USC

Graph Representation Learning for Optimization on Graphs

Bistra Dilkina

Assistant Professor of Computer Science, USC

Associate Director, USC Center of AI in Society

AI for
Sustainability and Social Good

Biodiversity Conservation

Disaster resilience

Public Health & Well-being

Design of policies to manage limited resources for best impact translate into
large-scale decision / optimization and learning problems, combining discrete and continuous effects

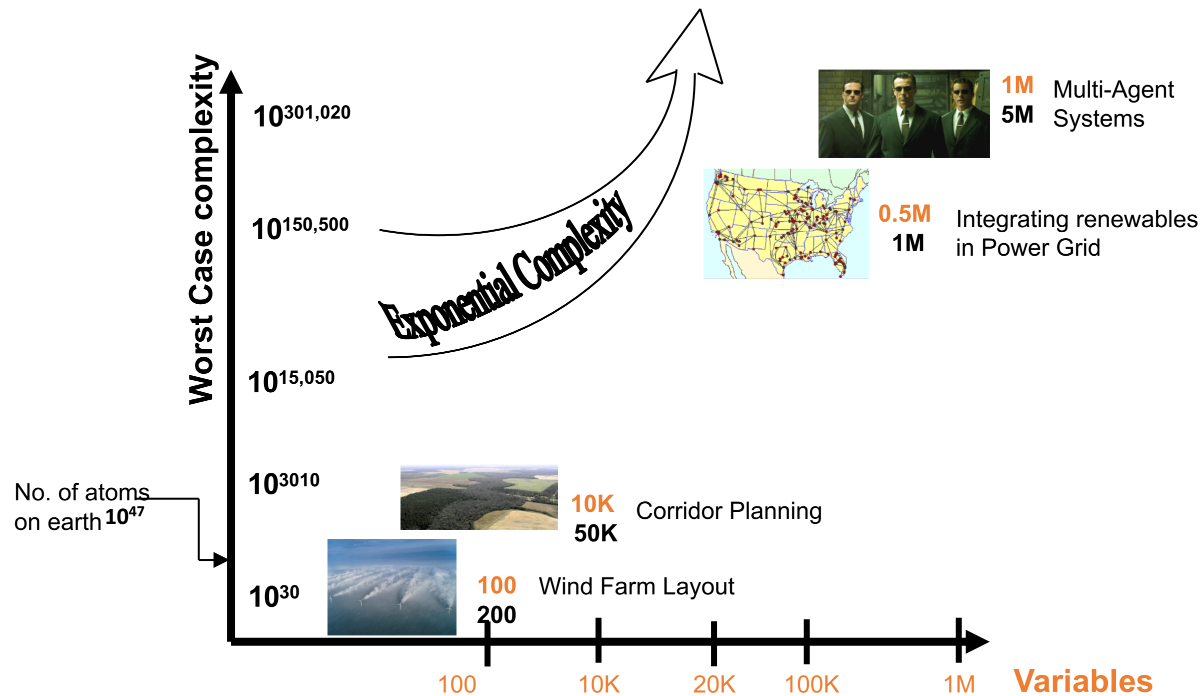
USC

ML ⟺ Combinatorial Optimization

- Exciting and growing research area
- Design discrete optimization algorithms with learning components
- Learning methods that incorporate the combinatorial decision making they inform

Constraint Reasoning and Optimization

Decision making problems of larger size and new problem structure drive the continued need to improve combinatorial solving methods



Constraint Reasoning and Optimization

Tackling NP-Hard problems	Design rationale
Exact algorithms	Tight formulations, good IP solvers
Approximation algorithms	Worst-case guarantees
Heuristics	Intuition, Empirical performance

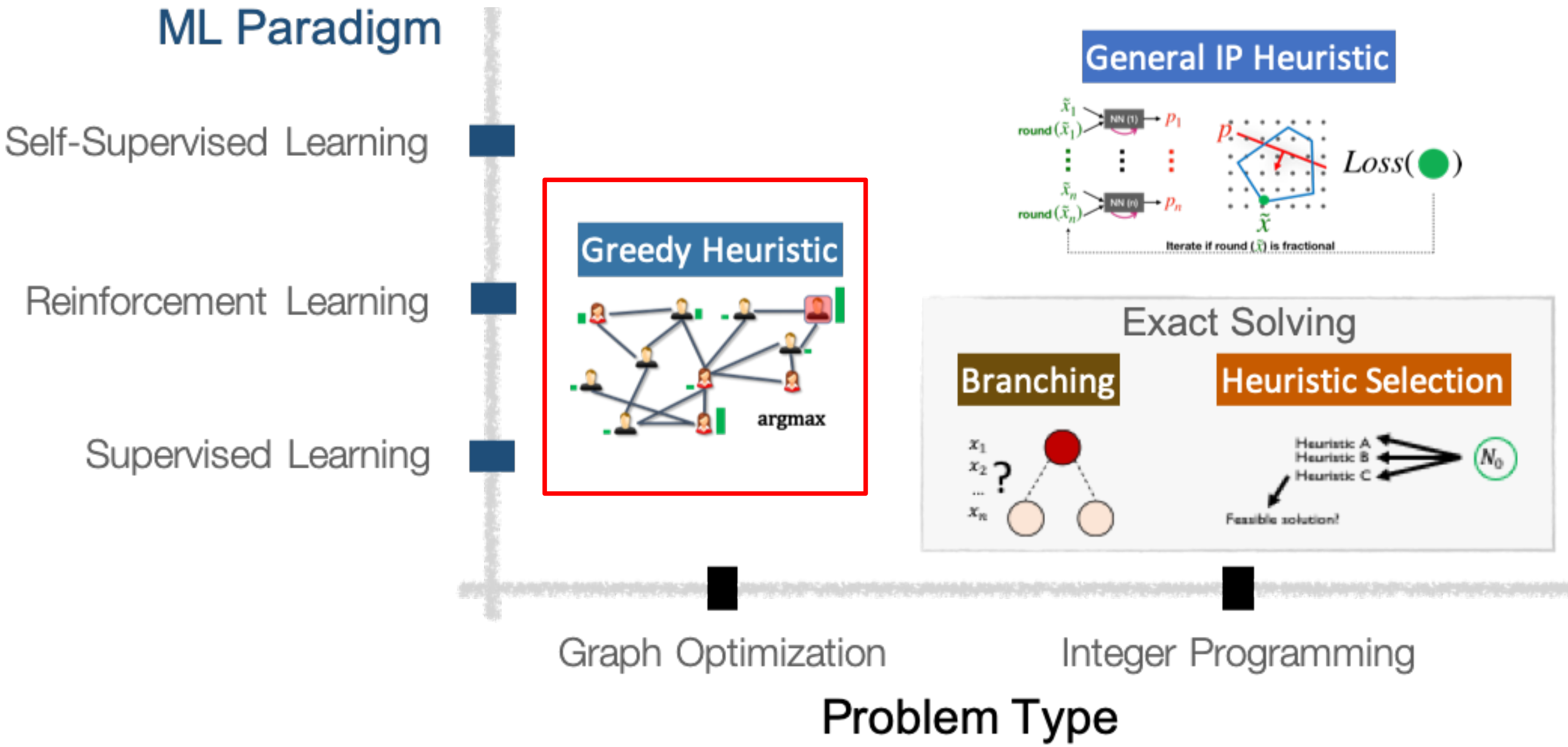
A realistic setting

- Same problem is solved repeatedly with slightly different data
- Delivery Company in Los Angeles:
  - Daily routing in the same area with slightly different customers

Opportunity:

Automatically tailor algorithms to a **family** of **instances** to **discover** novel **search strategies**

ML-Driven Discrete Algorithms



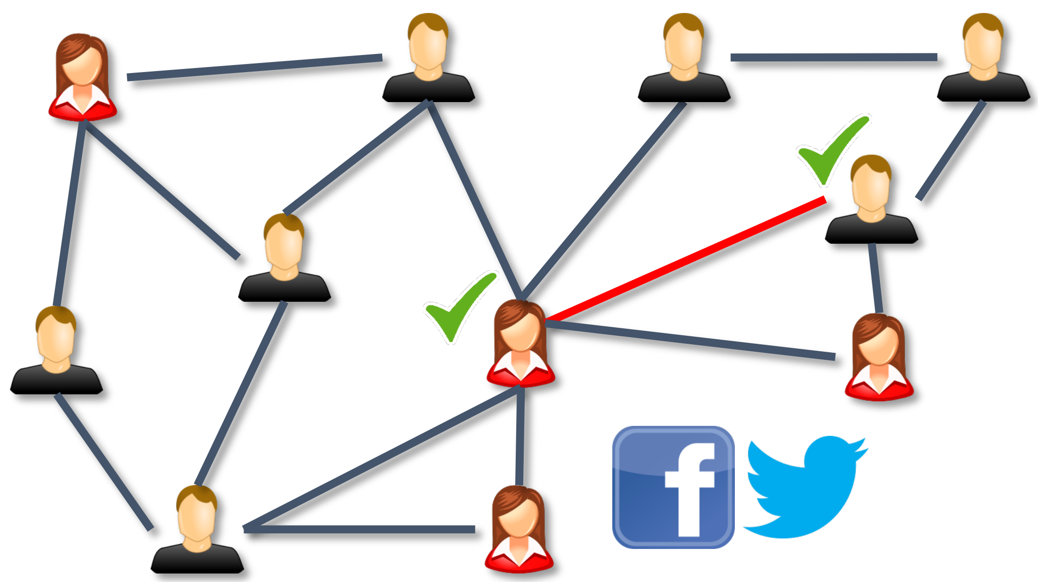
Elias B. Khalil*, Hanjun Dai*, Yuyu Zhang, Bistra Dilkina, Le Song. **Learning Combinatorial Optimization Algorithms over Graphs.** NeurIPS, 2017.

