Solving Combinatorial Optimization Problems with Graph Networks

Defeng Liu

Supervised by Andrea Lodi

2020.02.24

POLYTECHNIQUE MONTRÉAL

DATA SCIENCE FOR REAL-TIME DECISION-MAKING

Outline

- Introduction
- ML for combinatorial optimization
- Applications with GNNs

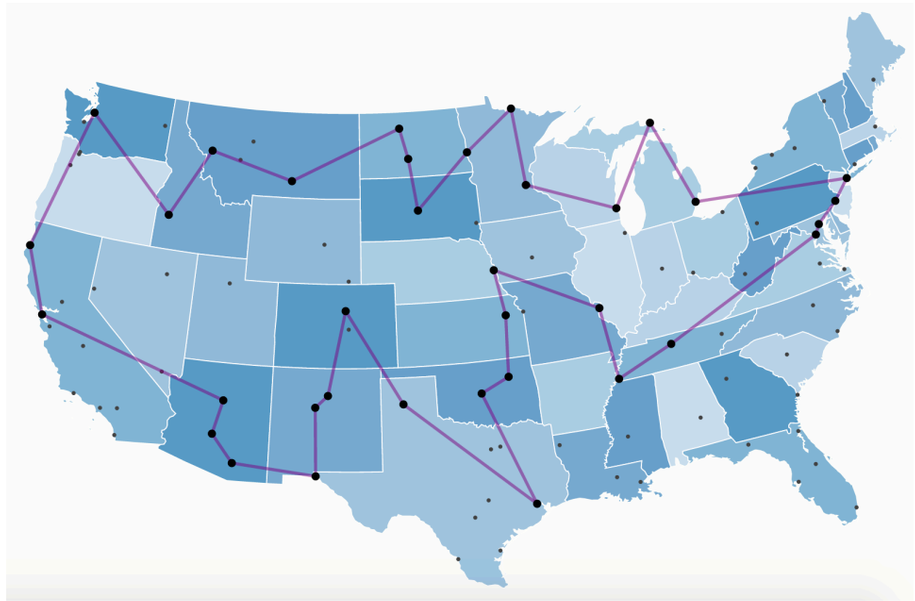
Outline

- **Introduction**
- ML for combinatorial optimization
- Applications with GNNs

Combinatorial Optimization

A large range of real world decision problems are ***Combinatorial Optimization(CO)*** problems:

- Examples: supply chain, packing and scheduling, etc…
- Techniques: branch and bound, cutting planes, heuristics, pre-processing …



The Traveling Salesman Problem
examples.gurobi.com

Loading a 20-foot shipping container
www.movehub.com

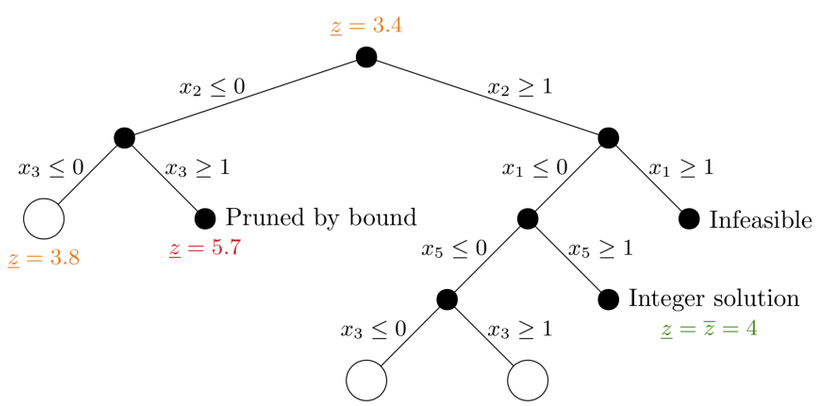
Combinatorial Optimization

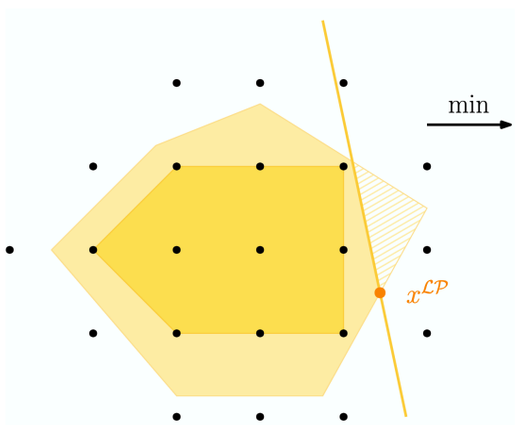
A large range of real world decision problems are ***Combinatorial Optimization(CO)*** problems:

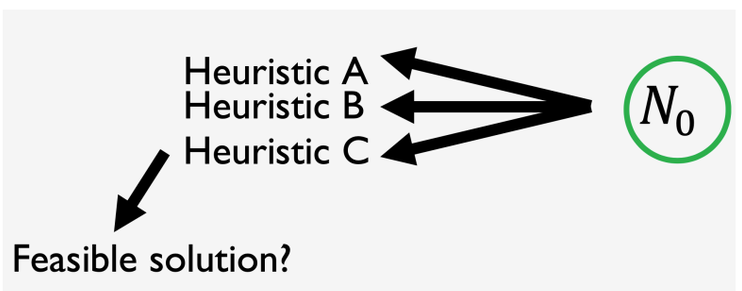
- Examples: supply chain, packing and scheduling, etc…
- Techniques: branch and bound, cutting planes, heuristics, pre-processing …

Many of them are NP-complete, practically we develop different methods:

- Exact methods, eg. branch and bound, cutting planes
- Approximation methods
- Heuristics, meta-heuristics





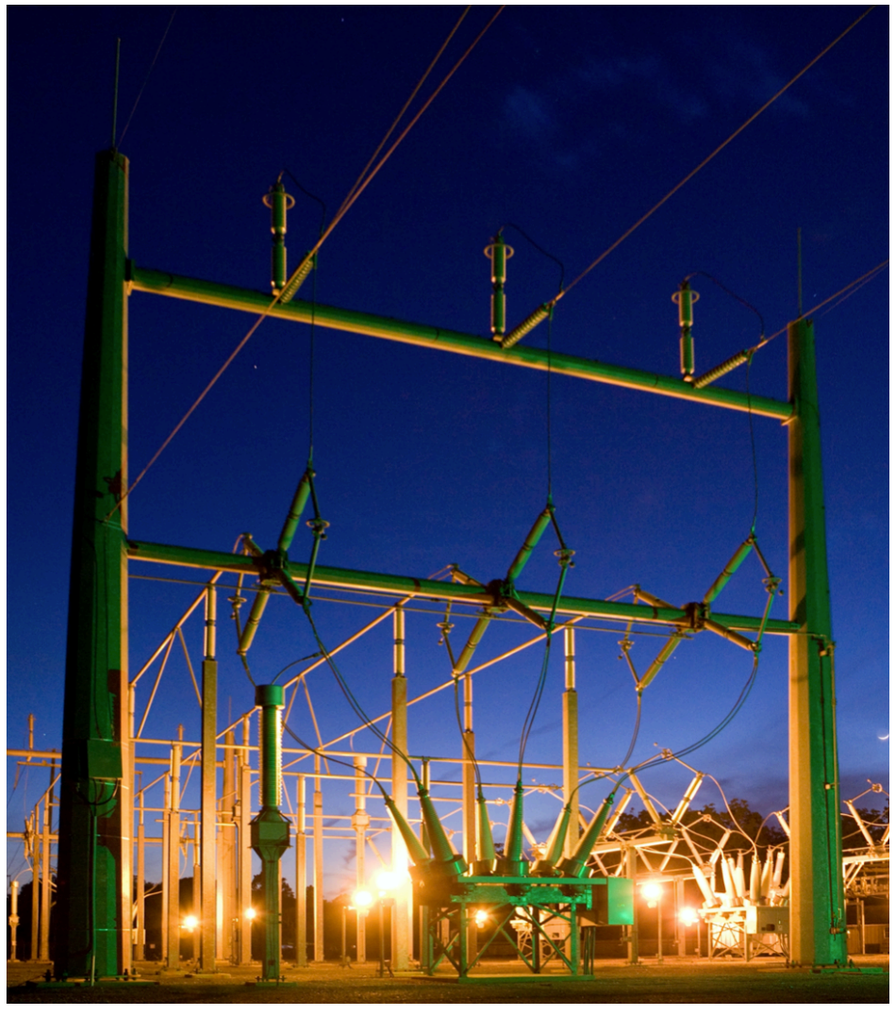


Motivation

- Although we have sophisticated solvers(*CPLEX*, *Gurobi*, etc), solving CO problems is still a challenging computation process:
  - For CO problems:
    - Challenge1: restricted computation/time budget
      - Desirable to compute fast solutions
  - For CO algorithms:
    - Challenge2: complex and heavy auxiliary tasks, ex. branching strategies, cuts selection
      - Cheaper approximators / new decision policies

Motivation

- In many real world applications, instances of the same problem are solved repeatedly.
    - Application in power systems [Xavier et al. 2019]
        - Schedule 3.8 trillion kWh production ($400 billion market annually in US)
        - Solved multiple times daily
        - 10x speed up combining ML and MIP



Hypothesis - What can be learned from data?

- In state space of CO problems, the instances in neighborhoods
  - have inherent structural commonalities,
  - show similar solving procedures.
  - their solutions are close to each other in solution space

- Valuable patterns/characteristics can be learned from data.
  - approximations of probability distributions of random elements
  - solutions at different levels of detail
  - decision policies in CO algorithms

Requirement

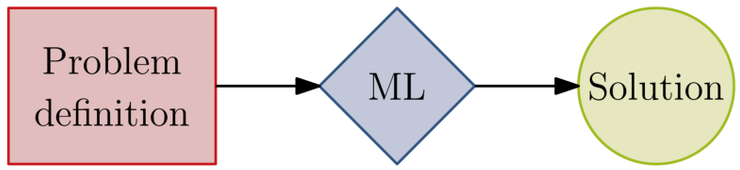
- We want to keep the guarantees provided by exact OR algorithms (feasibility, optimality)

Outline

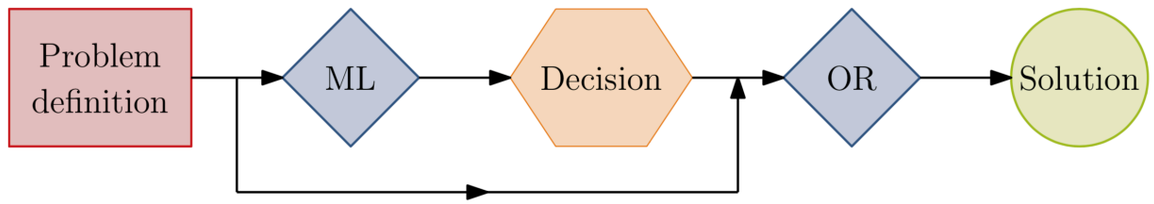
- Introduction
- **ML for combinatorial optimization**
- Applications with GNNs

Methodology - Paradigms

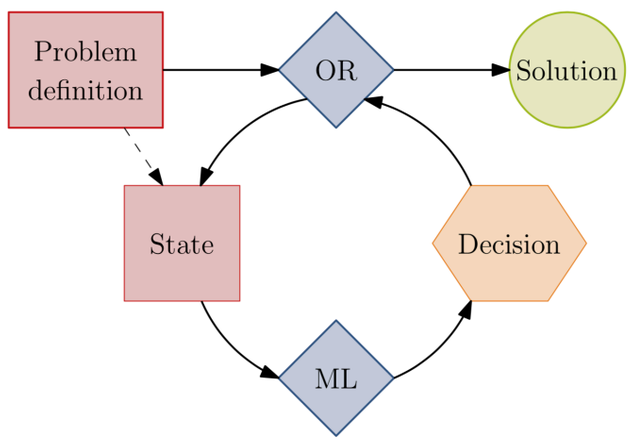
- **End to End Learning**
  - Build a machine learning model to directly output solutions.



