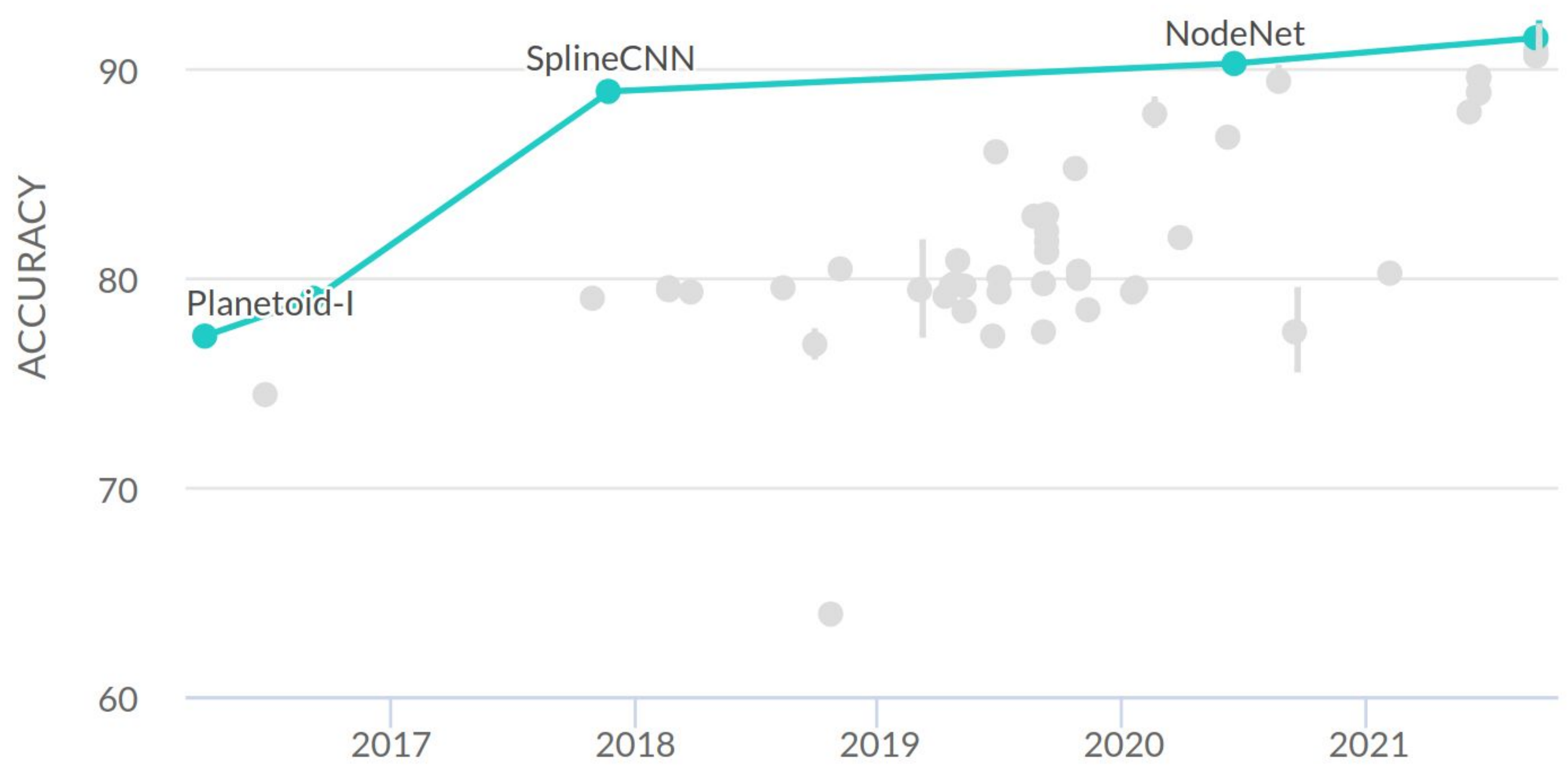
Given a graph with some nodes labeled, how do we assign labels to other nodes?

Well studied problem in social networks.

- *Examples*: document classification, webpage classification, user behavior on social networks …
- Datasets: Plantoid, OGB, …

Semi-supervised Node Classification

Node classification on Pubmed (Plantoid)



Leaderboard for ogbn-papers100M

The classification accuracy on the test and validation sets. The higher, the better.

Package: >=1.2.0

Rank	Method	Test Accuracy	Validation Accuracy	Contact	References	#Params	Hardware	Date
1	SAGN+SLE (4 stages)	0.6830 ± 0.0008	0.7163 ± 0.0007	Chuxiong Sun (CTRI)	Paper, Code	8,556,888	Tesla V100 (16GB GPU)	Sep 21, 2021
2	GAMLP+RLU	0.6825 ± 0.0011	0.7159 ± 0.0005	Wentao Zhang (PKU Tencent Joint Lab)	Paper, Code	16,308,751	Tesla V100 (32GB)	Aug 19, 2021
3	FSGNN	0.6807 ± 0.0006	0.7175 ± 0.0007	Sunil Kumar Maurya (TokyoTech, AIST)	Paper, Code	16,453,301	NVIDIA V100 (16GB)	Sep 16, 2021
4	SAGN+SLE	0.6800 ± 0.0015	0.7131 ± 0.0010	Chuxiong Sun	Paper, Code	8,556,888	Tesla V100 (16GB GPU)	Apr 19, 2021
5	GAMLP	0.6771 ± 0.0020	0.7117 ± 0.0014	Wentao Zhang (PKU Tencent Joint Lab)	Paper, Code	16,308,751	Tesla V100 (32GB)	Aug 22, 2021
6	TransformerConv	0.6736 ±	0.7172 ± 0.0005	Xiaonan Song (NVIDIA SAE	Paper,	883,378	NVIDIA DGX-2	Mar 4,

Given a graph with some nodes labeled, how do we assign labels to other nodes?

Well studied problem in social networks.

- *Examples*: document classification, webpage classification, user behavior on social networks …
- Datasets: Plantoid, OGB, …

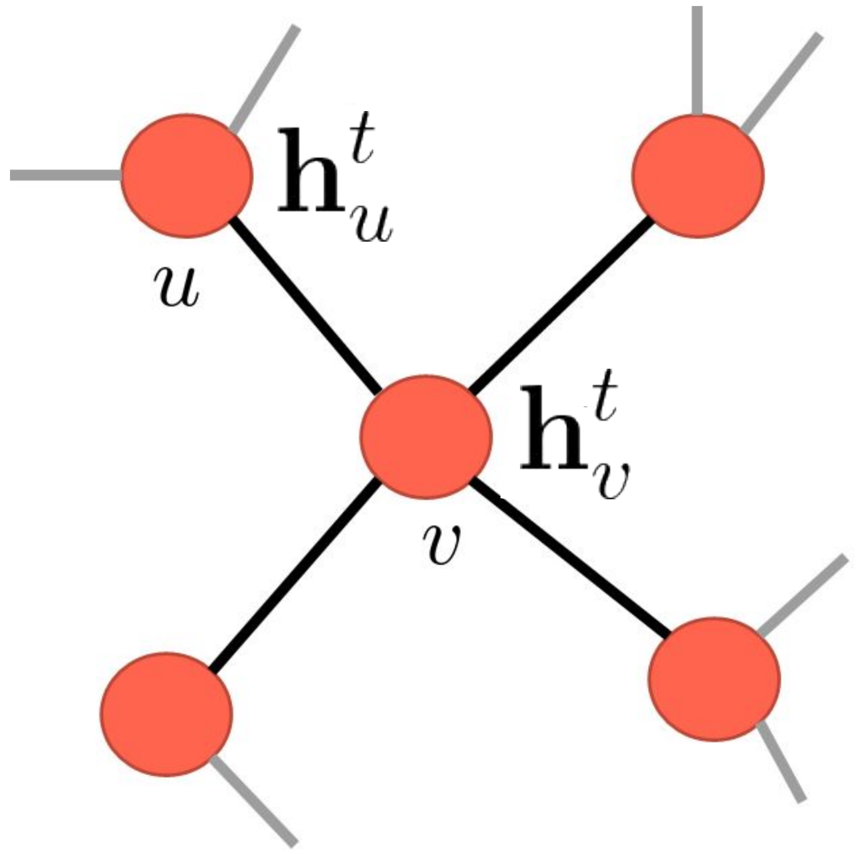
State-of-the-art approaches

- **Graph Neural Networks**

Graph Neural Networks

Basic idea: Iteratively aggregate representation and feature vectors of neighbors

$$\mathbf{h}_v^t = \mathrm{AGG}_t\left(\mathbf{h}_v^{t-1}, \left\{\left(\mathbf{h}_u^{t-1}, \kappa_{u,v}, \mathbf{h}_v^{t-1}\right) \mid w \in \mathcal{N}_{\mathcal{G}}(v)\right\}\right)$$



illustration

Read label after T iterations:

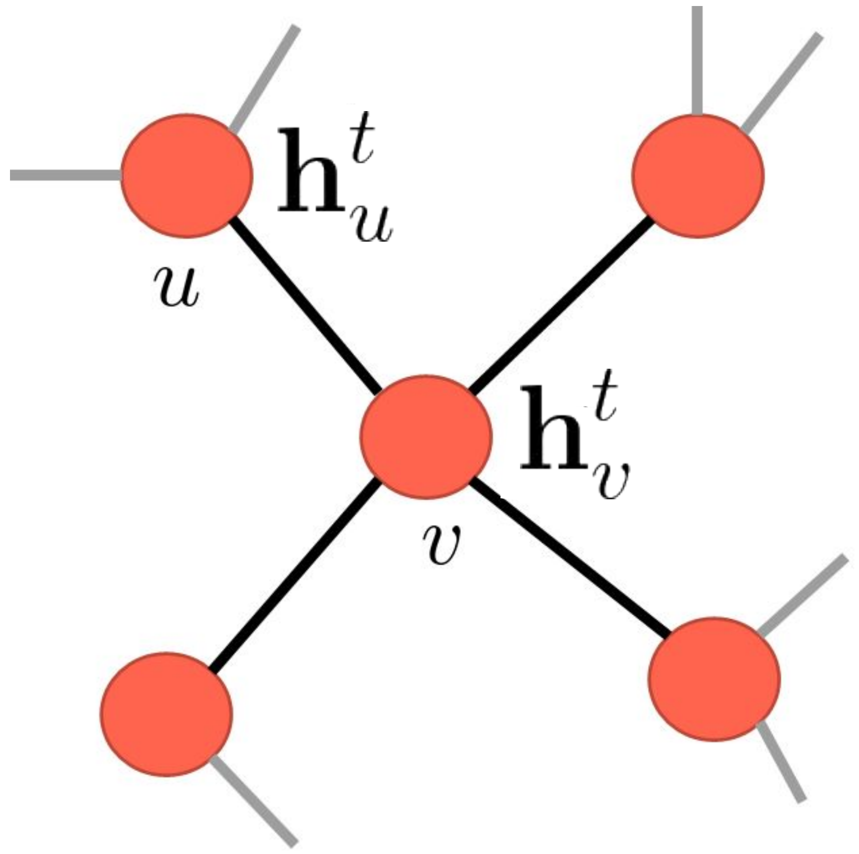
$$y_v = \mathrm{READ}(\mathbf{h}_v^T)$$

$$\mathbf{h}_v^0 = \mathbf{x}_v$$

Graph Neural Networks

Basic idea: Iteratively aggregate representation and feature vectors of neighbors

$$\mathbf{h}_v^t = \mathrm{AGG}_t\left(\mathbf{h}_v^{t-1}, \left\{\left(\mathbf{h}_u^{t-1}, \kappa_{u,v}, \mathbf{h}_v^{t-1}\right) \mid w \in \mathcal{N}_{\mathcal{G}}(v)\right\}\right)$$



illustration

Read label after T iterations:

$$y_v = \mathrm{READ}(\mathbf{h}_v^T)$$

$$\mathbf{h}_v^0 = \mathbf{x}_v$$

Simple idea, but increasing concerns about limited expressive power

Expressivity Bottlenecks of GNNs

Approximate any graph invariant/equivariant function is an open challenge

Invariance: output invariant to node permutation. *eg. graph classification*

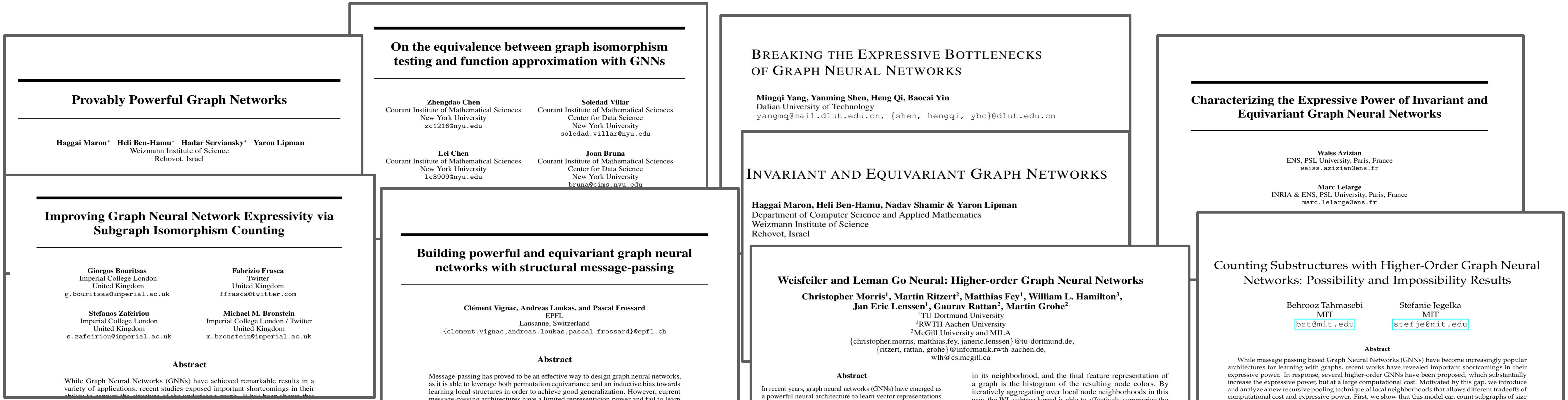
Equivariance: output commutes with node permutation. *eg. node classification*

Expressivity Bottlenecks of GNNs

Approximate any graph invariant/equivariant function is an open challenge

Invariance: output invariant to node permutation. *eg. graph classification*

Equivariance: output commutes with node permutation. *eg. node classification*



Provably Powerful Graph Networks

Haggai Maron* Heli Ben-Hamu* Hadar Serviansky* Yaron Lipman
Weizmann Institute of Science
Rehovot, Israel

On the equivalence between graph isomorphism testing and function approximation with GNNs

Zhengdao Chen
Courant Institute of Mathematical Sciences
New York University
zc1216@nyu.edu

Soledad Villar
Courant Institute of Mathematical Sciences
Center for Data Science
New York University
soledad.villar@nyu.edu

Lei Chen
Courant Institute of Mathematical Sciences
New York University
lc3909@nyu.edu

Joan Bruna
Courant Institute of Mathematical Sciences
Center for Data Science
New York University
bruna@cims.nyu.edu

BREAKING THE EXPRESSIVE BOTTLENECKS OF GRAPH NEURAL NETWORKS

Mingqi Yang, Yanming Shen, Heng Qi, Baocai Yin
Dalian University of Technology
yangmq@mail.dlut.edu.cn, {shen, hengqi, ybc}@dlut.edu.cn

Characterizing the Expressive Power of Invariant and Equivariant Graph Neural Networks

Waïss Azizian
ENS, PSL University, Paris, France
waiss.azizian@ens.fr

Marc Lelarge
INRIA & ENS, PSL University, Paris, France
marc.lelarge@ens.fr

Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting

Giorgos Bouritsas
Imperial College London
United Kingdom
g.bouritsas@imperial.ac.uk

Fabrizio Frasca
Twitter
United Kingdom
ffrasca@twitter.com

Stefanos Zafeiriou
Imperial College London
United Kingdom
s.zafeiriou@imperial.ac.uk

Michael M. Bronstein
Imperial College London / Twitter
United Kingdom
m.bronstein@imperial.ac.uk

Abstract

Building powerful and equivariant graph neural networks with structural message-passing

Clément Vignac, Andreas Loukas, and Pascal Frossard
EPFL
Lausanne, Switzerland
{clement.vignac,andreas.loukas,pascal.frossard}@epfl.ch

Abstract

INVARIANT AND EQUIVARIANT GRAPH NETWORKS

Haggai Maron, Heli Ben-Hamu, Nadav Shamir & Yaron Lipman
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science
Rehovot, Israel

Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks

Christopher Morris[1], Martin Ritzert[2], Matthias Fey[1], William L. Hamilton[3], Jan Eric Lenssen[1], Gaurav Rattan[2], Martin Grohe[2]
[1]TU Dortmund University
[2]RWTH Aachen University
[3]McGill University and MILA
{christopher.morris, matthias.fey, janeric.lenssen}@tu-dortmund.de,
{ritzert, rattan, grohe}@informatik.rwth-aachen.de,
wlh@cs.mcgill.ca

Abstract

Counting Substructures with Higher-Order Graph Neural Networks: Possibility and Impossibility Results

Behrooz Tahmasebi
MIT
bzt@mit.edu

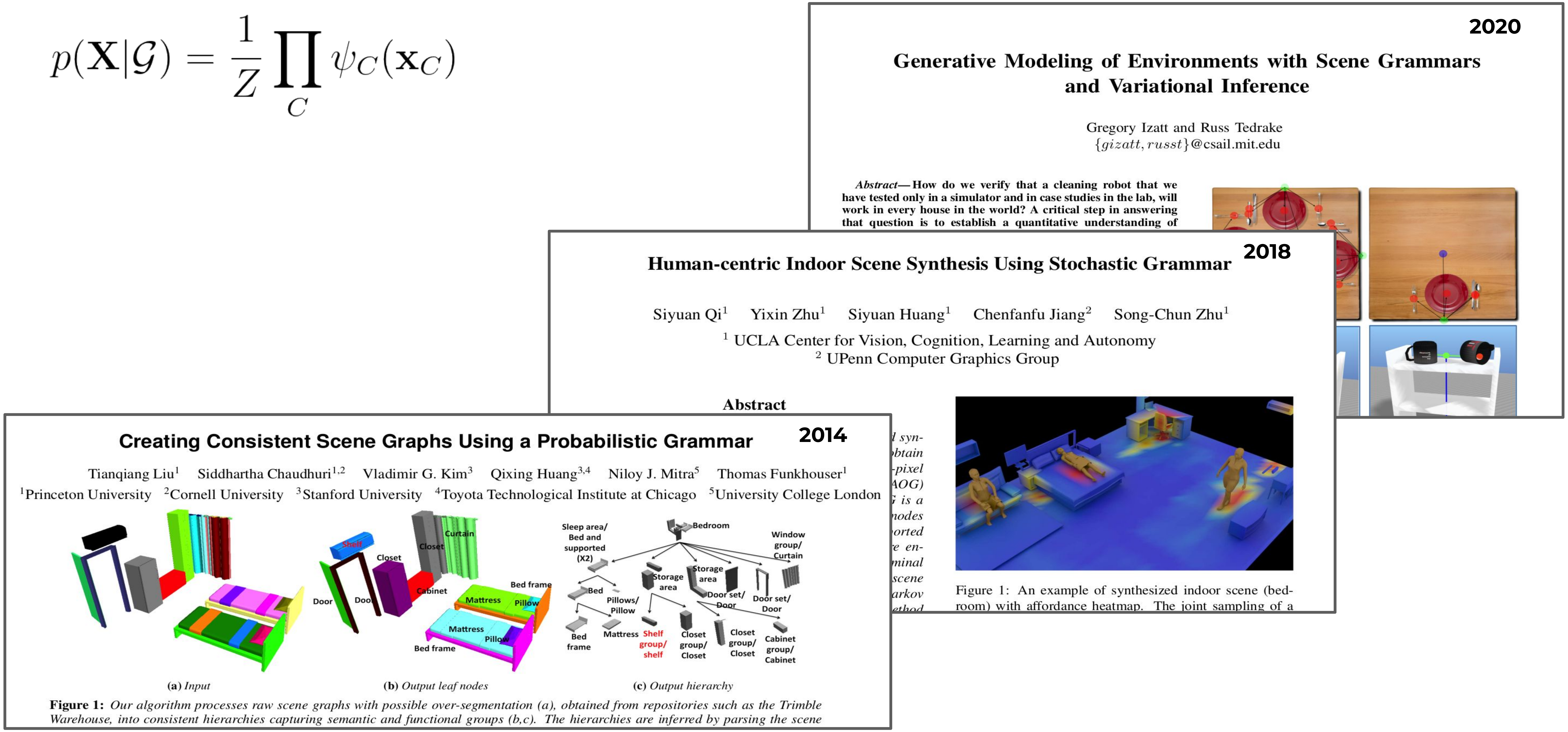
Stefanie Jegelka
MIT
stefje@mit.edu

Abstract

Models for Scene Graphs

Probabilistic graphical models have been used to describe scene graphs

$$p(\mathbf{X}|\mathcal{G}) = \frac{1}{Z}\prod_{C}\psi_C(\mathbf{x}_C)$$


2020

Generative Modeling of Environments with Scene Grammars and Variational Inference

Gregory Izatt and Russ Tedrake
$\{gizatt, russt\}$@csail.mit.edu

***Abstract*— How do we verify that a cleaning robot that we have tested only in a simulator and in case studies in the lab, will work in every house in the world? A critical step in answering that question is to establish a quantitative understanding of**

2018

Human-centric Indoor Scene Synthesis Using Stochastic Grammar

Siyuan Qi[1] Yixin Zhu[1] Siyuan Huang[1] Chenfanfu Jiang[2] Song-Chun Zhu[1]
[1] UCLA Center for Vision, Cognition, Learning and Autonomy
[2] UPenn Computer Graphics Group

Abstract

Figure 1: An example of synthesized indoor scene (bedroom) with affordance heatmap. The joint sampling of a

2014

Creating Consistent Scene Graphs Using a Probabilistic Grammar

Tianqiang Liu[1] Siddhartha Chaudhuri[1,2] Vladimir G. Kim[3] Qixing Huang[3,4] Niloy J. Mitra[5] Thomas Funkhouser[1]
[1]Princeton University [2]Cornell University [3]Stanford University [4]Toyota Technological Institute at Chicago [5]University College London

(a) *Input* (b) *Output leaf nodes* (c) *Output hierarchy*

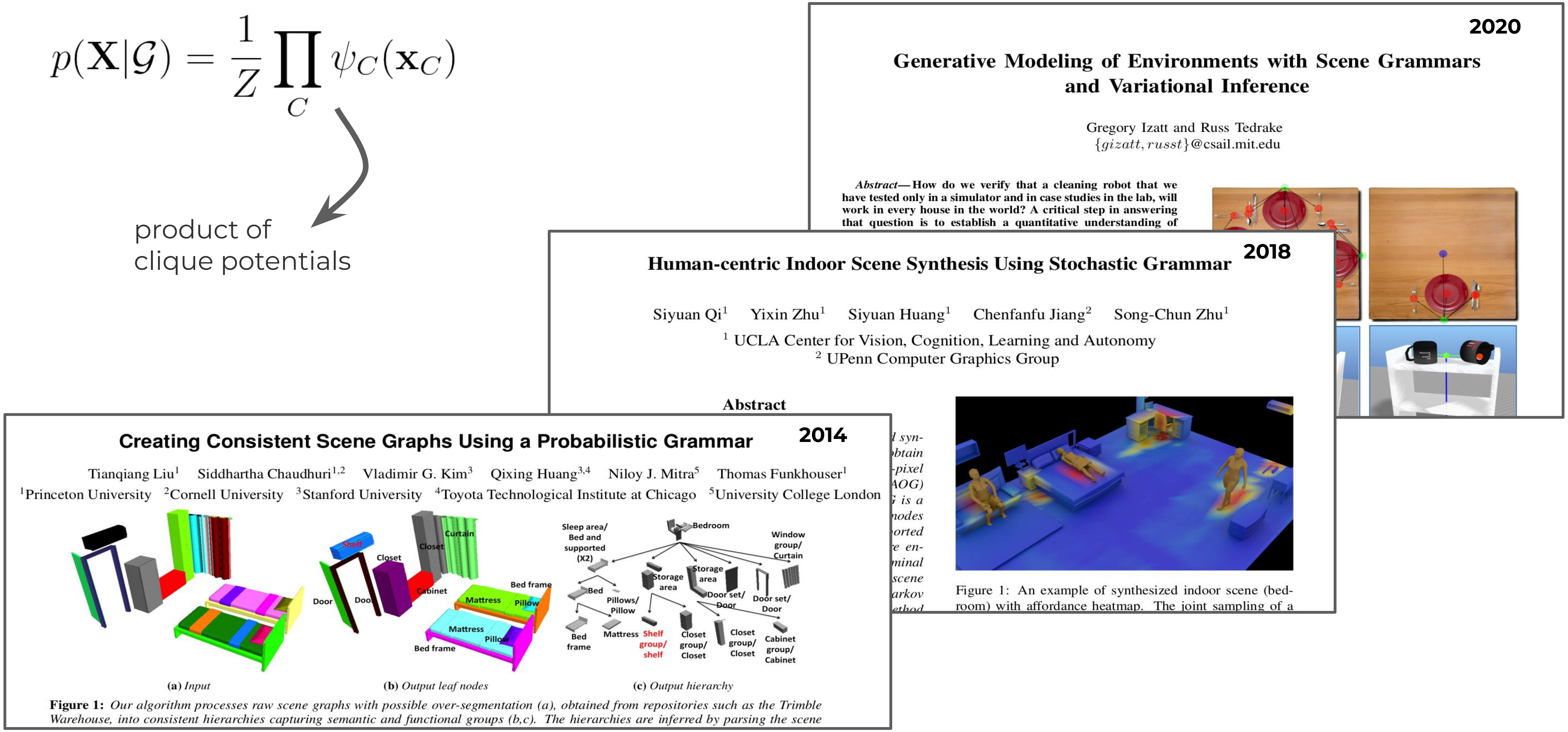
Figure 1: *Our algorithm processes raw scene graphs with possible over-segmentation (a), obtained from repositories such as the Trimble Warehouse, into consistent hierarchies capturing semantic and functional groups (b,c). The hierarchies are inferred by parsing the scene*

Models for Scene Graphs

Probabilistic graphical models have been used to describe scene graphs

$$p(\mathbf{X}|\mathcal{G}) = \frac{1}{Z}\prod_{C}\psi_C(\mathbf{x}_C)$$

product of
clique potentials



Models for Scene Graphs

Probabilistic graphical models have been used to describe scene graphs

$$p(\mathbf{X}|\mathcal{G}) = \frac{1}{Z}\prod_{C}\psi_C(\mathbf{x}_C)$$

product of
clique potentials

Example: Markov Random Field

$$p(\mathbf{x}_1, \ldots \mathbf{x}_4|\mathcal{G}) = \frac{1}{Z}e^{-||\mathbf{x}_1-\mathbf{x}_2||^2} \times e^{-||\mathbf{x}_2-\mathbf{x}_3||^2} \times e^{-||\mathbf{x}_3-\mathbf{x}_4||^2}$$

Models for Scene Graphs

Probabilistic graphical models have been used to describe scene graphs

$$p(\mathbf{X}|\mathcal{G}) = \frac{1}{Z}\prod_{C}\psi_C(\mathbf{x}_C)$$

product of
clique potentials

Example: Markov Random Field

$$p(\mathbf{x}_1,\ldots\mathbf{x}_4|\mathcal{G}) = \frac{1}{Z}e^{-||\mathbf{x}_1-\mathbf{x}_2||^2} \times e^{-||\mathbf{x}_2-\mathbf{x}_3||^2} \times e^{-||\mathbf{x}_3-\mathbf{x}_4||^2}$$