Algorithmic Template: Greedy

- **Minimum Vertex Cover:** Find smallest vertex subset S s.t. each edge has at least one end in S
  - Example: advertising optimization in social networks
  - 2-approx:

    greedily add vertices of edge

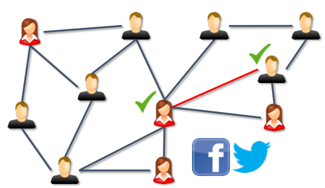
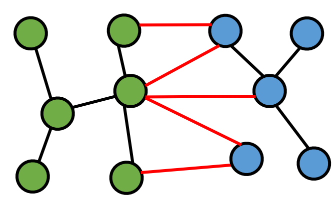
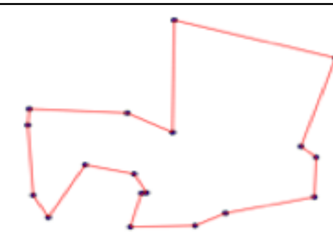
    with **max degree sum**



Learning Greedy Heuristics

Given: graph problem, family of graphs
Learn: a **scoring function** to guide a **greedy** algorithm

Problem	Minimum Vertex Cover	Maximum Cut	Traveling Salesman Problem
Domain	Social network snapshots	Spin glass models	Package delivery
Greedy operation	Insert nodes into cover	Insert nodes into subset	Insert nodes into sub-tour

Joint work with Elias Khalil, Hanjun Dai, Yuyu Zhang and Le Song [NIPS 2017]

Challenge #1: How to Learn

Possible approach: **Supervised learning**

- **Data**: collect (partial solution, next vertex) pairs

  features **label**

  from precomputed (near) optimal solutions

PROBLEM

Supervised learning → Need to compute good/optimal solutions to NP-Hard problems in order to learn!!

Reinforcement Learning Formulation

USC

Minimum Vertex Cover

$$\min_{x_i \in \{0,1\}} \sum_{i \in \mathcal{V}} x_i$$

$$s.t.\ x_i + x_j \geq 1, \forall (i,j) \in \mathcal{E}$$

Start with **COVER** = empty

Repeat until all edges covered:

1. Compute score for each vertex
2. Select vertex with **largest score**
3. Add best vertex to **COVER**

Reward: $r^t = -1$

State $\boldsymbol{S}$: current partial solution

Action value function: $\widehat{\boldsymbol{Q}}(\boldsymbol{S}, \boldsymbol{v})$

Greedy policy: $v^* = argmax_v\ \hat{Q}(S, v)$

Update state S

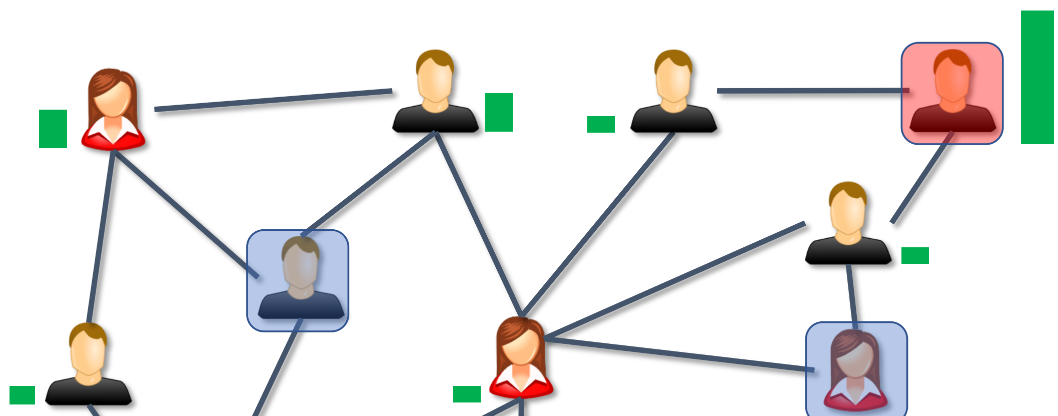
SOLUTION

Improve policy by learning from experience → no need to compute optima

Challenge #2: How to Represent

- **Action value function:** $\hat{Q}(S_t, v; \Theta)$
  - Estimate of goodness of vertex v in state S_t

- **Representation of v: Feature engineering**
  - Degree, 2-hop neighborhood size, other centrality measures…



PROBLEMS

1- **Task-specific engineering** needed

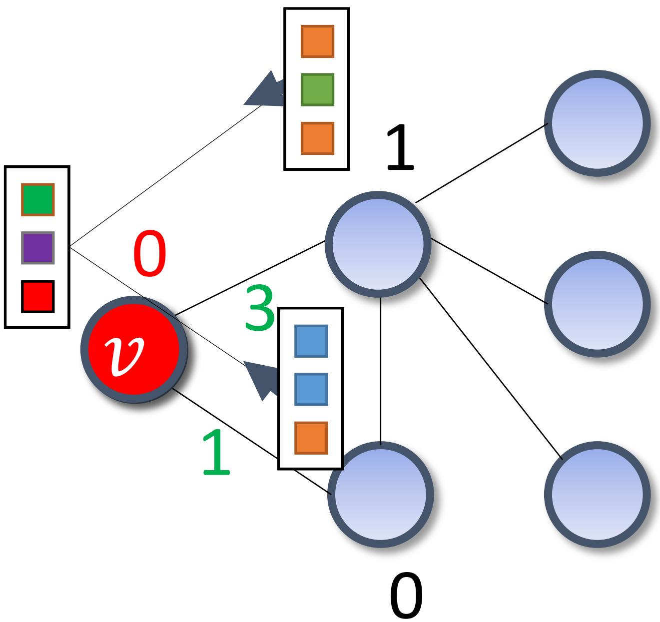
2- Hard to tell **what is a good feature**

3- Difficult to generalize across **diff. graph sizes**

Deep Representation Learning

structure2vec

Dai, Hanjun, Bo Dai, and Le Song. "Discriminative embeddings of latent variable models for structured data." *ICML*. 2016.



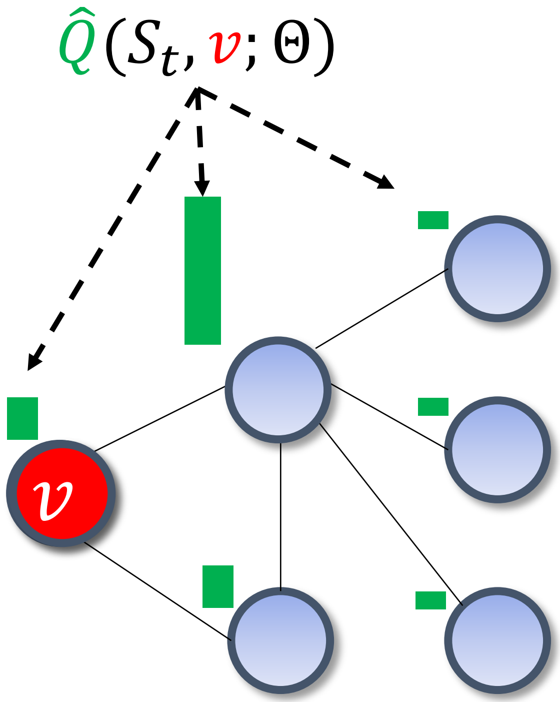
Graph embedding

$$\mu_v^{(t+1)} \leftarrow \text{relu}\Big(\underbrace{\theta_1 x_v +}_{\text{Node's own tag } x_v} \underbrace{\theta_2 \sum_{u\in\mathcal{N}(v)} \mu_u^{(t)}}_{\text{Neighbors' features}} + \underbrace{\theta_3 \sum_{u\in\mathcal{N}(v)} \text{relu}(\theta_4\, w(v,u))}_{\text{Neighbors' edge weights}}\Big)$$