- **Learning Properties**
  - Build a machine learning model to parametrize your OR algorithm.



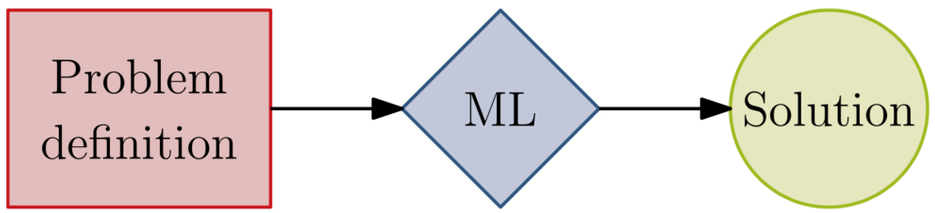
- **Learning Repeated Decisions**
  - The machine learning model is used repeatedly by the OR algorithm to assist it in its solving process.



More details can be found in the proposal and the survey [Bengio et al. 2018]

Methodology - Paradigms

- **End to End Learning**



- Learning TSP solutions
  [*Bello et al. 2017*][*Kool and Welling 2018*][*Emami and Ranka 2018*]
  [*Vinyals et al. 2015*][*Nowak et al. 2017*]

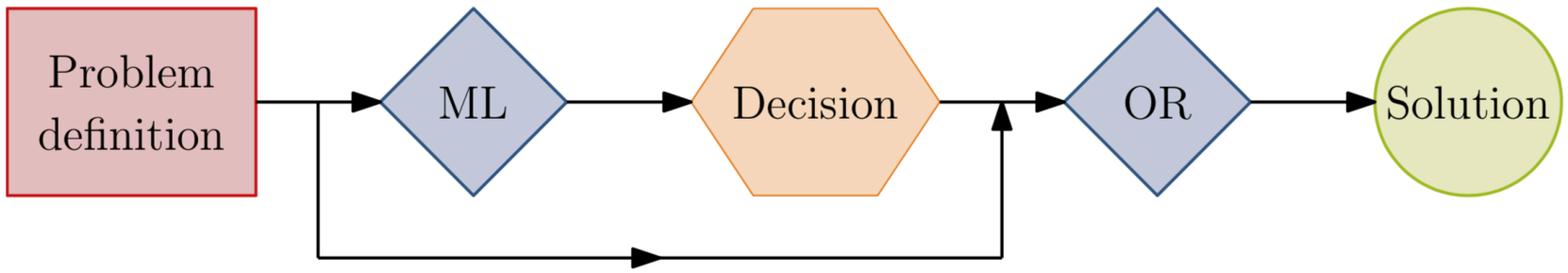
- Predict aggregated solutions to MILP under partial information
  [*Larsen et al. 2018*]

- Approximate obj value to SDP (for cut selection)
  [*Baltean-Lugojan et al. 2018*]

Slides supported by Antoine Prouvost

Methodology - Paradigms

- **Learning Properties**



- Use a decomposition method
  [*Kruber et al. 2017*]

- Linearize an MIQP
  [*Bonami et al. 2018*]

- Provide initial cancer treatment plans to inverse optimization
  [*Mahmood et al. 2018*]

Slides supported by Antoine Prouvost

Methodology - Paradigms

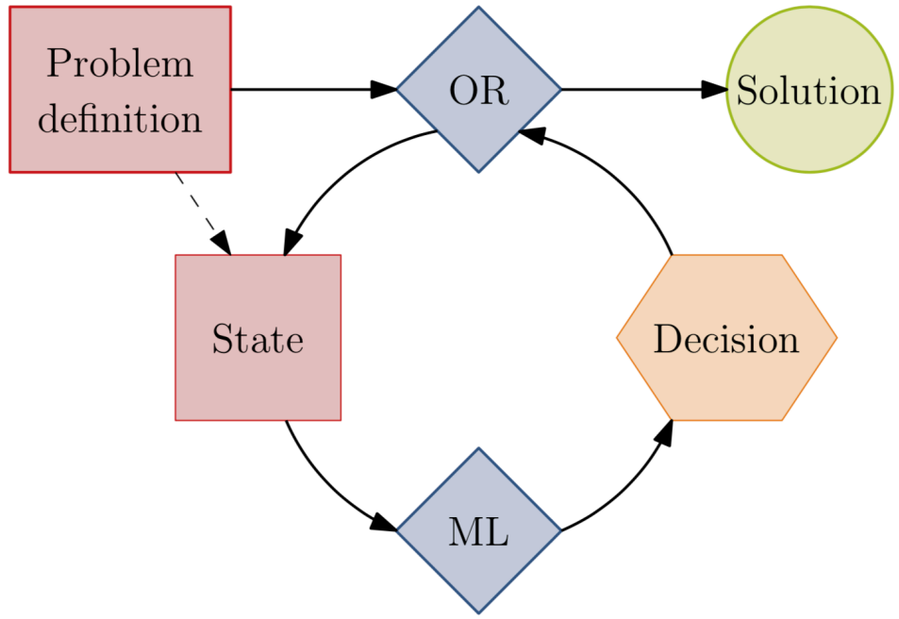
- **Learning Repeated Decisions**

  - Learning where to run heuristics in B&B
    [*Khalil et al. 2017b*]

  - Learning to branch
    [*Lodi and Zarpellon 2017*] (survey)

  - Learning gradient descent
    e.g. [*Andrychowicz et al. 2016*]

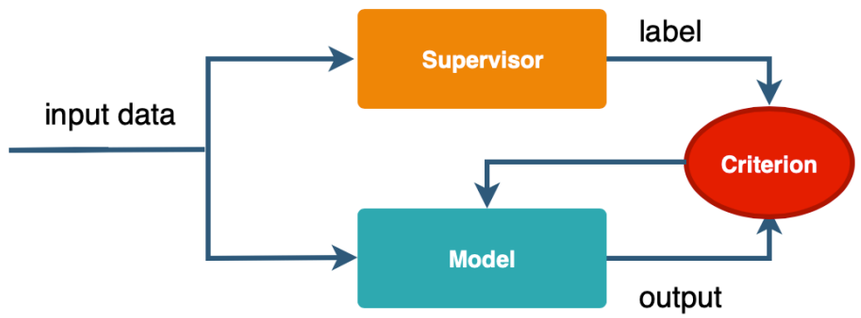
  - Learning cutting plane selection
    [*Baltean-Lugojan et al. 2018*]



Slides supported by Antoine Prouvost

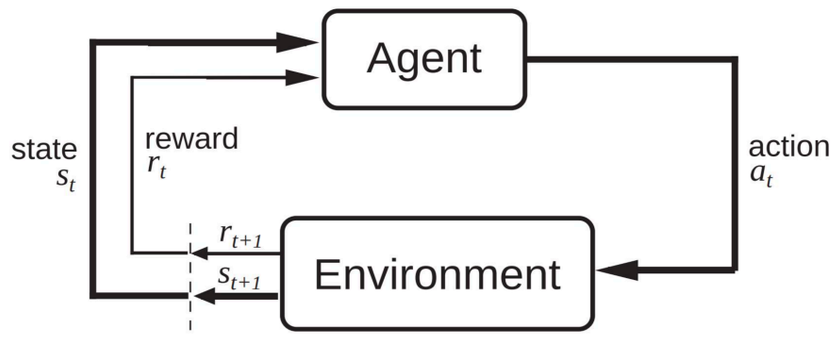
Methodology - Learning Mechanisms

- By label/demonstration:
  - approximating branching heuristics [*Khalil et al. 2016*]
  - predicting outcome of cuts [*Baltean-Lugojan et al. 2018*]
  - approximating tactical solutions [*Larsen et al. (2019)*]



Supervised Learning

- By experience
  - learning to solve TSP [*Bello et al. 2016, Kool et al. 2018*]
  - learning to solve problems on graphs [*Khalil et al. 2017a*]
  - learning to select cuts [*Tang et al. 2019*]



RL framework by *Sutton et al., 1998*

Methodology - Representation

Applications of Deep Neural Networks:

- *Multilayer Perceptron (MLPs)*
  - fixed-sized data [*Larsen et al. 2019*]
- *Recurrent Neural Networks (RNNs)*
  - sequencial data [*Vinyals et al. 2015, Bello et al. 2016,*]
- *Graph Neural Networks (GNNs)*
  - graph space data [*Khalil et al. 2017a, Kool et al. 2018, Gasse et al. 2019*]
- *Convolutional Neural Networks (CNNs)*
  - pixel space data [*Silver et al., 2016, 2018, Miki et all. 2019*]

More details can be found in the proposal and the survey [Bengio et al. 2018]

Outline

- Introduction
- ML for combinatorial optimization
- **Applications with GNNs**

Learn heuristics for graph opt. problems

Khalil et al. "Learning combinatorial optimization algorithms over graphs." *NIPS*. 2017.

Learn heuristics for graph opt. problems

Khalil et al. "Learning combinatorial optimization algorithms over graphs." *NIPS*. 2017.

- Problem statement

Given a **graph optimization problem** G
and a **distribution** $\mathcal{D}$ of problem instances,
can we **learn better greedy heuristics** that
generalize to unseen instances from $\mathcal{D}$?

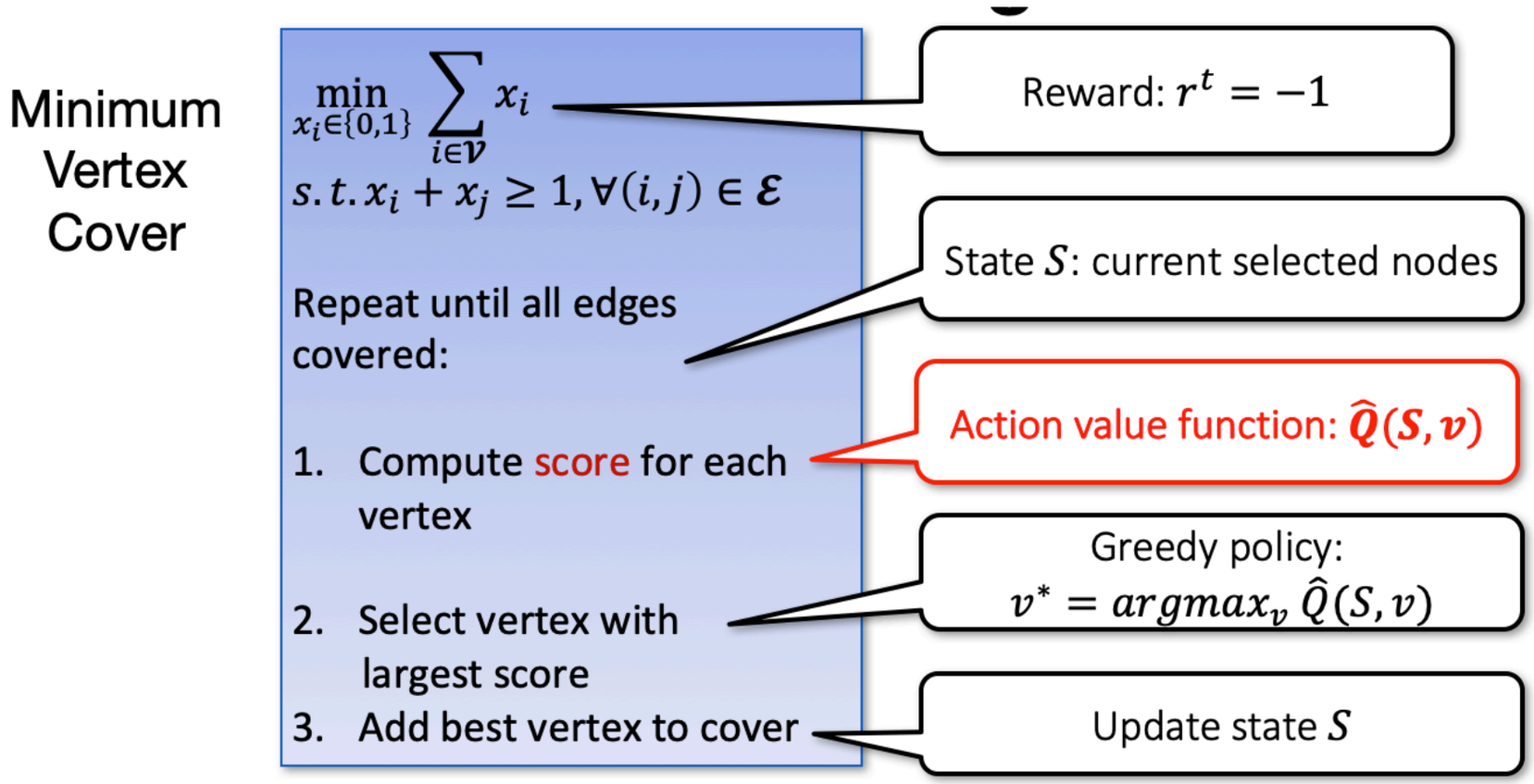
Minimum Vertex Cover	**Maximum Cut**	**Traveling Salesman Prob.**
Insert nodes into cover	Insert nodes into subset	Insert nodes into sub-tour