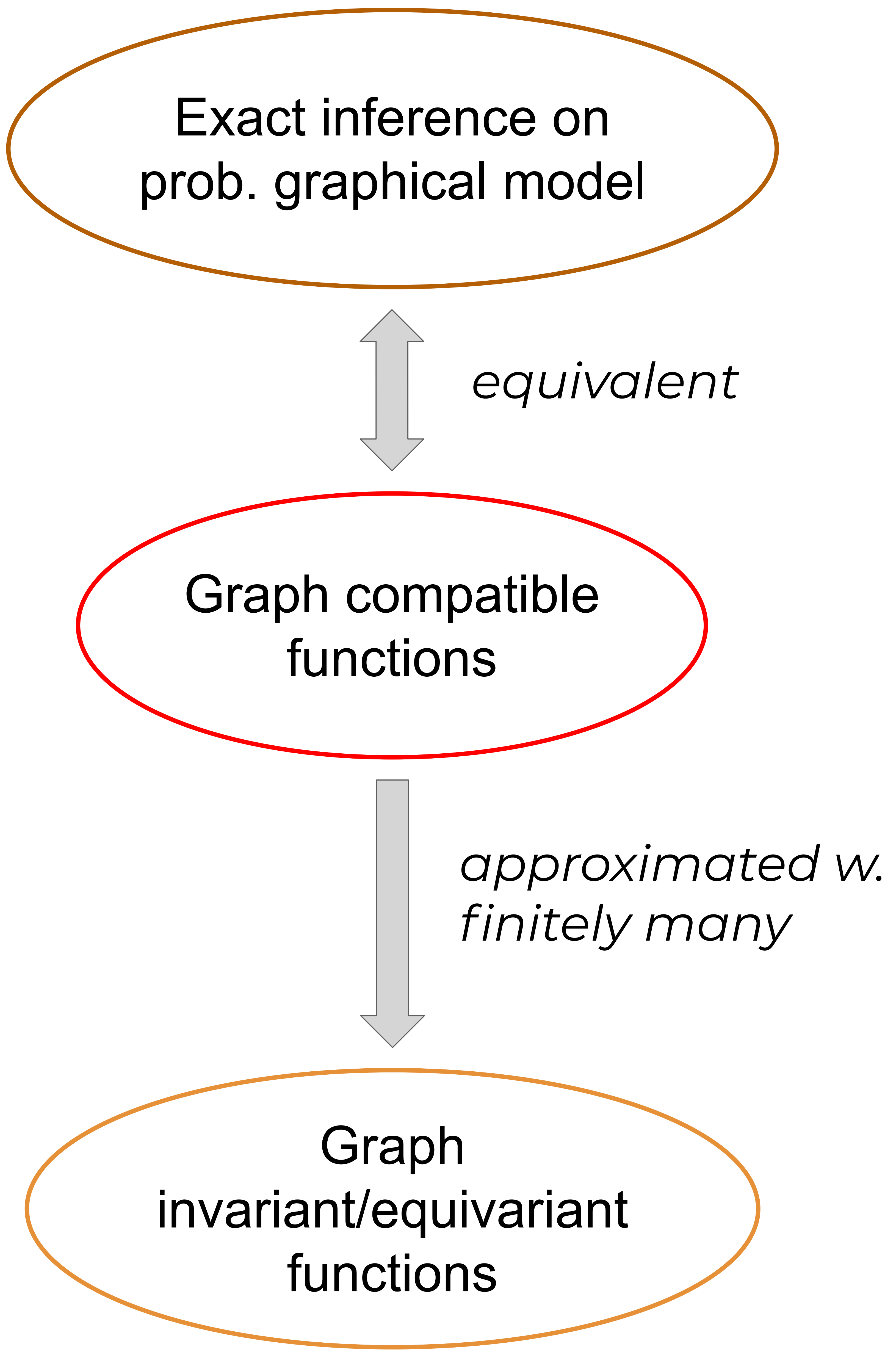
Exact inference is NP-hard and exponential in graph treewidth

Contributions

- Graph compatible functions
- Neural Tree architecture
- Approximation Results
- Experiments

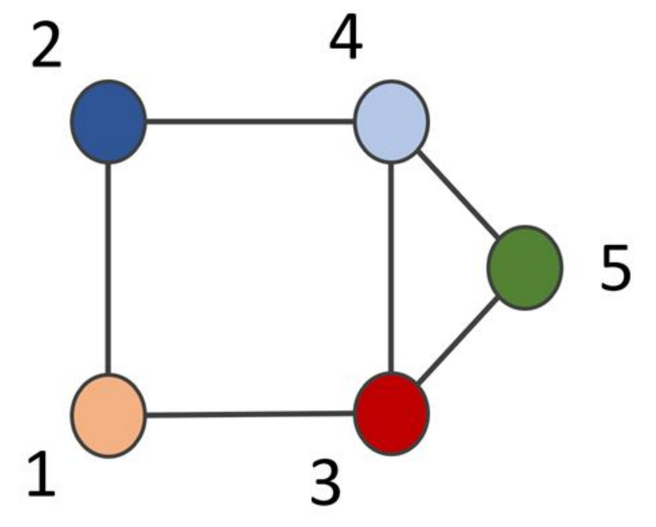
Contributions

- Graph compatible functions
- Neural Tree architecture
- Approximation Results
- Experiments



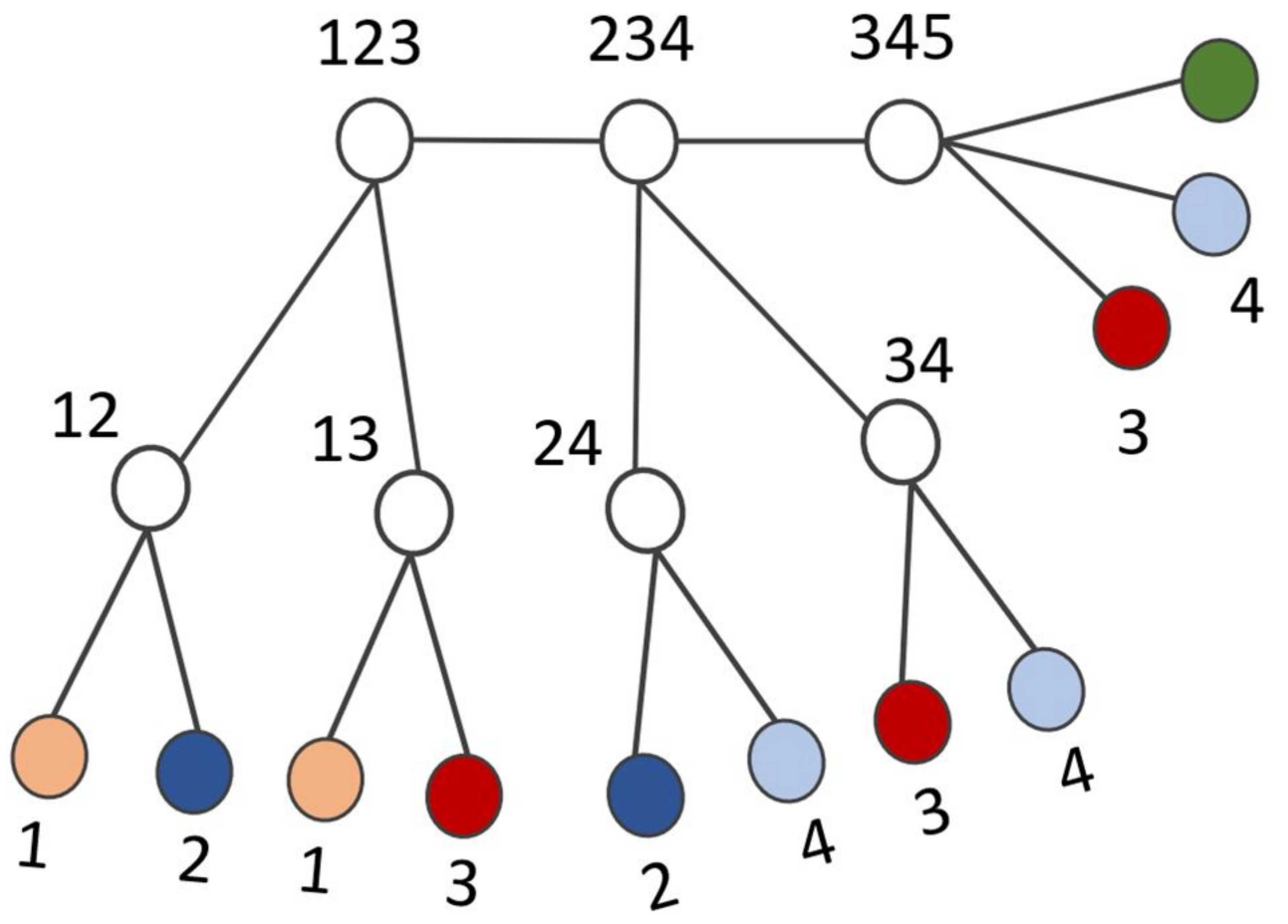
Contributions

- Graph compatible functions
- Neural Tree architecture
- Approximation Results
- Experiments



Input graph with node attributes (colors)

Generate a tree structured graph called *H-tree*



Generated tree structured graph

Neural Tree is *message passing on H-tree*

Contributions

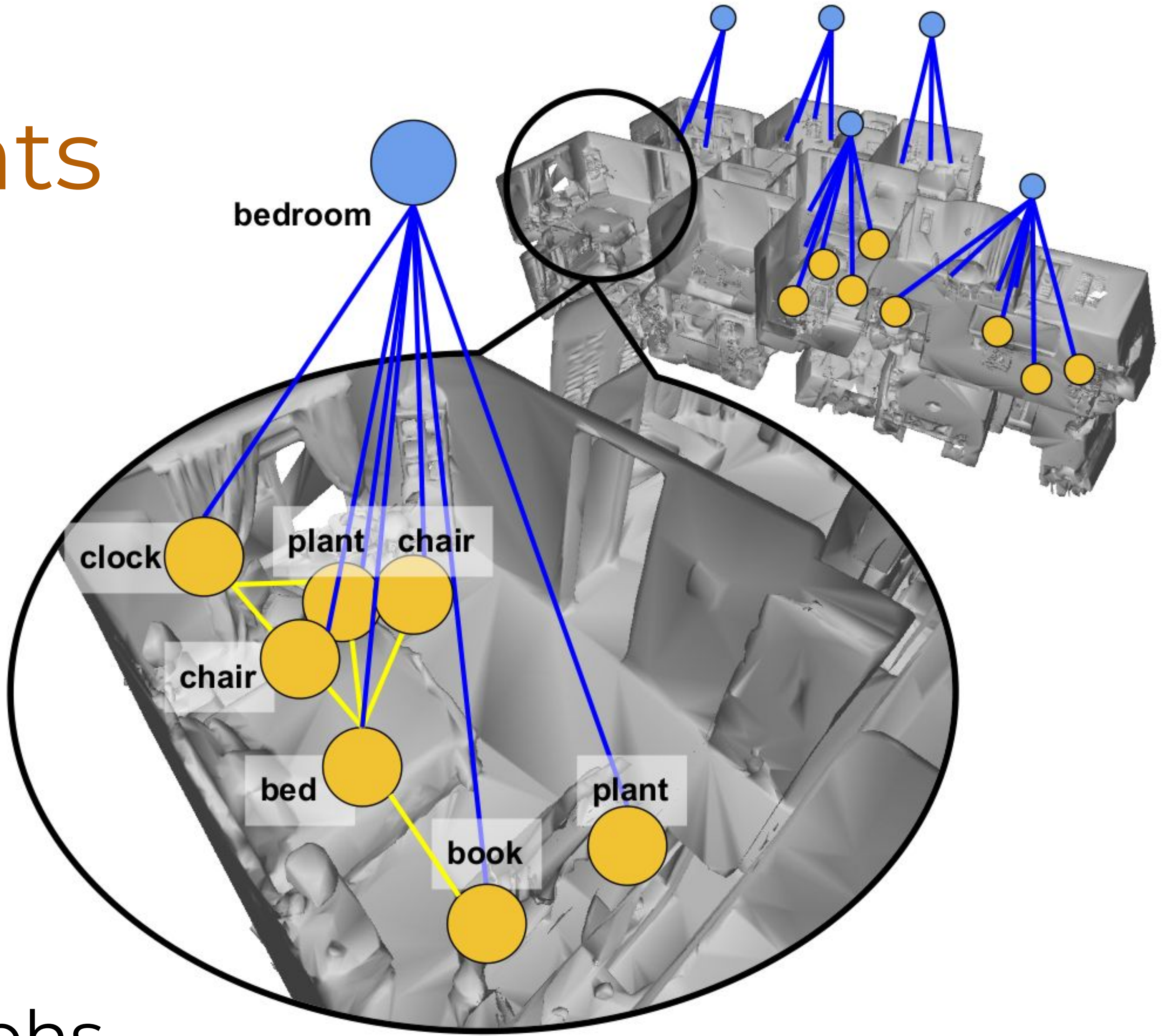
- Graph compatible functions
- Neural Tree architecture
- Approximation Results
- Experiments

Any (smooth) graph compatible function can be approximated by a Neural Tree with number of weights/parameters

$$N = \mathcal{O}\left(\underbrace{n}_{\text{num. nodes}} \cdot \left(\underbrace{\text{tw}(G)}_{\text{treewidth}} / \underbrace{\epsilon}_{\text{approx. distance}} \right)^{c \cdot \text{tw}(G)} \right)$$

Contributions

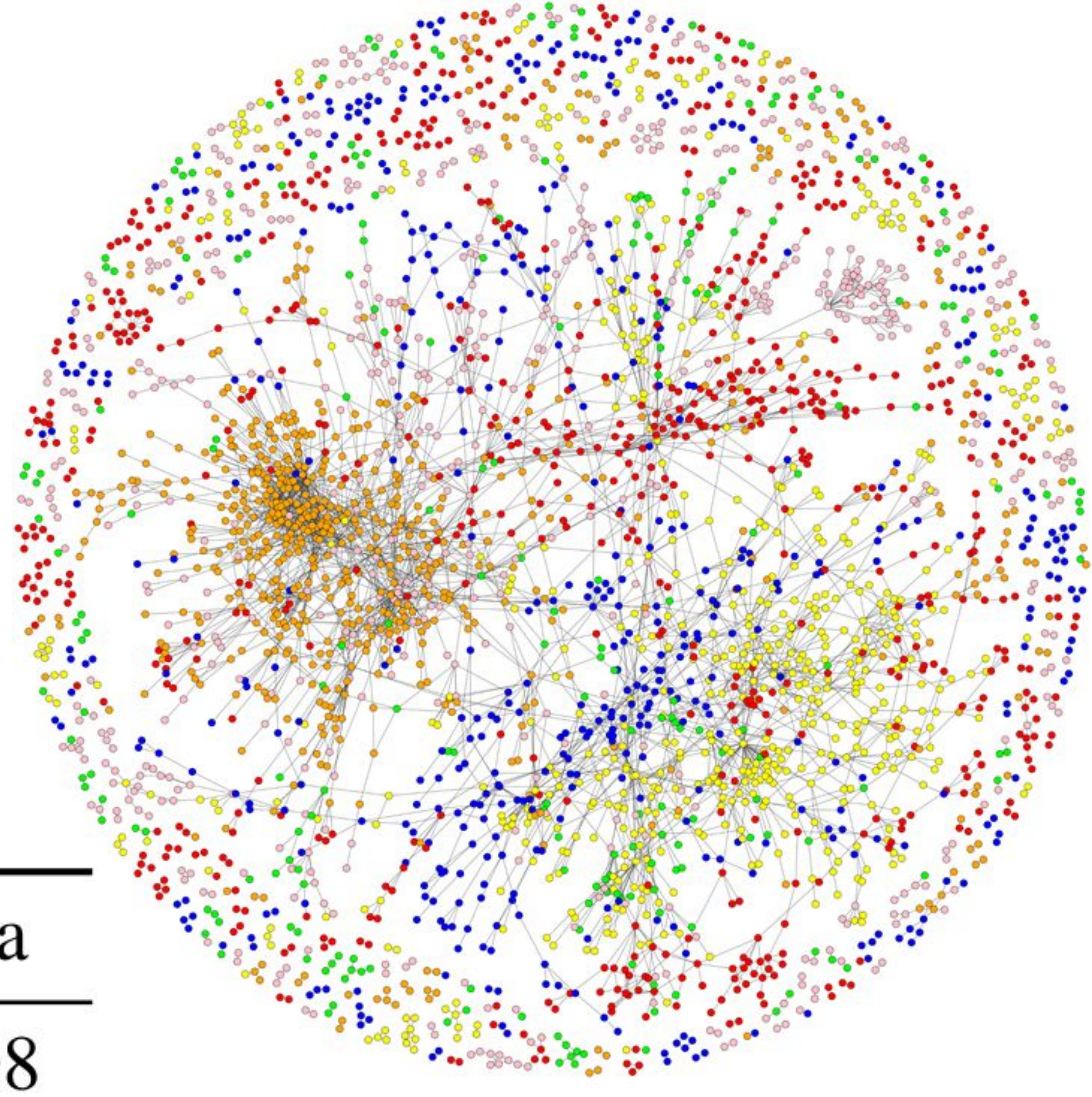
- Graph compatible functions
- Neural Tree architecture
- Approximation Results
- Experiments



3D Scene Graphs
(smaller graphs with low treewidth)

Citation Networks
(large treewidth graphs)

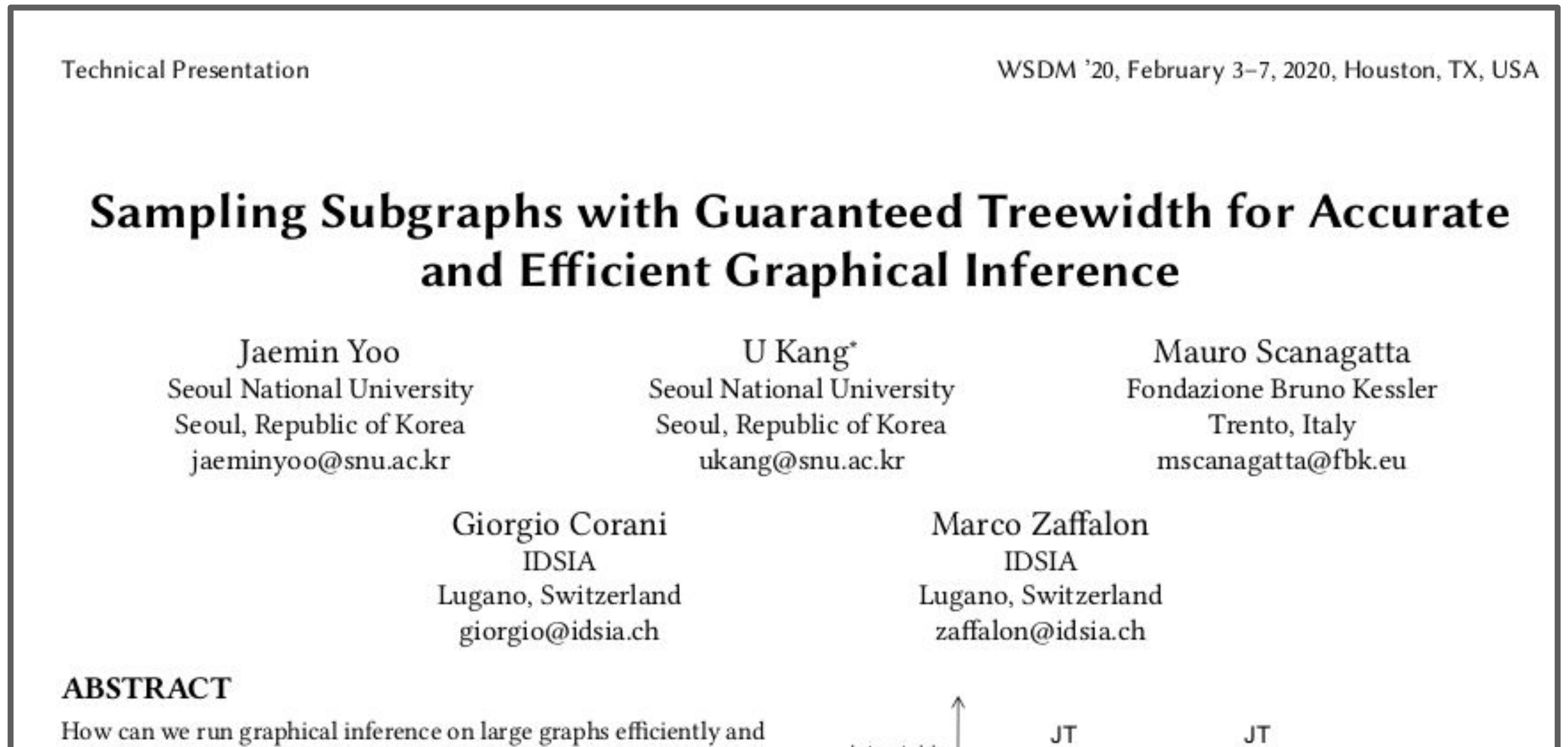
	PubMed	CiteSeer	Cora
Nodes	19,717	3,327	2,708
Edges	44,338	4,732	5,429
Classes	3	6	7



CiteSeer graph

Scalable Neural Tree

bounded treewidth subgraph sampling + Neural Tree



Technical Presentation WSDM '20, February 3–7, 2020, Houston, TX, USA

Sampling Subgraphs with Guaranteed Treewidth for Accurate and Efficient Graphical Inference

Jaemin Yoo, Seoul National University, Seoul, Republic of Korea, jaeminyoo@snu.ac.kr
U Kang*, Seoul National University, Seoul, Republic of Korea, ukang@snu.ac.kr
Mauro Scanagatta, Fondazione Bruno Kessler, Trento, Italy, mscanagatta@fbk.eu
Giorgio Corani, IDSIA, Lugano, Switzerland, giorgio@idsia.ch
Marco Zaffalon, IDSIA, Lugano, Switzerland, zaffalon@idsia.ch

ABSTRACT

How can we run graphical inference on large graphs efficiently and

In the remaining time ...

- Graph compatible functions
- Neural Tree architecture
- Approximation Results
- Experiments

Graph Compatible Function

A function on input node features

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n)$$

is said to be graph compatible w.r.t. graph $\mathcal{G}$ if it can be written as

$$f(\mathbf{X}) = \sum_C \theta_C(\mathbf{x}_C)$$

sum over all cliques in the graph

function over node features in the clique

Graph Compatible Function

A function on input node features

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n)$$

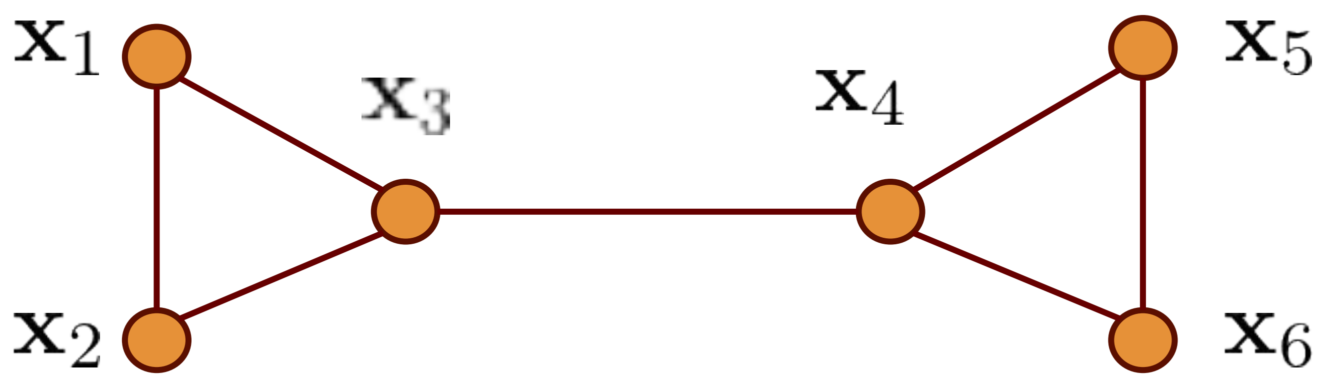
is said to be graph compatible w.r.t. graph $\mathcal{G}$ if it can be written as

$$f(\mathbf{X}) = \sum_{C} \theta_C(\mathbf{x}_C)$$

sum over all cliques in the graph

function over node features in the clique

Examples



$$f(\mathbf{x}_1, \ldots \mathbf{x}_6) = \mathbf{x}_1\mathbf{x}_2\mathbf{x}_3 + \mathbf{x}_3\mathbf{x}_4 + \mathbf{x}_4\mathbf{x}_5\mathbf{x}_6$$

Graph Compatible Function

A function on input node features

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n)$$

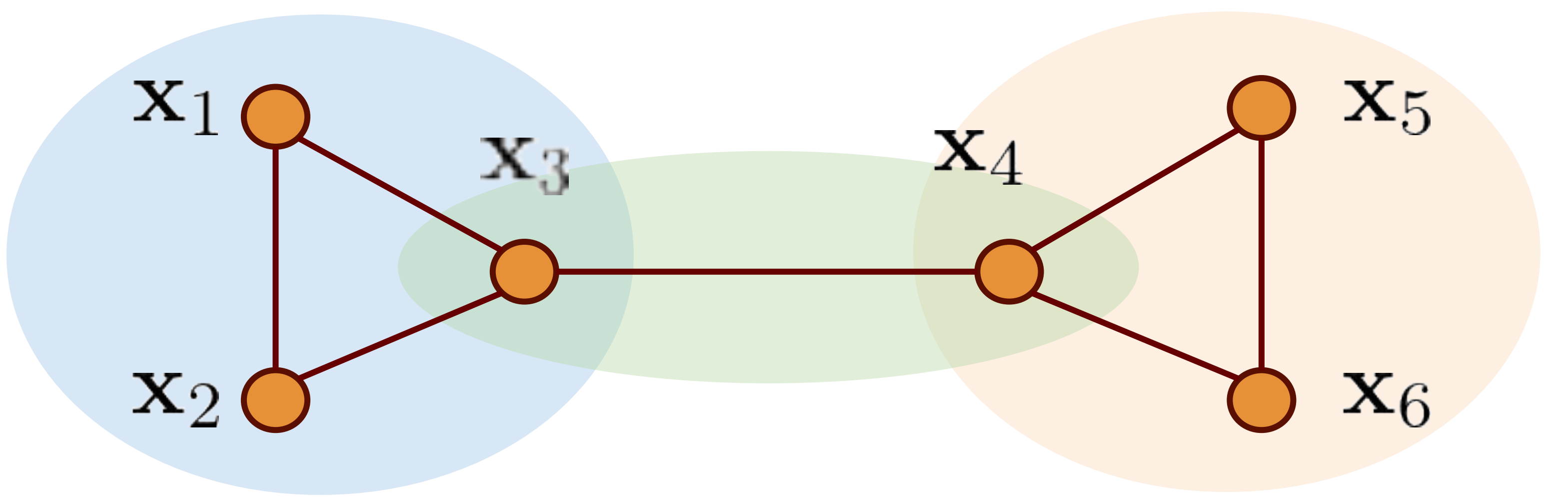
is said to be graph compatible w.r.t. graph $\mathcal{G}$ if it can be written as

$$f(\mathbf{X}) = \sum_C \theta_C(\mathbf{x}_C)$$

sum over all cliques in the graph

function over node features in the clique

Examples



$$f(\mathbf{x}_1, \ldots \mathbf{x}_6) = \mathbf{x}_1\mathbf{x}_2\mathbf{x}_3 + \mathbf{x}_3\mathbf{x}_4 + \mathbf{x}_4\mathbf{x}_5\mathbf{x}_6$$