$\boxed{\boldsymbol{\Theta}\text{: model parameters}}$

Repeat embedding T times

Deep Representation Learning

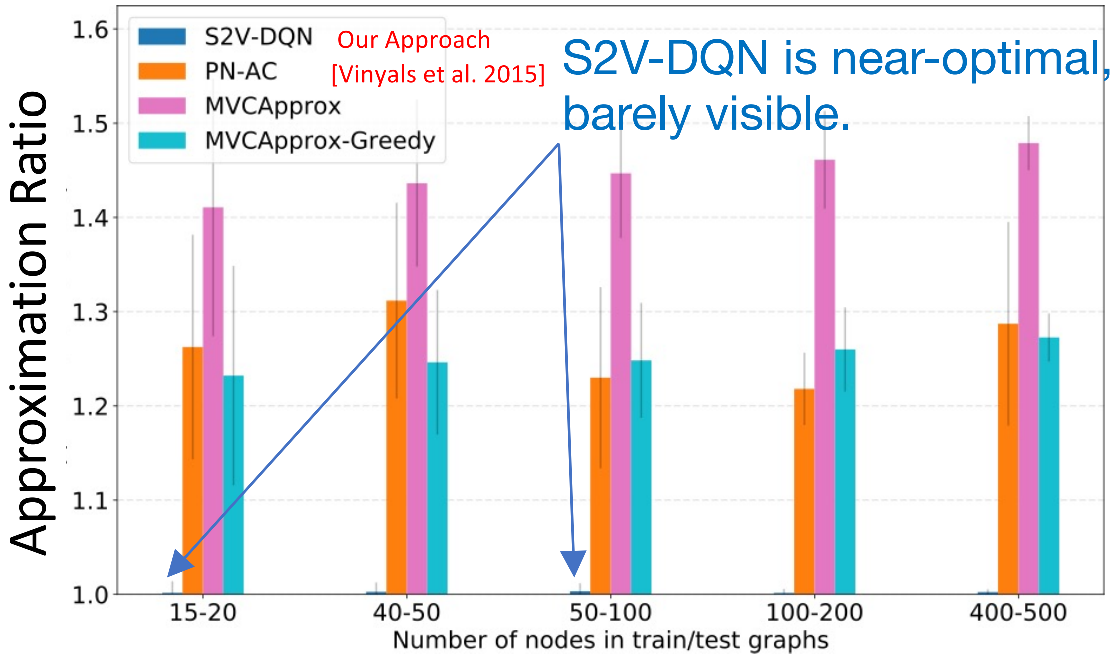


Compute Q-value:

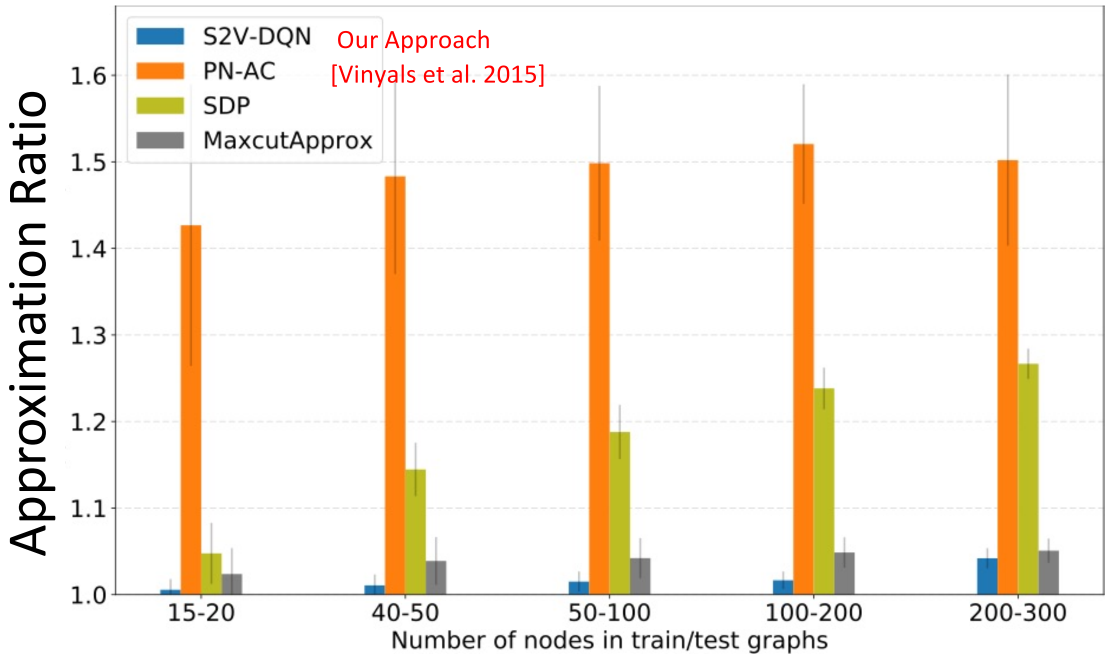
$$\widehat{Q}(h(S), v; \Theta) = \theta_5^\top \operatorname{relu}([\theta_6 \underbrace{\sum_{u \in V} \mu_u^{(T)}}_{\text{Sum-pooling over nodes}}, \theta_7 \, \mu_v^{(T)}])$$

Θ: model parameters

Minimum Vertex Cover - BA

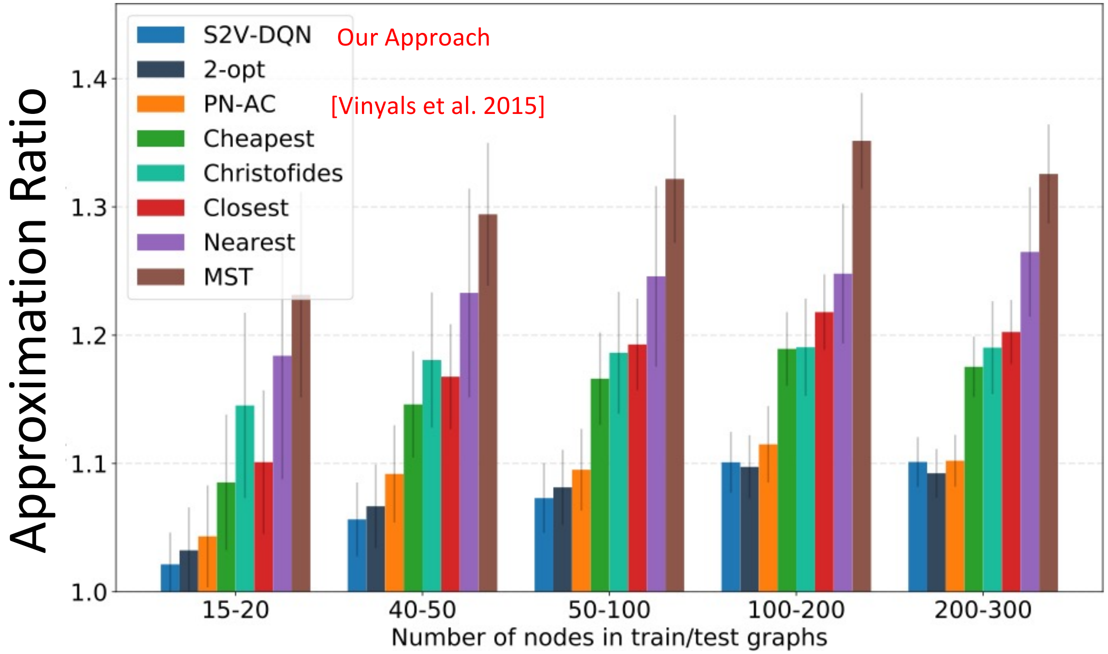


MaxCut - BA

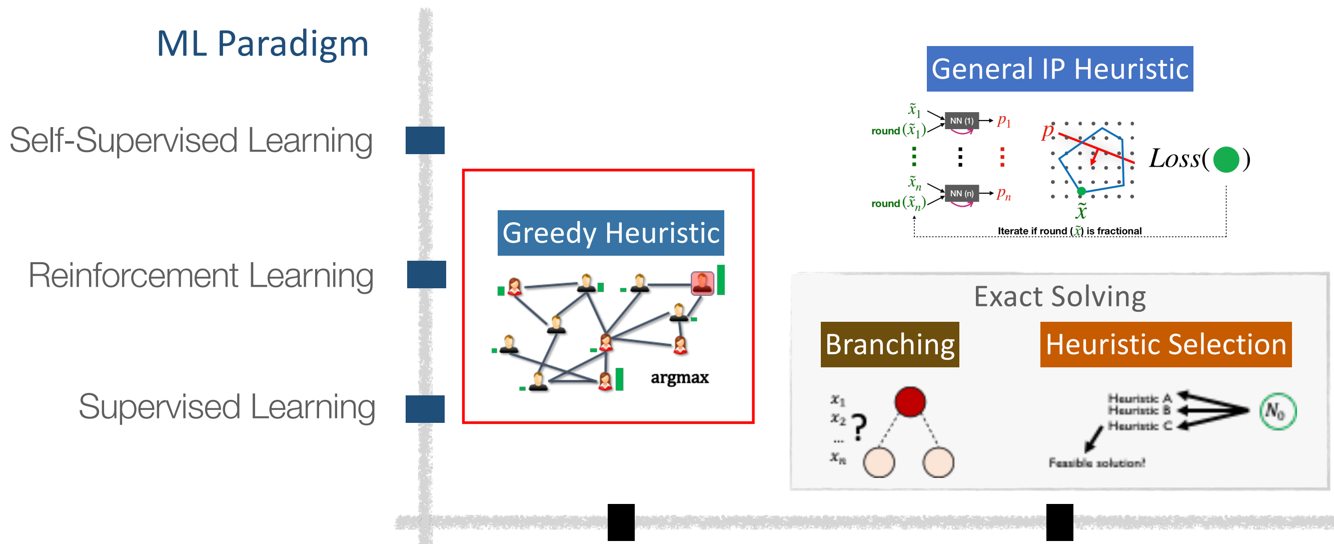


S2V-DQN
Our Approach
PN-AC
[Vinyals et al. 2015]
SDP
MaxcutApprox
Approximation Ratio
1.0
1.1
1.2
1.3
1.4
1.5
1.6
15-20
40-50
50-100
100-200
200-300
Number of nodes in train/test graphs

