Machine Learning for Combinational Optimization

@GeminiLight

2021/09/03

Contents

ML for CO

Combinatorial Optimization

Various Applications

Non-European Structure

Complex Constrains

Considerable Variants

2.1 Graph Matching (GM)	2.2 Quadratic Assignment Problem (QAP)
2.3 Travelling Salesman Problem (TSP)	2.4 Maximal Cut
2.5 Vehicle Routing Problem (VRP)	2.6 Computing Resource Allocation
2.7 Job Shop Scheduling Problem (JSSP)	2.8 Bin Packing Problem (BPP)
2.9 Graph Edit Distance (GED)	2.10 Graph Coloring
2.11 Maximal Common Subgraph (MCS)	2.12 Influence Maximization
2.13 Maximal Independent Set (MIS)	2.14 Mixed Integer Programming
2.15 Causal Discovery	2.16 Game Theoretic Semantics
2.17 Boolean Satisfiability (SAT)	2.18 Differentiable Optimization
2.19 Car Dispatch	

Image by Thinklab-SJTU from Github

Travelling Salesman Problem (TSP)

Given

A list of cities and the distances between each pair of cities

Task

Find an route to return to the origin city

Constraint

Each city is visited exactly once

Objective

Minimizes the tour length

Formulation

$$min \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} c_{ij} x_{ij}.$$

$$s.t. \ 0 \leq x_{ij} \leq 1, i, j = 0, \cdots, n$$

$$u_i \in Z, i = 0, \cdots, n.$$

$$\sum_{i=0, i \neq j}^{n} x_{ij} = 1, j = 0, \cdots, n.$$

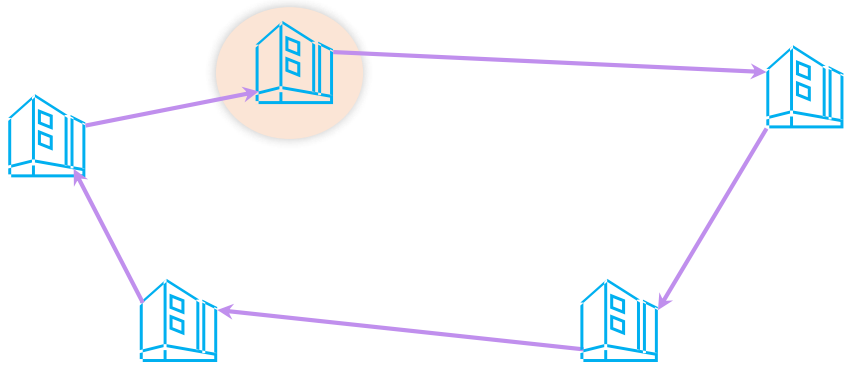
$$\sum_{j=0, j \neq i}^{n} x_{ij} = 1, i = 0, \cdots, n.$$

$$u_i - u_j + n x_{ij} \leq n - 1, 1 \leq i \neq j \leq n$$

c_{ij} the cost from i to j

x_{ij} a binary indicate whether i connect j or not

Example



Vehicle Routing Problem (VRP)

Given

A set of customers needing items
A depot where vehicles start and end

Task

Find routes to return to the origin city

Constraint

Each route starts and ends at the depot
Each customer is visited exactly once
Total demand <= Vehicle capacity

Objective

Minimize the total vehicle distance and the total number of vehicles used

Formulation

$$min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

c_{ij} the cost from i to j

x_{ij} a binary indicate whether i connect j or not

$$\sum_{i \in V} x_{ij} = 1 \forall j \in V \setminus \{0\}$$

$$\sum_{j \in V} x_{ij} = 1 \forall i \in V \setminus \{0\}$$

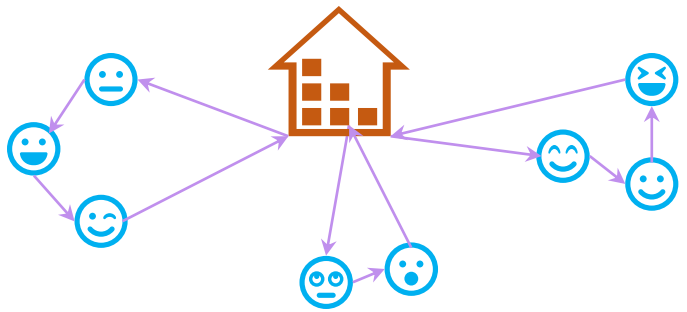
$$\sum_{i \in V} x_{i0} = K$$

$$\sum_{j \in V} x_{0j} = K$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq r(S), \forall S \subseteq V \setminus \{0\}, S \neq \emptyset$$