Graph Compatible Function

A function on input node features

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n)$$

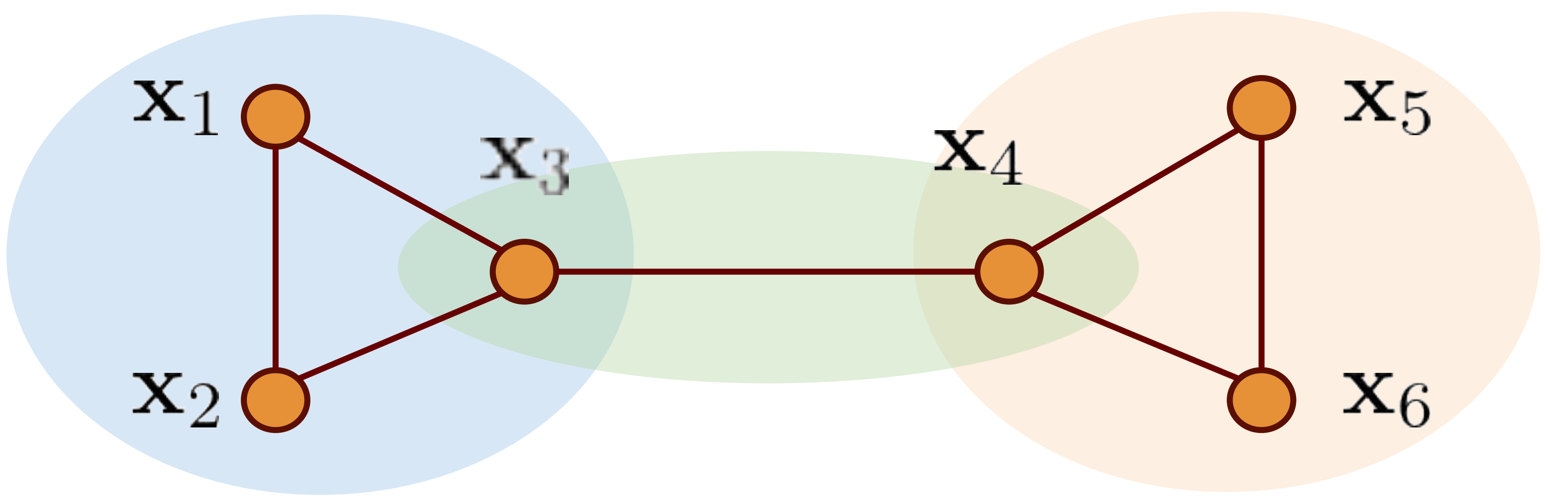
is said to be graph compatible w.r.t. graph $\mathcal{G}$ if it can be written as

$$f(\mathbf{X}) = \sum_C \theta_C(\mathbf{x}_C)$$

sum over all cliques in the graph

function over node features in the clique

Examples



$$f(\mathbf{x}_1, \ldots \mathbf{x}_6) = \mathbf{x}_1\mathbf{x}_2\mathbf{x}_3 + \mathbf{x}_3\mathbf{x}_4 + \mathbf{x}_4\mathbf{x}_5\mathbf{x}_6$$

$$f(\mathbf{x}_1, \ldots \mathbf{x}_4) = ||\mathbf{x}_1 - \mathbf{x}_2|| + ||\mathbf{x}_2 - \mathbf{x}_3|| + ||\mathbf{x}_3 - \mathbf{x}_4||$$

Graph Compatible Function

A function on input node features

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n)$$

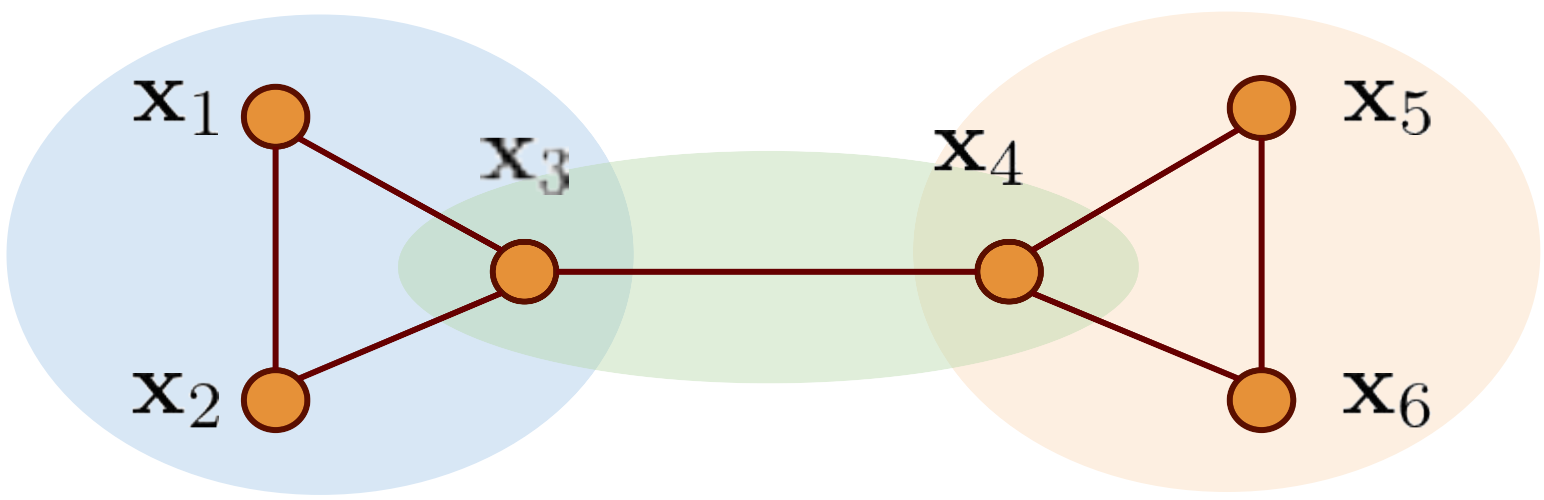
is said to be graph compatible w.r.t. graph $\mathcal{G}$ if it can be written as

$$f(\mathbf{X}) = \sum_{C} \theta_C(\mathbf{x}_C)$$

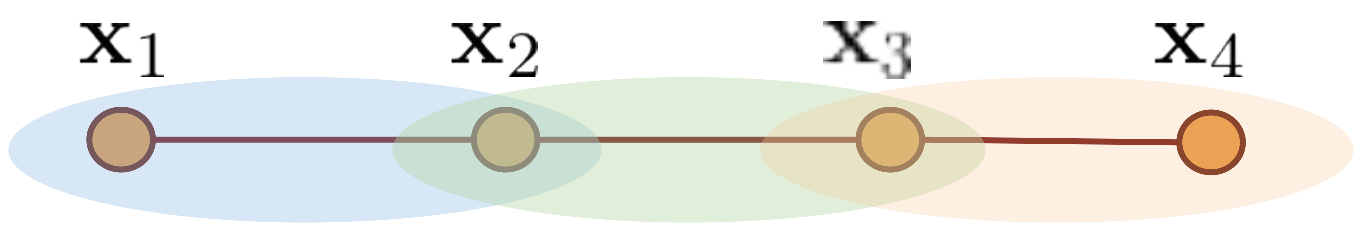
sum over all cliques in the graph

function over node features in the clique

Examples



$$f(\mathbf{x}_1, \ldots \mathbf{x}_6) = \mathbf{x}_1\mathbf{x}_2\mathbf{x}_3 + \mathbf{x}_3\mathbf{x}_4 + \mathbf{x}_4\mathbf{x}_5\mathbf{x}_6$$



$$f(\mathbf{x}_1, \ldots \mathbf{x}_4) = ||\mathbf{x}_1 - \mathbf{x}_2|| + ||\mathbf{x}_2 - \mathbf{x}_3|| + ||\mathbf{x}_3 - \mathbf{x}_4||$$

Relevance: Graphical Models

Graph compatible functions arise naturally in probabilistic graphical models

$$p(\mathbf{X}|\mathcal{G}) = \frac{1}{Z}\prod_{C}\psi_C(\mathbf{x}_C) = \exp\{\underbrace{-f(\mathbf{X})} - \log(Z)\}$$

prob. distribution over graph — $p(\mathbf{X}|\mathcal{G})$

clique potential — ψ_C

graph compatible function — $-f(\mathbf{X})$

Relevance: Graphical Models

Graph compatible functions arise naturally in probabilistic graphical models

$$p(\mathbf{X}|\mathcal{G}) = \frac{1}{Z}\prod_{C}\psi_C(\mathbf{x}_C) = \exp\{-f(\mathbf{X}) - \log(Z)\}$$

prob. distribution over graph — $p(\mathbf{X}|\mathcal{G})$

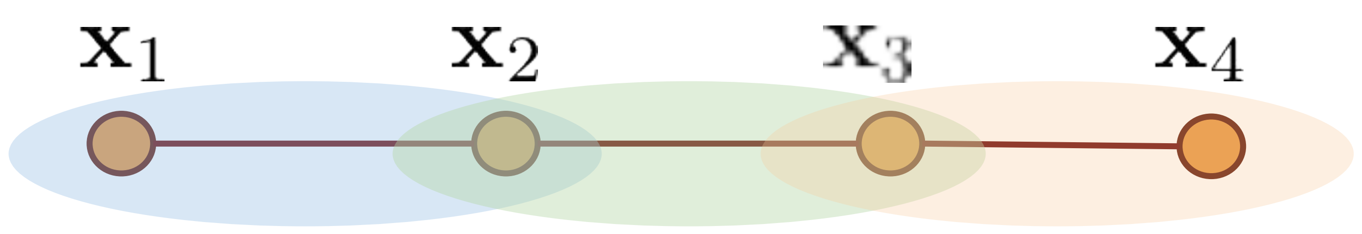
clique potential — ψ_C

graph compatible function — $f(\mathbf{X})$

Example: Markov Random Field

$$p(\mathbf{x}_1, \ldots \mathbf{x}_4|\mathcal{G}) = \frac{1}{Z}e^{-||\mathbf{x}_1 - \mathbf{x}_2||} \times e^{-||\mathbf{x}_2 - \mathbf{x}_3||} \times e^{-||\mathbf{x}_3 - \mathbf{x}_4||}$$

$$= \exp\{-f(\mathbf{X}) - \log(Z)\}$$



$$f(\mathbf{x}_1, \ldots \mathbf{x}_4) = ||\mathbf{x}_1 - \mathbf{x}_2|| + ||\mathbf{x}_2 - \mathbf{x}_3|| + ||\mathbf{x}_3 - \mathbf{x}_4||$$

Relevance: Approximating Inference

Graph compatible functions arise naturally in probabilistic graphical models

$$p(\mathbf{X}|\mathcal{G}) = \frac{1}{Z}\prod_{C}\psi_C(\mathbf{x}_C) = \exp\{-\underbrace{f(\mathbf{X})} - \log(Z)\}$$

prob. distribution over graph

clique potential

graph compatible function

Implication:

Approximating graph compatible functions ⟷ Approximating exact inference on probabilistic graphical models

(see Appendix A in the paper).

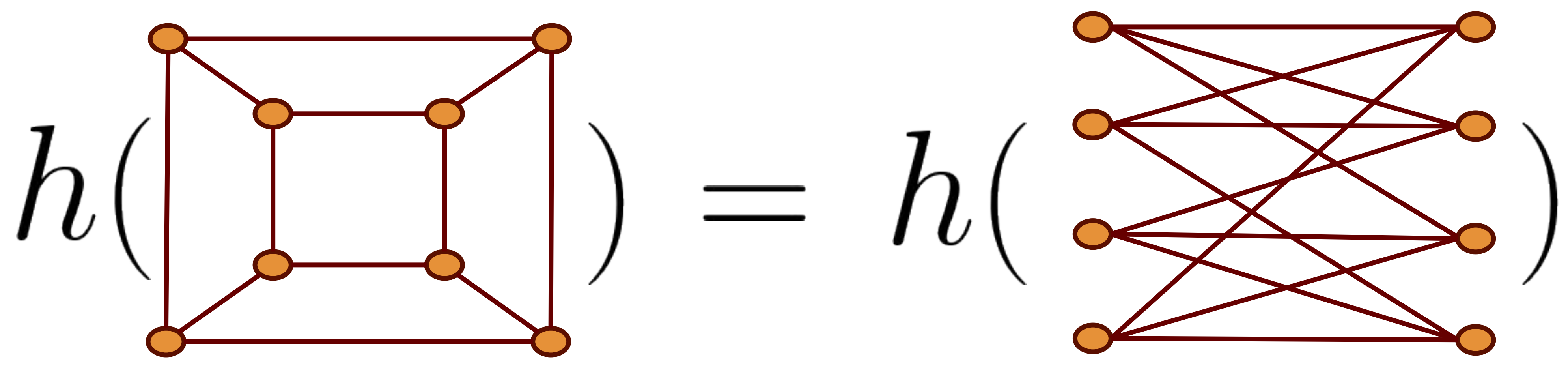
Relevance: Approx. Invariant/Equivariant Functions

We prove that graph compatible functions can be used to approximate graph invariant/equivariant functions.

Invariant function

$$h(\mathbf{X}^{\sigma}, \mathcal{G}^{\sigma}) = h(\mathbf{X}, \mathcal{G})$$

for all node permutations σ



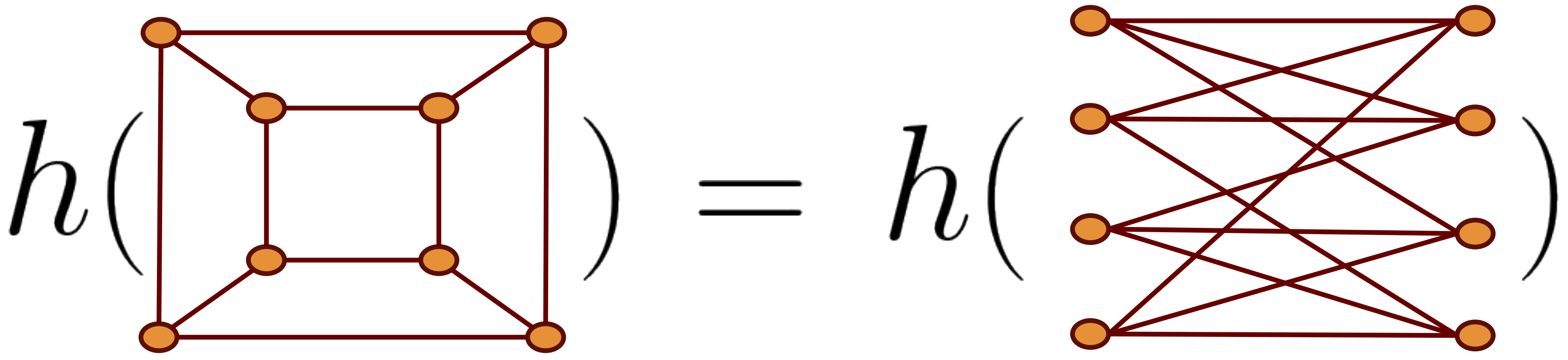
Relevance: Approx. Invariant/Equivariant Functions

We prove that graph compatible functions can be used to approximate graph invariant/equivariant functions.

Invariant function

$$h(\mathbf{X}^{\sigma}, \mathcal{G}^{\sigma}) = h(\mathbf{X}, \mathcal{G})$$

for all node permutations σ



Theorem:

For every invariant function h and an $\epsilon > 0$ there exists M graph compatible functions f^i such that

$$\sup_{\mathbf{X} \in \mathbb{X}} \left| h(\mathbf{X}) - \sum_{i=1}^{M} \phi\left(f^i(\mathbf{X})\right) \right| < \epsilon$$

some non-linear function $\phi : \mathbb{R} \to \mathbb{R}$

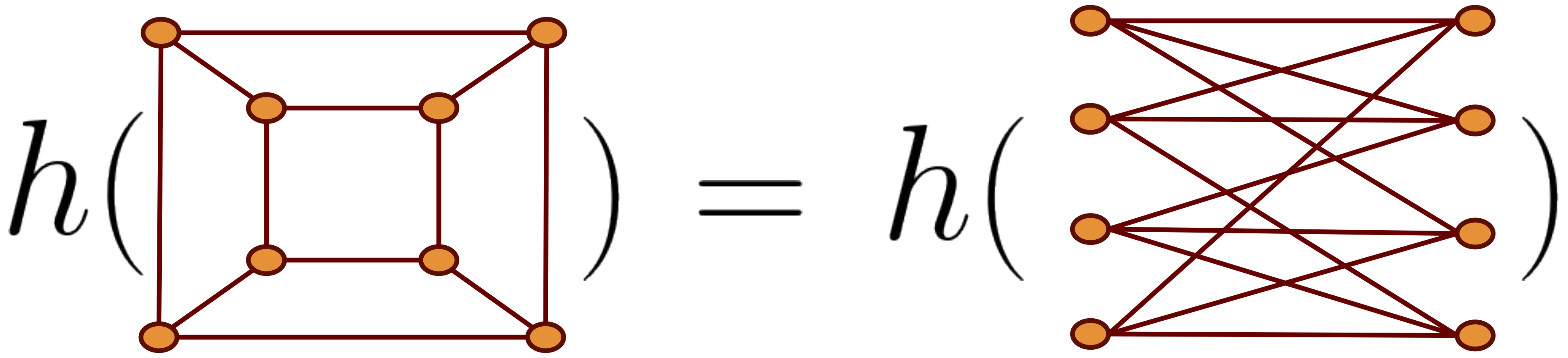
Relevance: Approx. Invariant/Equivariant Functions

We prove that graph compatible functions can be used to approximate graph invariant/equivariant functions.

Invariant function

$$h(\mathbf{X}^\sigma, \mathcal{G}^\sigma) = h(\mathbf{X}, \mathcal{G})$$

for all node permutations σ



Theorem:

For every invariant function h and an $\epsilon > 0$ there exists M graph compatible functions f^i such that

$$\sup_{\mathbf{X}\in\mathbb{X}} \left| h(\mathbf{X}) - \sum_{i=1}^{M} \phi\left(f^i(\mathbf{X})\right) \right| < \epsilon$$

some non-linear function $\phi : \mathbb{R} \to \mathbb{R}$

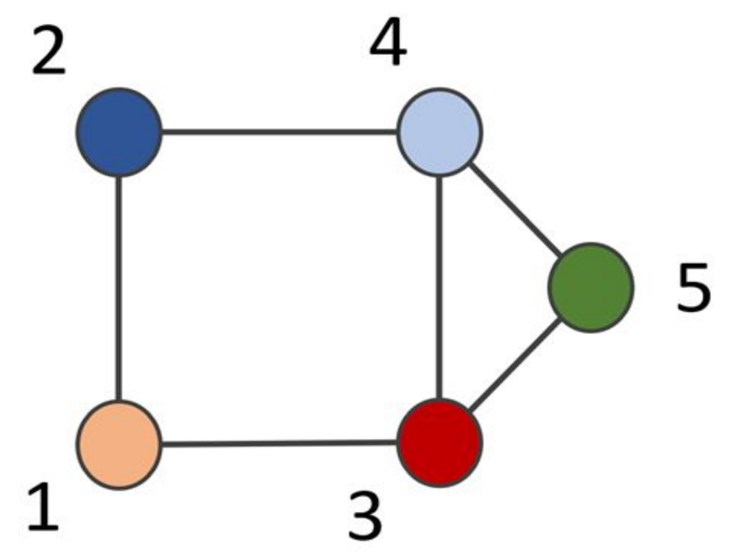
Similar result holds for equivariant functions

Neural Tree Architecture

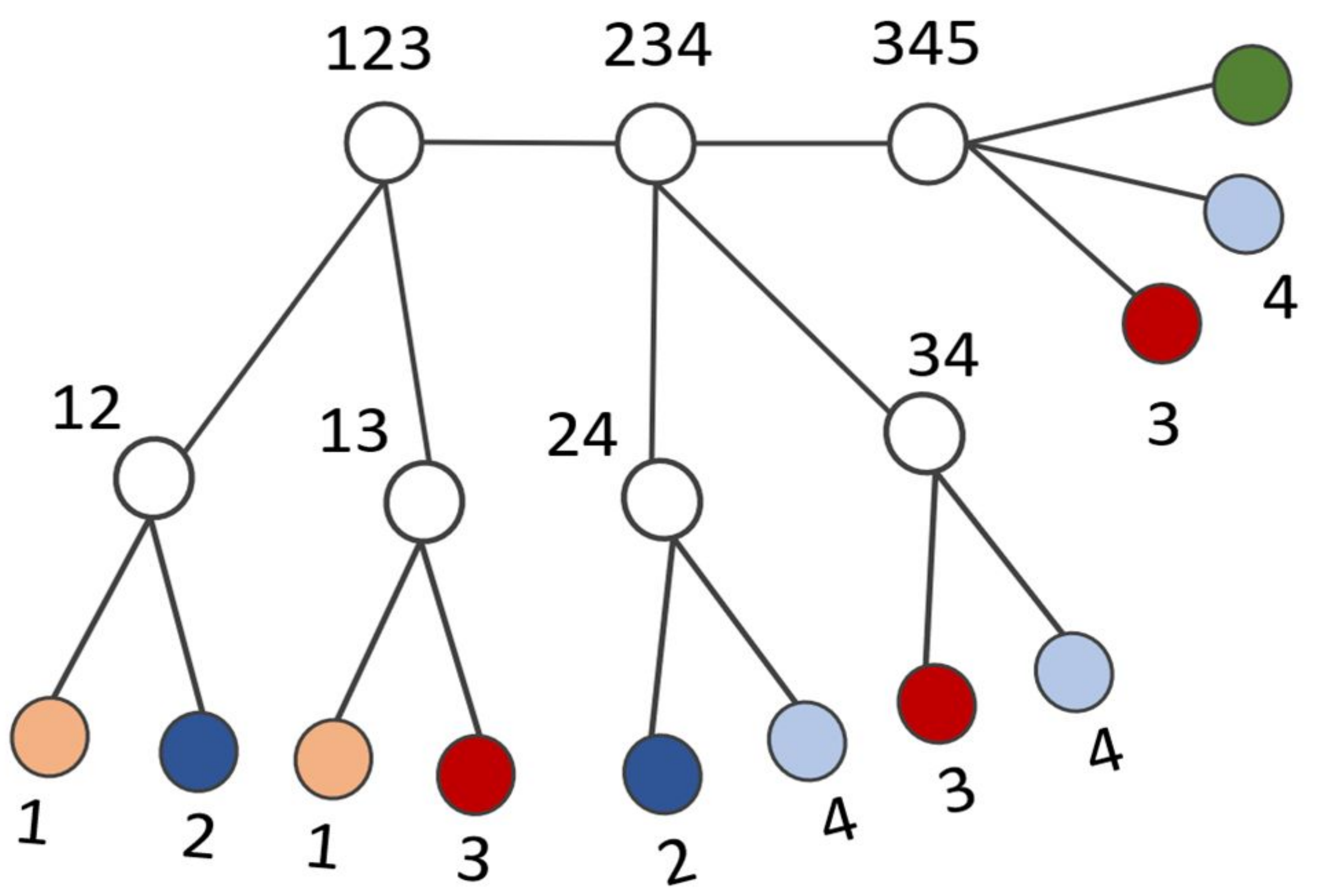
An architecture to approximate any (smooth) graph compatible function

Basic Idea

1. Convert graph to a tree, called H-tree
2. Neural Tree arch. = Message passing on the H-tree



Input graph with node attributes (colors)



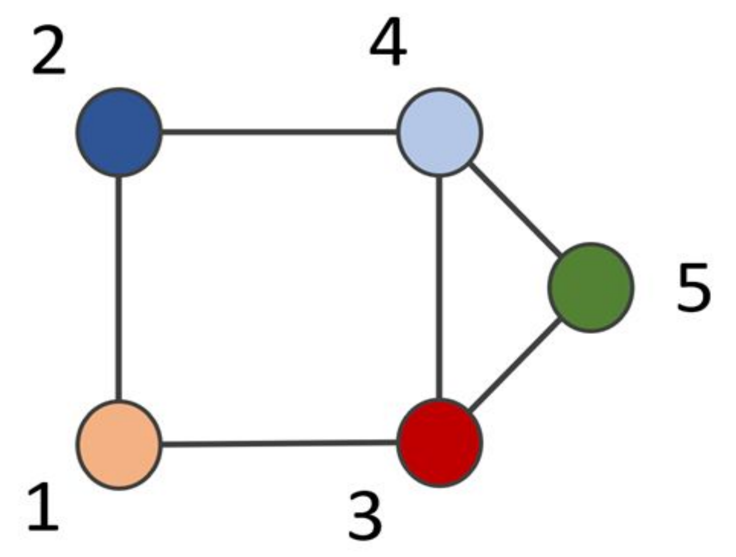
H-tree

Neural Tree Architecture

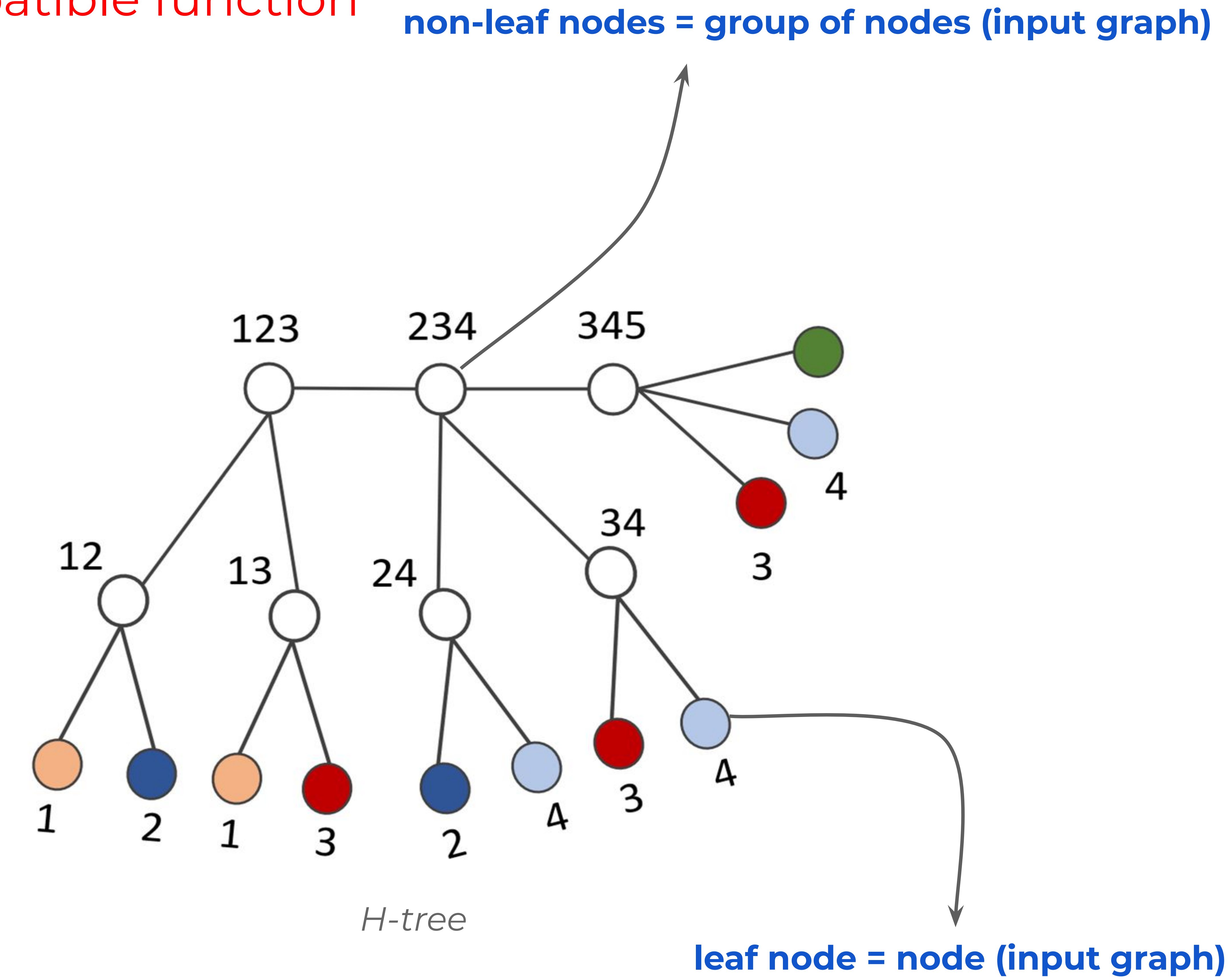
An architecture to approximate any (smooth) graph compatible function

Basic Idea

1. Convert graph to a tree, called H-tree
2. Neural Tree arch. = Message passing on the H-tree



Input graph with node attributes (colors)