TSP - clustered



Learning-Driven Algorithm Design



Takeaways

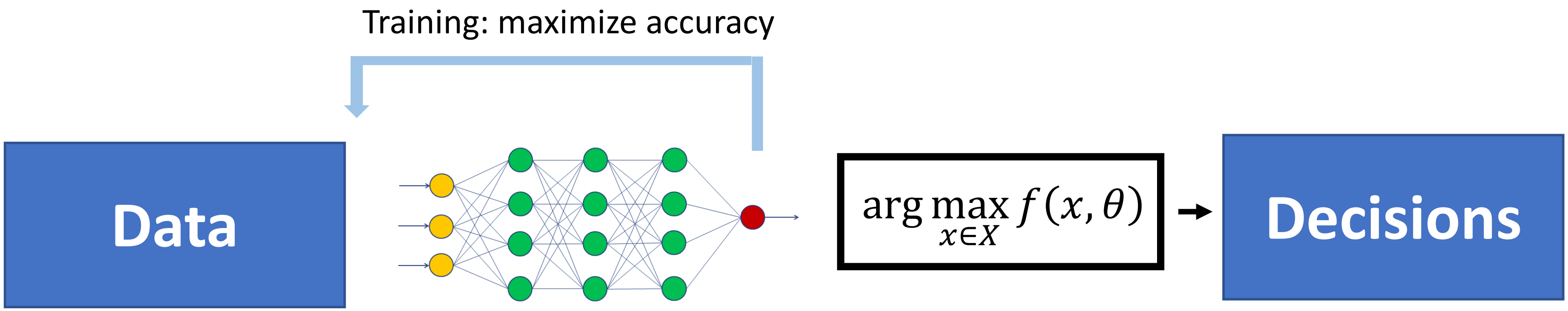
- RL tailors greedy search to family of graph instances
- Learn features jointly with greedy policy
- Human priors encoded via meta-algorithm (Greedy)

The data-decisions pipeline

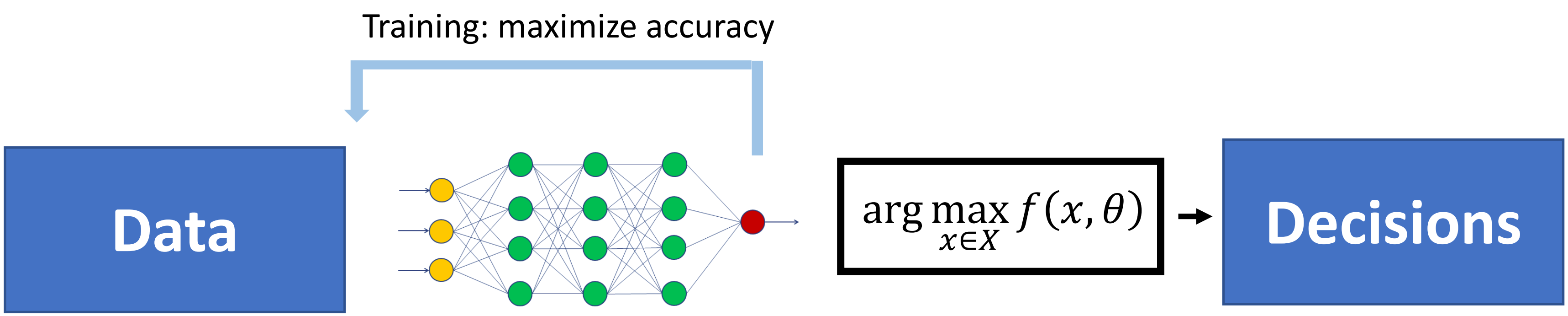
Many real-world applications of AI involve a common template:

[Horvitz and Mitchell 2010; Horvitz 2010]