More details can be found from https://www.cc.gatech.edu/~hdai8/slides/hanjun_hotcse.pptx.

Learn heuristics for graph opt. problems

Khalil et al. "Learning combinatorial optimization algorithms over graphs." *NIPS*. 2017.

- RL Q-learning framework

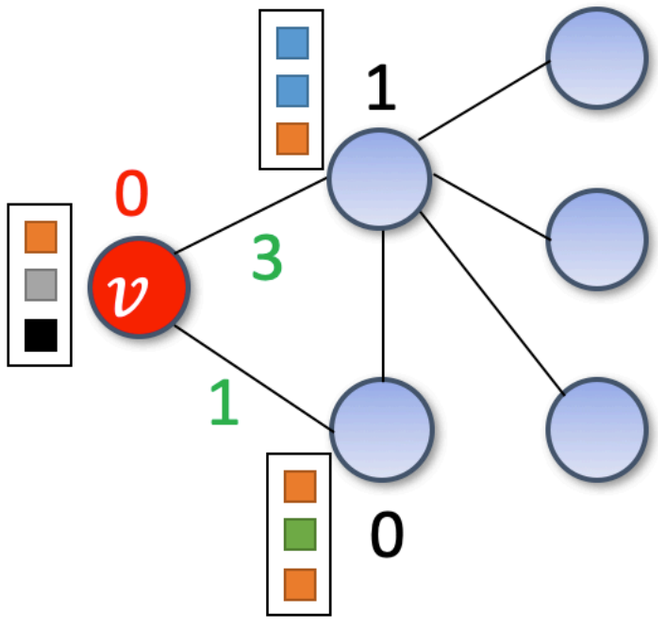


More details can be found from https://www.cc.gatech.edu/~hdai8/slides/hanjun_hotcse.pptx.

Learn heuristics for graph opt. problems

Khalil et al. "Learning combinatorial optimization algorithms over graphs." *NIPS*. 2017.

- Graph embedding (structure2vector)



Repeat embedding T times:

Updating feature vector

$$\mu_v^{(t+1)} \leftarrow \text{relu}\Big(\theta_1 x_v + \theta_2 \sum_{u \in \mathcal{N}(v)} \mu_u^{(t)} + \theta_3 \sum_{u \in \mathcal{N}(v)} \text{relu}(\theta_4\, w(v,u))\Big)$$

Node's own tag x_v

Neighbors' features

Neighbors' edge weights

Non-linearity: $\text{relu}(x) = \max(0, x)$

Θ: model parameters

More details can be found from https://www.cc.gatech.edu/~hdai8/slides/hanjun_hotcse.pptx.

Learn heuristics for graph opt. problems

Khalil et al. "Learning combinatorial optimization algorithms over graphs." *NIPS*. 2017.

- Results on realistic instances

Table 3: Realistic data experiments, results summary. Values are average approximation ratios.

Problem	Dataset	S2V-DQN	Best Competitor	2nd Best Competitor
MVC	MemeTracker	**1.0021**	1.2220 (MVCApprox-Greedy)	1.4080 (MVCApprox)
MAXCUT	Physics	**1.0223**	1.2825 (MaxcutApprox)	1.8996 (SDP)
TSP	TSPLIB	**1.0475**	1.0947 (2-opt)	1.1771 (Cheapest)

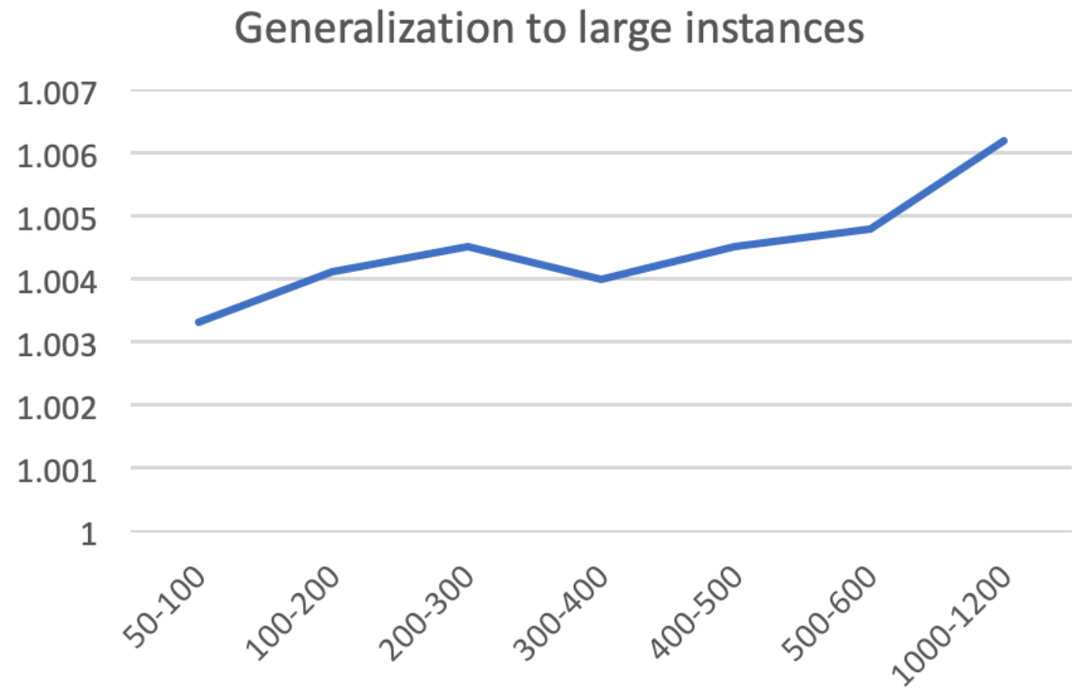
More details can be found from https://www.cc.gatech.edu/~hdai8/slides/hanjun_hotcse.pptx.

Learn heuristics for graph opt. problems

Khalil et al. "Learning combinatorial optimization algorithms over graphs." *NIPS*. 2017.

- Generalization to larger instances

- Train on small graphs with 50-100 nodes
- Generalize to not only graphs from same distribution
- But also larger graphs
- Approximation ratio < 1.007



More details can be found from https://www.cc.gatech.edu/~hdai8/slides/hanjun_hotcse.pptx.

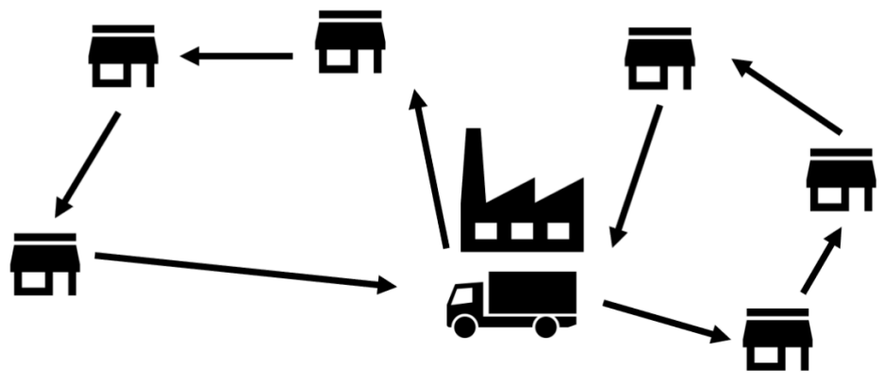
Learn heuristics for routing problems

Kool et all. "Attention, learn to solve routing problems!.” ICLR. 2019.

Learn heuristics for routing problems

Kool et all. "Attention, learn to solve routing problems!." ICLR. 2019.

- Problem statement



- Simplest form of the routing problem
  - One capacitated vehicle
  - Multiple customers
  - Vehicle must return to the depot to refill
  - Objective: Minimize total distance

Previous work: Neural Combinatorial Optimization with RL

Bello et al. "Neural combinatorial optimization with reinforcement learning."(2016).

- Policy model: Pointer Network

$$P_\theta(solution|problem)$$

- RL learning:
  - learn from experience by exploring the solution space, no label needed
  - cost: the tour length of the solution
  - training : adjust θ with RL to minimize the objective(the expected cost of solutions)

$$J(\theta|problem) = E_{solution\sim P_{\theta^*}(.|problem)}[cost(solution|problem)]$$

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\ J(\theta|problem)$$

- Inference: sample solutions from

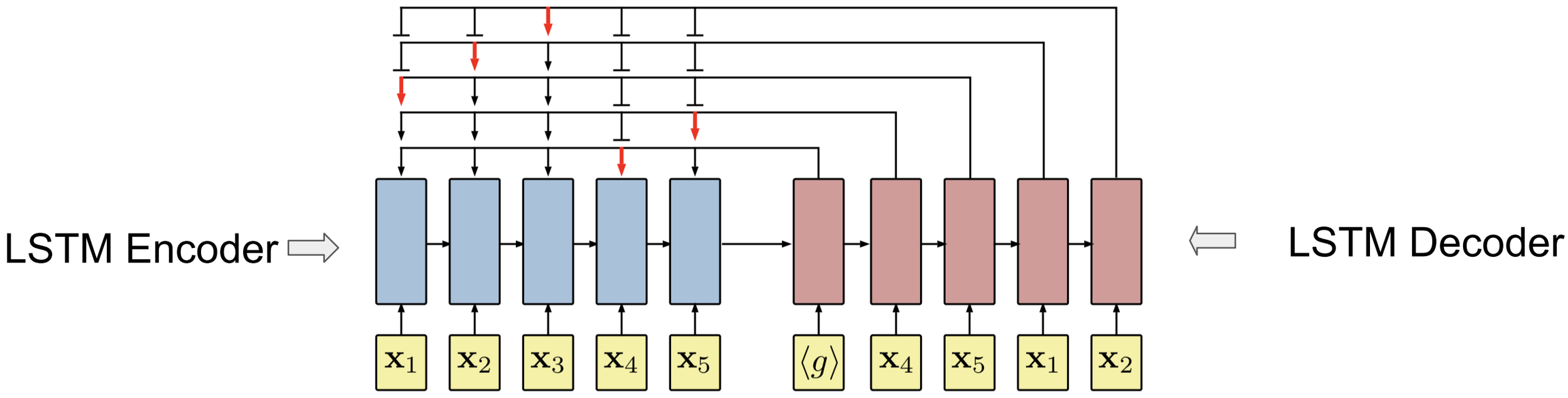
$$solution \sim P_{\theta^*}(.|problem)$$

Previous work: Neural Combinatorial Optimization with RL

Bello et al. "Neural combinatorial optimization with reinforcement learning."(2016).