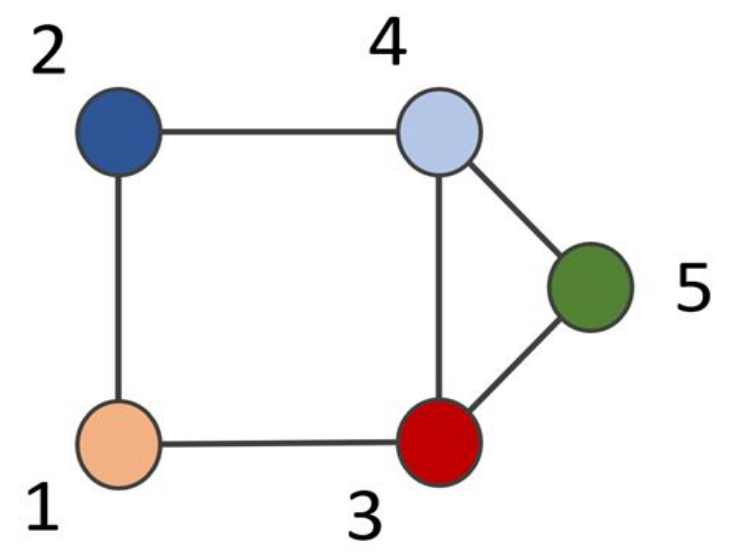
H-tree

Neural Tree Architecture

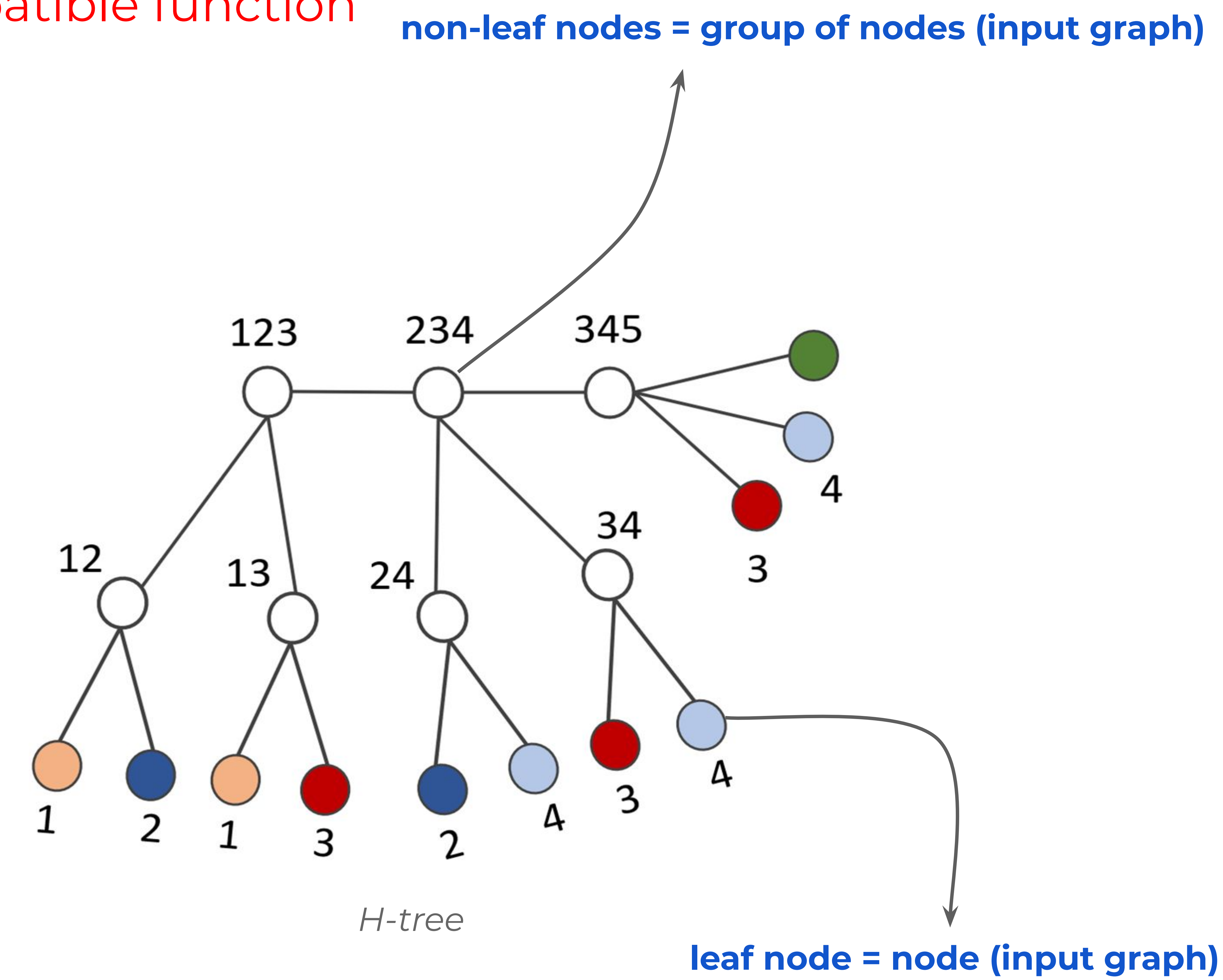
An architecture to approximate any (smooth) graph compatible function

Basic Idea

1. Convert graph to a tree, called H-tree
2. Neural Tree arch. = Message passing on the H-tree



Input graph with node attributes (colors)



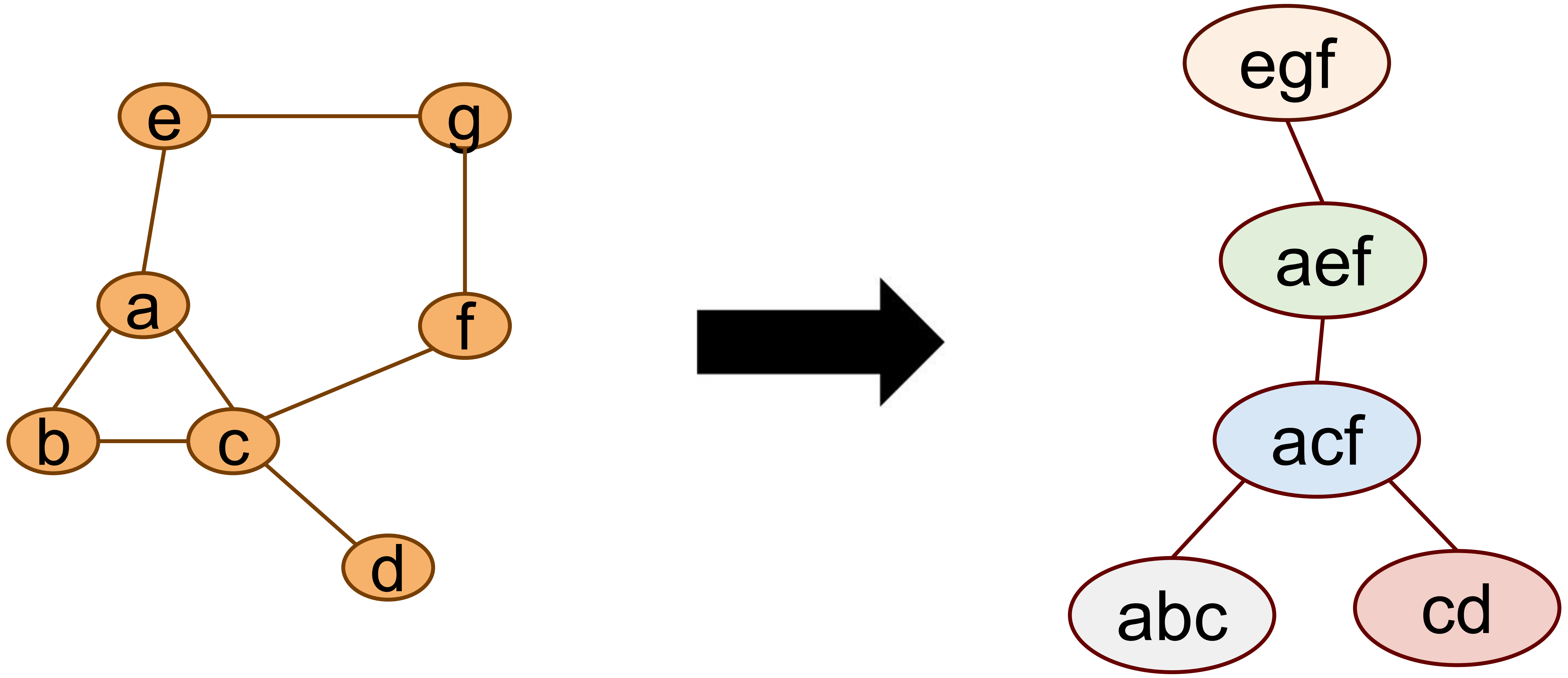
H-tree

The H-tree is constructed by successive **tree decomposition**

Recall: Tree Decomposition

Tree decomposition is a *tree structured graph* such that

Graph



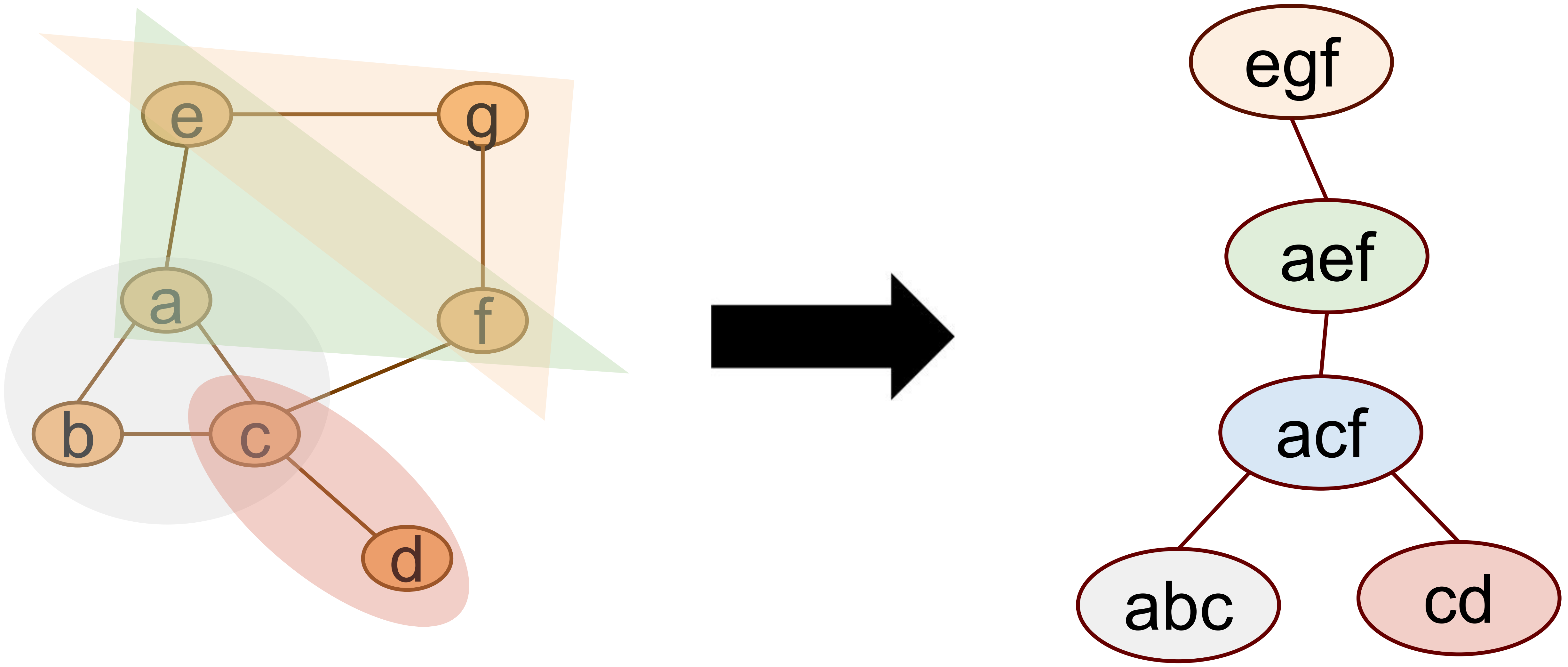
each node (bag) in the *tree graph* corresponds to a group of nodes in the input graph

Tree decomposition

Recall: Tree Decomposition

Tree decomposition is a *tree structured graph* such that

Graph



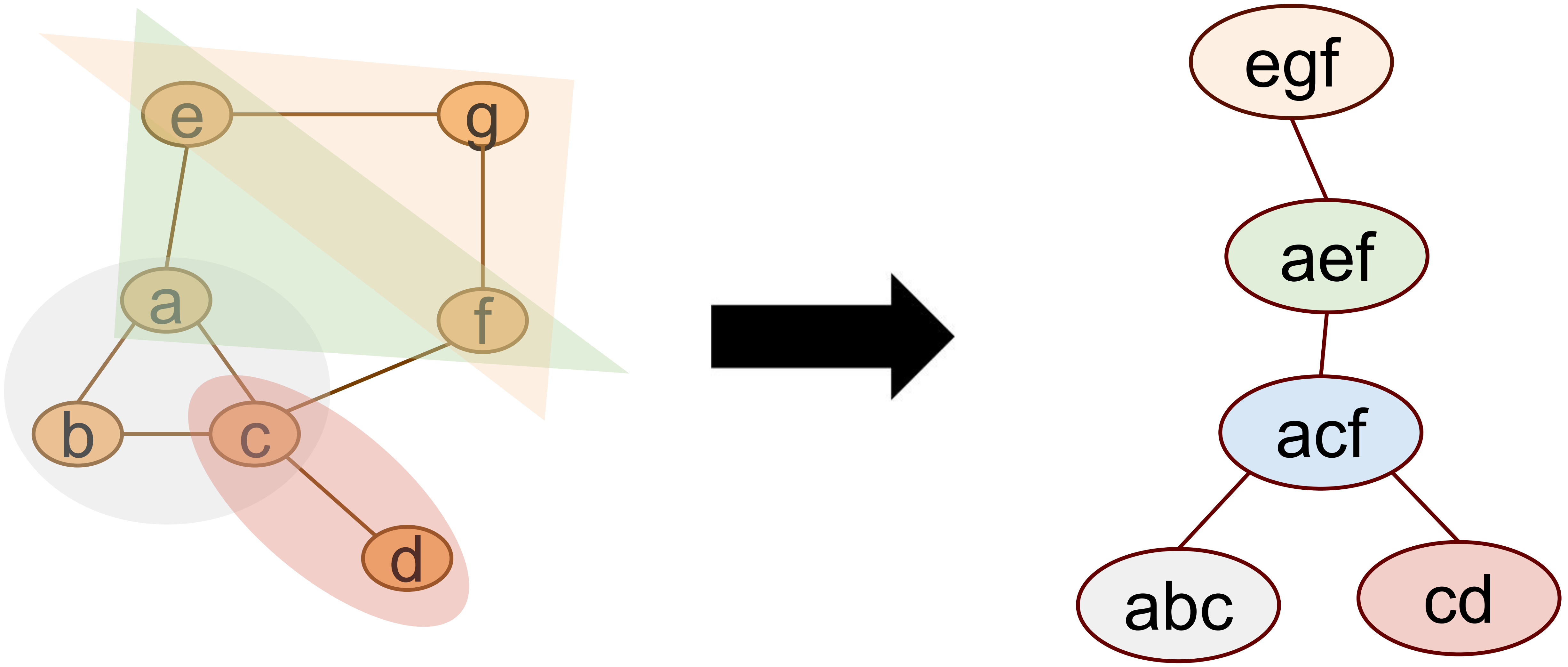
each node (bag) in the *tree graph* corresponds to a group of nodes in the input graph

Tree decomposition

Recall: Tree Decomposition

Tree decomposition is a *tree structured graph* such that

Graph



each node (bag) in the *tree graph* corresponds to a group of nodes in the input graph

Tree decomposition

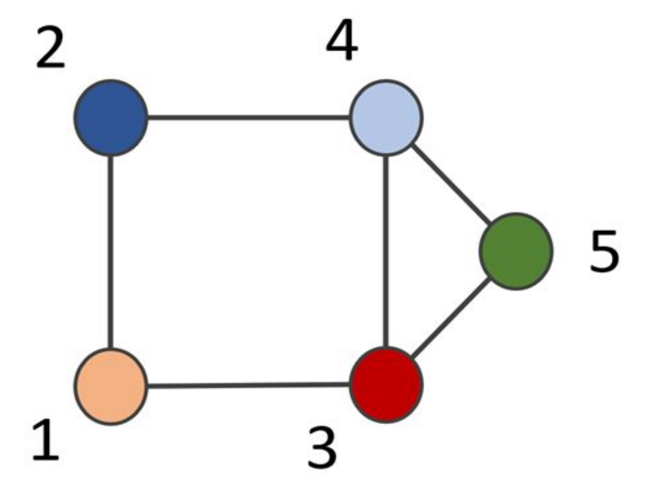
A well studied structure in graph optimization, combinatorial optimization, and probabilistic graphical models.

Junction Tree Algorithm:

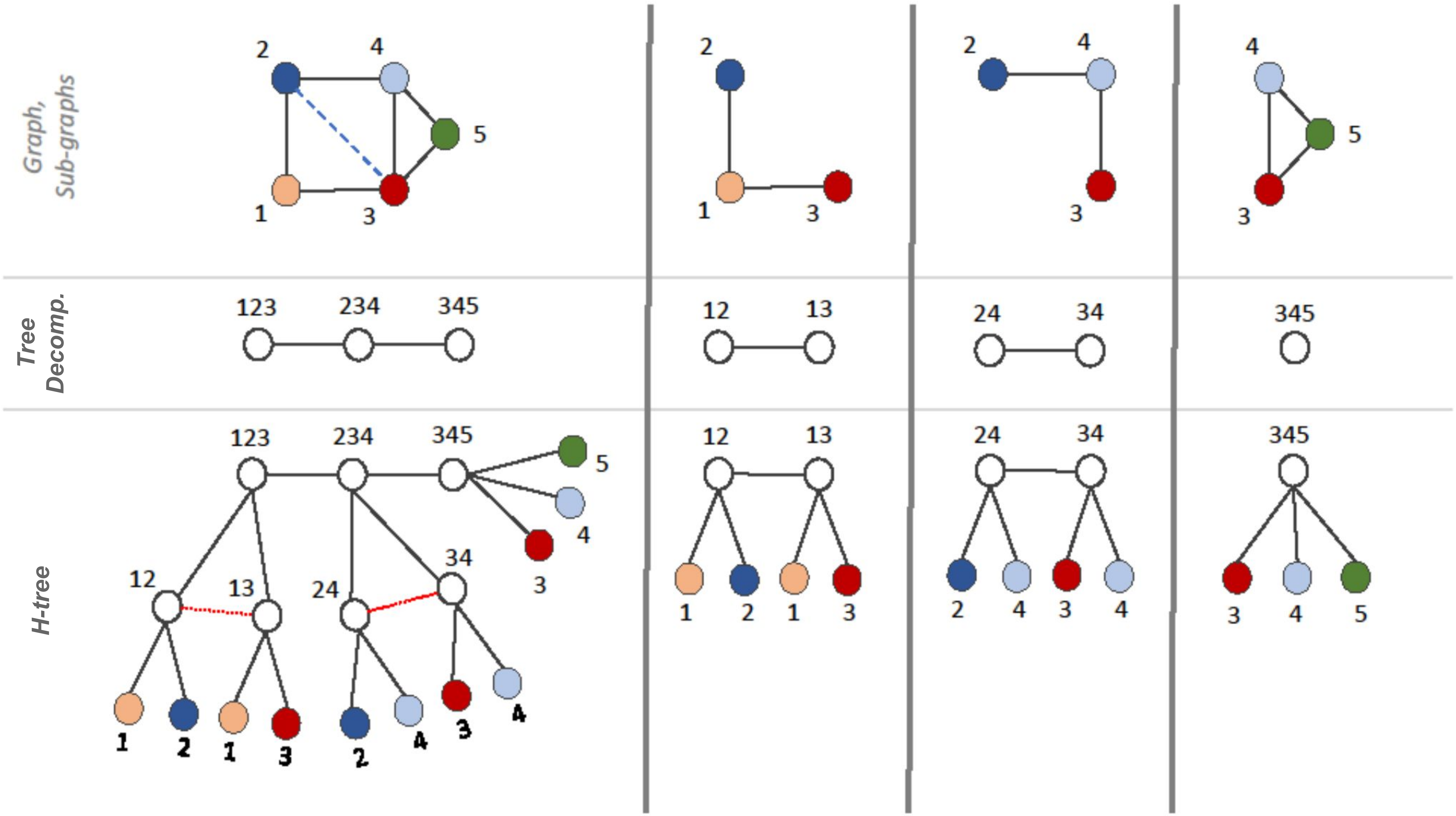
- Message passing on tree decomposition of input graph
- Algorithm for exact inference

H-tree Construction

Successive tree decomposition

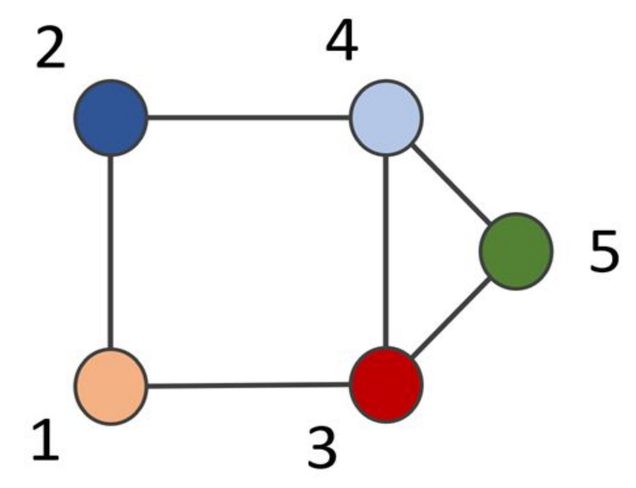


Input graph with node attributes (colors)



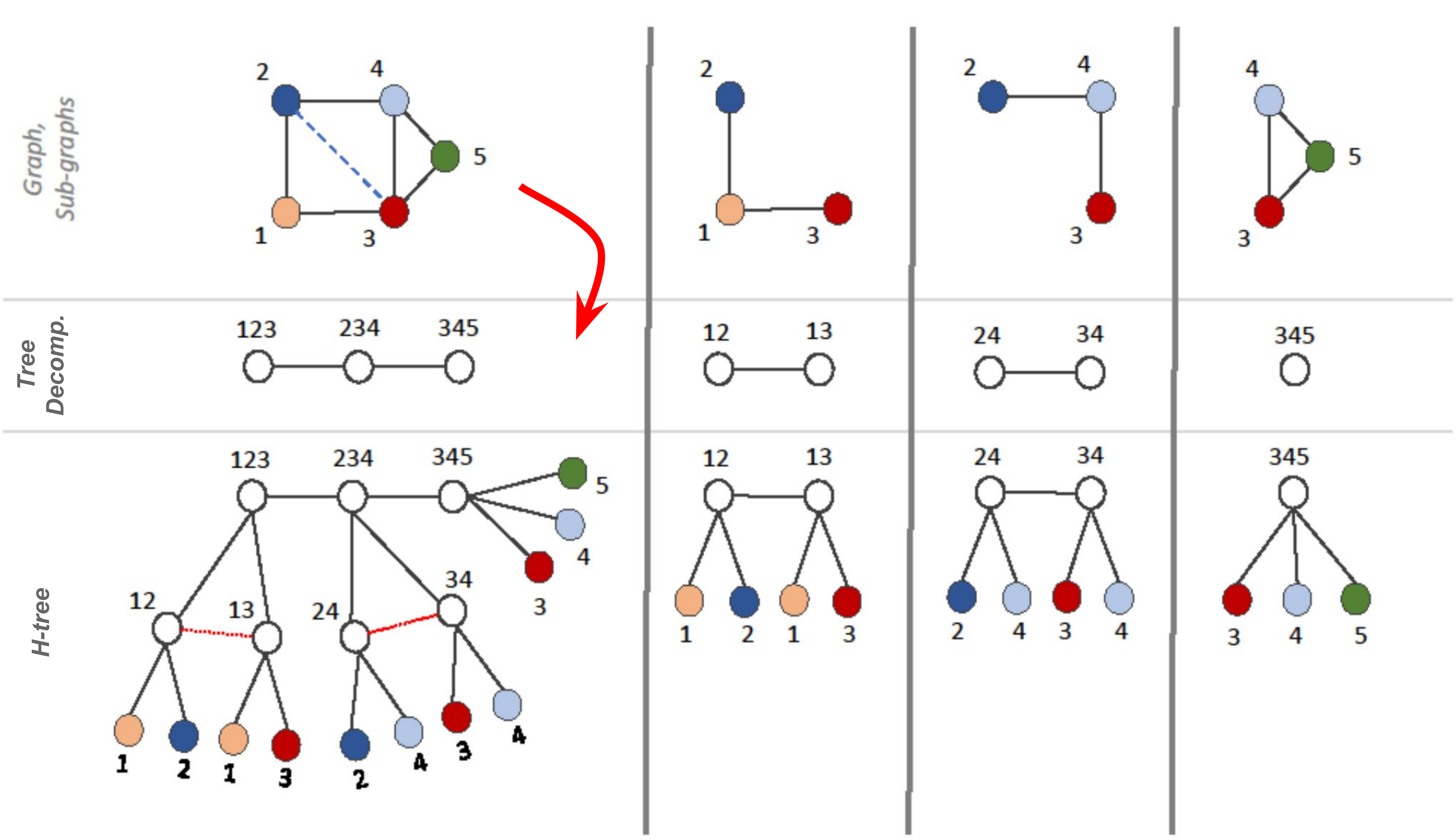
H-tree Construction

Successive tree decomposition



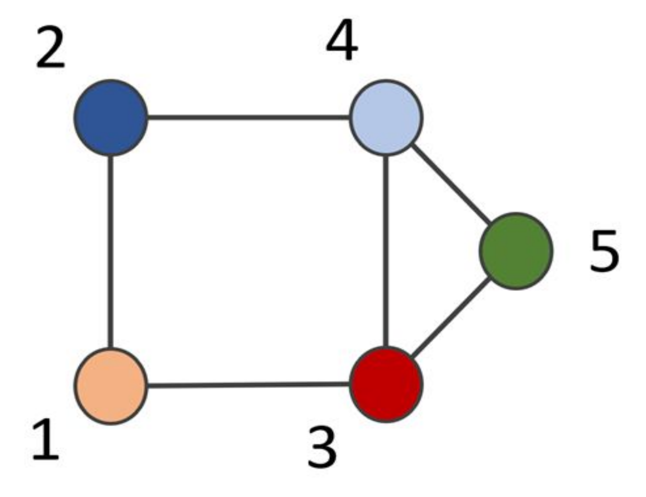
Input graph with node attributes (colors)