Observe data → Predictions → Decisions

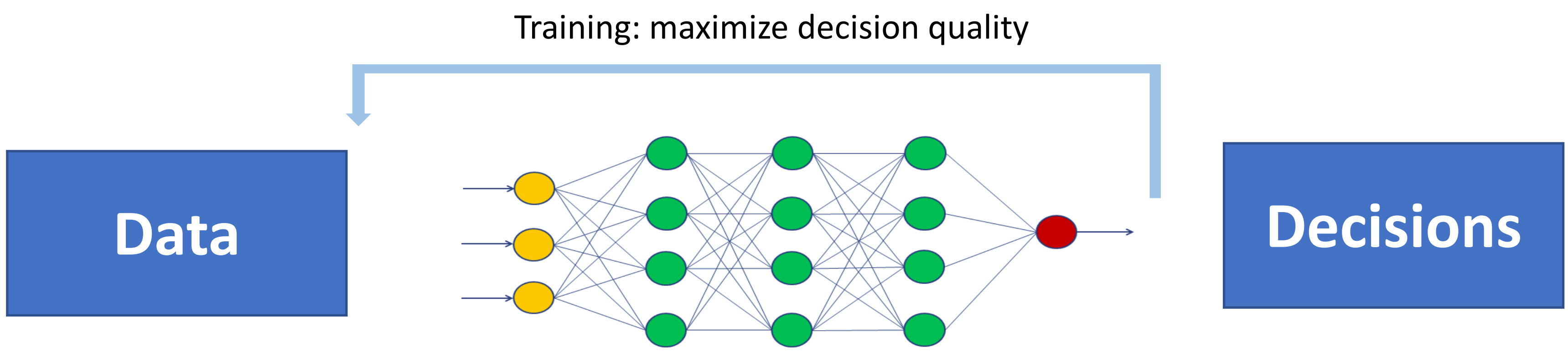


Standard two stage: predict then optimize

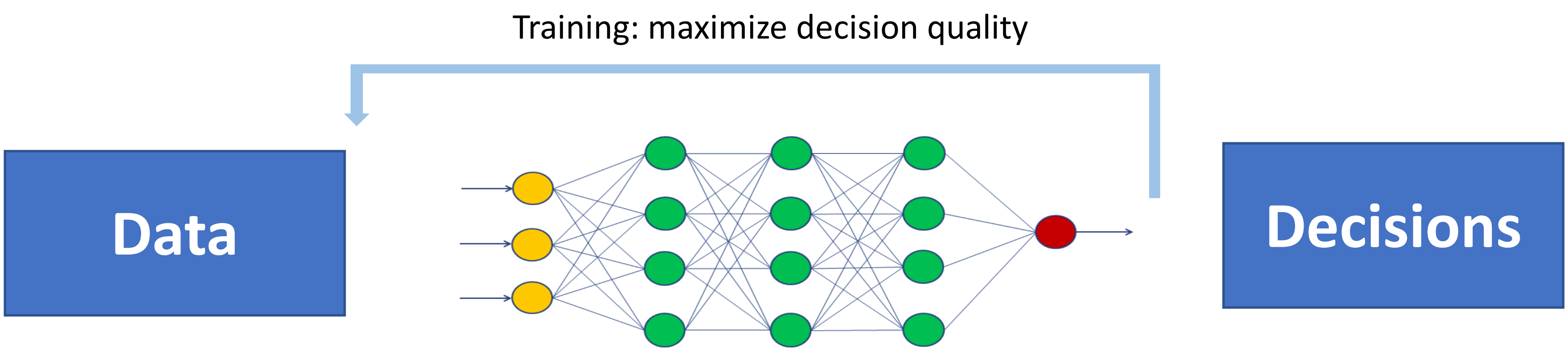


Standard two stage: predict then optimize

Challenge: misalignment between “accuracy” and decision quality

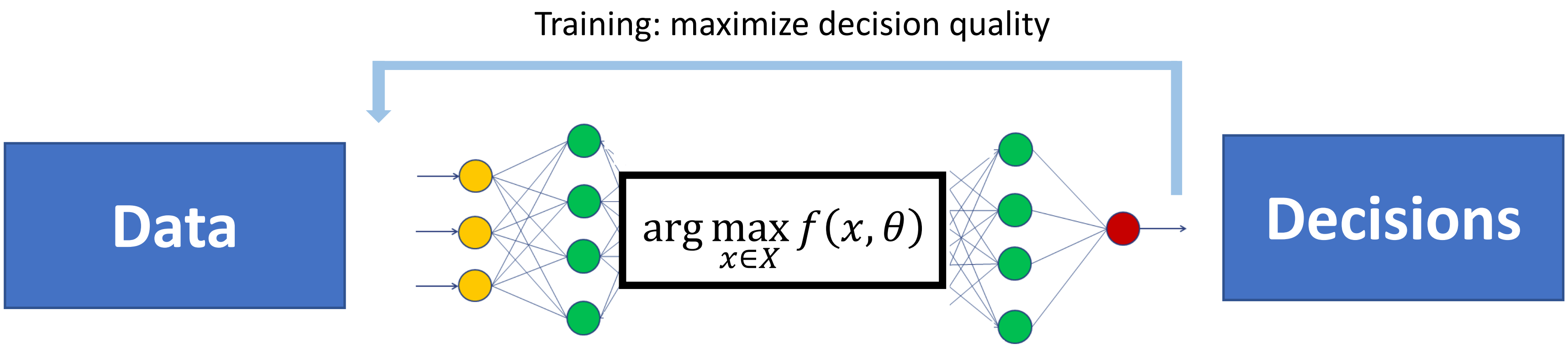


Pure end to end: predict decisions directly from input

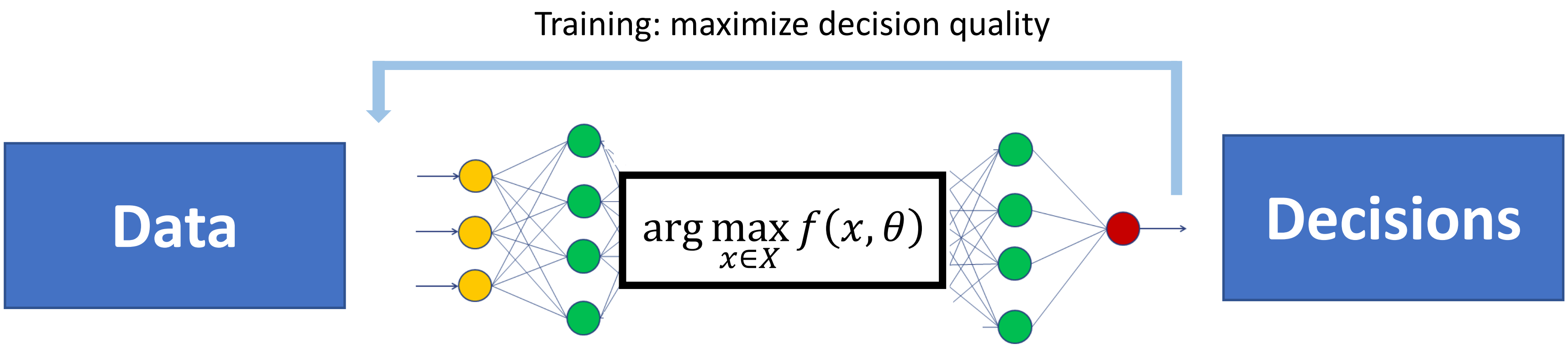


Pure end to end: **predict decisions directly from input**

Challenge: optimization is hard to encode in a NN



Decision-focused learning: differentiable optimization during training

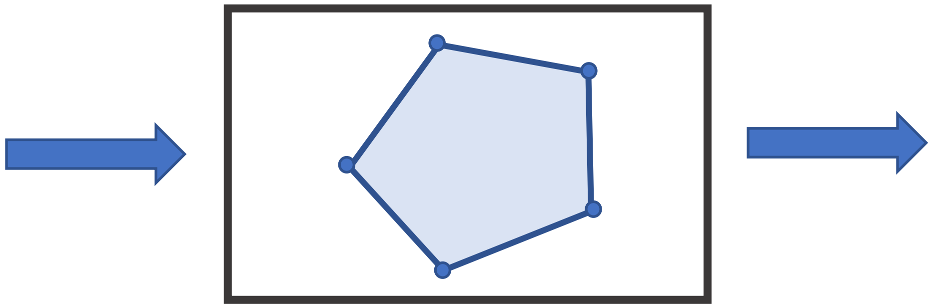


Decision-focused learning: differentiable optimization during training

Challenge: how to make optimization differentiable?

Relax + differentiate

Forward pass: run a solver



Backward pass: sensitivity analysis via KKT conditions

Convex QPs *[Amos and Kolter 2018, Donti et al 2018]*
Linear and submodular programs *[Wilder, Dilkina, Tambe 2019]*
MAXSAT (via SDP relaxation) *[Wang, Donti, Wilder, Kolter 2019]*
MIPs *[Ferber, Wilder, Dilkina, Tambe 2019]*

Some problems don't have good relaxations
Slow to solve continuous optimization problem
Slow to backprop through – $O(n^3)$

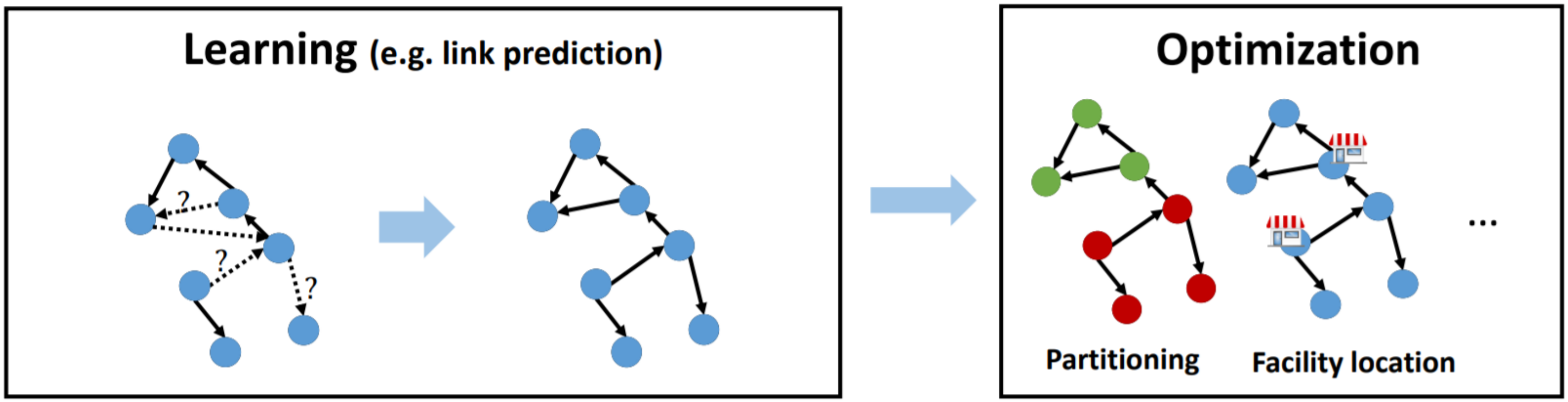
Our Alternative

- Learn a **representation** that maps the original problem to a simpler (efficiently differentiable) **proxy problem**.

- **Instantiation for a class of graph problems**: k-means clustering in embedding space.

Bryan Wilder, Eric Ewing, Bistra Dilkina, Milind Tambe.
End to End Learning and Optimization on Graphs.
NeurIPS, 2019.