- Policy model: Pointer Network

$$P_\theta(solution|problem)$$



Pointer network(Vinyals et al.,2015b)

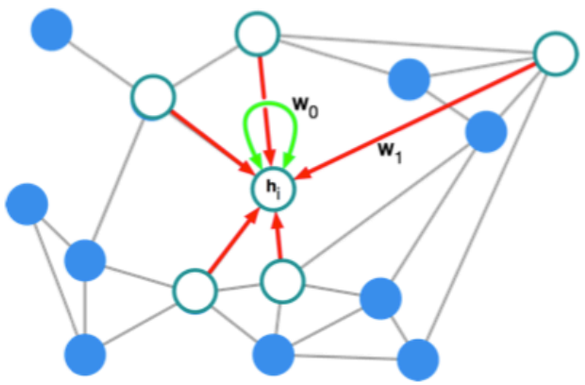
Learn heuristics for routing problems

Kool et all. "Attention, learn to solve routing problems!." ICLR. 2019.

- Policy model: Pointer Network

$$P_\theta(solution|problem)$$

Graph convolutions



+

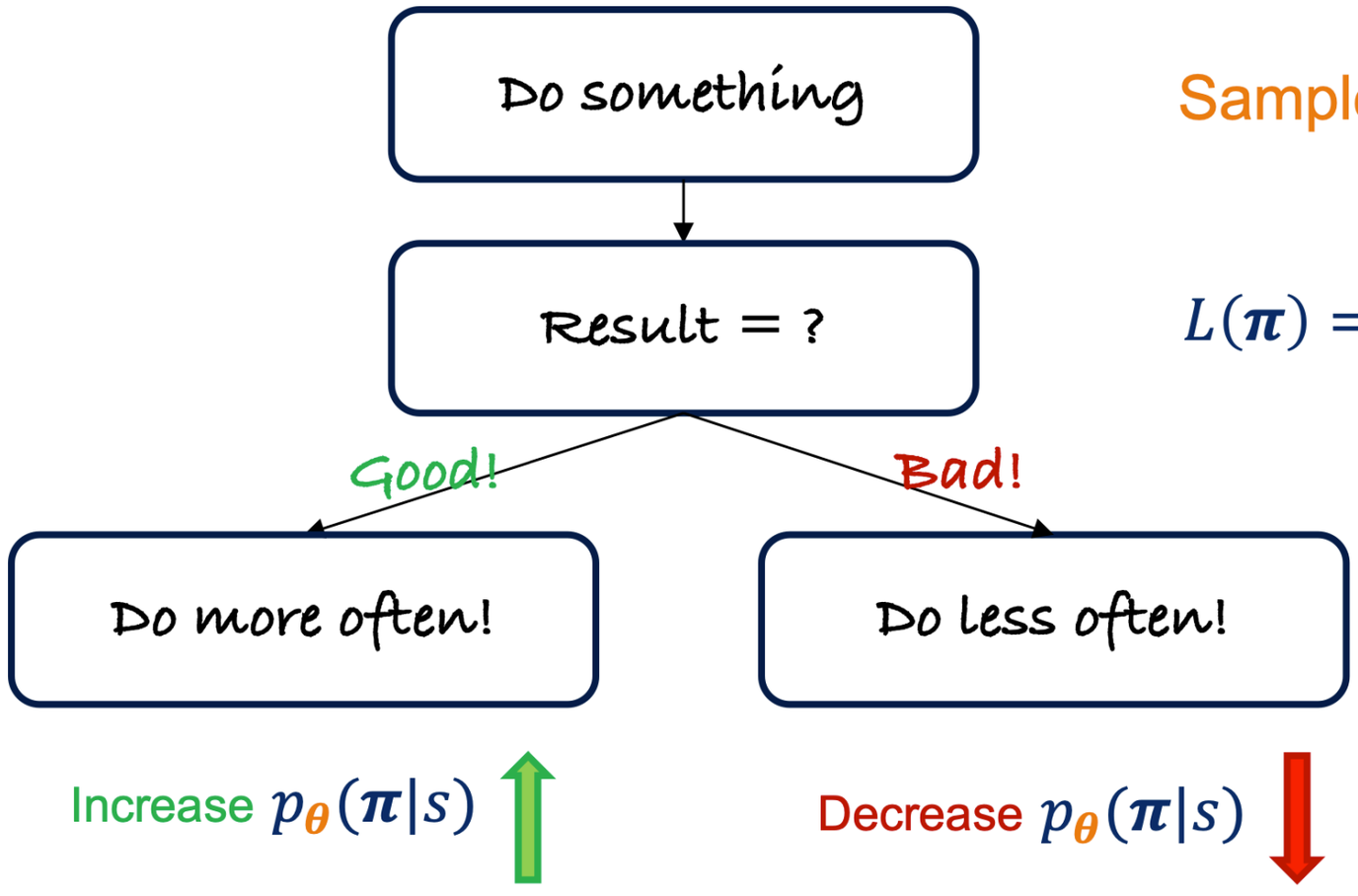
Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Learn heuristics for routing problems

Kool et all. "Attention, learn to solve routing problems!." ICLR. 2019.

- Learning mechanism: Reinforce with rollout baseline



Sample $\pi \sim p_\theta(\cdot\,|s)$

$L(\pi) = 7.43$

We need a *baseline* to compare against: *rollout* earlier model

$\pi^{bl} \sim p_{\theta^{bl}}(\cdot\,|s)$ **(greedy!)**

$L(\pi^{bl}) = 6.89$

Learn heuristics for routing problems

Kool et all. "Attention, learn to solve routing problems!." ICLR. 2019.

- Results: GNNs-Attention model + rollout baseline
  - Improves over classical heuristics!
  - Improves over prior learned heuristics!
    - Attention Model improves
    - Rollout helps significantly
- Gets close to single-purpose SOTA (20 to 100 nodes)!
  - TSP 0.34% to 4.53% (greedy)
  - TSP 0.08% to 2.26% (best of 1280 samples)

Learn to branch

Gasse, et al. "Exact combinatorial optimization with graph convolutional neural networks." Neurips 2019

Learn to branch

Gasse, et al. "Exact combinatorial optimization with graph convolutional neural networks." Neurips 2019

- Mixed-Integer Linear Program (MILP)
-

$$\begin{aligned} \arg\min_{\mathbf{x}} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \\ & \mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p}. \end{aligned}$$

- $\mathbf{c} \in \mathbb{R}^n$ the objective coefficients
- $\mathbf{A} \in \mathbb{R}^{m \times n}$ the constraint coefficient matrix
- $\mathbf{b} \in \mathbb{R}^m$ the constraint right-hand-sides
- $\mathbf{l}, \mathbf{u} \in \mathbb{R}^n$ the lower and upper variable bounds
- $p \leq n$ integer variables

https://github.com/gasse/gasse.github.io/blob/master/files/NeurIPS_2019_learn2branch_slides.pdf

Learn to branch

Gasse, et al. "Exact combinatorial optimization with graph convolutional neural networks." Neurips 2019

- Linear Relaxation

$$\begin{aligned} \arg\min_{\mathbf{x}} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \\ & \mathbf{x} \in \mathbb{R}^n. \end{aligned}$$

Convex problem, efficient algorithms (e.g., simplex).

- $\mathbf{x}^\star \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$ (lucky) $\rightarrow$ solution to the original MILP
- $\mathbf{x}^\star \notin \mathbb{Z}^p \times \mathbb{R}^{n-p} \rightarrow$ lower bound to the original MILP

https://github.com/gasse/gasse.github.io/blob/master/files/NeurIPS_2019_learn2branch_slides.pdf

Learn to branch

Gasse, et al. "Exact combinatorial optimization with graph convolutional neural networks." Neurips 2019

- Linear Relaxation

$$\begin{aligned} \arg\min_{\mathbf{x}} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \\ & \mathbf{x} \in \mathbb{R}^n. \end{aligned}$$

Convex problem, efficient algorithms (e.g., simplex).

- $\mathbf{x}^\star \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$ (lucky) $\rightarrow$ solution to the original MILP
- $\mathbf{x}^\star \notin \mathbb{Z}^p \times \mathbb{R}^{n-p} \rightarrow$ lower bound to the original MILP

https://github.com/gasse/gasse.github.io/blob/master/files/NeurIPS_2019_learn2branch_slides.pdf

Learn to branch

Gasse, et al. "Exact combinatorial optimization with graph convolutional neural networks." Neurips 2019

- Branch & Bound

Split the LP recursively over a non-integral variable, i.e. $\exists i \leq p \mid x_i^\star \notin \mathbb{Z}$

$$x_i \leq \lfloor x_i^\star \rfloor \quad \vee \quad x_i \geq \lceil x_i^\star \rceil.$$

Lower bound (L): minimal among leaf nodes.
Upper bound (U): minimal among leaf nodes with integral solution.

Stopping criterion:

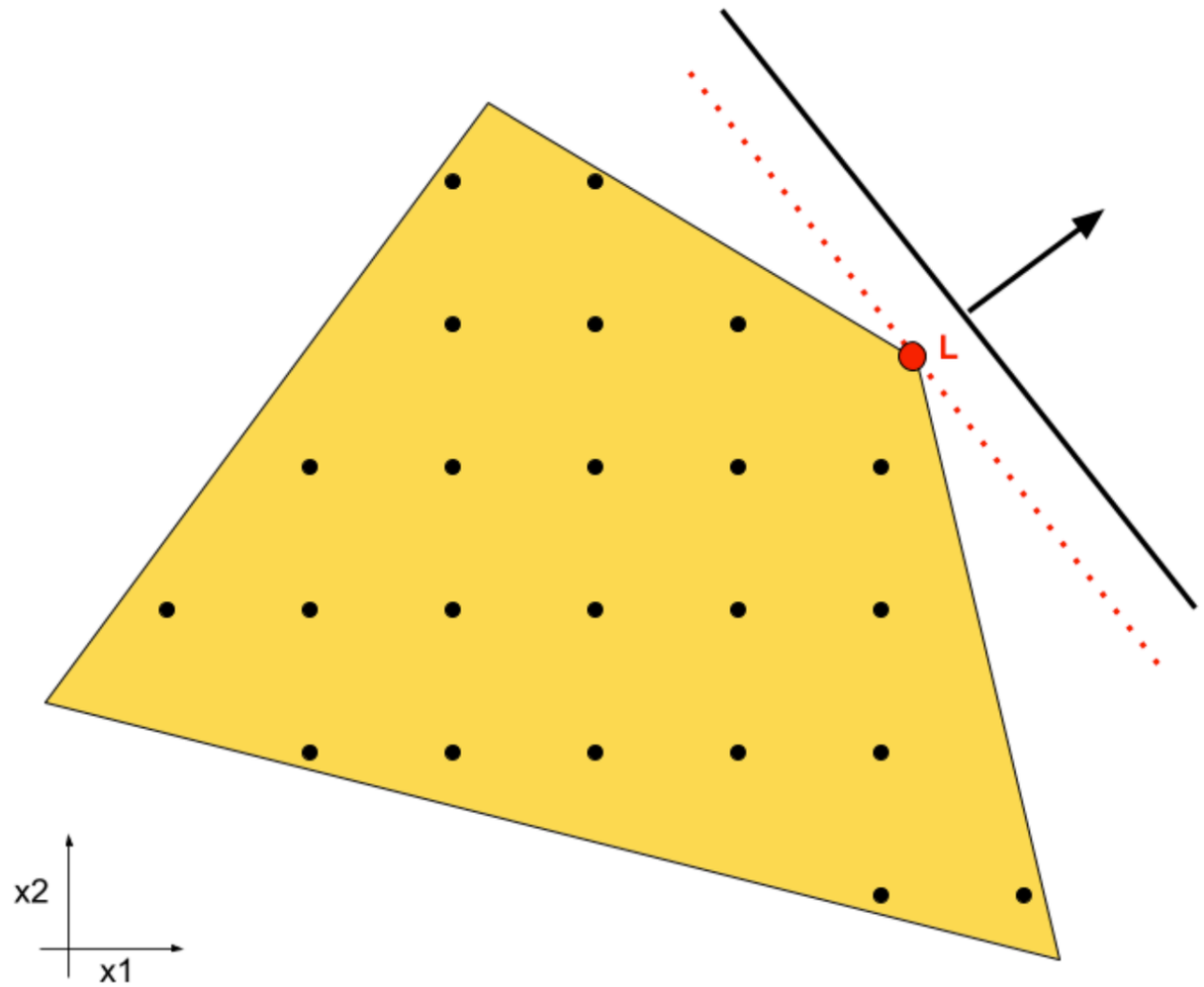
- L = U (optimality certificate)
- L = ∞ (infeasibility certificate)
- L - U < threshold (early stopping)

https://github.com/gasse/gasse.github.io/blob/master/files/NeurIPS_2019_learn2branch_slides.pdf