$$x_{ij} \in \{0,1\} \forall i, j \in V$$

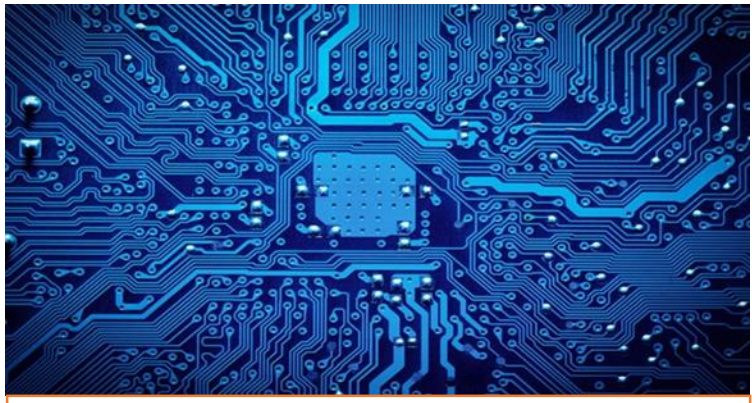
Example



Applications of TSP & VRP

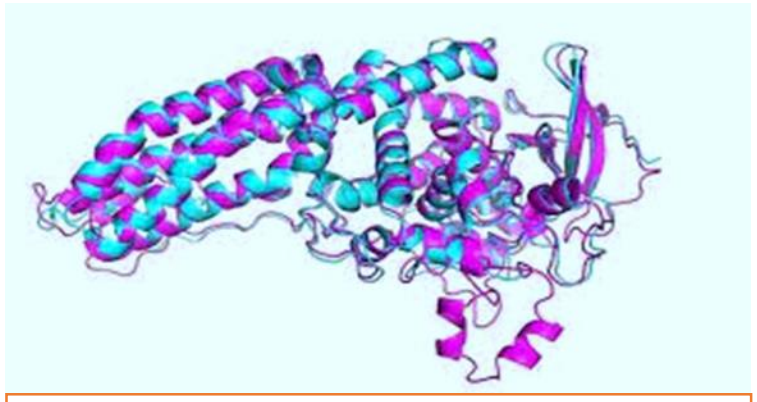
Energy transportation

Distribution logistics



Integrated circuit

Industrial production



Biological protein

Network management

*All images are from Google

Maximal Independent Set (MIS)

Given

A graph $G(V, E, w)$

Task

Find a subset of nodes $S \subseteq V$

Constraint

No two nodes are connected by an edge

Objective

Minimizes $|S|$

Formulation

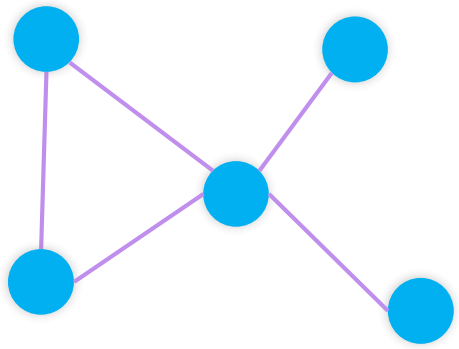
choice variables

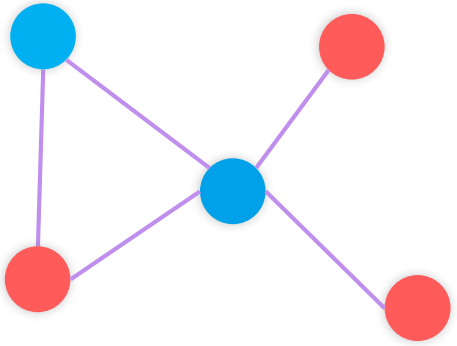
$x_v \in \{0,1\} \forall v \in V$

$$\text{maximize} \sum_{v \in V} w(v)x(v)$$

$$x(u) + x(v) \leq 1 (u, v) \in E$$
$$x(v) \in [0,1] v \in V$$

Example





Minimum vertex cover (MVC)