Graph learning + graph optimization



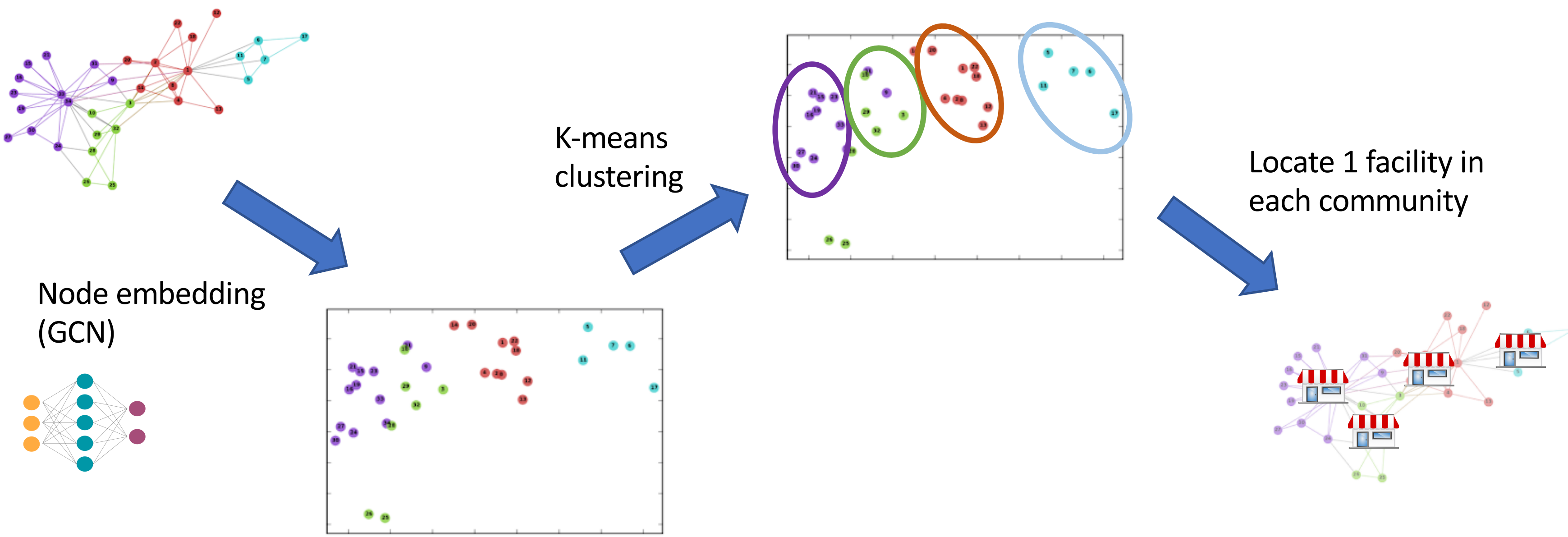
Problem classes

- **Partition the nodes into K disjoint groups**
  - Community detection, maxcut, …
- **Select a subset of K nodes**
  - Facility location, influence maximization, …
- Methods of choice are often combinatorial/discrete

Approach

- Observation: **clustering nodes** is a good proxy
  - Partitioning: correspond to well-connected subgroups
  - Facility location: put one facility in each community
- Observation: graph learning approaches already embed into R^n

ClusterNet Approach



Differentiable K-means

Forward pass

$$\mu_k = \frac{\sum_j r_{jk} x_j}{\sum_j r_{jk}}$$

Update cluster centers

$$r_{jk} = \frac{\exp(-\beta ||x_j - \mu_k||)}{\sum_\ell \exp(-\beta ||x_j - \mu_\ell||)}$$

Softmax update to node assignments

Differentiable K-means

Backward pass

- Option 1: differentiate through the fixed-point condition

$$\mu^t = \mu^{t+1}$$

  - Prohibitively slow, memory-intensive

Differentiable K-means

Backward pass

- Option 1: differentiate through the fixed-point condition

$$\mu^t = \mu^{t+1}$$

  - Prohibitively slow, memory-intensive
- Option 2: unroll the entire series of updates
  - Cost scales with # iterations
  - Have to stick to differentiable operations

Differentiable K-means

Backward pass

- Option 1: differentiate through the fixed-point condition

$$\mu^t = \mu^{t+1}$$

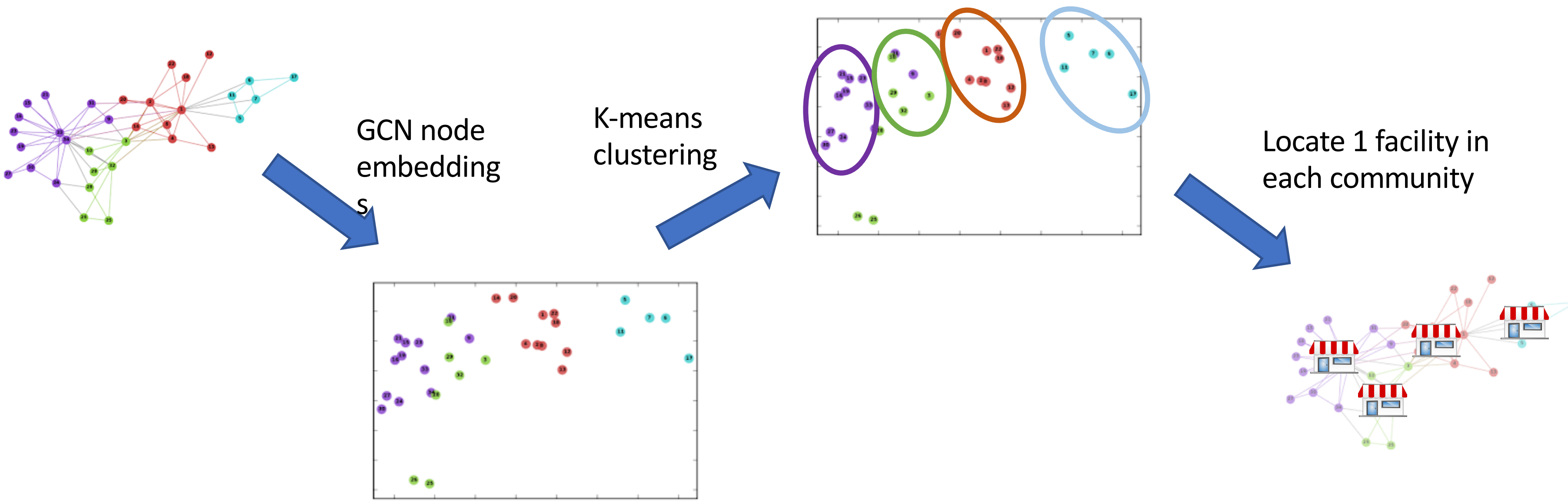
  - Prohibitively slow, memory-intensive
- Option 2: unroll the entire series of updates
  - Cost scales with # iterations
  - Have to stick to differentiable operations
- **Option 3: get the solution, then unroll one update**
  - Do anything to solve the forward pass
  - Linear time/memory, implemented in vanilla pytorch

Differentiable K-means

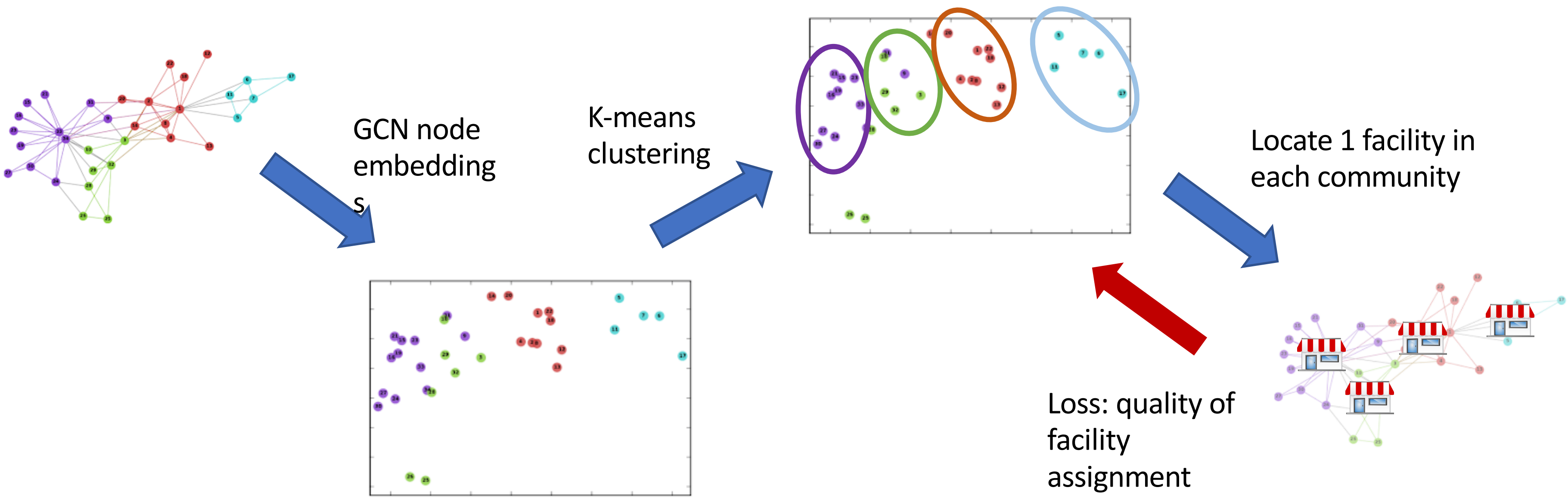
Theorem [informal]: provided the clusters are sufficiently balanced and well-separated, the Option 3 approximate gradients converge exponentially quickly to the true ones.

Idea: show that this corresponds to approximating a particular term in the analytical fixed-point gradients.