Step 1: Tree decomposition of the input graph



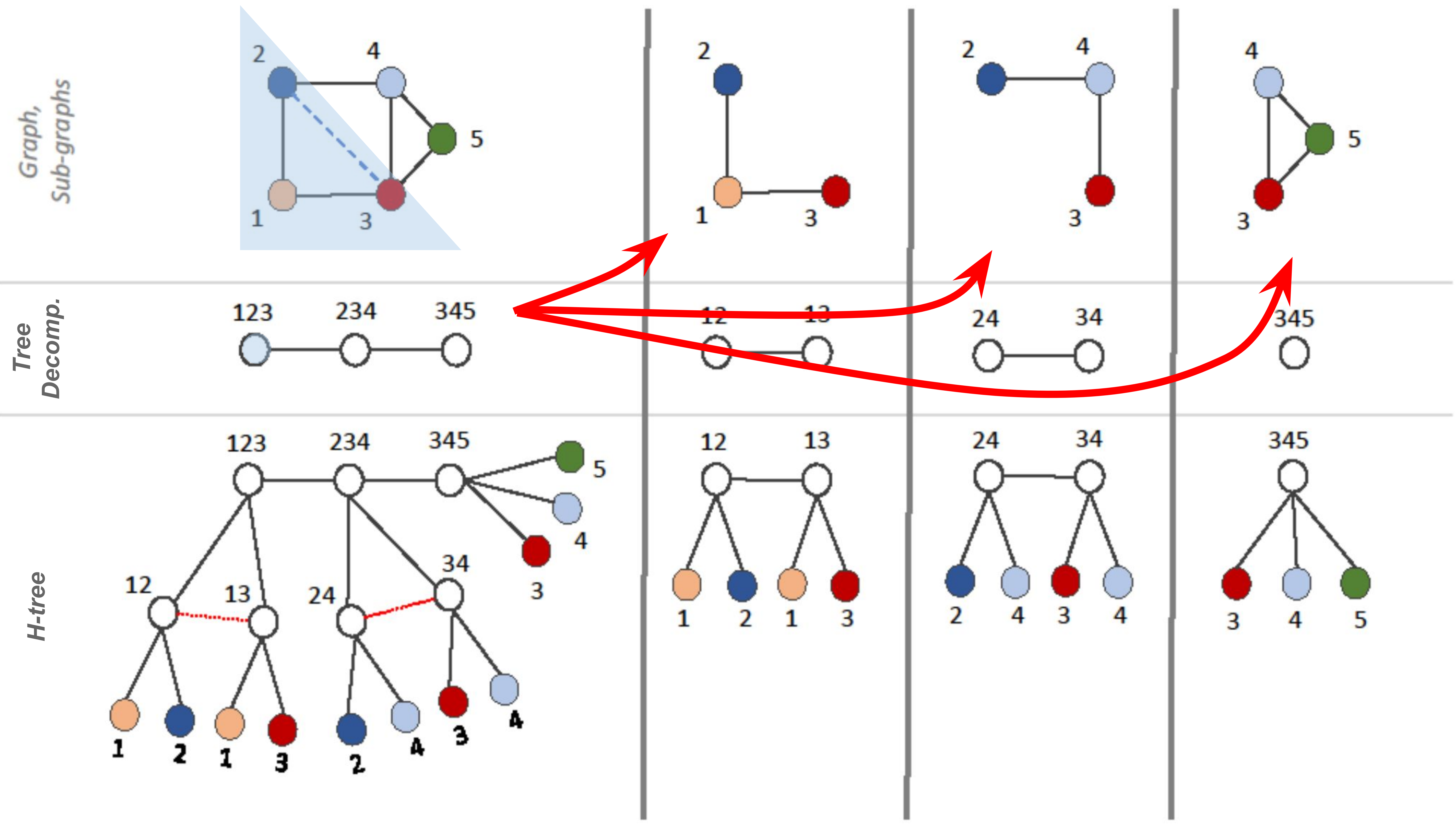
H-tree Construction

Successive tree decomposition



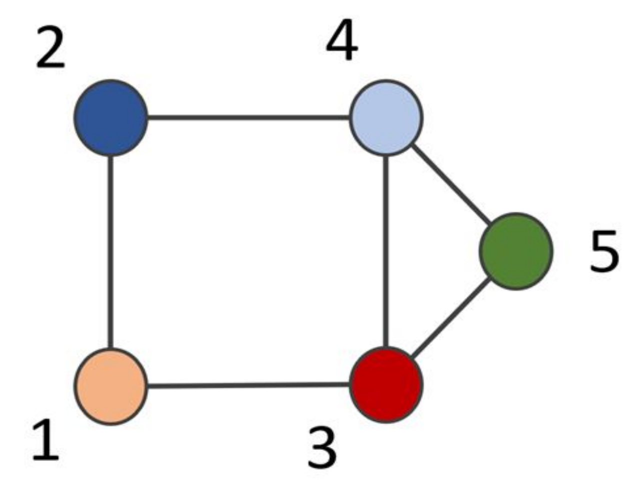
Input graph with node attributes (colors)

Step 2: Get subgraph corresponding to each bag

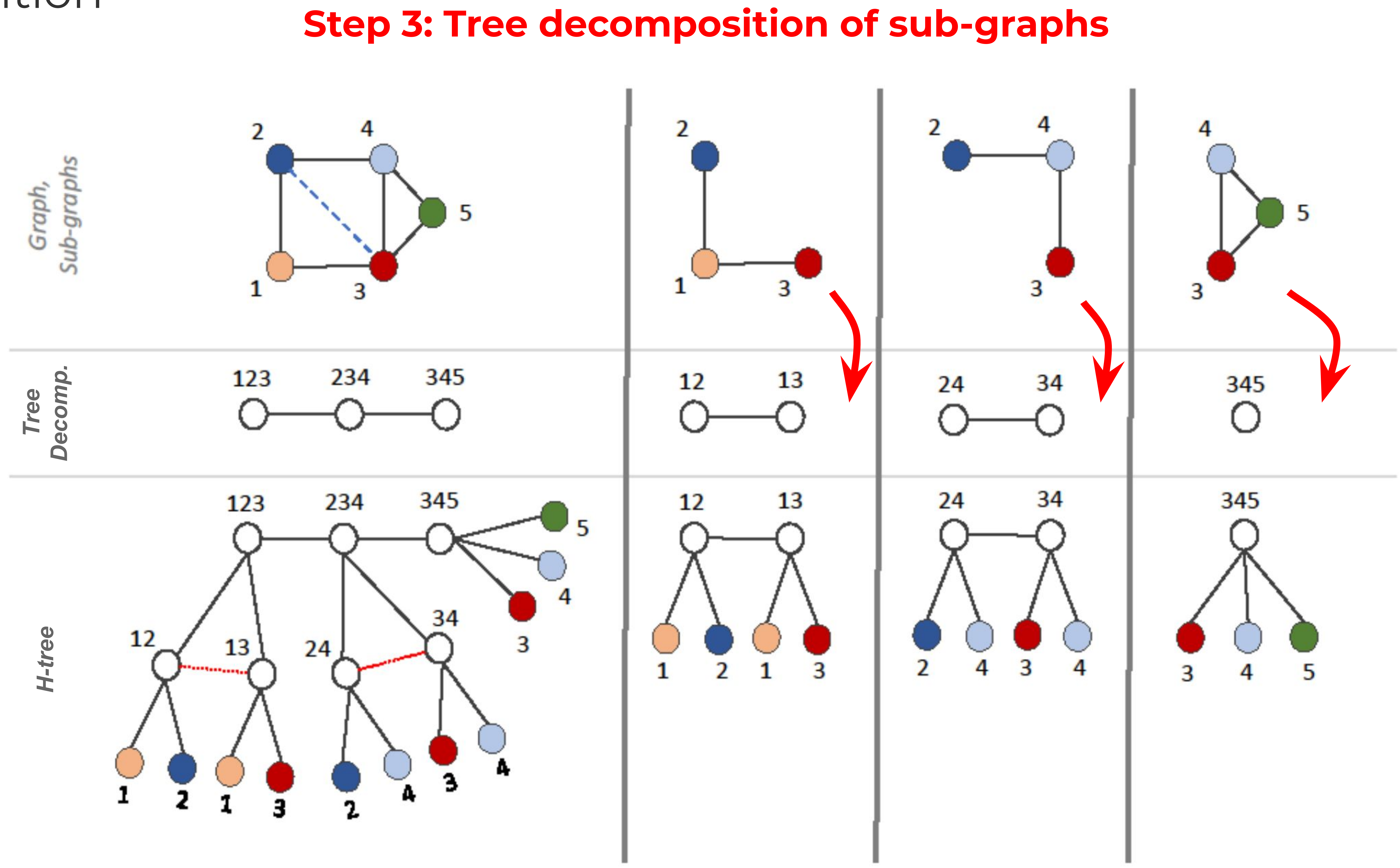


H-tree Construction

Successive tree decomposition



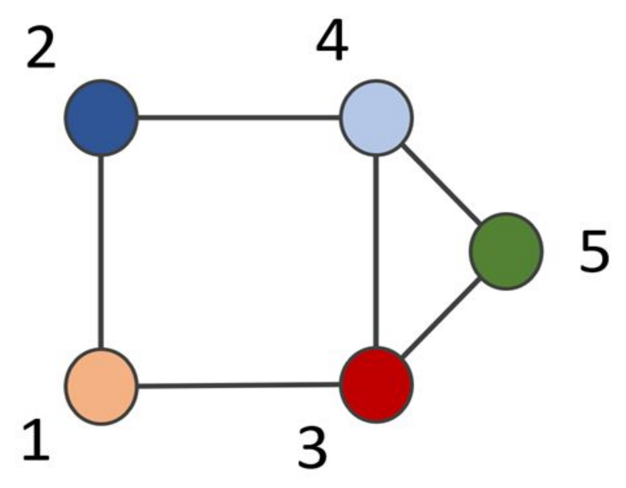
Input graph with node attributes (colors)



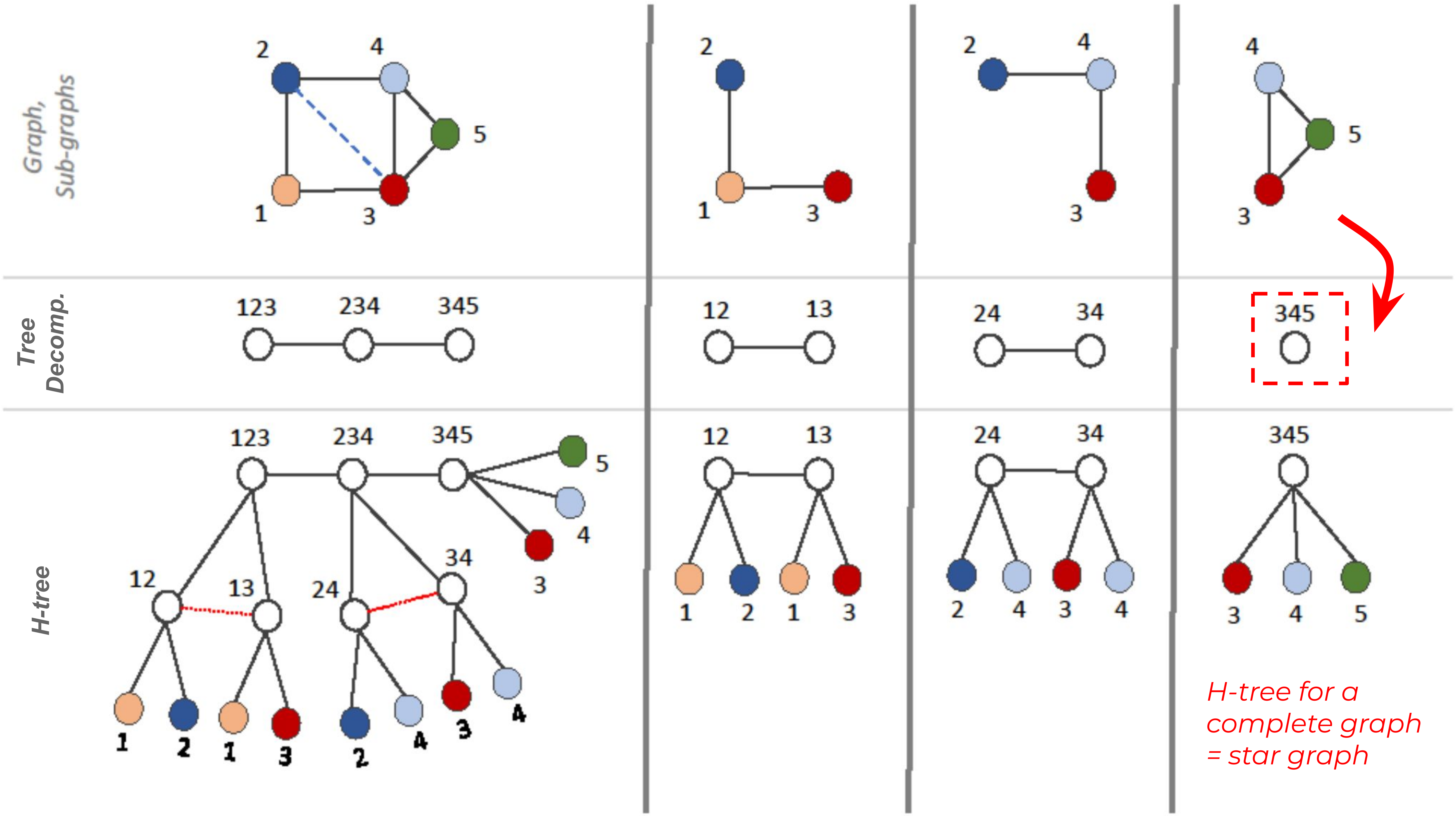
H-tree Construction

Successive tree decomposition

Iterate till all the sub-graphs are complete graphs

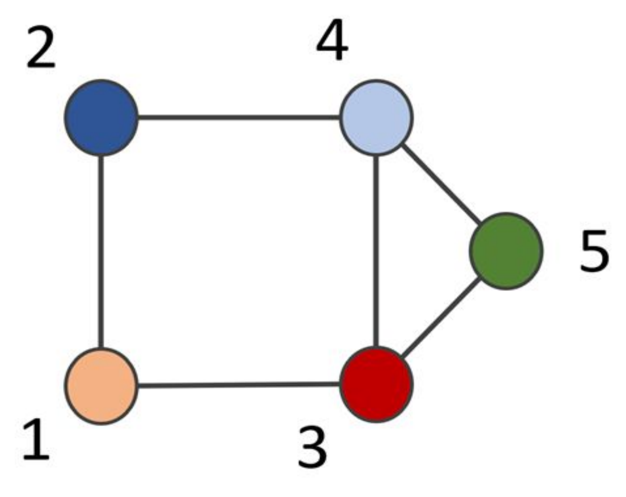


Input graph with node attributes (colors)



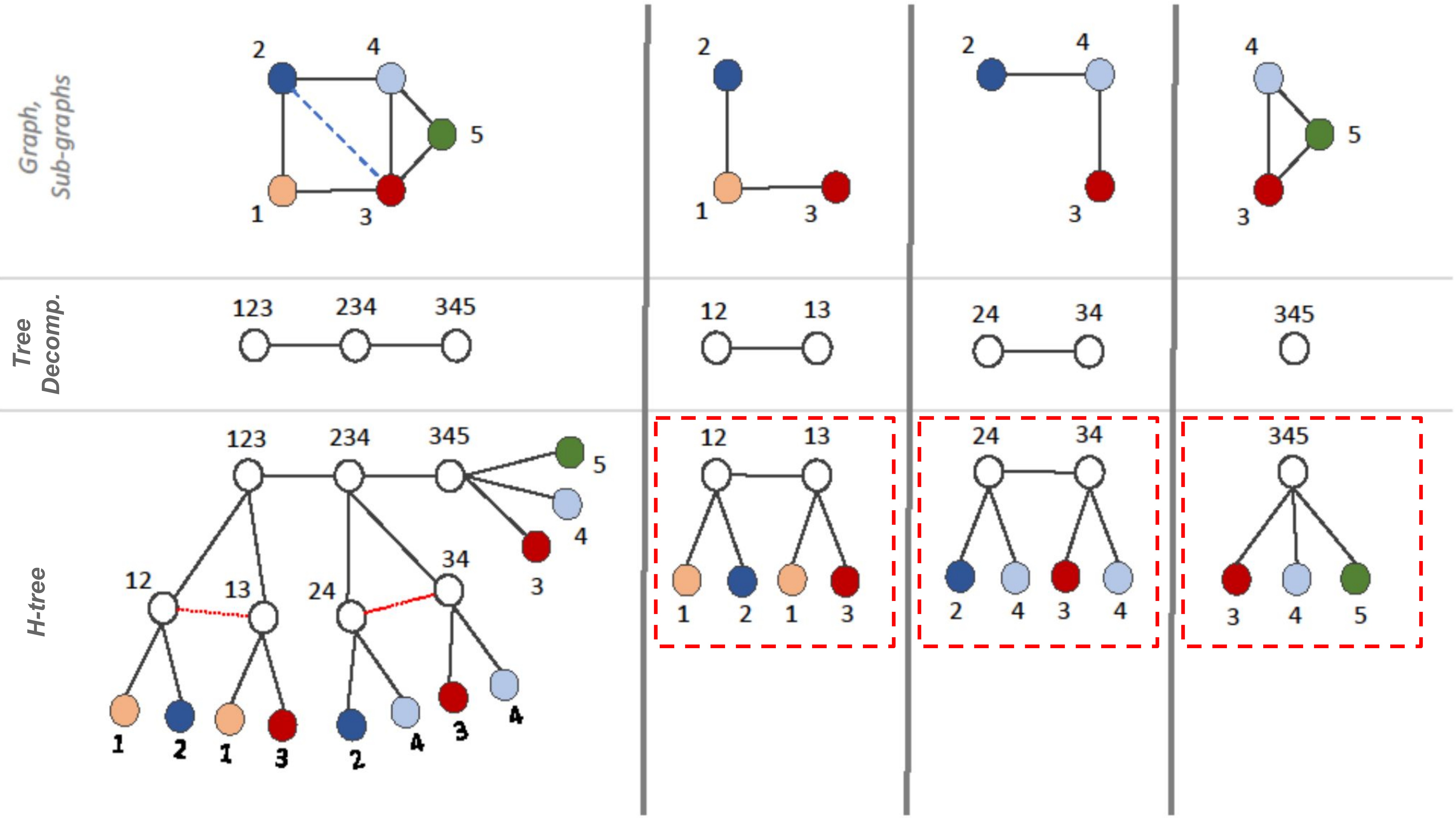
H-tree Construction

Successive tree decomposition



Input graph with node attributes (colors)

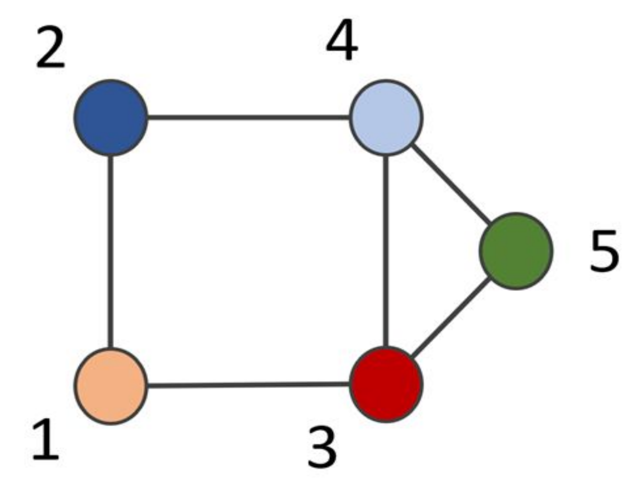
Iterate till all the sub-graphs are complete graphs



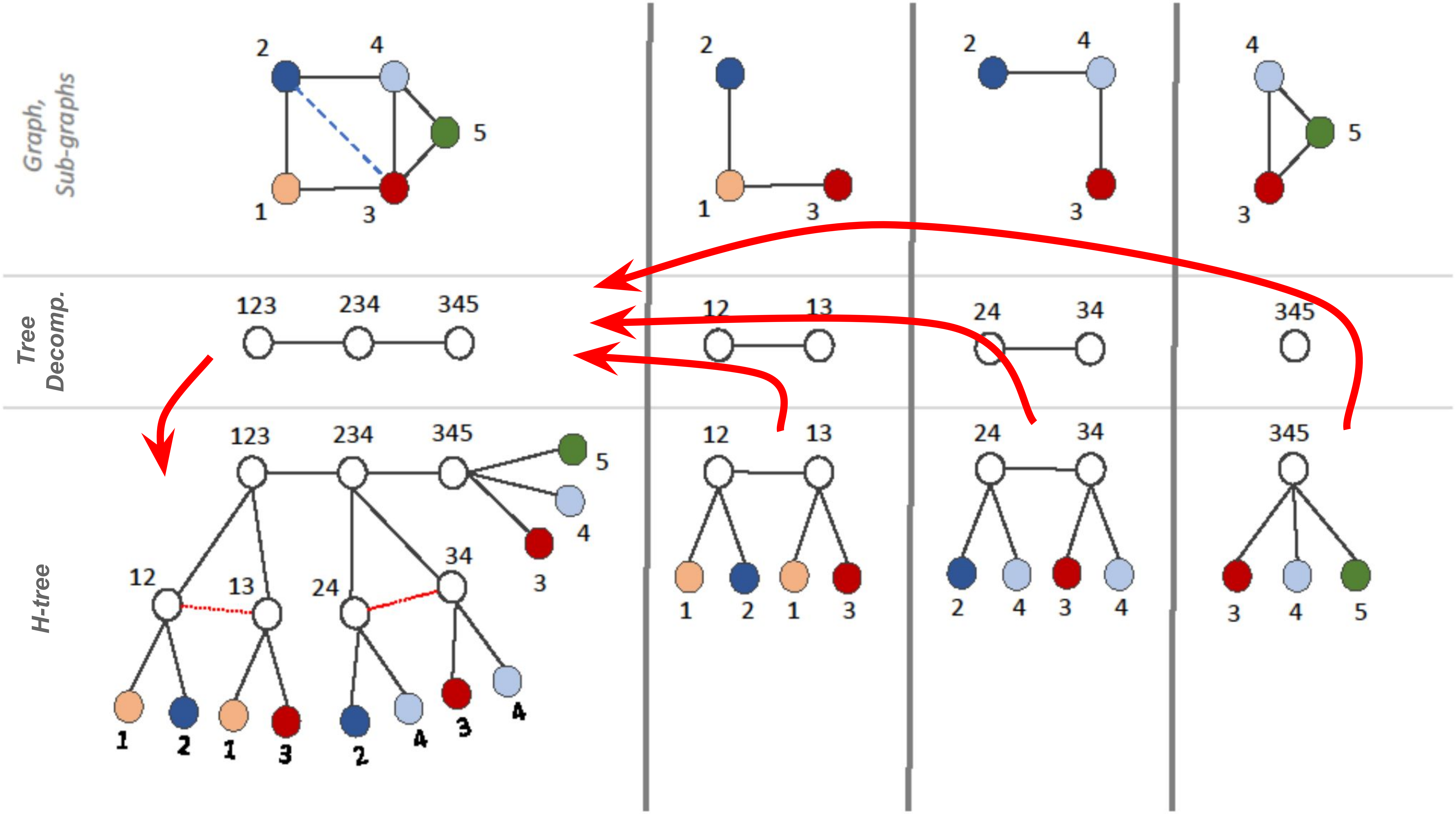
H-tree Construction

Successive tree decomposition

Finally, Connect all H-trees of sub-graphs to the tree-decomposition of the input graph



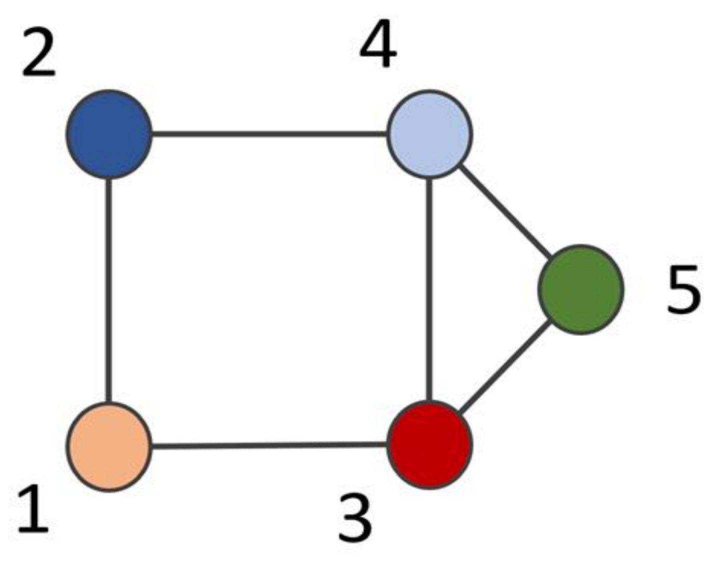
Input graph with node attributes (colors)



Neural Tree Architecture

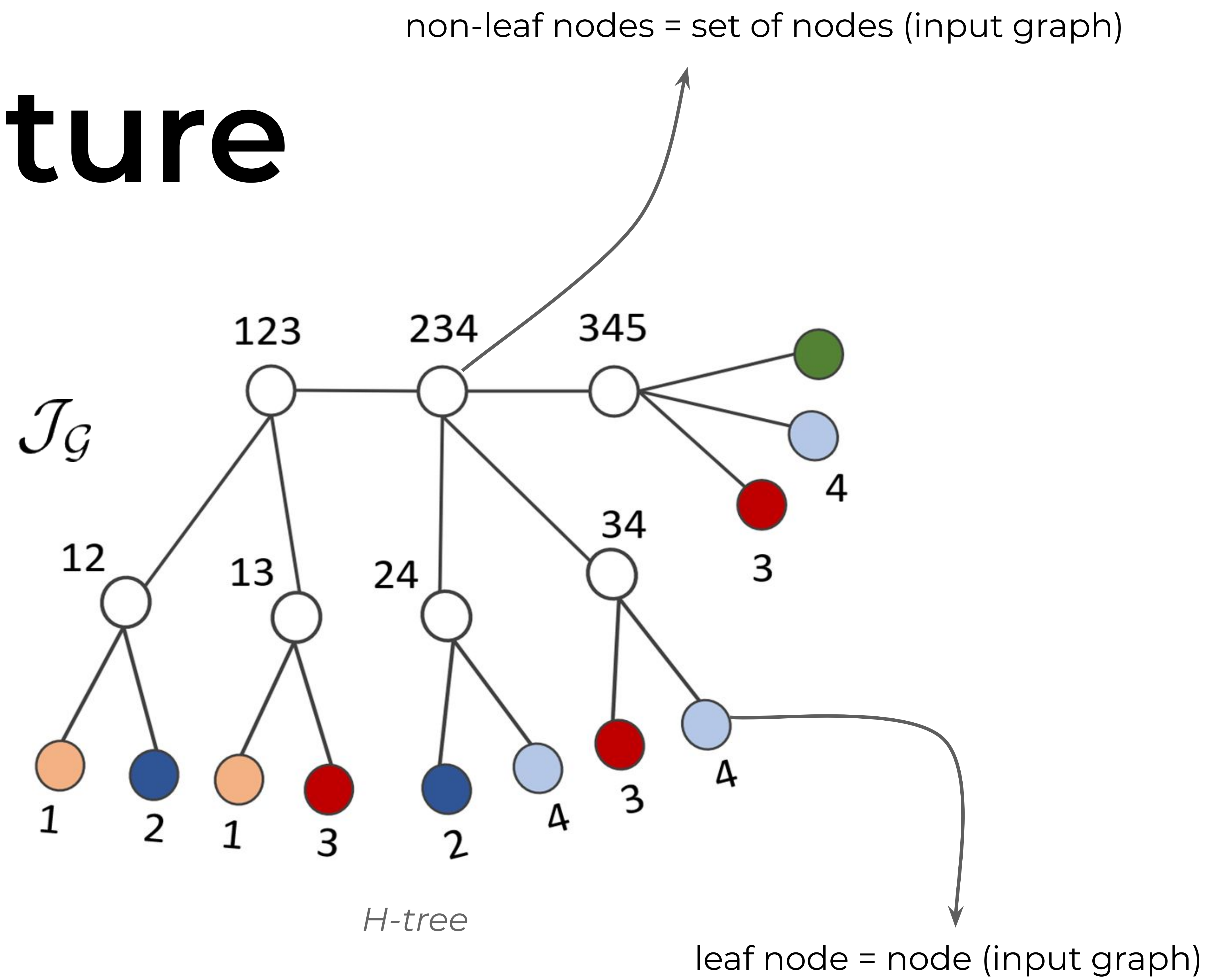
1. Convert graph to a tree, called H-tree

$(\mathcal{G}, \mathbf{X})$



Input graph with node attributes (colors)

successive Tree Decomposition



$\mathcal{J}_{\mathcal{G}}$

H-tree

2. Neural Tree arch. = Message passing on the H-tree

$$\mathbf{h}_u^t = \mathrm{AGG}_t \left(\mathbf{h}_u^{t-1}, \{ \left(\mathbf{h}_w^{t-1}, \kappa_{w,u}, \mathbf{h}_u^{t-1} \right) \mid w \in \mathcal{N}_{\mathcal{J}_{\mathcal{G}}}(u) \} \right) \quad \text{with} \quad \mathbf{h}_u^0 = \begin{cases} \mathbf{x}_{k(u)} & \text{if } u \text{ is a leaf node} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

Aggregation function

Neighbors of u in H-tree

Approximation Results

Theorem: For any (smooth) graph compatible function

$$f(\mathbf{X}) = \sum_C \theta_C(\mathbf{x}_C)$$

and an $\epsilon > 0$, there exists a Neural Tree model $g(\mathbf{X})$ with N weights/parameters such that

a. $||f - g||_\infty < \epsilon$

b. $N = \mathcal{O}\left(n \cdot (\mathrm{tw}(G)/\epsilon)^{c \cdot \mathrm{tw}(G)}\right)$

num. nodes in the graph

treewidth of the tree-decomposition used

Approximation Results

1-Lipschitz continuous

Theorem: For any (smooth) graph compatible function

$$f(\mathbf{X}) = \sum_C \theta_C(\mathbf{x}_C)$$

and an $\epsilon > 0$, there exists a Neural Tree model $g(\mathbf{X})$ with N weights/parameters such that

a. $||f - g||_\infty < \epsilon$

b. $N = \mathcal{O}\left(n \cdot (\mathrm{tw}(G)/\epsilon)^{c \cdot \mathrm{tw}(G)}\right)$