Given

A graph $G(V, E, w)$

Task

Find a subset of nodes $S \subseteq V$

Constraint

Every edge is covered
i.e. $(u, v) \in E \Leftrightarrow u \in S$ or $v \in S$

Objective

Minimizes $|S|$

Formulation

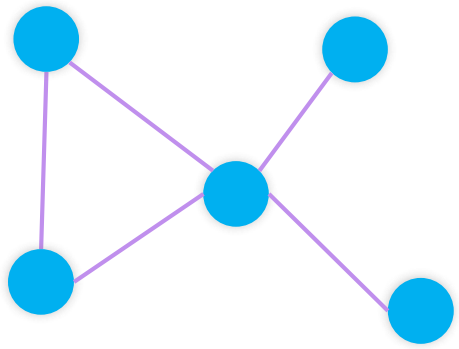
$$minimize \sum_{v \in V} x_v$$

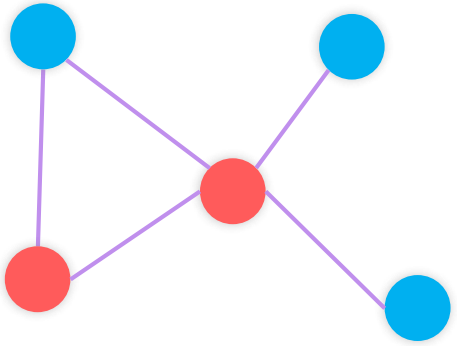
choice variables

$x_v \in \{0,1\} \forall v \in V$

$x_u + x_v \geq 1$ for each edge $\{u, v\} \in E$
$x_v \in \{0,1\}$ for each $v \in V$

Example





Maximum Cut (MAXCUT)

Given

A graph $G(V, E, w)$

Task

Find a subset of nodes $S \subseteq V$

Objective

Minimizes the weight of the cut-set $\sum_{(u,v)\in C} w(u, v)$

Cut-set $C \subseteq E$

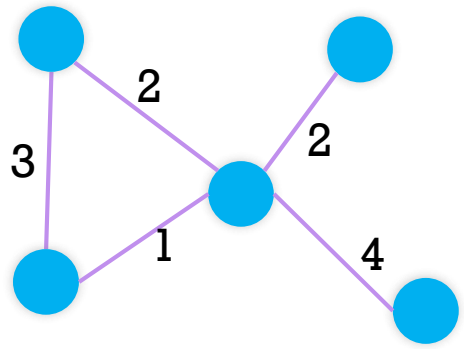
The set of edges with one end in S and the other end in V/S.

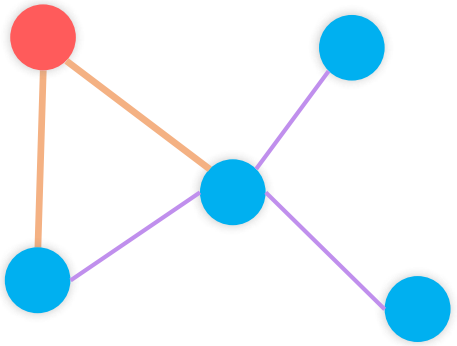
Formulation

$$Max \frac{1}{2} \sum_{1 \le i < j \le n} w_{ij}(1 - y_i y_j)$$

$$w_{ij} := \begin{cases} 1, & if (i,j) \in E \\ 0, & if (i,j) \notin E \end{cases} \qquad y_i \in \{-1, 1\}, \forall i \in V$$

Example





Job-shop Scheduling Problem (JSSP)

Given

A set of jobs J and a set of machines M.