Learn to branch

Gasse, et al. "Exact combinatorial optimization with graph convolutional neural networks." Neurips 2019

- Branch & Bound
-

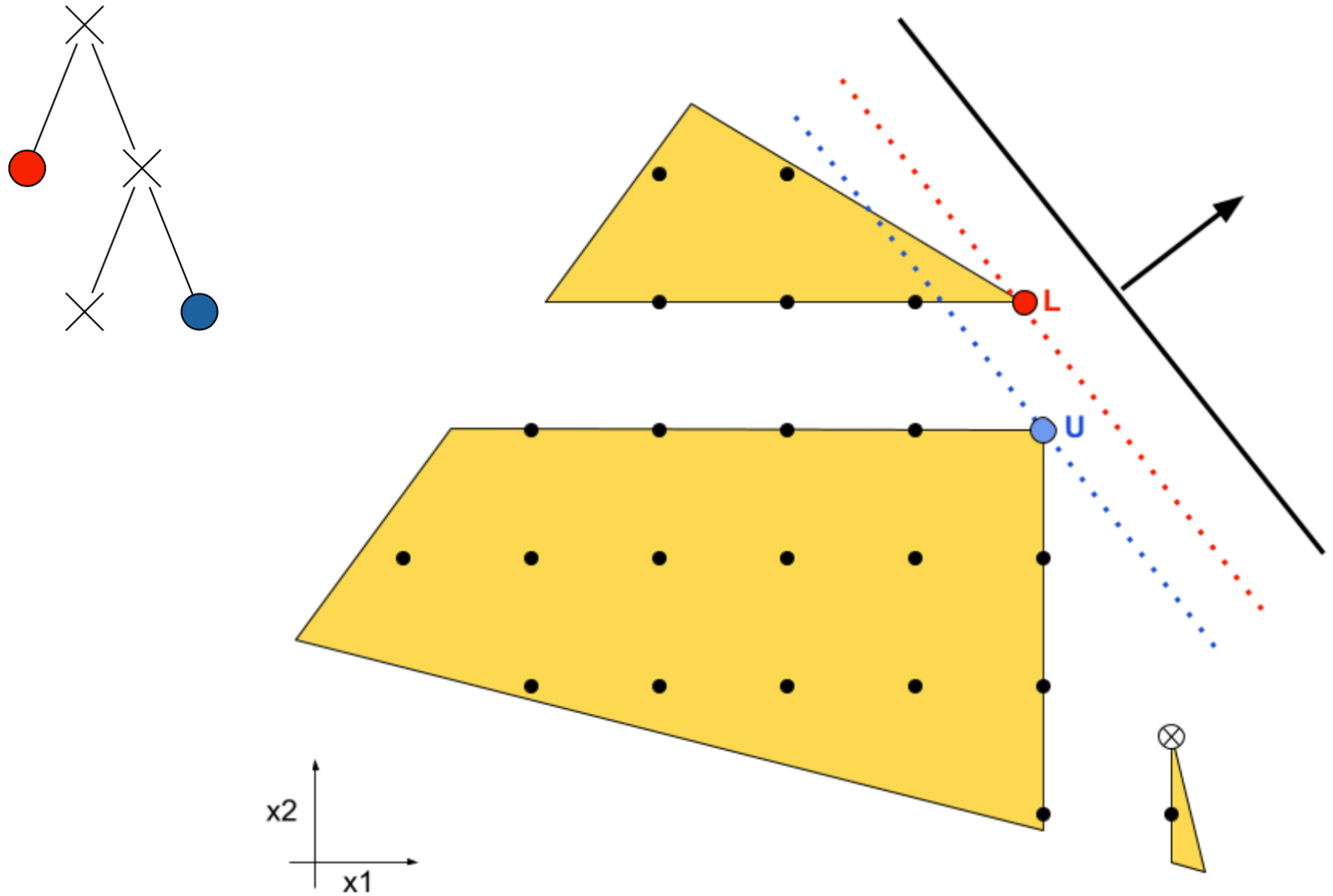


https://github.com/gasse/gasse.github.io/blob/master/files/NeurIPS_2019_learn2branch_slides.pdf

Learn to branch

Gasse, et al. "Exact combinatorial optimization with graph convolutional neural networks." Neurips 2019

- Branch & Bound
-

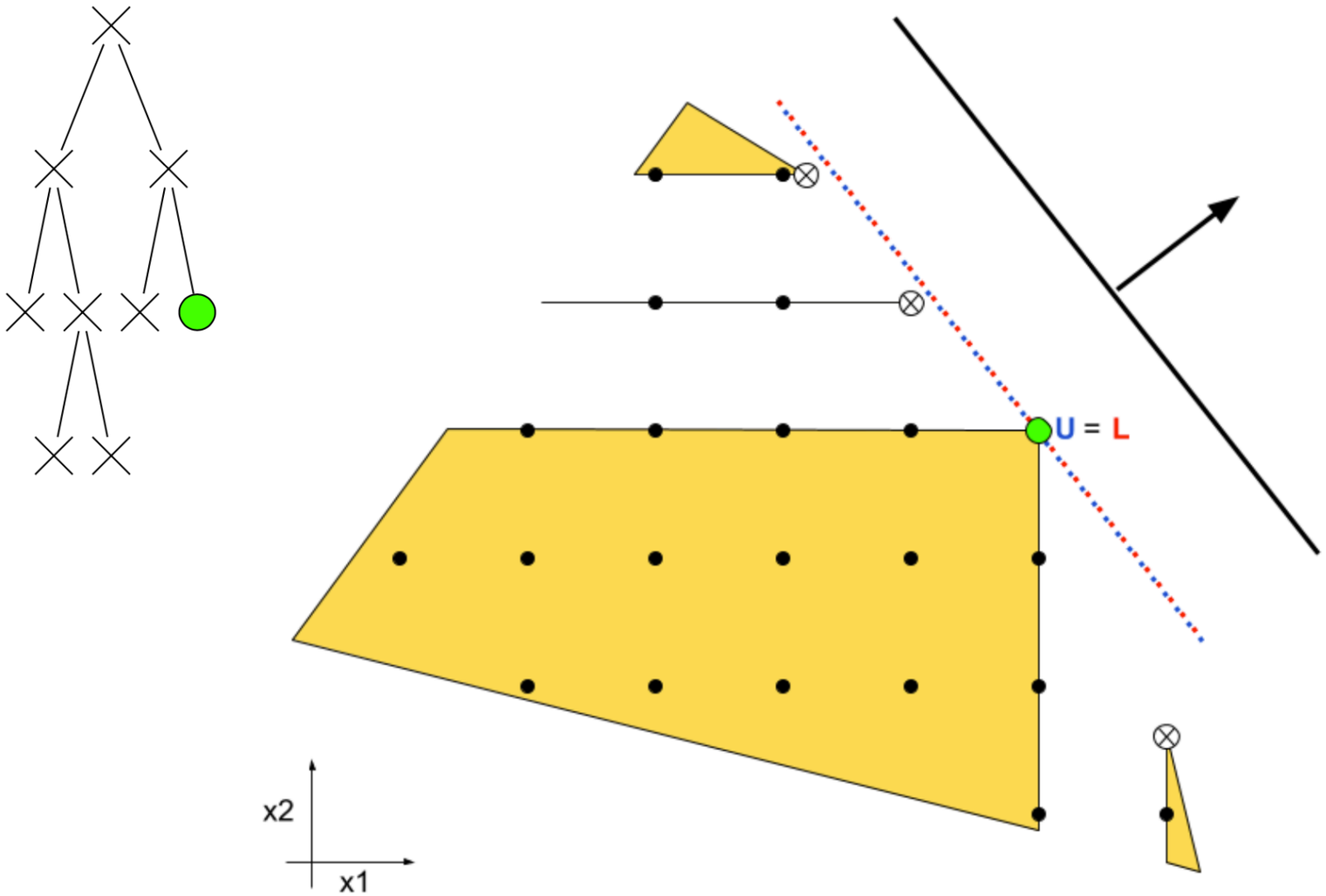


https://github.com/gasse/gasse.github.io/blob/master/files/NeurIPS_2019_learn2branch_slides.pdf

Learn to branch

Gasse, et al. "Exact combinatorial optimization with graph convolutional neural networks." Neurips 2019

- Branch & Bound
-



https://github.com/gasse/gasse.github.io/blob/master/files/NeurIPS_2019_learn2branch_slides.pdf

Learn to branch

Gasse, et al. "Exact combinatorial optimization with graph convolutional neural networks." Neurips 2019

- Expert branching rules : state-of-the-art

Strong branching: one-step forward looking (greedy)

- ▶ solve both LPs for each candidate variable
- ▶ pick the variable resulting in tightest relaxation
- \+ small trees
- − computationally expensive

Pseudo-cost: backward looking

- ▶ keep track of tightenings in past branchings
- ▶ pick the most promising variable
- \+ very fast, almost no computations
- − cold start

Reliability pseudo-cost: best of both worlds

- ▶ compute SB scores at the beginning
- ▶ gradually switches to pseudo-cost (+ other heuristics)
- \+ best overall solving time trade-off (on MIPLIB)

https://github.com/gasse/gasse.github.io/blob/master/files/NeurIPS_2019_learn2branch_slides.pdf

Learn to branch

Gasse, et al. "Exact combinatorial optimization with graph convolutional neural networks." Neurips 2019

- Imitation learning

Full Strong Branching (FSB): good branching rule, but expensive.
Can we learn a fast, good-enough approximation ?