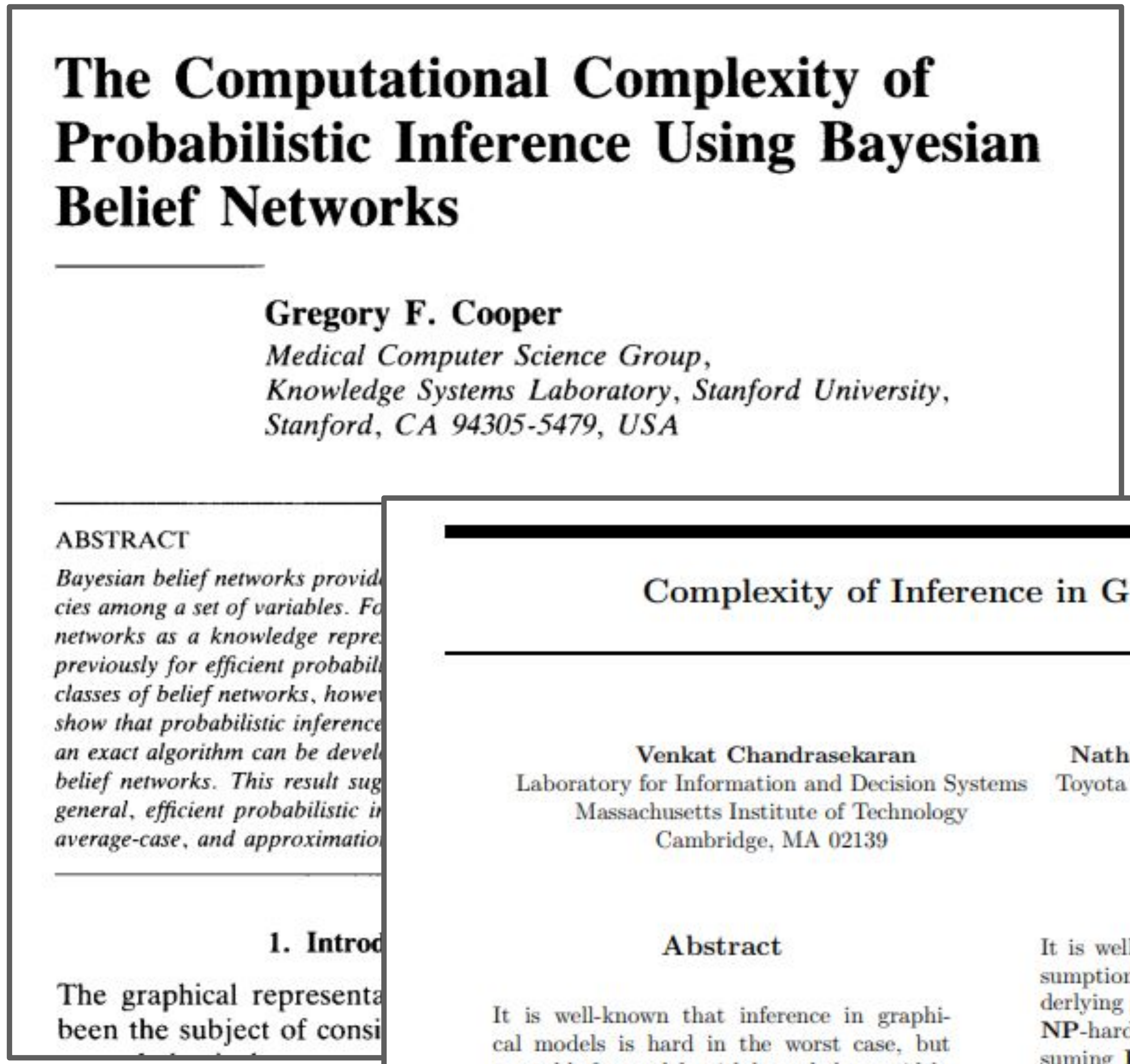
num. nodes in the graph

treewidth of the tree-decomposition used

Remarks

- **Parameter complexity of Neural Tree**
  - Linearly in graph size
  - Exponentially in graph treewidth

- **Complexity of exact inference on graphical models**
  - NP-hard
  - Exponential in graph treewidth



The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks

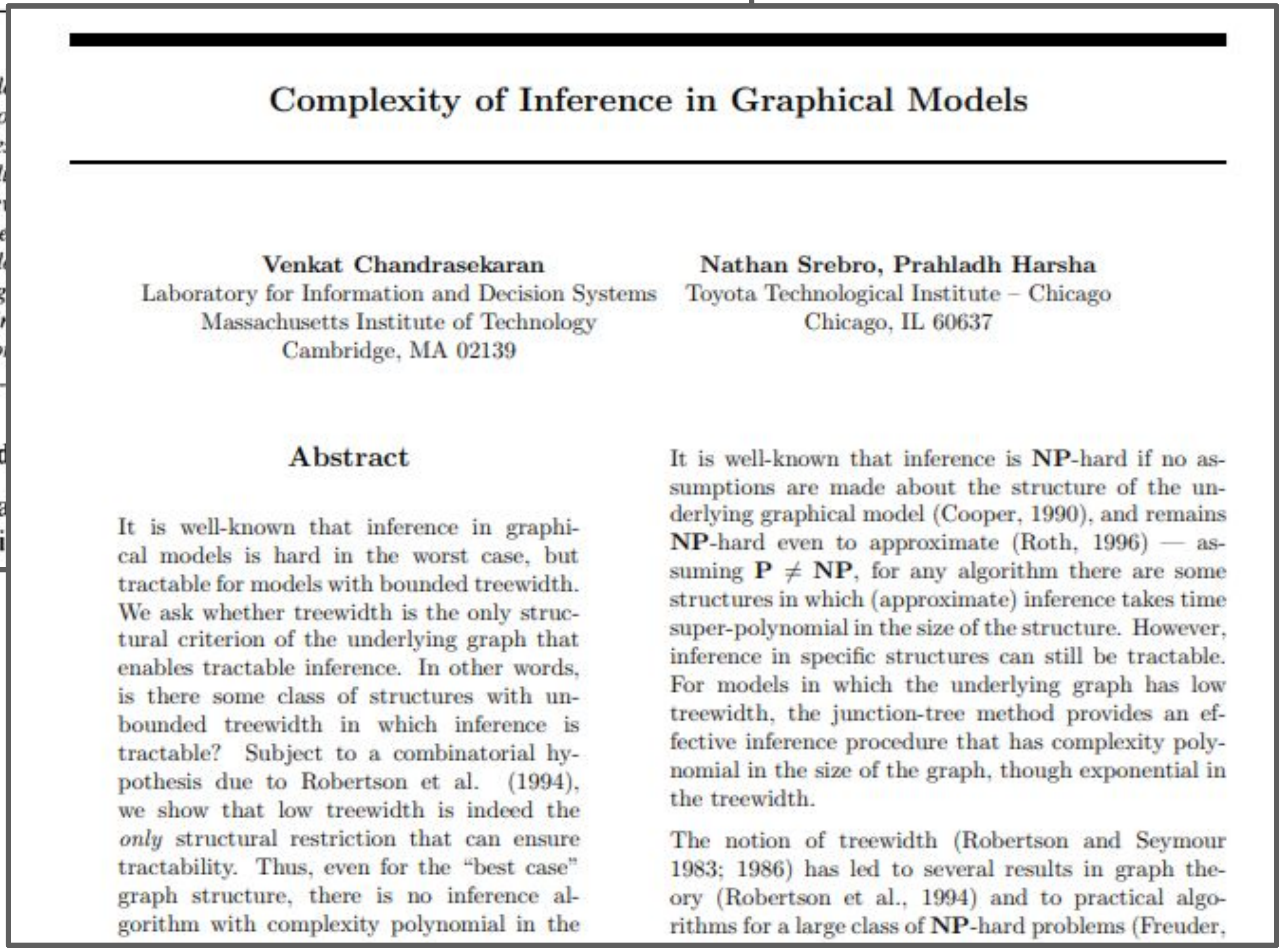
Gregory F. Cooper
Medical Computer Science Group,
Knowledge Systems Laboratory, Stanford University,
Stanford, CA 94305-5479, USA

ABSTRACT

Bayesian belief networks provid
cies among a set of variables. Fo
networks as a knowledge repre
previously for efficient probabili
classes of belief networks, howe
show that probabilistic inference
an exact algorithm can be devel
belief networks. This result sug
general, efficient probabilistic i
average-case, and approximatio

1. Introd

The graphical representa
been the subject of consi



Complexity of Inference in Graphical Models

Venkat Chandrasekaran
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139

Nathan Srebro, Prahladh Harsha
Toyota Technological Institute – Chicago
Chicago, IL 60637

Abstract

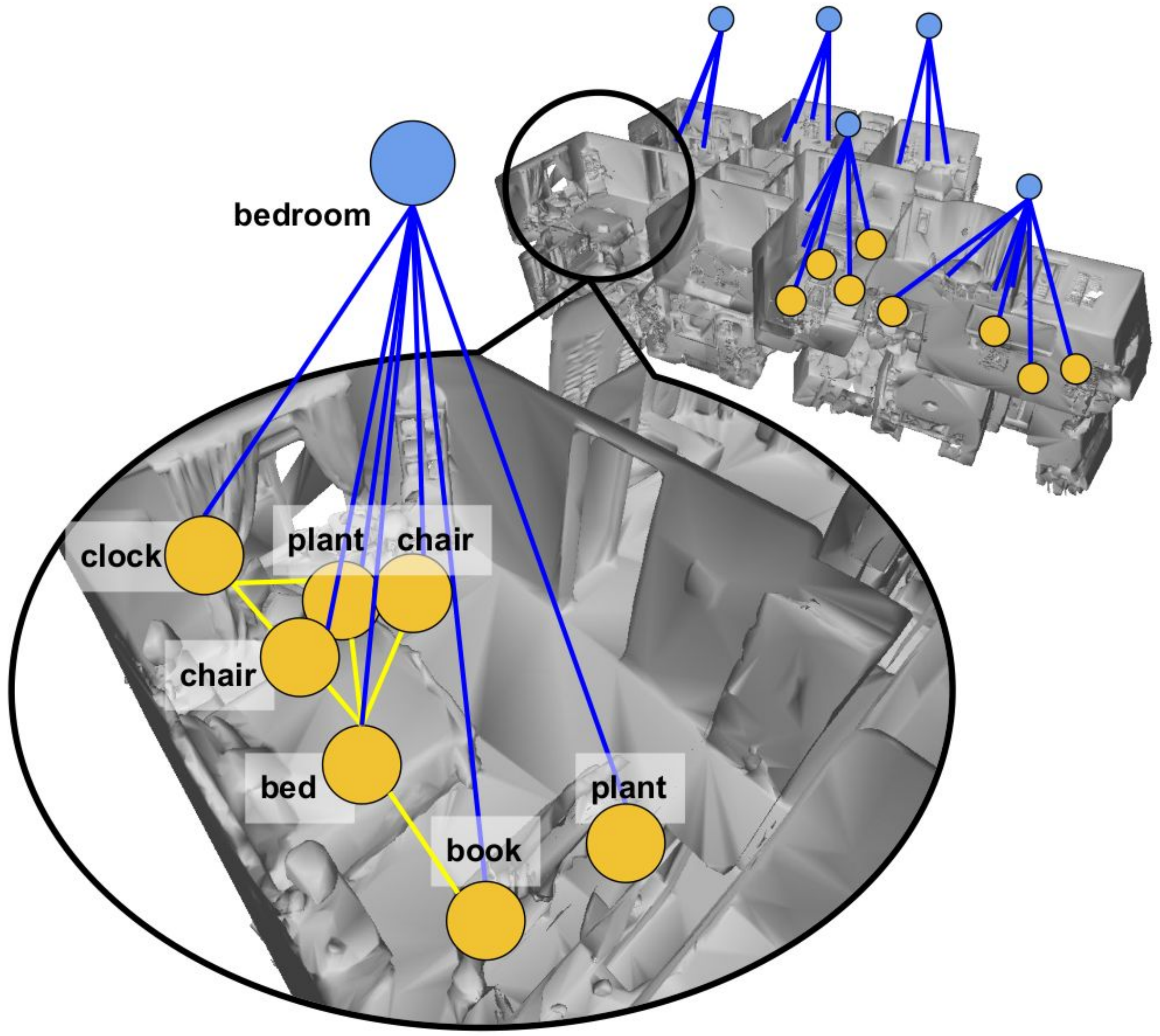
It is well-known that inference in graphical models is hard in the worst case, but tractable for models with bounded treewidth. We ask whether treewidth is the only structural criterion of the underlying graph that enables tractable inference. In other words, is there some class of structures with unbounded treewidth in which inference is tractable? Subject to a combinatorial hypothesis due to Robertson et al. (1994), we show that low treewidth is indeed the *only* structural restriction that can ensure tractability. Thus, even for the "best case" graph structure, there is no inference algorithm with complexity polynomial in the

It is well-known that inference is **NP**-hard if no assumptions are made about the structure of the underlying graphical model (Cooper, 1990), and remains **NP**-hard even to approximate (Roth, 1996) — assuming **P** $\neq$ **NP**, for any algorithm there are some structures in which (approximate) inference takes time super-polynomial in the size of the structure. However, inference in specific structures can still be tractable. For models in which the underlying graph has low treewidth, the junction-tree method provides an effective inference procedure that has complexity polynomial in the size of the graph, though exponential in the treewidth.

The notion of treewidth (Robertson and Seymour 1983; 1986) has led to several results in graph theory (Robertson et al., 1994) and to practical algorithms for a large class of **NP**-hard problems (Freuder,

Experiments

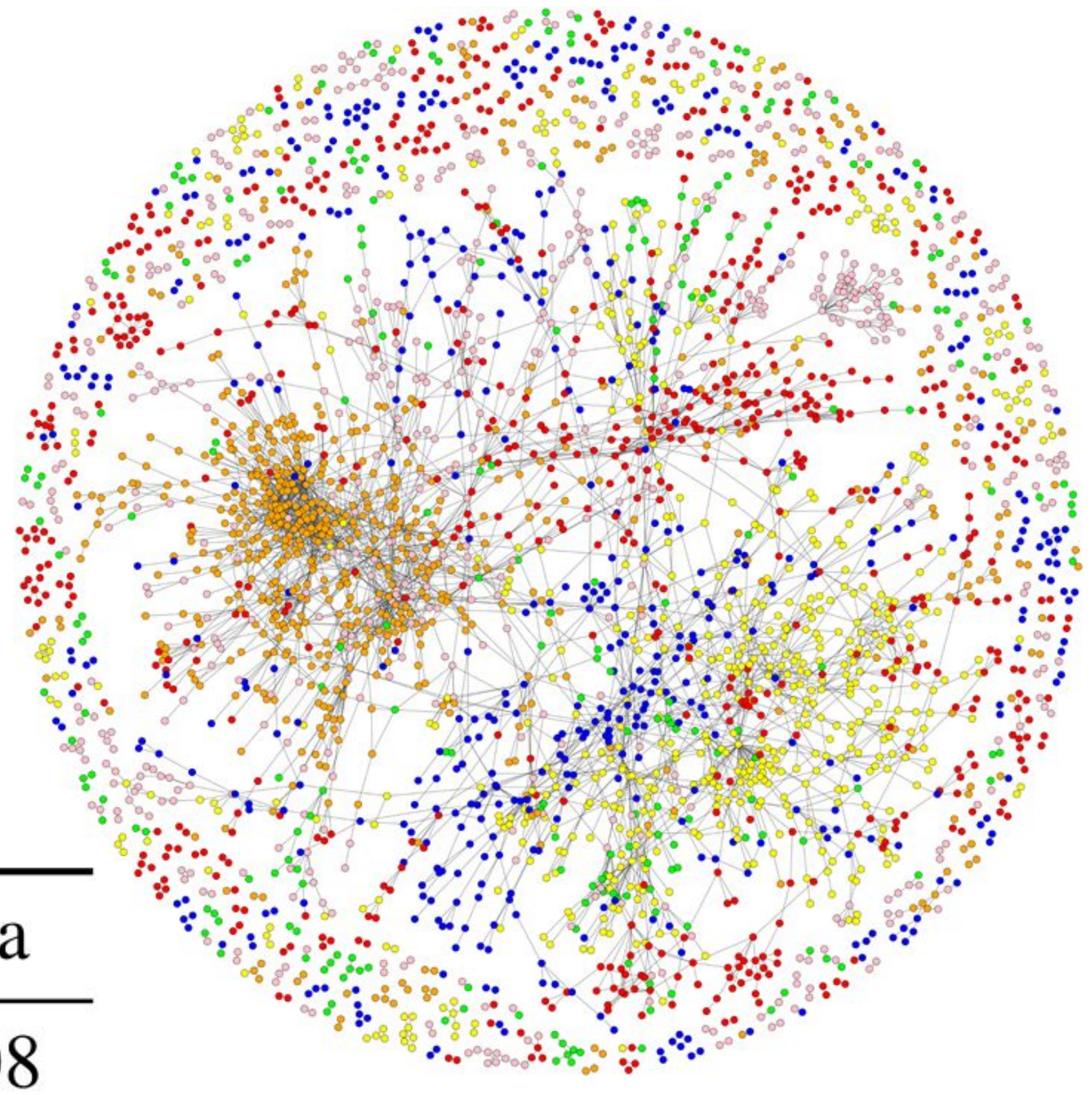
- 3D Scene Graphs
- Citation Networks



3D Scene Graphs
(smaller graphs with low treewidth)

Citation Networks
(large treewidth graphs)

	PubMed	CiteSeer	Cora
Nodes	19,717	3,327	2,708
Edges	44,338	4,732	5,429
Classes	3	6	7

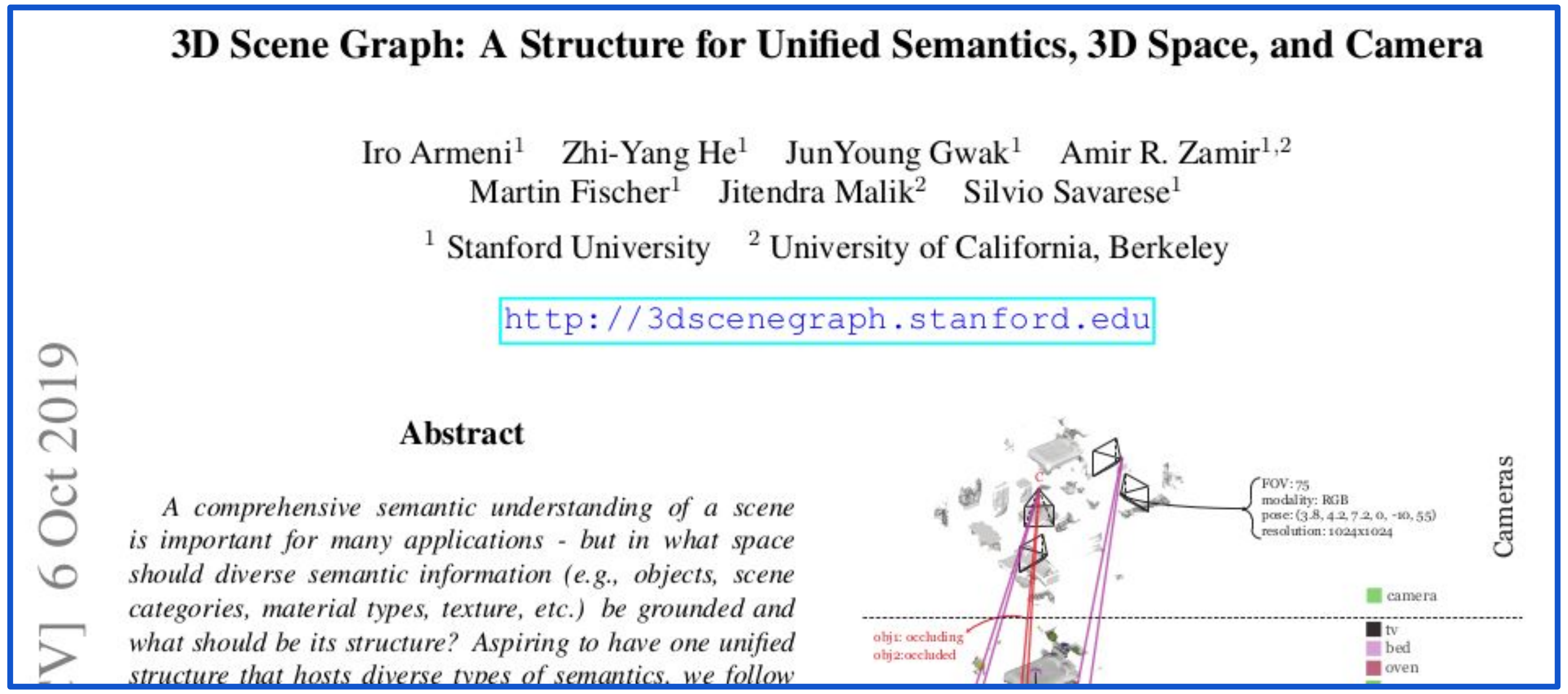


CiteSeer graph

3D Scene Graphs

Stanford 3D Scene Graph dataset

- 482 rooms with 15 categories
- 2338 objects with 35 categories



3D Scene Graph: A Structure for Unified Semantics, 3D Space, and Camera

Iro Armeni[1] Zhi-Yang He[1] JunYoung Gwak[1] Amir R. Zamir[1,2]
Martin Fischer[1] Jitendra Malik[2] Silvio Savarese[1]

[1] Stanford University [2] University of California, Berkeley

http://3dscenegraph.stanford.edu

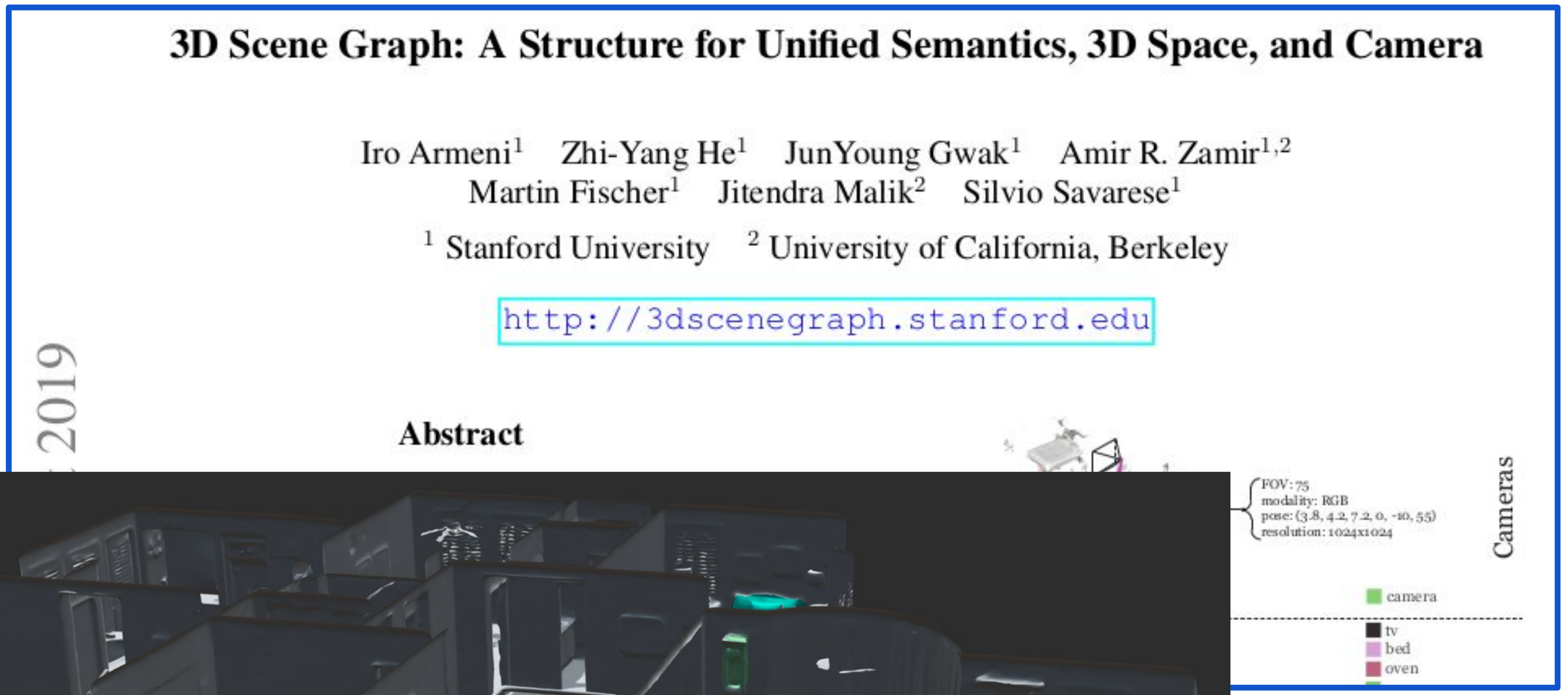
Abstract

A comprehensive semantic understanding of a scene is important for many applications - but in what space should diverse semantic information (e.g., objects, scene categories, material types, texture, etc.) be grounded and what should be its structure? Aspiring to have one unified structure that hosts diverse types of semantics, we follow

3D Scene Graphs

Stanford 3D Scene Graph dataset

- 482 rooms with 15 categories
- 2338 objects with 35 categories


3D Scene Graph: A Structure for Unified Semantics, 3D Space, and Camera

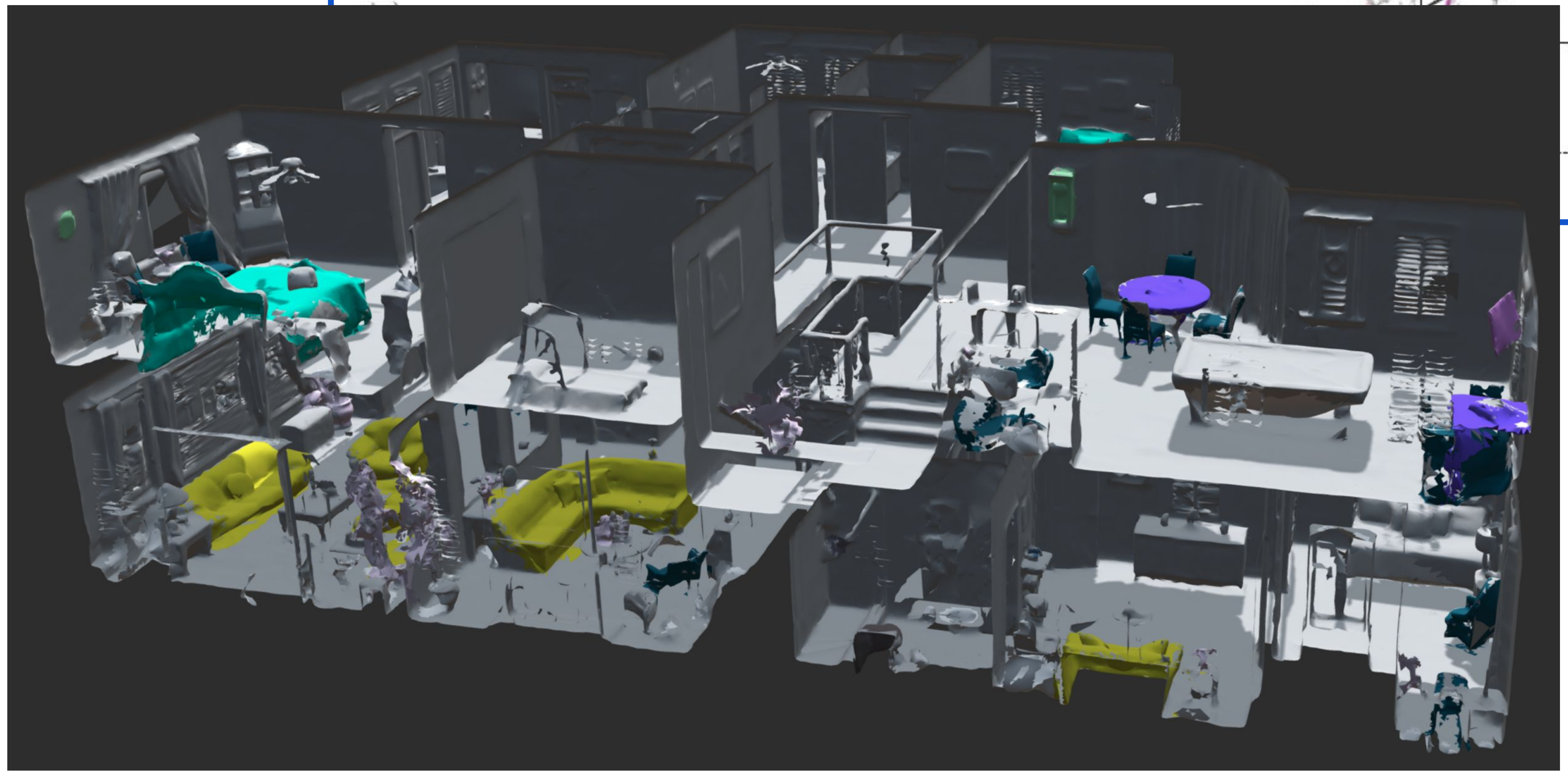
Iro Armeni[1] Zhi-Yang He[1] JunYoung Gwak[1] Amir R. Zamir[1,2]
Martin Fischer[1] Jitendra Malik[2] Silvio Savarese[1]