Task

Find a subset of nodes $S \subseteq V$

Constraints

Each job J_i must go through n_i machines in M in a specific order

Objective

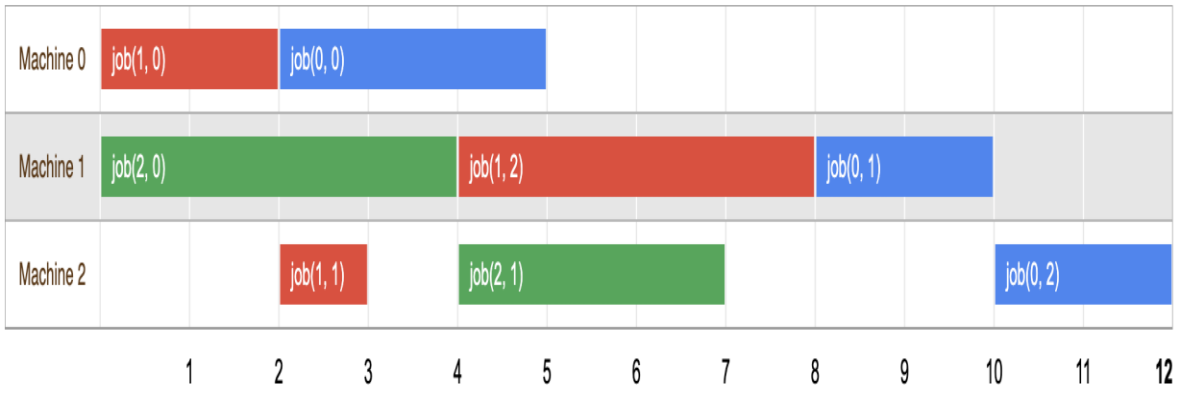
Minimizes the weight of the cut-set

$\sum_{(u,v)\in C} w(u,v)$

Formulation

Example

Image from Google



Quadratic Assignment Problem (QAP)

Given

Two sets of equal size
P ("facilities") and L ("locations")
A weight function $w: P \times P$
A distance function $d: L \times L$

Task

Find $P \rightarrow L$ ("assignment")

Objective

Minimizes the assignment cost

Formulation

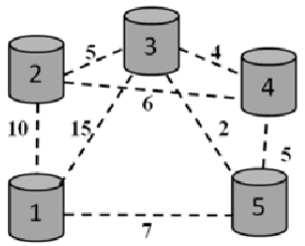
$$Min \sum_{a,b \in P} w(a,b) \cdot d(f(a), f(b))$$

$w(a,b)$ The weight between a and b

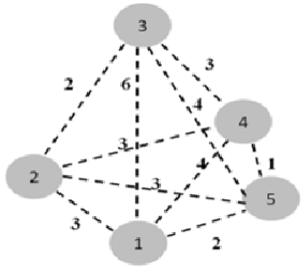
$d(f(a), f(b))$ The distance between $f(a)$ and $f(b)$

Example

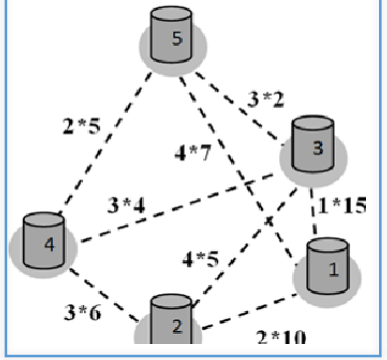
Facilities with weight



Locations with distance



Optimal distance



MIP & Branch and Bound

$$\min_x c^T x \quad \text{Objective function}$$

$$s.t. \quad Ax \leq b \quad \text{Linear constraints}$$

$$x_i \in \mathbb{Z} \quad \forall i \quad \text{Integrality constraint}$$

$$\max z = 4x_1 + 3x_2$$

$$\text{s. t.} \begin{cases} 3x_1 + 4x_2 \leqslant 12 \\ 4x_1 + 2x_2 \leqslant 9 \\ x_1, x_2 \geqslant 0 \\ x_1, x_2 \text{ 取整数} \end{cases}$$