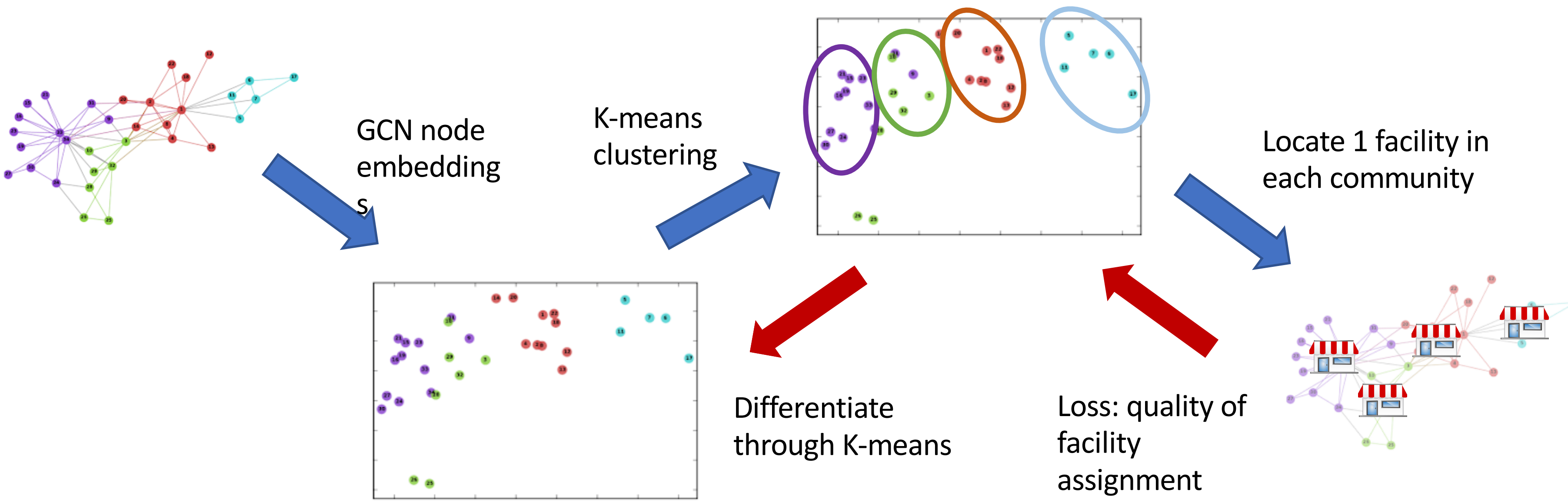
ClusterNet Approach



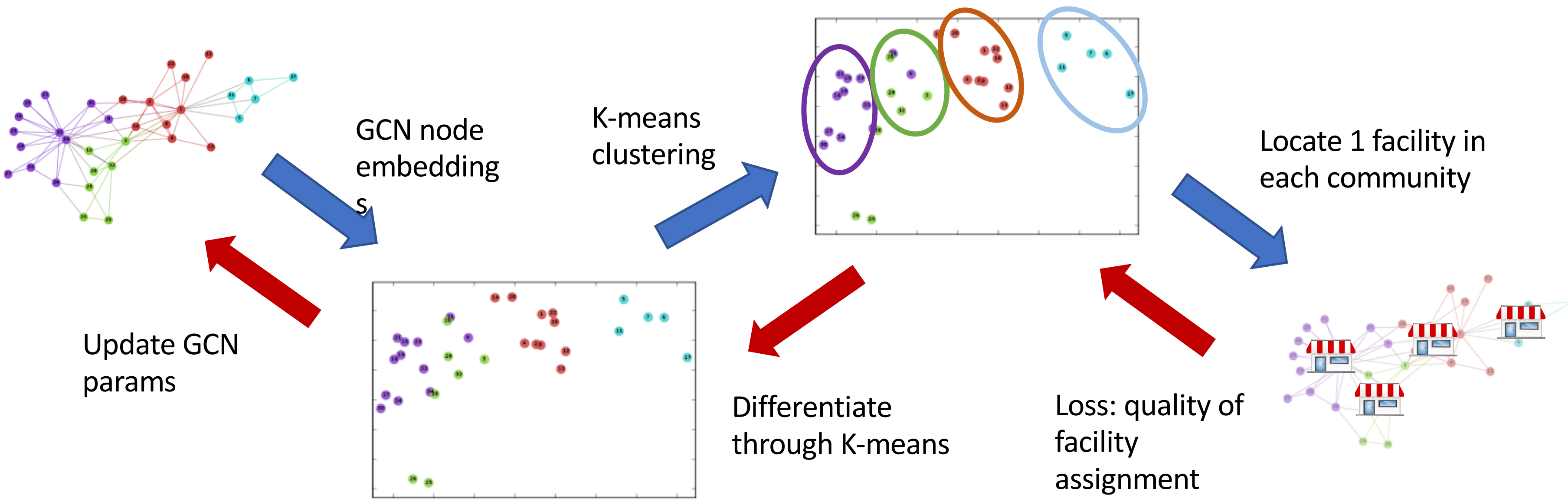
ClusterNet Approach



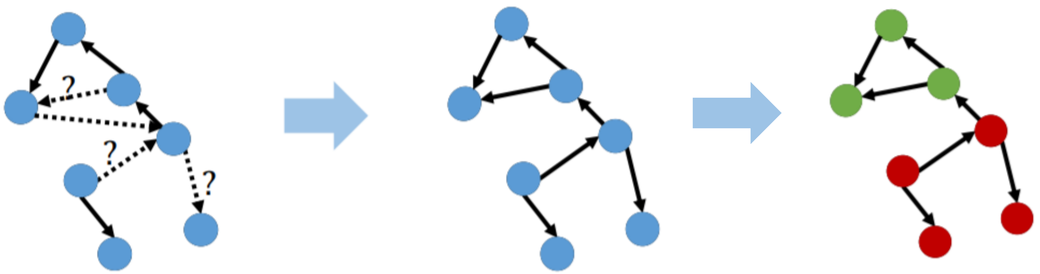
ClusterNet Approach



ClusterNet Approach



Example: community detection



Observe partial graph

Predict unseen edges

Find communities

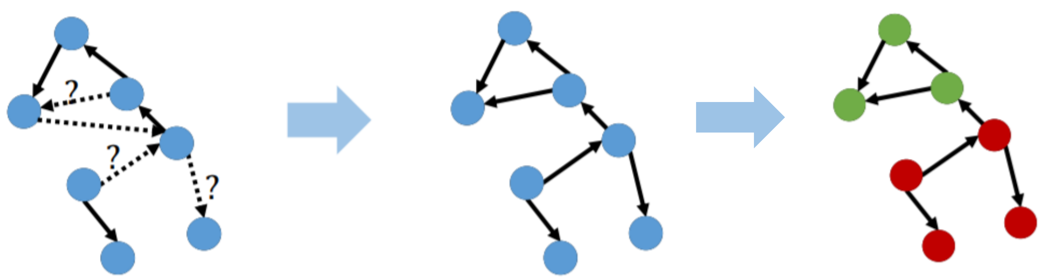
max **modularity**

$$\max_r \frac{1}{2m} \sum_{u,v \in V} \sum_{k=1}^{K} \left[A_{u,v} - \frac{d_u d_v}{2m} \right] r_{uk} r_{vk}$$

$$r_{uk} \in \{0,1\} \quad \forall u \in V, k = 1 \dots K$$

$$\sum_{k=1}^{K} r_{uk} = 1 \quad \forall u \in V$$

Example: community detection



Observe partial graph

Predict unseen edges

Find communities

max **modularity**

$$\max_{r} \frac{1}{2m} \sum_{u,v \in V} \sum_{k=1}^{K} \left[A_{u,v} - \frac{d_u d_v}{2m} \right] r_{uk} r_{vk}$$

$$r_{uk} \in \{0,1\} \quad \forall u \in V, k = 1 \dots K$$

$$\sum_{k=1}^{K} r_{uk} = 1 \quad \forall u \in V$$

- **Useful in scientific discovery** (social groups, functional modules in biological networks)
- In applications, **two-stage approach is common**: [Yan & Gegory '12, Burgess et al '16, Berlusconi et al '16, Tan et al '16, Bahulker et al '18…]

Experiments

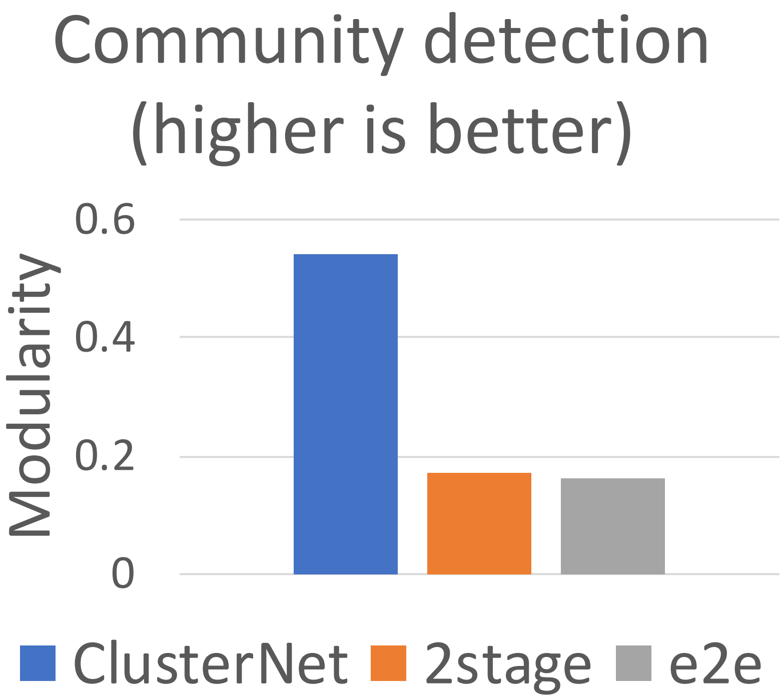
- **Learning problem**: link prediction
- **Optimization:** community detection and facility location problems
- Train **GCNs** as predictive component