[1] Stanford University [2] University of California, Berkeley

http://3dscenegraph.stanford.edu

2019

Abstract



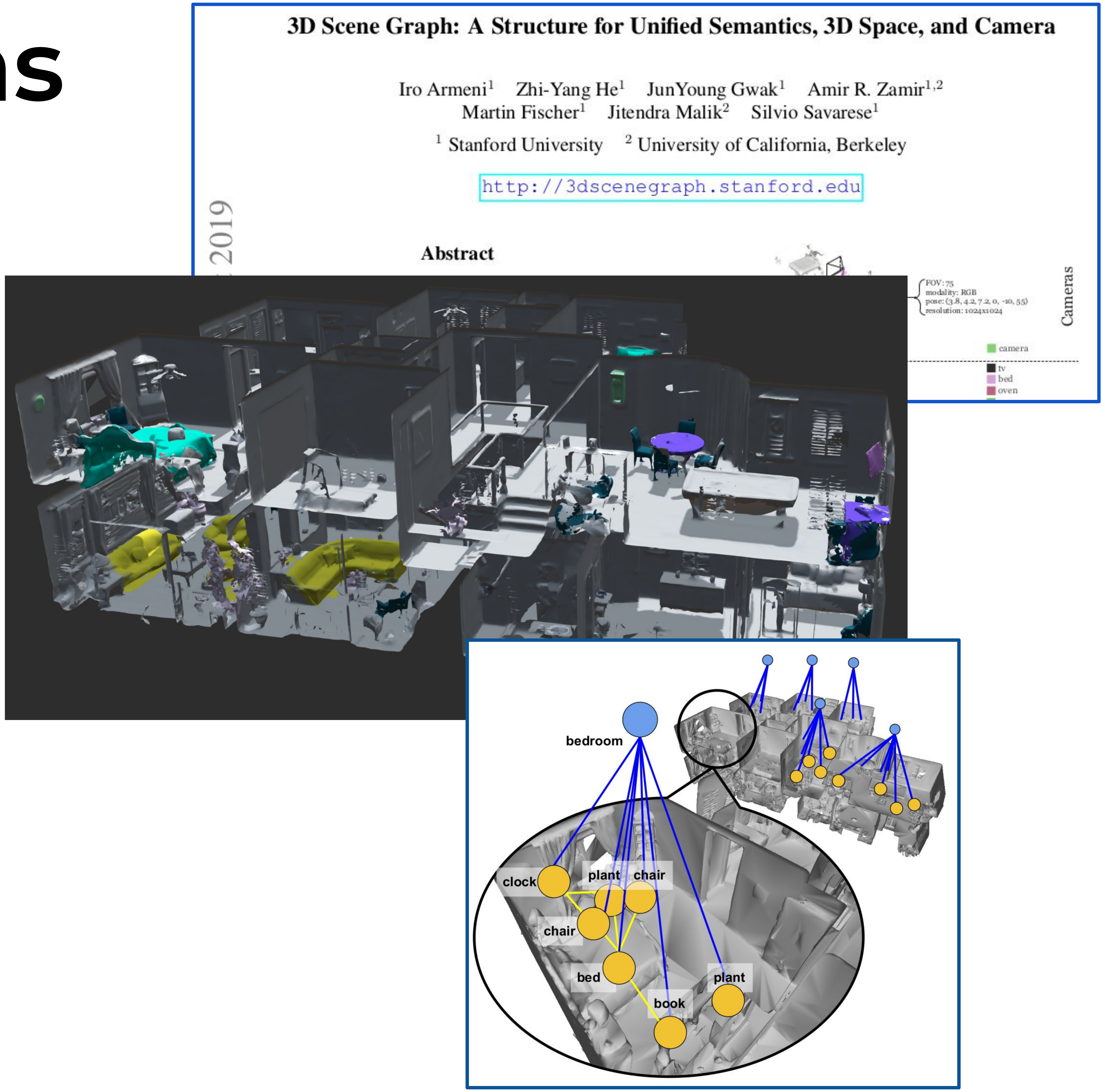
3D Scene Graphs

Stanford 3D Scene Graph dataset

- 482 rooms with 15 categories
- 2338 objects with 35 categories

We construct 3D Scene Graph by

- Connecting nearby objects

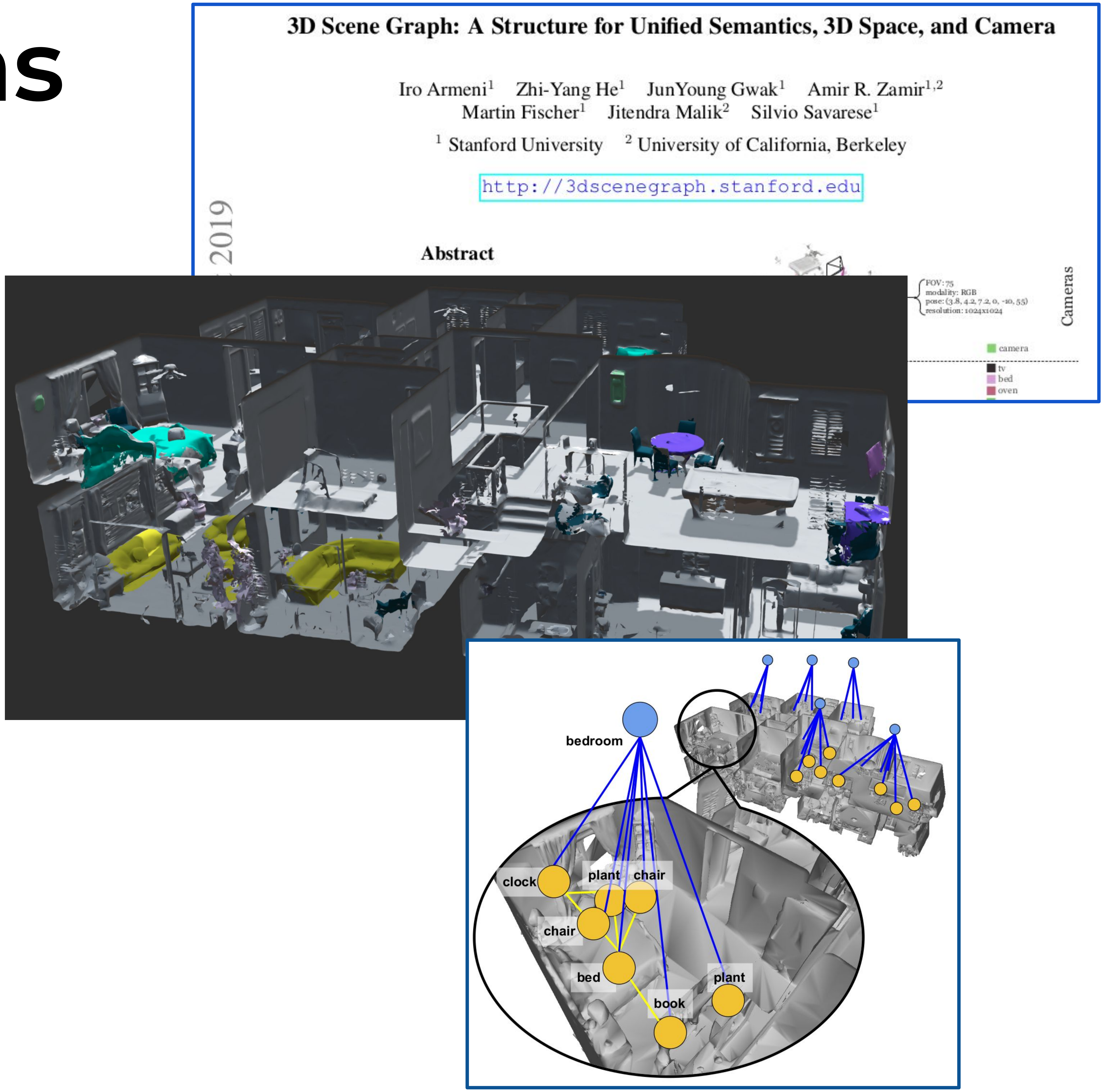


3D Scene Graphs

Stanford 3D Scene Graph dataset
- 482 rooms with 15 categories
- 2338 objects with 35 categories

We construct 3D Scene Graph by
- Connecting nearby objects

Input feature for each node
- centroid and bounding box dimension



Experiments and Results

Neural Tree vs traditional GNN

Compare Neural Tree message passing with GNN message passing, with same aggregation function

Test Accuracy*

AGG_t	Input graph	Neural Tree
GCN	40.88 ± 2.28 %	$\mathbf{50.63} \pm 2.25$ %
GraphSAGE	59.54 ± 1.35 %	$\mathbf{63.57} \pm 1.54$ %
GAT	46.56 ± 2.21 %	$\mathbf{62.16} \pm 2.03$ %
GIN	49.25 ± 1.15 %	$\mathbf{63.53} \pm 1.38$ %

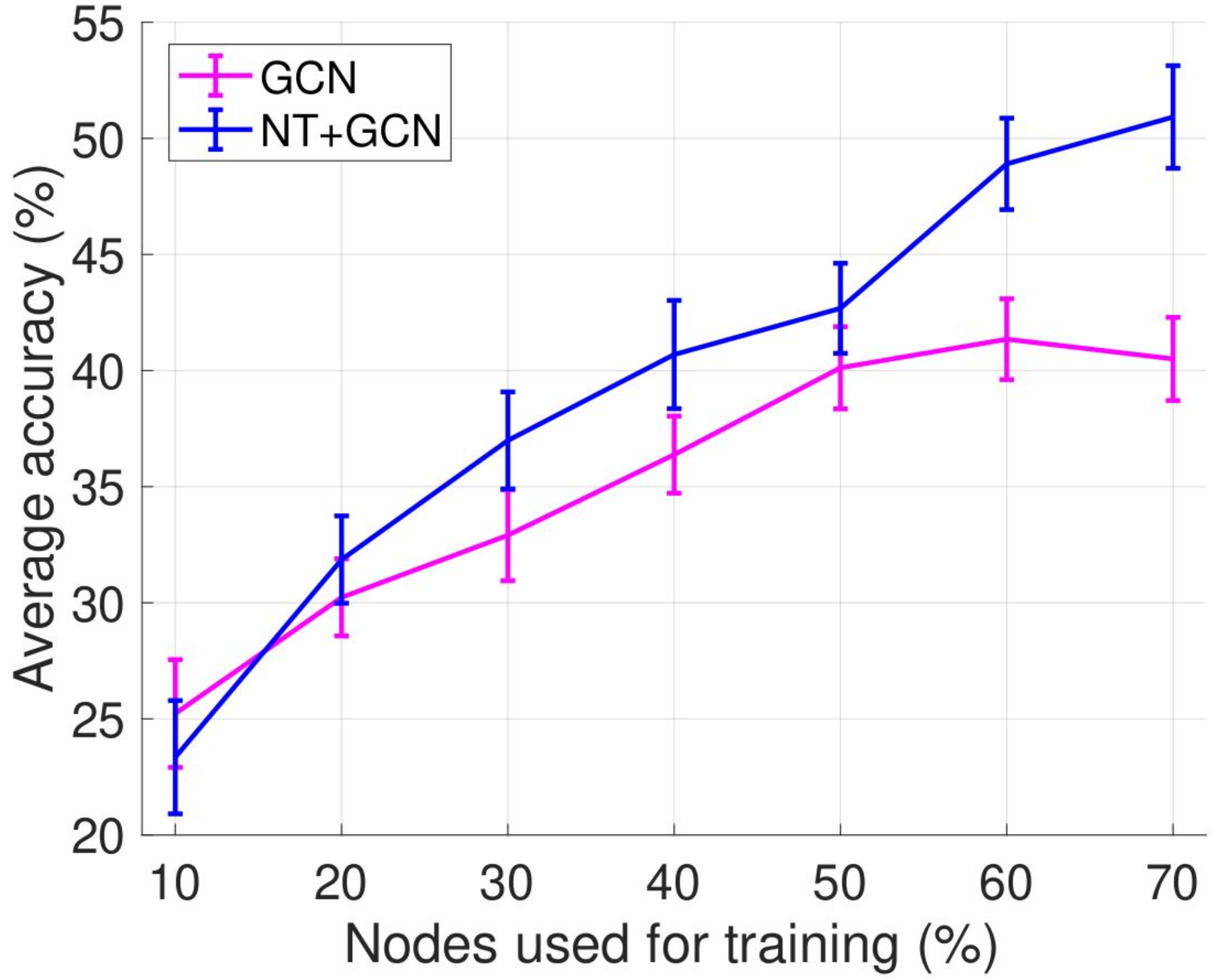
- Neural Tree always performs better than traditional GNN
- Always better to do message passing on H-tree

*Random train/val/test (70/10/20) split

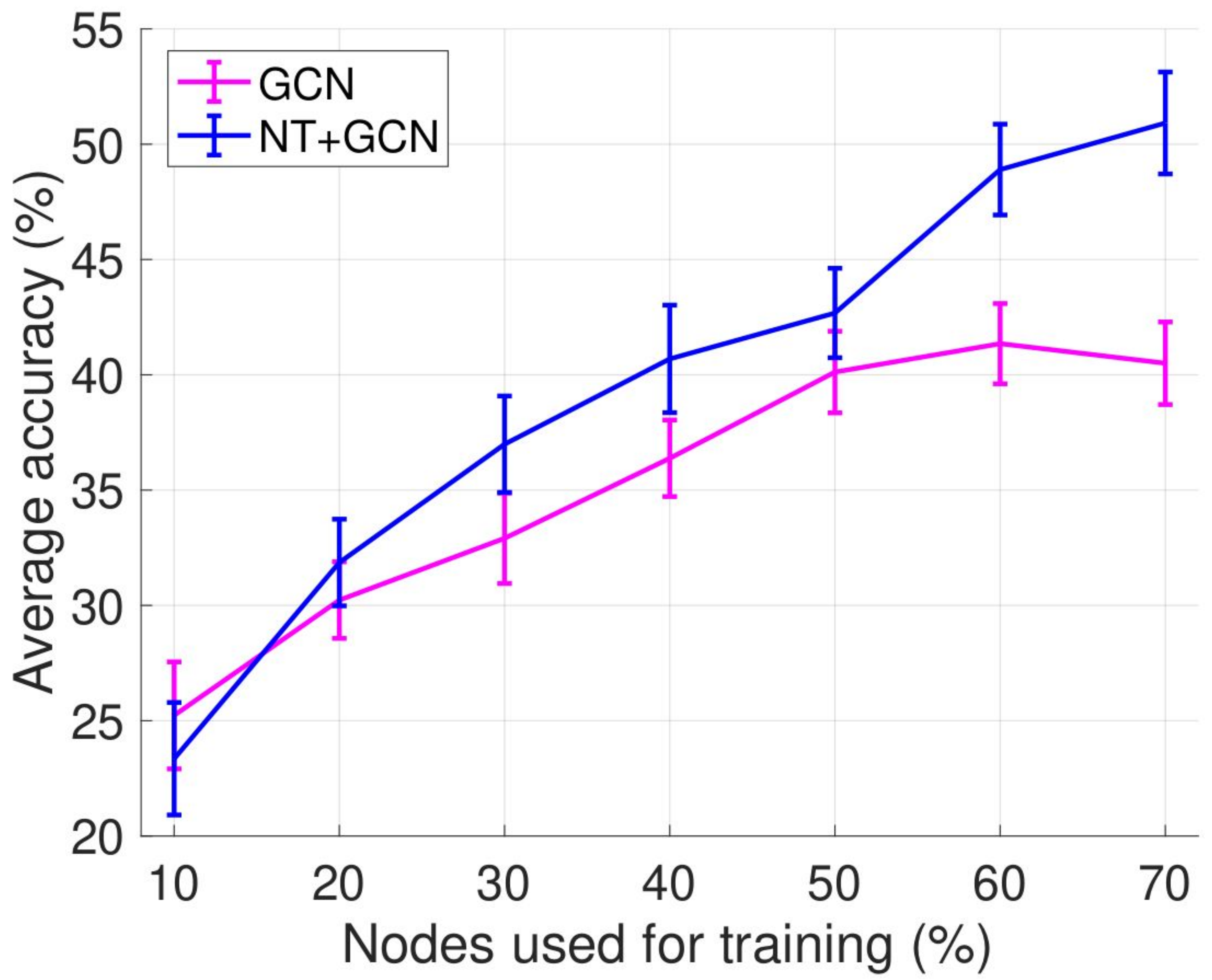
Experiments and Results

Increased training data

- Sharper increase with increasing training data for Neural Trees
- Performance of traditional GNN caps out

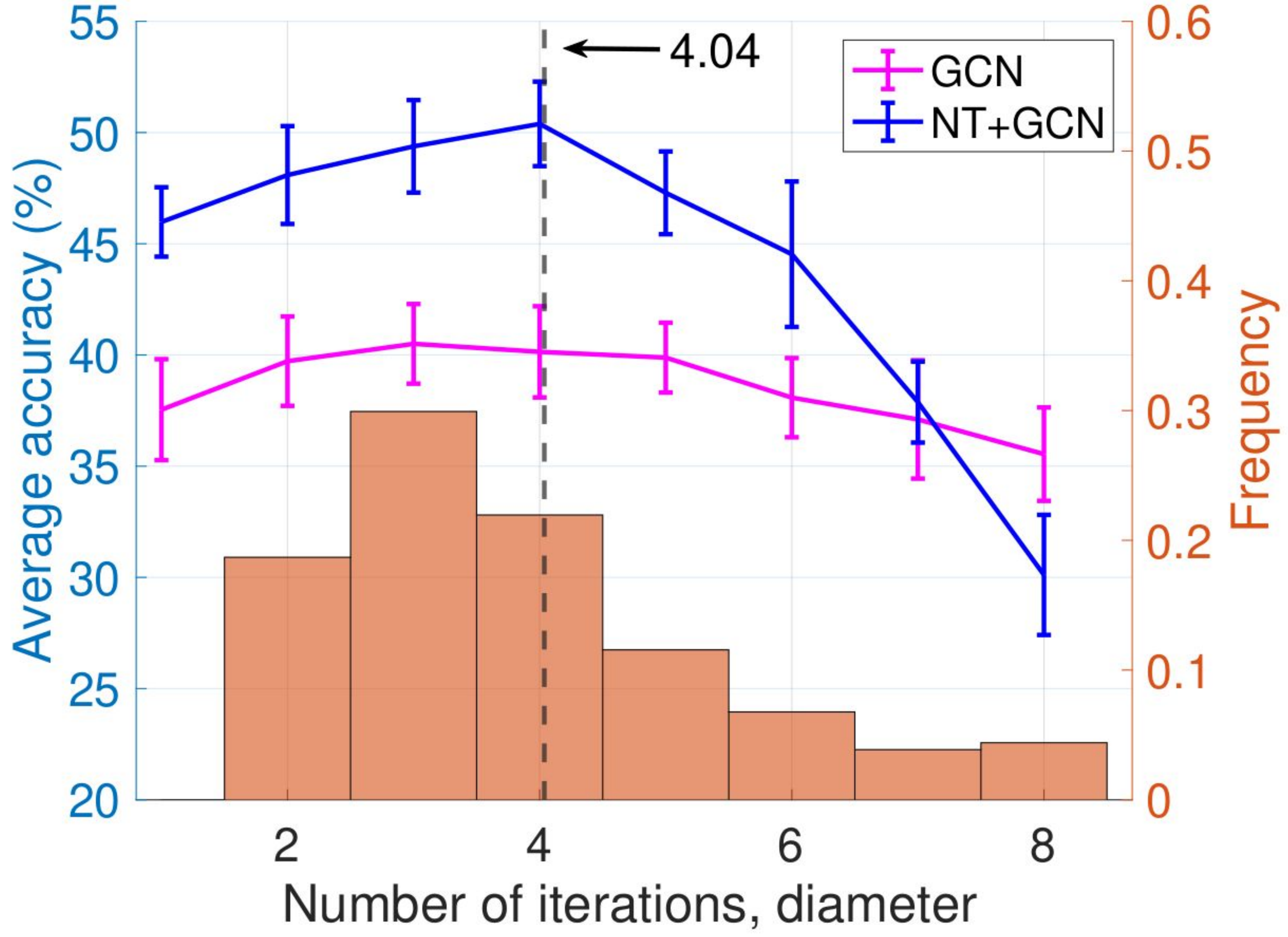


Experiments and Results



Increased training data

- Sharper increase with increasing training data for Neural Trees
- Performance of traditional GNN caps out



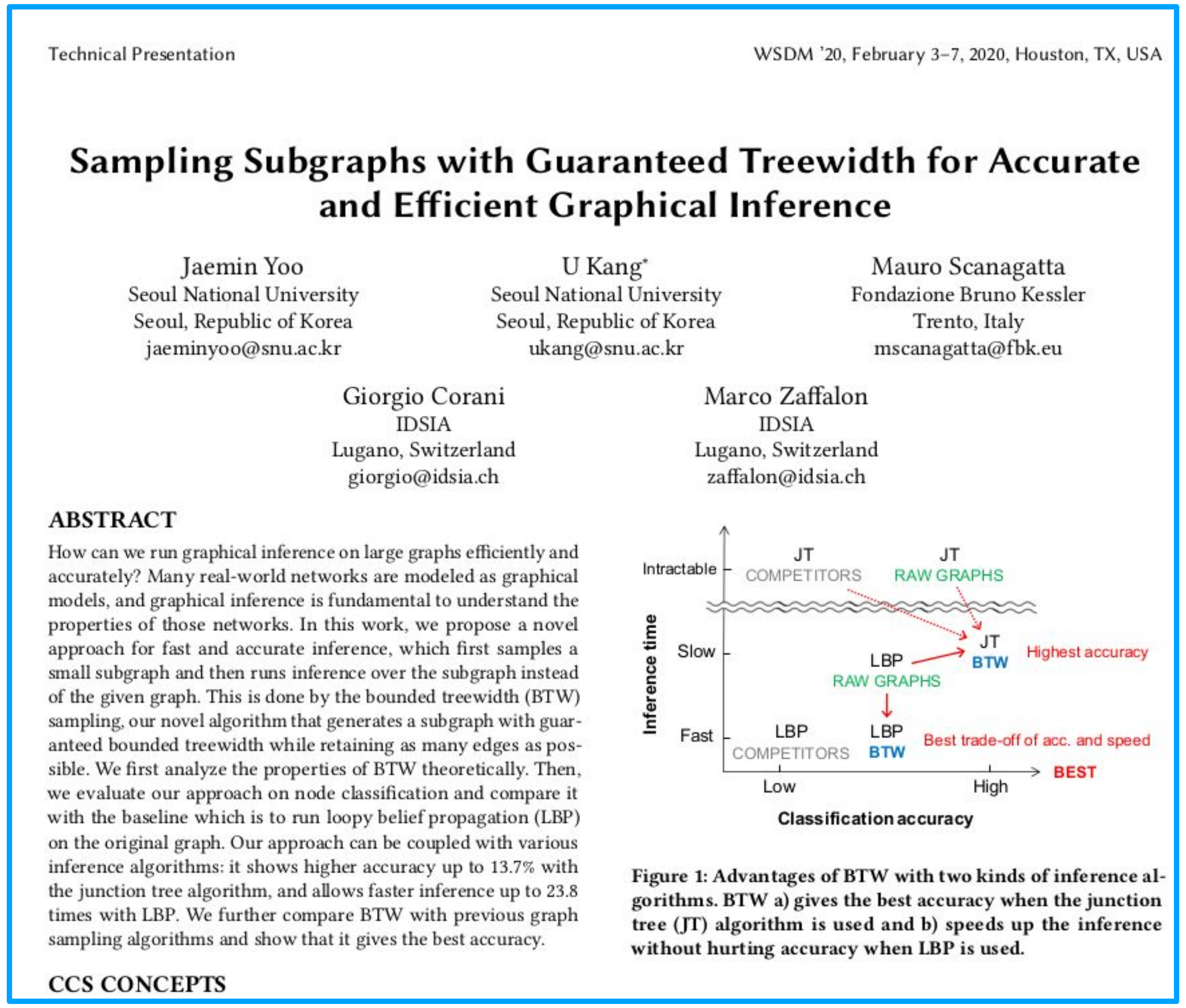
Number of message passing iterations

- Max attained at roughly the ave. diameter of the H-tree

Citation Networks

Does Neural Tree scale?

Bounded treewidth subgraph sampling + Neural Tree


Technical Presentation WSDM '20, February 3–7, 2020, Houston, TX, USA

Sampling Subgraphs with Guaranteed Treewidth for Accurate and Efficient Graphical Inference

Jaemin Yoo
Seoul National University
Seoul, Republic of Korea
jaeminyoo@snu.ac.kr

U Kang*
Seoul National University
Seoul, Republic of Korea
ukang@snu.ac.kr

Mauro Scanagatta
Fondazione Bruno Kessler
Trento, Italy
mscanagatta@fbk.eu

Giorgio Corani
IDSIA
Lugano, Switzerland
giorgio@idsia.ch

Marco Zaffalon
IDSIA
Lugano, Switzerland
zaffalon@idsia.ch

ABSTRACT

How can we run graphical inference on large graphs efficiently and accurately? Many real-world networks are modeled as graphical models, and graphical inference is fundamental to understand the properties of those networks. In this work, we propose a novel approach for fast and accurate inference, which first samples a small subgraph and then runs inference over the subgraph instead of the given graph. This is done by the bounded treewidth (BTW) sampling, our novel algorithm that generates a subgraph with guaranteed bounded treewidth while retaining as many edges as possible. We first analyze the properties of BTW theoretically. Then, we evaluate our approach on node classification and compare it with the baseline which is to run loopy belief propagation (LBP) on the original graph. Our approach can be coupled with various inference algorithms: it shows higher accuracy up to 13.7% with the junction tree algorithm, and allows faster inference up to 23.8 times with LBP. We further compare BTW with previous graph sampling algorithms and show that it gives the best accuracy.

Figure 1: Advantages of BTW with two kinds of inference algorithms. BTW a) gives the best accuracy when the junction tree (JT) algorithm is used and b) speeds up the inference without hurting accuracy when LBP is used.