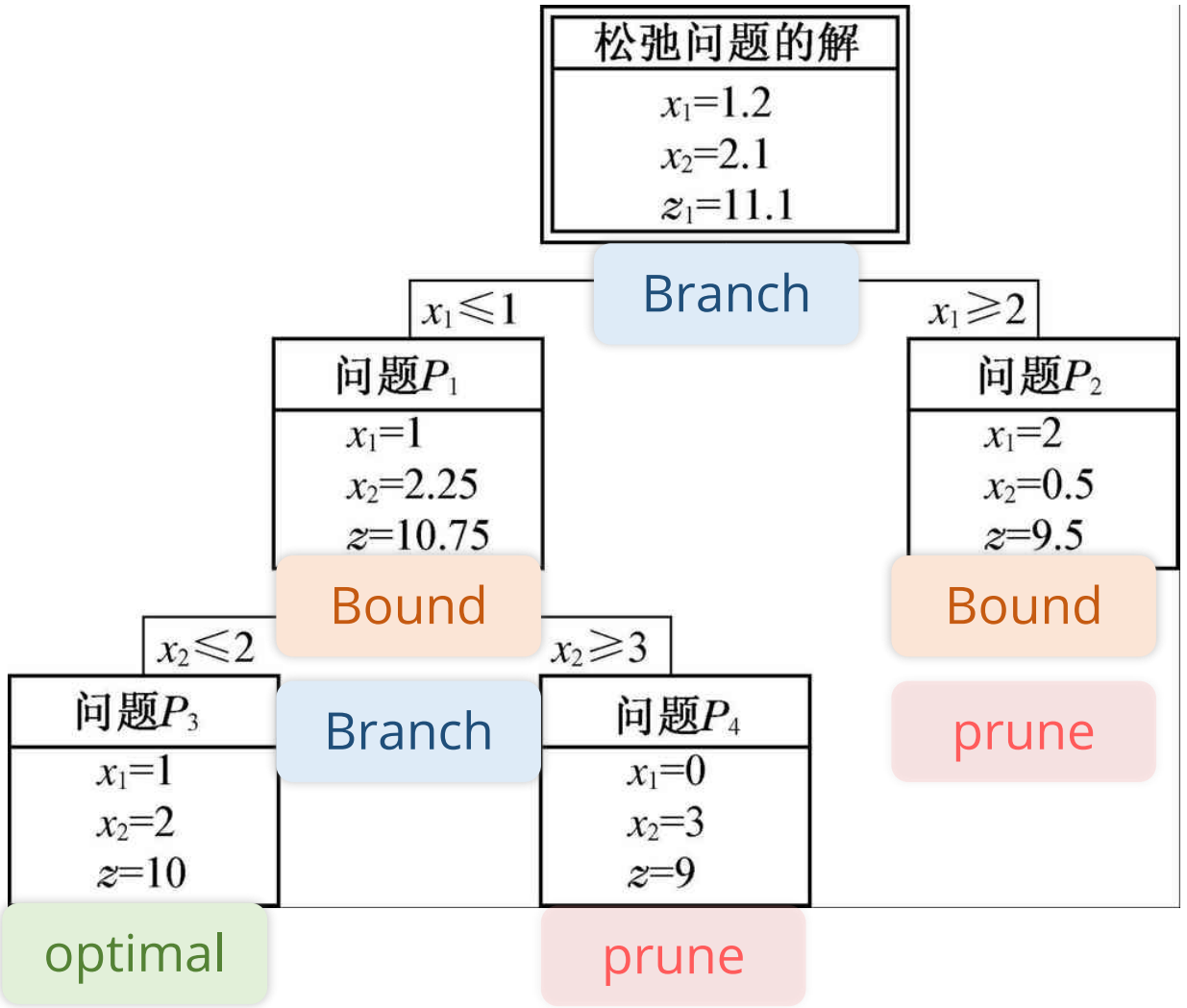


Image from http://www.tjxzj.net/1657.html

Combinatorial Optimization

The majority are NP-hard!

Various Applications	Economics	Industry	Management
	Communication	Biology	Healthcare
Non-European Structure	Directed / Undirected	Weight / Unweighted	
	Dynamic / Static	Bipartite	
Complex Constrains	Node-level (visit)	Edge-level (connect)	
	Pair-level (mapping)	Graph-level (structure)	
Considerable Variants	Offline & Online	Single & Multi	
	Optimization objective	Considered constrains	

Traditional Methods for COPs

Exact approach

PROS: Find the global optimal solution
CONS: Not applicable for large problems

Enumeration → to reduce the search depth

- heuristic search needs a heuristic function
- branch-and-bound needs a bounding function

Approximate approach

PROS: Find an approximate solution usually in polynomial time
CONS: Difficult to design appropriate strategies

Heuristic approach

PROS: Can find a suboptimal solution quickly
CONS:

- Usually rely on manual heuristics
- So targeted pertinence that it is difficult to generalize
- Hard to tune when and where to apply heuristics
- Requires significant specialized knowledge and trial-and-error