Behavioural cloning

- collect $\mathcal{D} = \{(\mathbf{s}, a^\star), \dots\}$ from the expert agent (FSB)
- estimate $\pi^\star(a \,|\, \mathbf{s})$ from $\mathcal{D}$

\+ no reward function, supervised learning, well-behaved

\− will never surpass the expert...

https://github.com/gasse/gasse.github.io/blob/master/files/NeurIPS_2019_learn2branch_slides.pdf

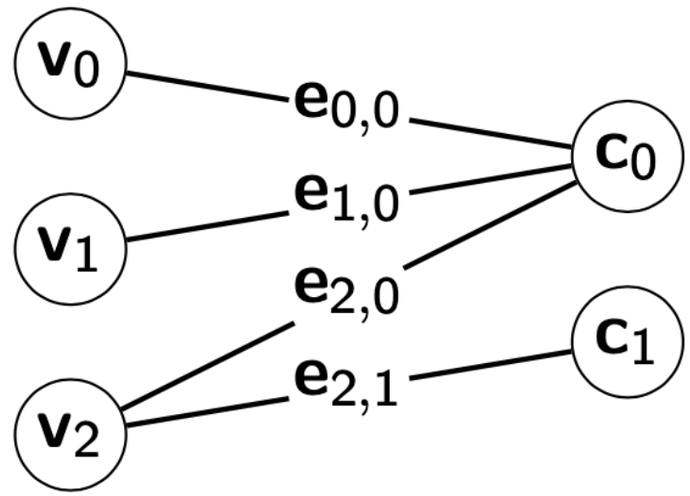
Learn to branch

Gasse, et al. "Exact combinatorial optimization with graph convolutional neural networks." Neurips 2019

- GNNs model

Natural representation : variable / constraint bipartite graph

$$\begin{aligned} \arg\min_{\mathbf{x}} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \\ & \mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p}. \end{aligned}$$



- $\mathbf{v}_i$: variable features (type, coef., bounds, LP solution...)
- $\mathbf{c}_j$: constraint features (right-hand-side, LP slack...)
- $\mathbf{e}_{i,j}$: non-zero coefficients in $\mathbf{A}$

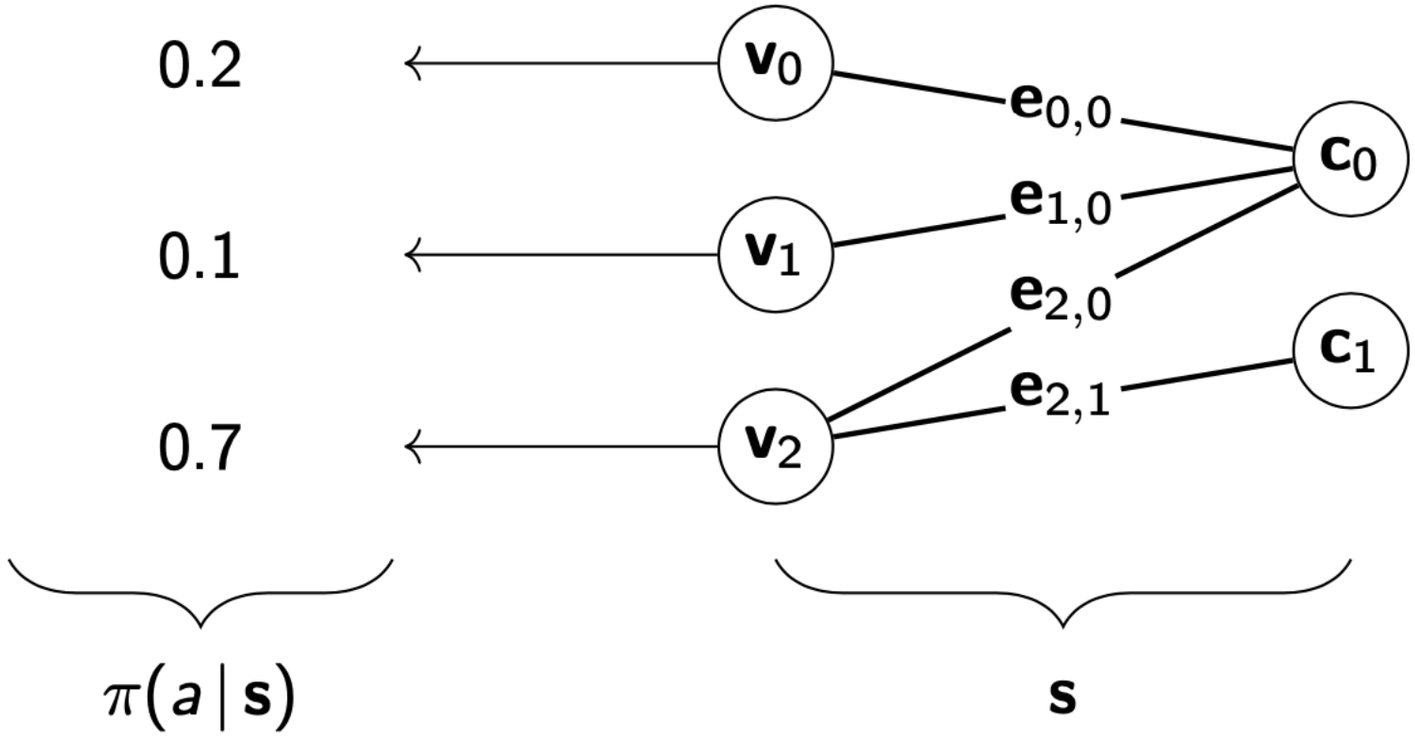
https://github.com/gasse/gasse.github.io/blob/master/files/NeurIPS_2019_learn2branch_slides.pdf

Learn to branch

Gasse, et al. "Exact combinatorial optimization with graph convolutional neural networks." Neurips 2019

- GNNs model

Neighbourhood-based updates: $\mathbf{v}_i \leftarrow \sum_{j \in \mathcal{N}_i} \mathbf{f}_\theta(\mathbf{v}_i, \mathbf{e}_{i,j}, \mathbf{c}_j)$



https://github.com/gasse/gasse.github.io/blob/master/files/NeurIPS_2019_learn2branch_slides.pdf

Learn to branch

Gasse, et al. "Exact combinatorial optimization with graph convolutional neural networks." Neurips 2019

- Experiment (minimum set covering)

	Easy			Medium			Hard		
Model	Time	Wins	Nodes	Time	Wins	Nodes	Time	Wins	Nc
FSB	17.30	0 / 100	17	411.34	0 / 90	171	3600.00	0 / 0	n
RPB	8.98	0 / 100	**54**	60.07	0 / 100	1741	1677.02	4 / 65	47
XTrees	9.28	0 / 100	187	92.47	0 / 100	2187	2869.21	0 / 35	59
SVMrank	8.10	1 / 100	165	73.58	0 / 100	1915	2389.92	0 / 47	42
λ-MART	7.19	14 / 100	167	59.98	0 / 100	1925	2165.96	0 / 54	45
GCNN	**6.59**	**85** / 100	134	**42.48**	**100** / 100	**1450**	**1489.91**	**66** / 70	**29**

3 problem sizes

- 500 rows, 1000 cols (easy), training distribution
- 1000 rows, 1000 cols (medium)
- 2000 rows, 1000 cols (hard)

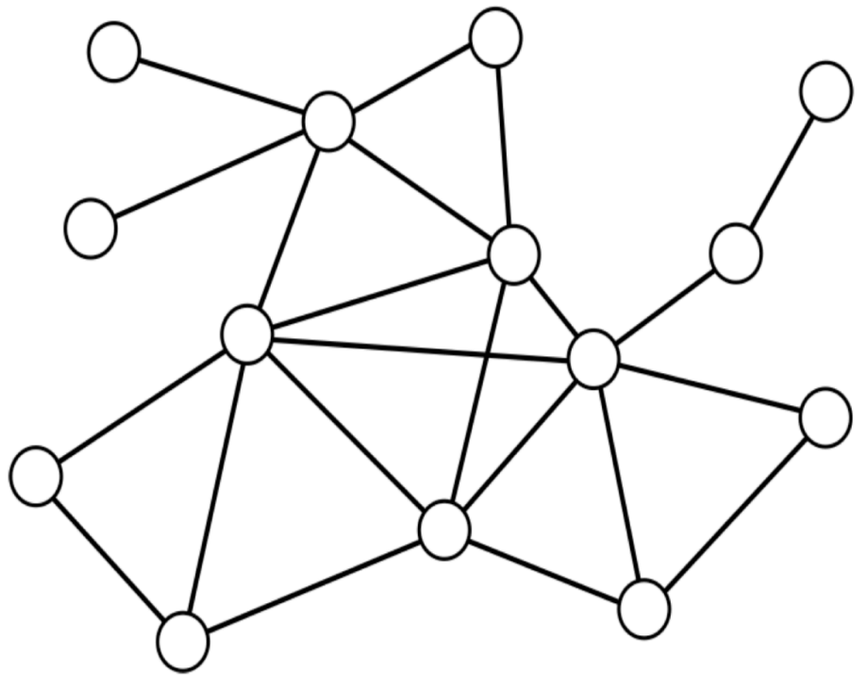
https://github.com/gasse/gasse.github.io/blob/master/files/NeurIPS_2019_learn2branch_slides.pdf

Learn to build chordal graphs

Liu et all. "Learning chordal extensions." arXiv preprint arXiv:1910.07600 (2019).

Chordal Graphs

A *chordal graph* is an undirected graph with the property that every cycle of length greater than three has a chord (an edge between nonconsecutive vertices in the cycle) [*Vandenberghe et al., 2015*].

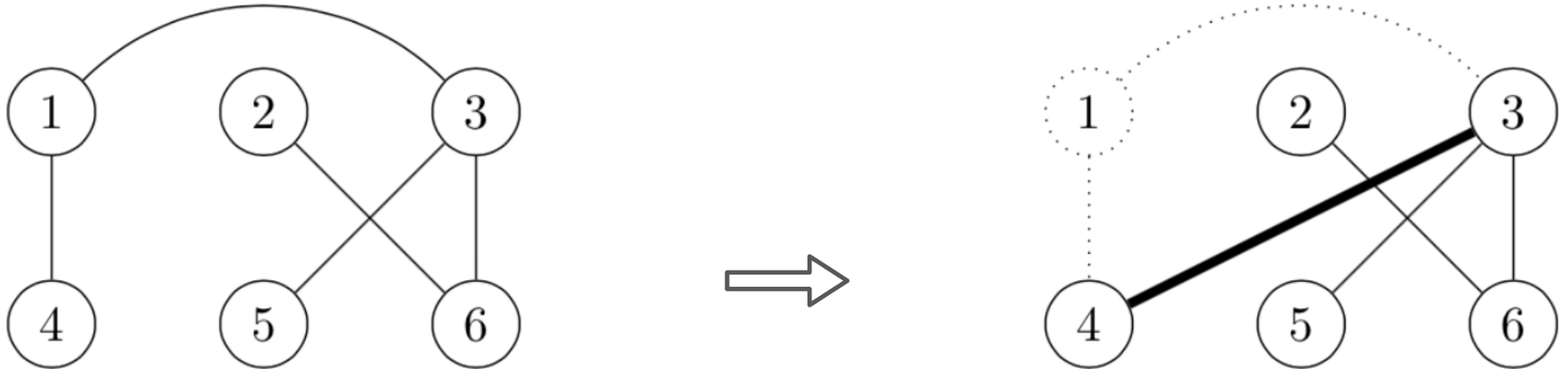


example: a chordal graph

Applications: semidefinite optimization, probabilistic graphical models, dynamic programming.

Graph Elimination and Chordal Extensions

Any undirected graph can be converted to a *chordal graph* by *graph elimination*. The resulting graph is known as *chordal extension.*



An example of graph elimination. Consider an undirected graph G=(V, E), and a vertex v ∈ V. The elimination graph is obtained by removing vertex v and its incident edges from G, and then adding all edges (w, w') where w and w' were adjacent to v in G.