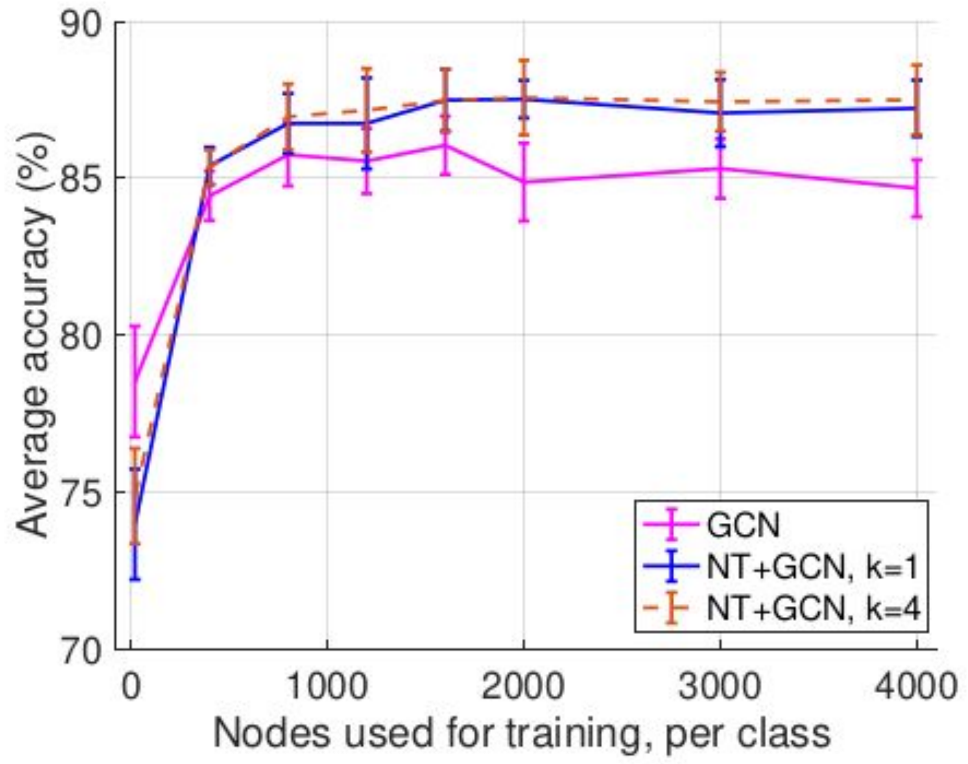
CCS CONCEPTS

Citation Networks

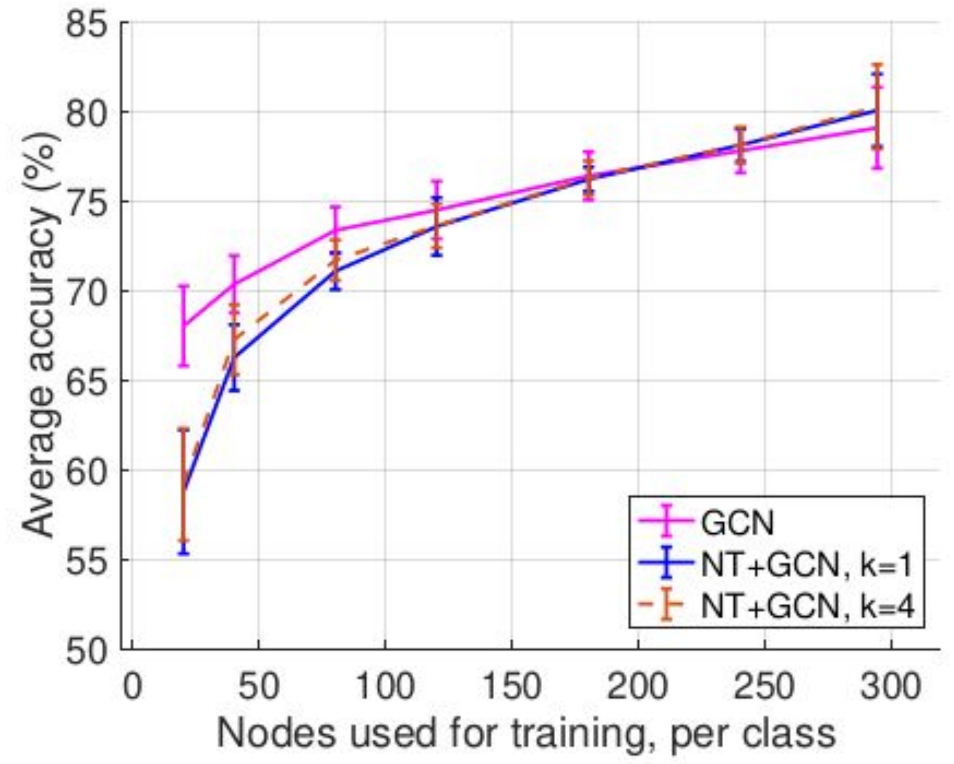
Tree width bounds k = 1 and 4

Does Neural Tree scale?

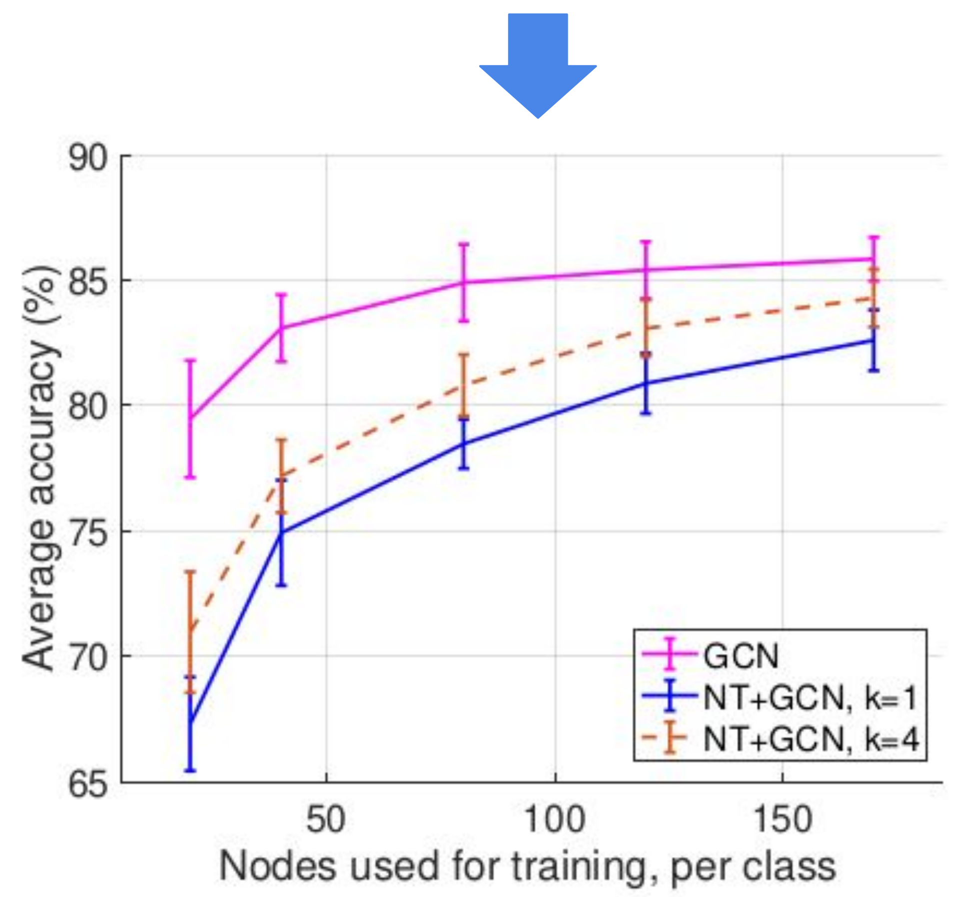
Bounded treewidth subgraph sampling + Neural Tree



(a) NT+GCN on PubMed

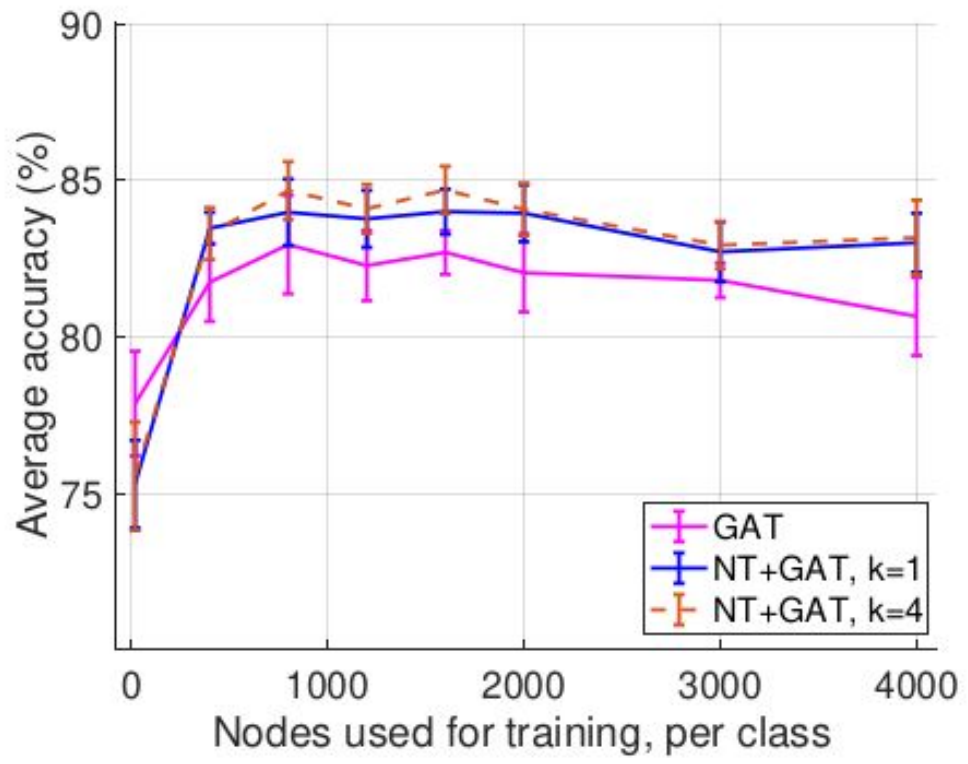


(b) NT+GCN on CiteSeer

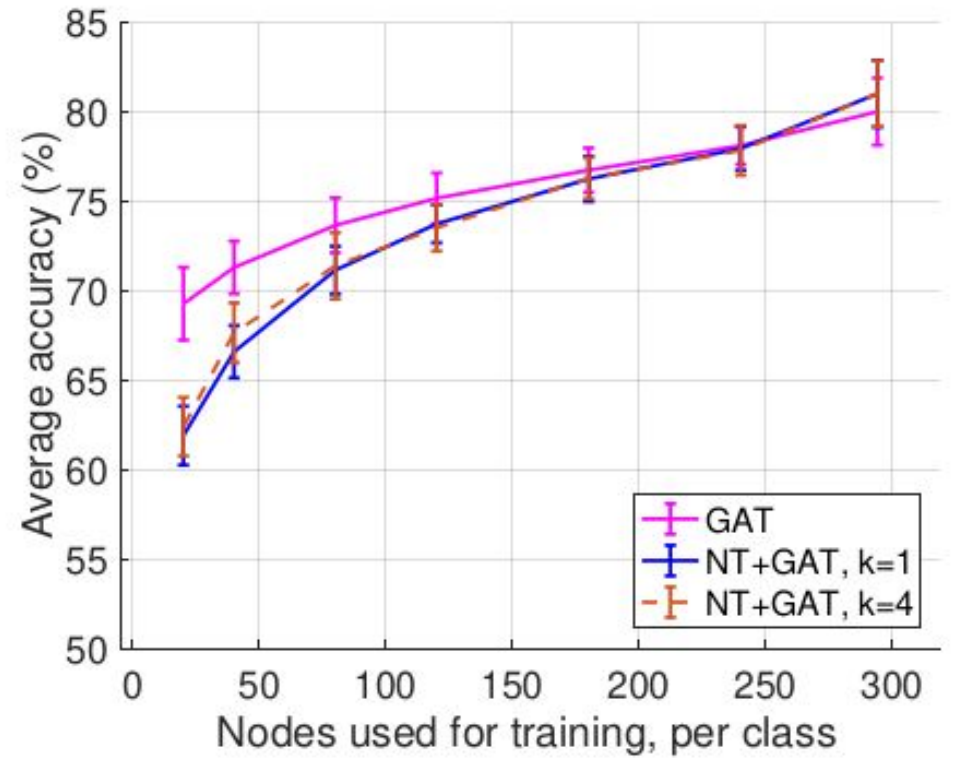


(c) NT+GCN on Cora

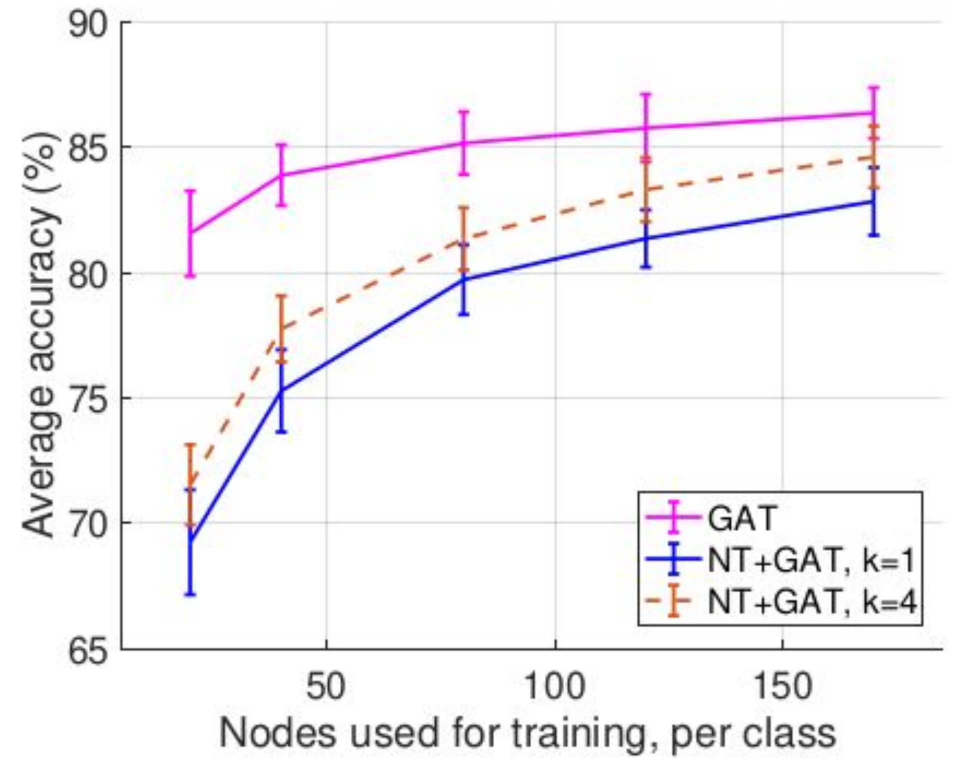
- Neural Tree attains the same performance as GNN, even for small treewidth bound
- Neural Tree is data hungry. Does not perform well with less training data.



(d) NT+GAT on PubMed



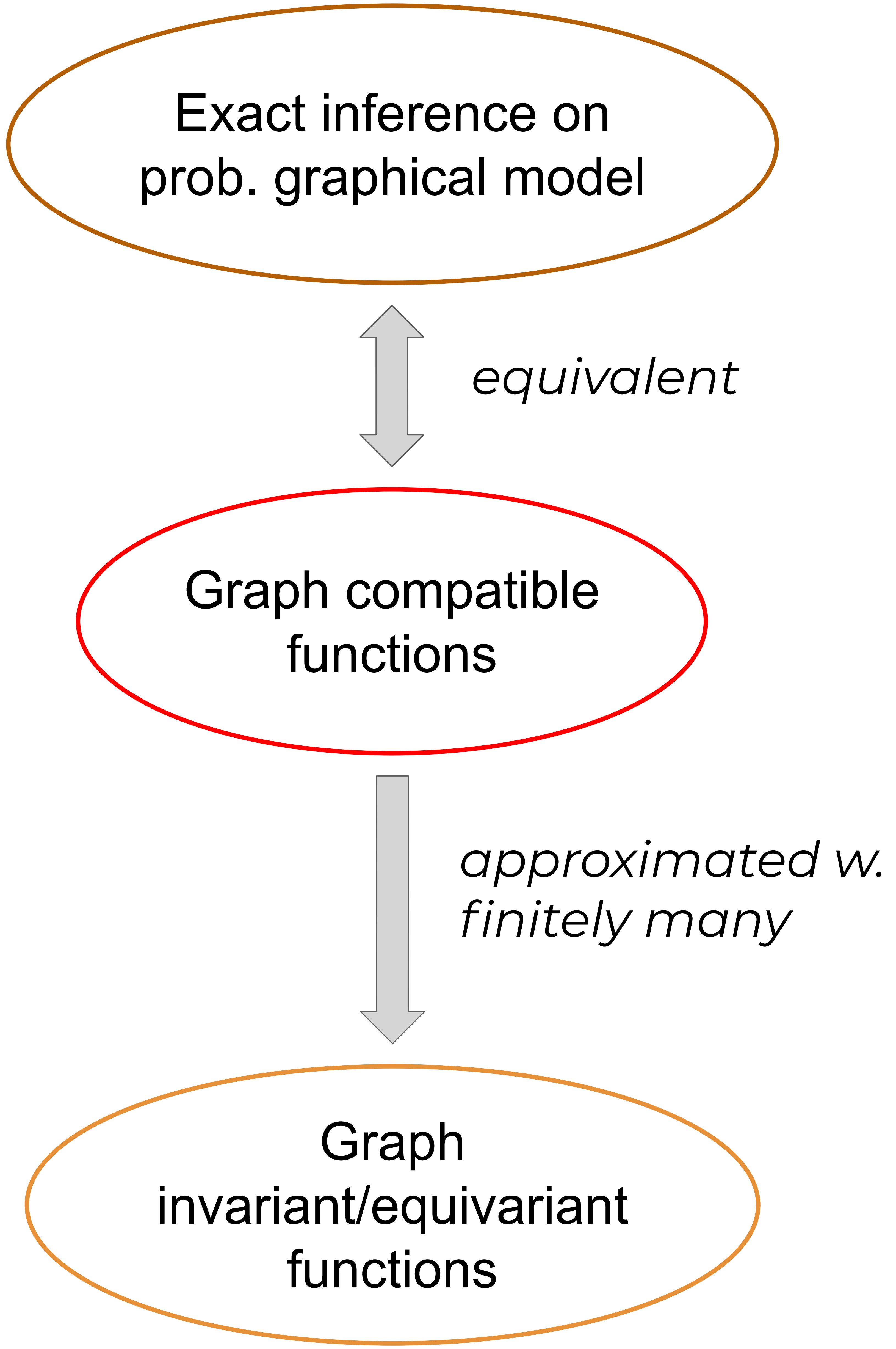
(e) NT+GAT on CiteSeer



(f) NT+GAT on Cora

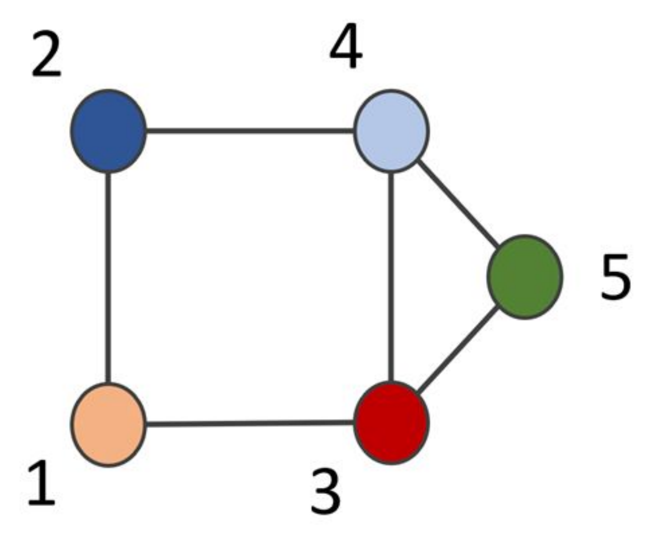
Conclusion

- Graph compatible functions
- Neural Tree architecture
- Approximation Results
- Scalable Neural Tree



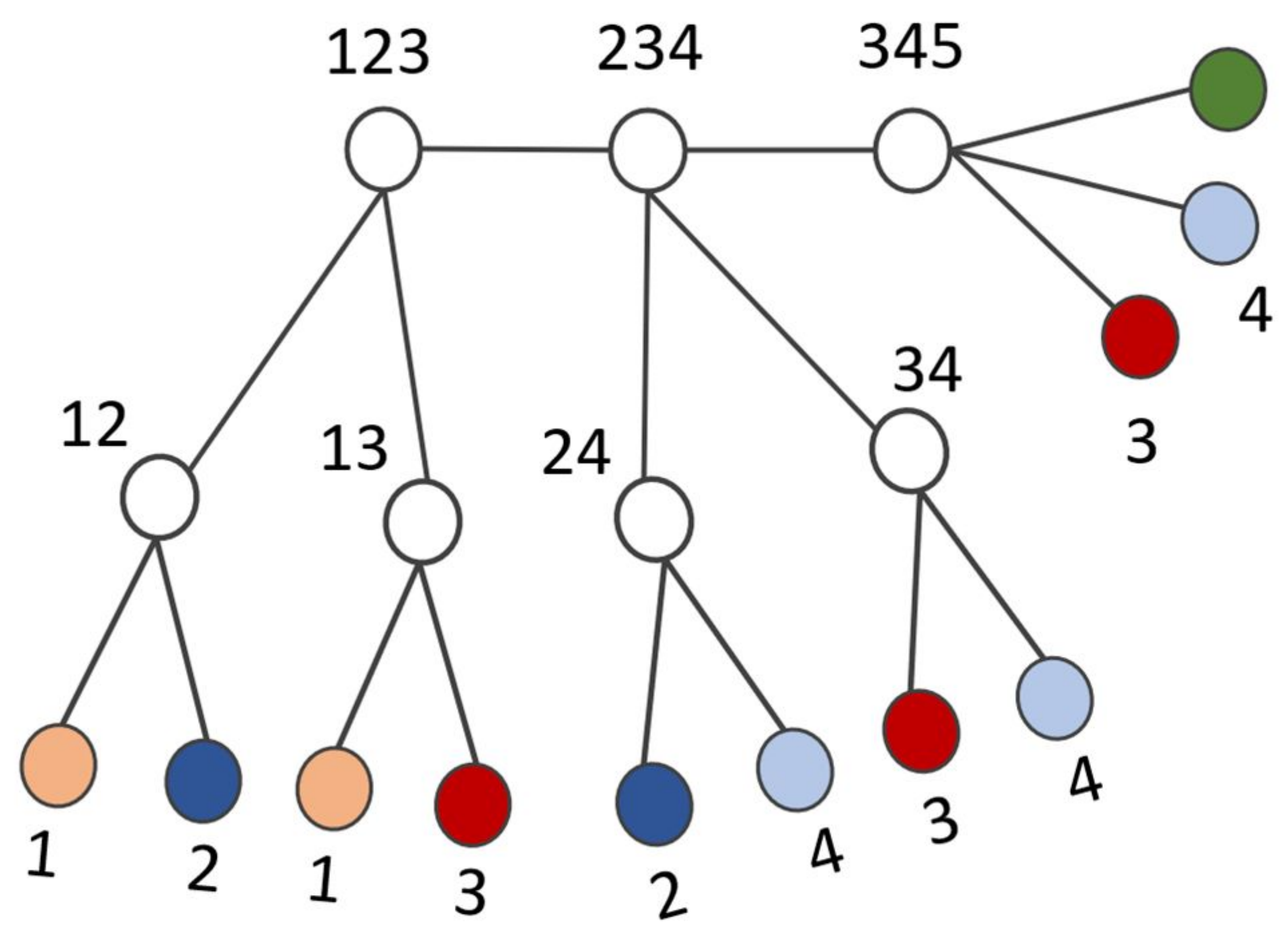
Conclusion

- Graph compatible functions
- Neural Tree architecture
- Approximation Results
- Scalable Neural Tree



Input graph with node attributes (colors)

Generate a tree structured graph called *H-tree*



Generated tree structured graph

Neural Tree is *message passing on H-tree*

Conclusion

- Graph compatible functions
- Neural Tree architecture
- Approximation Results
- Scalable Neural Tree

Invariant/equivariant function approximation

Graph Compatible functions

Graph treewidth

Parameter complexity

$$N = \mathcal{O}\left(n \cdot (\mathrm{tw}(G)/\epsilon)^{c \cdot \mathrm{tw}(G)}\right)$$

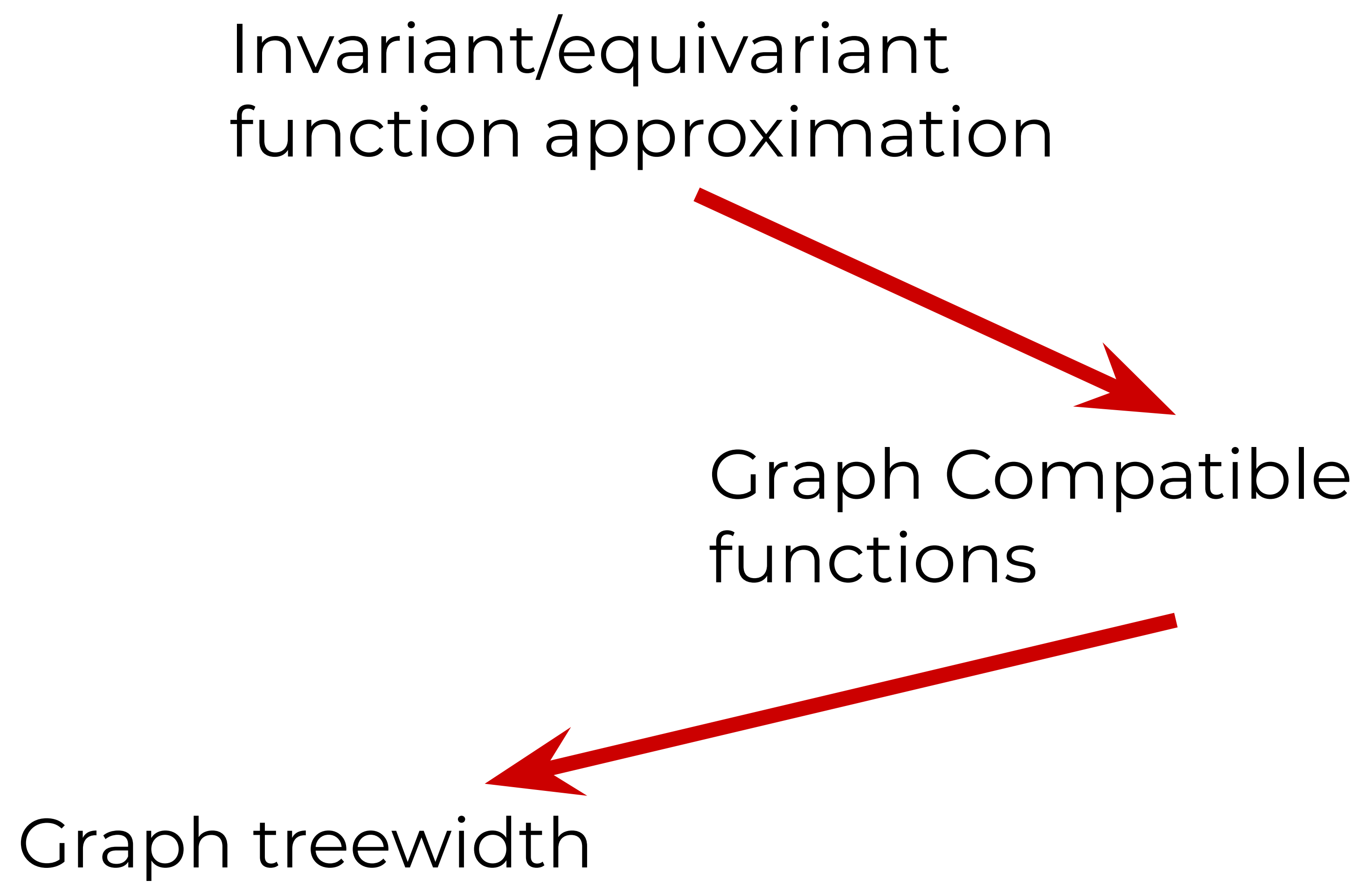
num. nodes

treewidth

approx. distance

Conclusion

- Graph compatible functions
- Neural Tree architecture
- Approximation Results
- Scalable Neural Tree



Result may be repurposed to other GNNs that extract hierarchical features

Conclusion

- Graph compatible functions
- Neural Tree architecture
- Approximation Results
- Scalable Neural Tree

Bounded treewidth subgraph sampling + Neural Tree

Conclusion

- Graph compatible functions
- Neural Tree architecture
- Approximation Results
- Scalable Neural Tree

Neural Tree is a general architecture

Remains to be applied to graph classification, link prediction, etc.

Questions, Comments, Related Work

Rajat Talak	talak@mit.edu
Siyi Hu	siyi@mit.edu
Lisa Peng	lisapeng@mit.edu
Luca Carlone	lcarlone@mit.edu