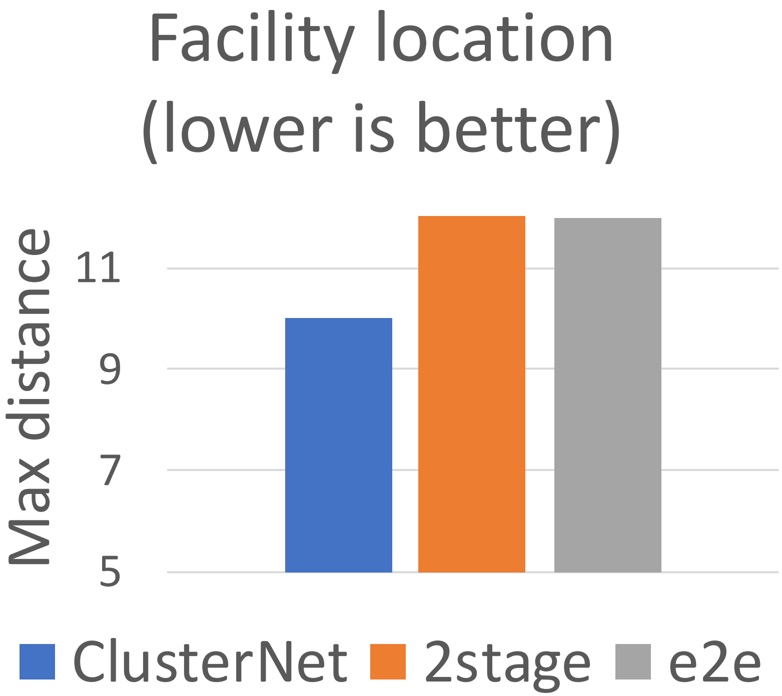
Experiments

- **Learning problem**: link prediction
- **Optimization:** community detection and facility location problems
- Train **GCNs** as predictive component

- **Comparison**
  - Two stage: GCN + expert-designed algorithm (**2Stage**)
  - Pure end to end: Deep GCN to predict optimal solution (**e2e**)

Results: single-graph link prediction

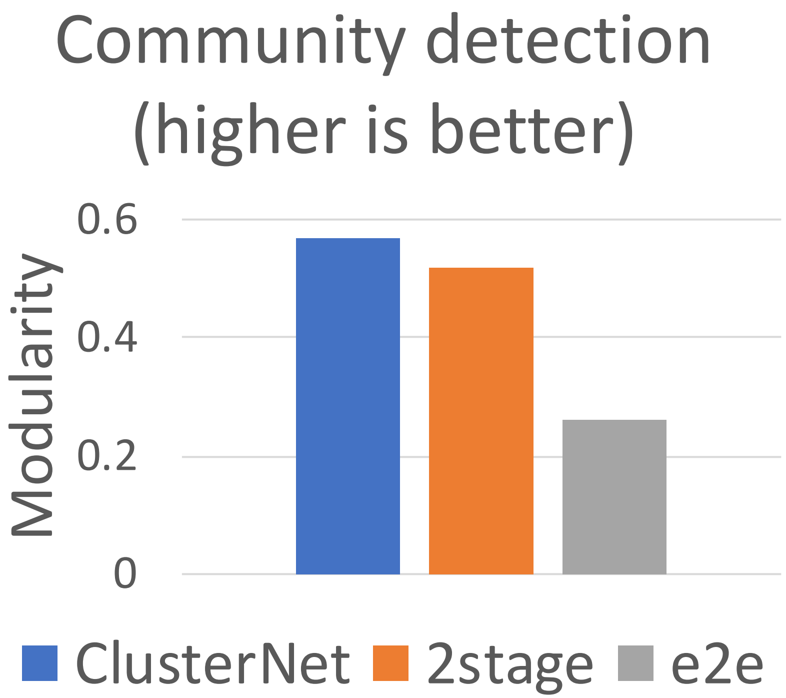


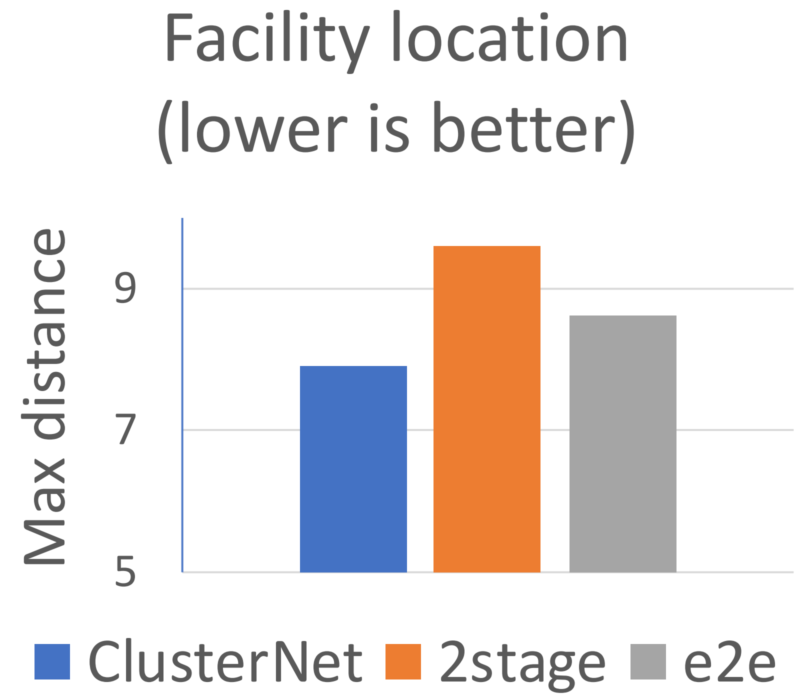


Representative example from **cora**, citeseer, protein interaction, facebook, adolescent health networks

Community algos: CNM, Newman, SpectralClustering
Facility Locations algos: greedy, gonzalez2approx

Results: generalization across graphs





ClusterNet learns generalizable strategies for optimization!

Results: optimization only ClusterNet as a solver

	Optimization				
	cora	cite.	prot.	adol	fb
ClusterNet	**0.71**	**0.76**	**0.52**	0.55	**0.80**
GCN-e2e	0.07	0.08	0.14	0.15	0.15
Train-CNM	0.08	0.34	0.05	**0.60**	**0.80**
Train-Newman	0.20	0.22	0.29	0.30	0.47
Train-SC	0.15	0.08	0.07	0.46	0.79

ClusterNet learns an effective graph optimization solver!

Takeaways

- Good decisions require integrating learning and optimization
- Pure end-to-end methods miss out on useful structure
- Even simple optimization primitives provide good inductive bias

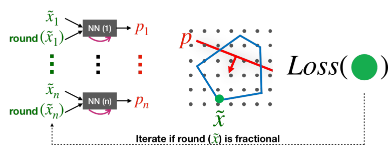
ML ⟺ Combinatorial Optimization

- Exciting and growing research area

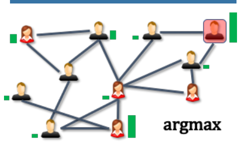
USC

Infusing Discrete Optimization with Machine Learning

General IP Heuristic

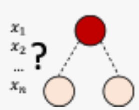


Greedy Heuristic

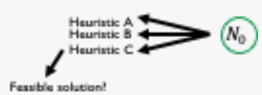


Exact Solving

Branching



Heuristic Selection



Graph Optimization

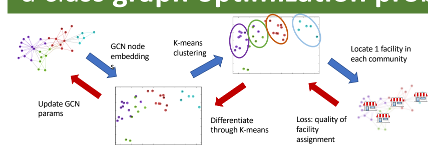
Integer Programming

Problem Type

Augment discrete optimization algorithms with learning components

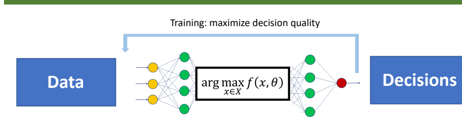
Infusing ML with Constrained Decision Making

ClusterNET: Differentiable kmeans for a class **graph optimization problems**



MIPaaL: **MIP** as a layer in Neural Networks

Decision-focused learning for **submodular optimization and LP**



Learning methods that incorporate the combinatorial decisions they inform

USC

ML ⟺ Combinatorial Optimization

- Exciting and growing research area
- Design discrete optimization algorithms with learning components
- Learning methods that incorporate the combinatorial decision making they inform

ExxonMobil

NSF

Microsoft AI for Earth

Georgia Tech

DEPARTMENT of the INTERIOR SE CLIMATE SCIENCE CENTER

Thank you!