Motivation

Given a sparse linear system

$$\Phi x = b$$

If Φ is symmetric and positive definite, we can compute its **Cholesky factorization**

$$\Phi = LL^T$$


André-Louis Cholesky

Then the original problem is solved by

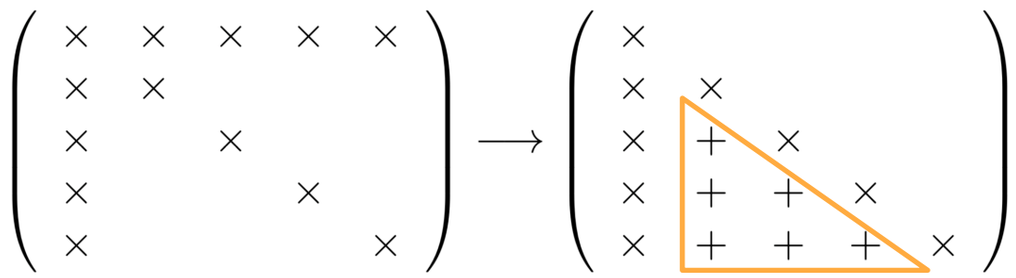
$$z = L^{-1}b,$$
$$x = L^{-T}z.$$

Motivation

Cholesky factorization:

$$\Phi = LL^T$$

:(Cholesky factor L might suffer from significant **fill-in**



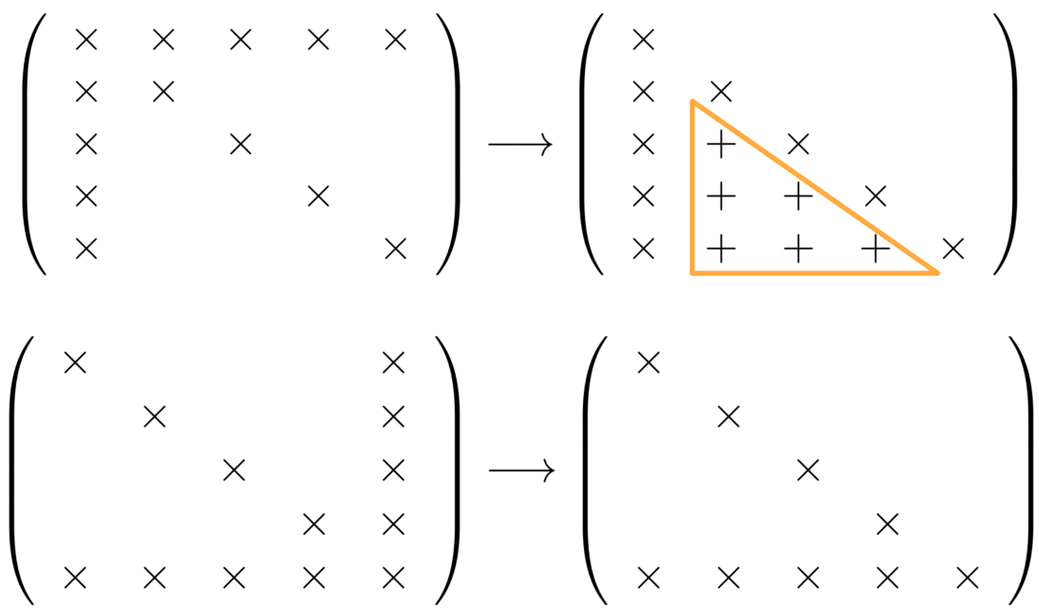
Motivation

Cholesky factorization:

$$\Phi = LL^T$$

:(Cholesky factor L might suffer from significant **fill-in**

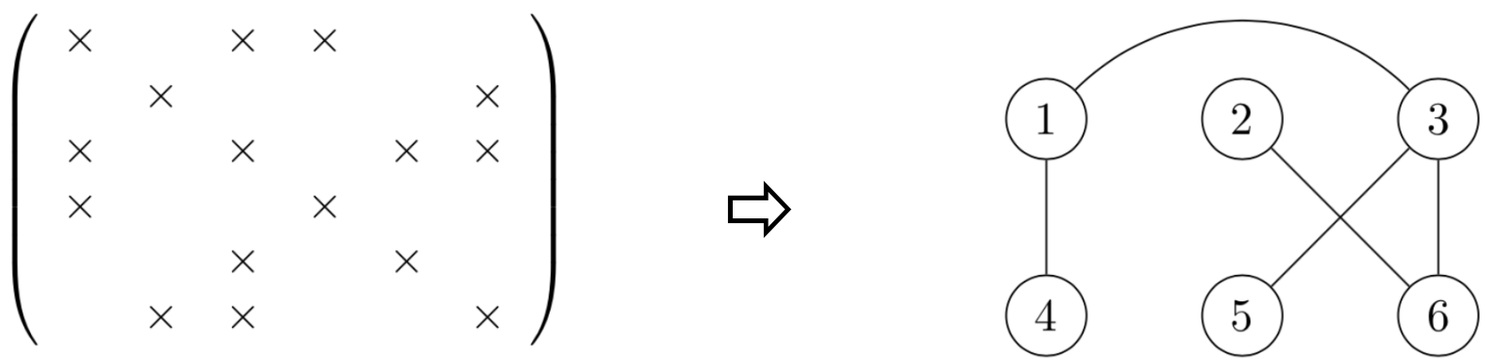
:) Fill-in can be reduced by re-ordering the rows and columns of the matrix



Motivation

For any symmetric matrix Φ, its ordered graph is defined as

$$V^{\Phi} = \{1, \ldots, n\},$$
$$E^{\Phi} = \{(i,j) | \Phi_{i,j} \neq 0, i \neq j\}$$

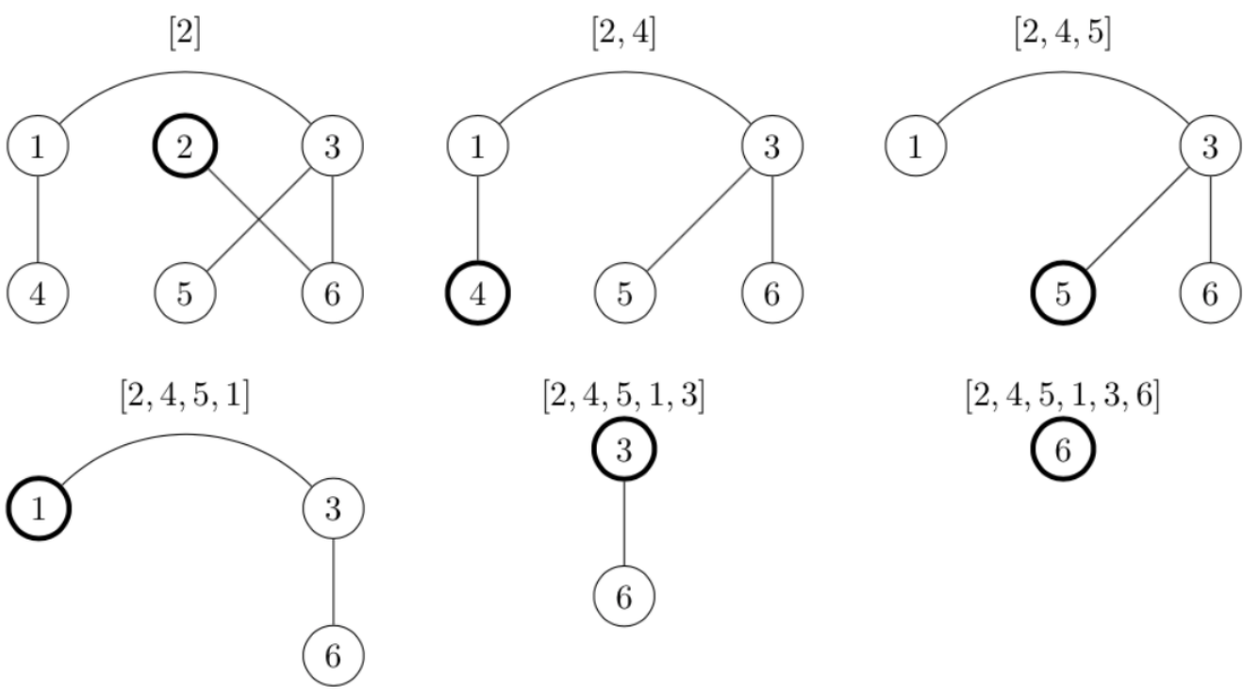


Objective: find an ordering that minimizes fill-in. (NP-complete hard)

Ordering Heuristics

Fill-reducing Heuristics: *minimum degree, nested dissection, and band reduction [Davis, 2006].*

Minimum degree : choosing a vertex of minimum degree to be eliminated



Can we leverage ML to learn a better ordering heuristic?

ML Framework for learning heuristics

- Representation (Policy):

$$P_\theta(solution|problem)$$

- Learning:
  - learn θ from a dataset
    - imitation learning
    - reinforcement learning

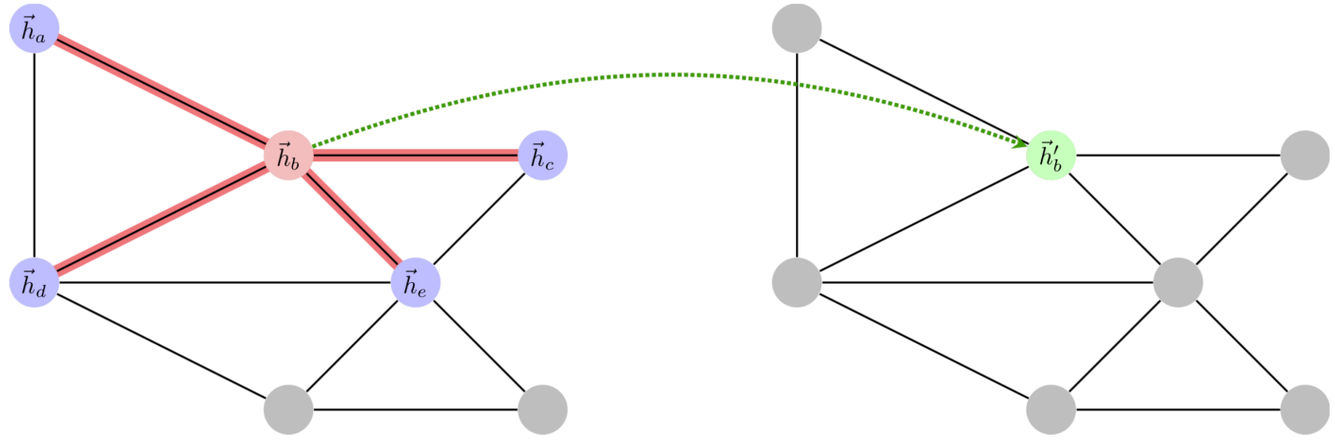
- Inference: sample solutions from

$$solution \sim P_{\theta*}(.|problem)$$

Learning Setting: Representation

- GNNs Policy Function Approximation:

$$P_\theta(solution|problem)$$



Learning Setting: Imitation Learning

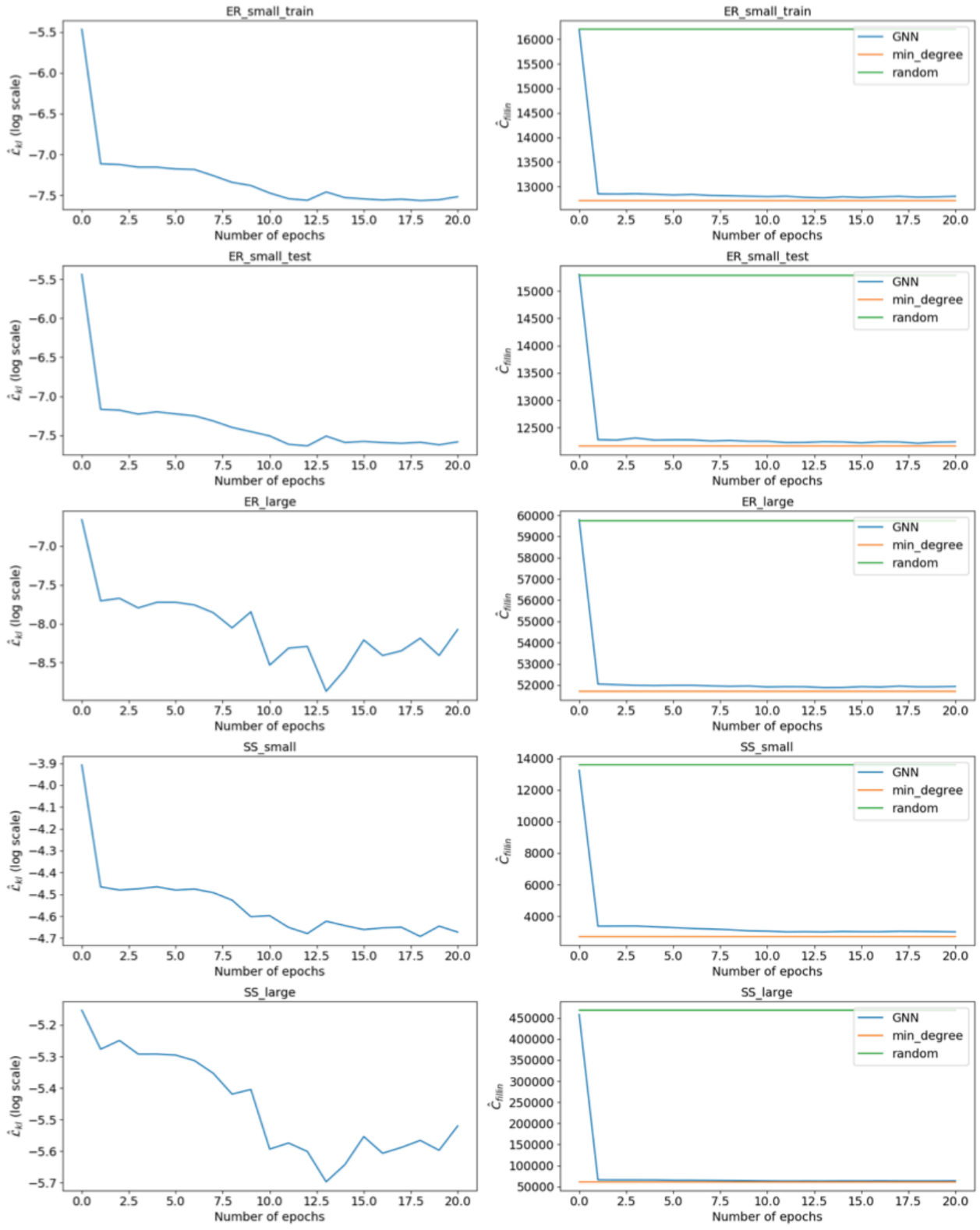
- Supervised Learning
  - learn from dataset(problem, label solution), so need labels!
  - training : Adjust θ by maximizing the likelihood of the label solutions

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, P_\theta(label\ solution|problem)$$

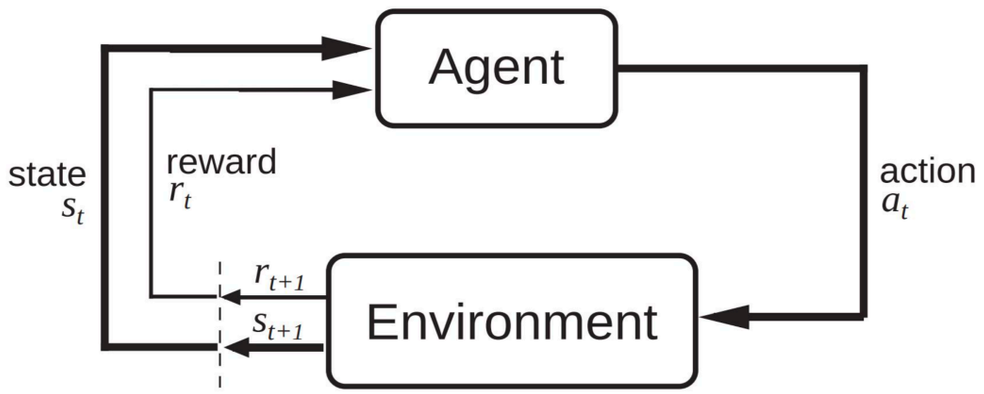
- Idea: can the policy model do the same as the heuristics?
  - Imitation learning:
    - get labels from an heuristic expert
    - train the policy model with it iteratively
  - Experimental set-up
    - 200 Erdos-Renyi random graphs
    - heuristic experts: minimum degree

Preliminary Results:

- Imitation Learning
  - Training:
    - ER graphs small(100-200)
  - Test:
    - ER graphs small(100-200)
    - ER graphs large(1000-2000)
    - Realistic SS dataset (50-500)
    - Realistic SS dataset (1000-2000)



Learning Setting: Reinforcement Learning(RL)

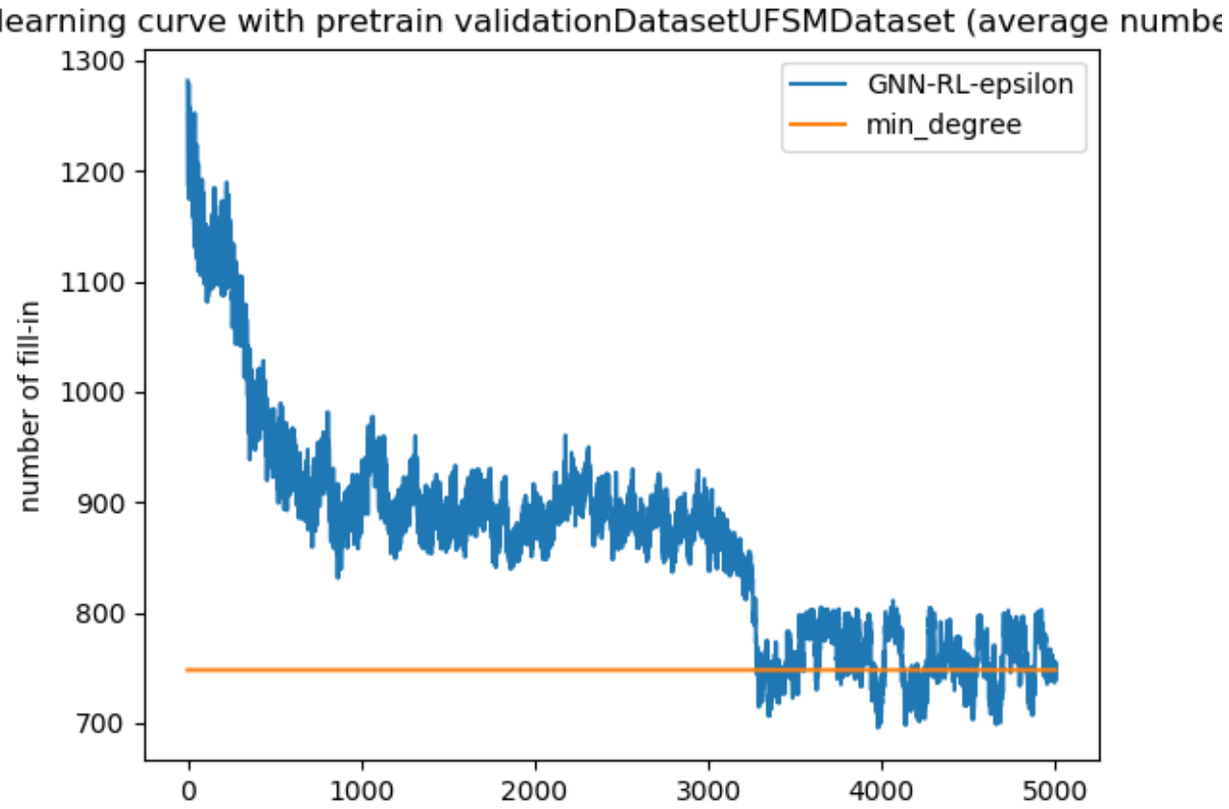


Reinforcement learning framework by Sutton and Barto(1998)

- MDP Formulation:
  - ***State space*** : set of undirected graphs
  - ***Action space*** : the set of vertices
  - ***Policy*** : maps a graph G = (V, E) to a vertex v ∈ V to be eliminated.
  - ***Reward/Return(r)***: the number of edges inserted (flexible)

Preliminary Results:

- RL with Pre-training

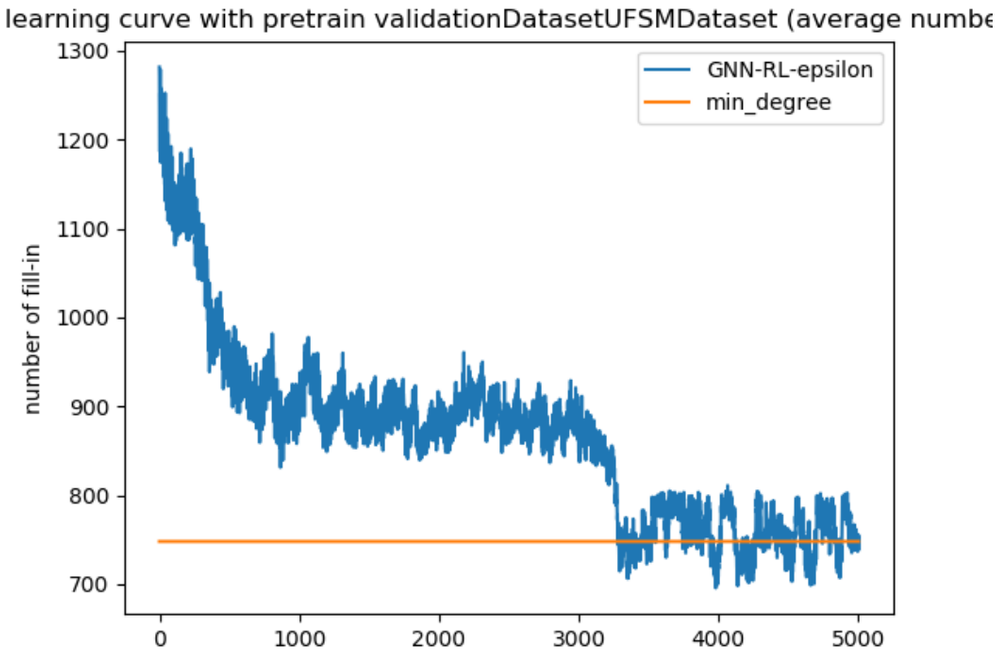


RL-PolicyGradient training over
SS dataset

Discussion

- Why does policy model converges to that local min:
  - flat local min landscape of policy function
  - not enough exploration
  - not enough model capacity

- Next step:
  - RL refinement: more exploration to find better local min
  - analysis of model capacity to get theoretical bounds
  - learning a critic function to reduce variance
    - need to design suitable policy evaluation networks



RL-PolicyGradient training over *SS* dataset

Summary

Applications of GNNs on Combinatorial Optimization:

- Graph/network optimization problems
  - *Khalil, et al. "Learning combinatorial optimization algorithms over graphs." NIPS. 2017*
  - *Kool, et all. "Attention, learn to solve routing problems!." ICLR. 2019.*
  - *Liu, et all. "Learning chordal extensions." arXiv preprint arXiv:1910.07600 (2019).*

- General mixed integer linear programming problems
  - *Gasse, et al. "Exact combinatorial optimization with graph convolutional neural networks." Neurips 2019*
  - *Ding, et al. "Accelerating Primal Solution Findings for Mixed Integer Programs Based on Solution Prediction." arXiv preprint arXiv:1906.09575(2019)*

Thank you for your attention!