Meta-heuristic approach

PROS: Partly universalize and can jump from local optimization solution
CONS:

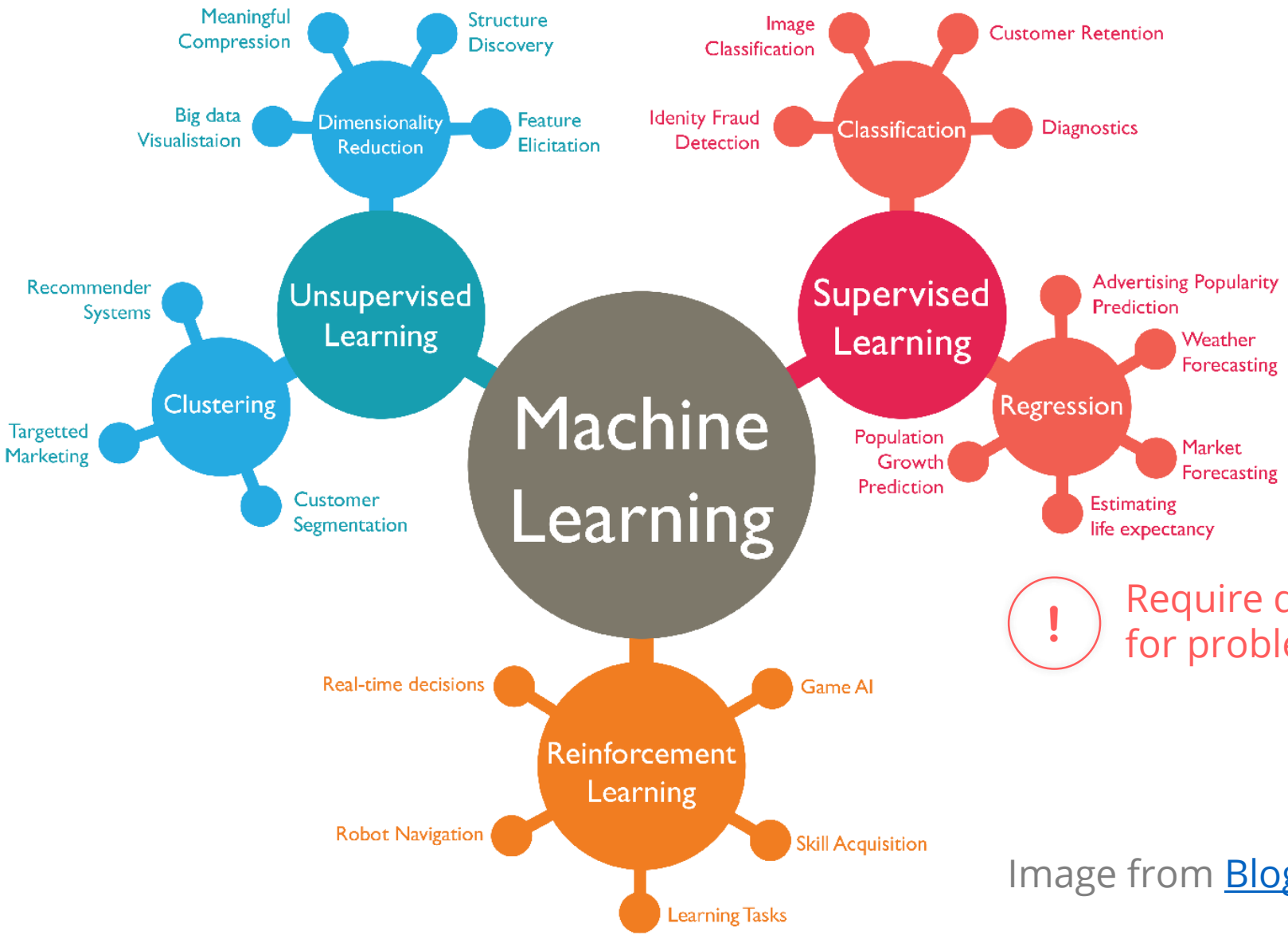
- The initial solution is usually required
- The convergence within polynomial time is not guaranteed

i.e. Genetic programming
Evolutionary methods

Contents

ML for CO

Categories of ML



! Require quality labels for problem instances

Image from Blog by Abdul Rahid

Reinforcement Learning

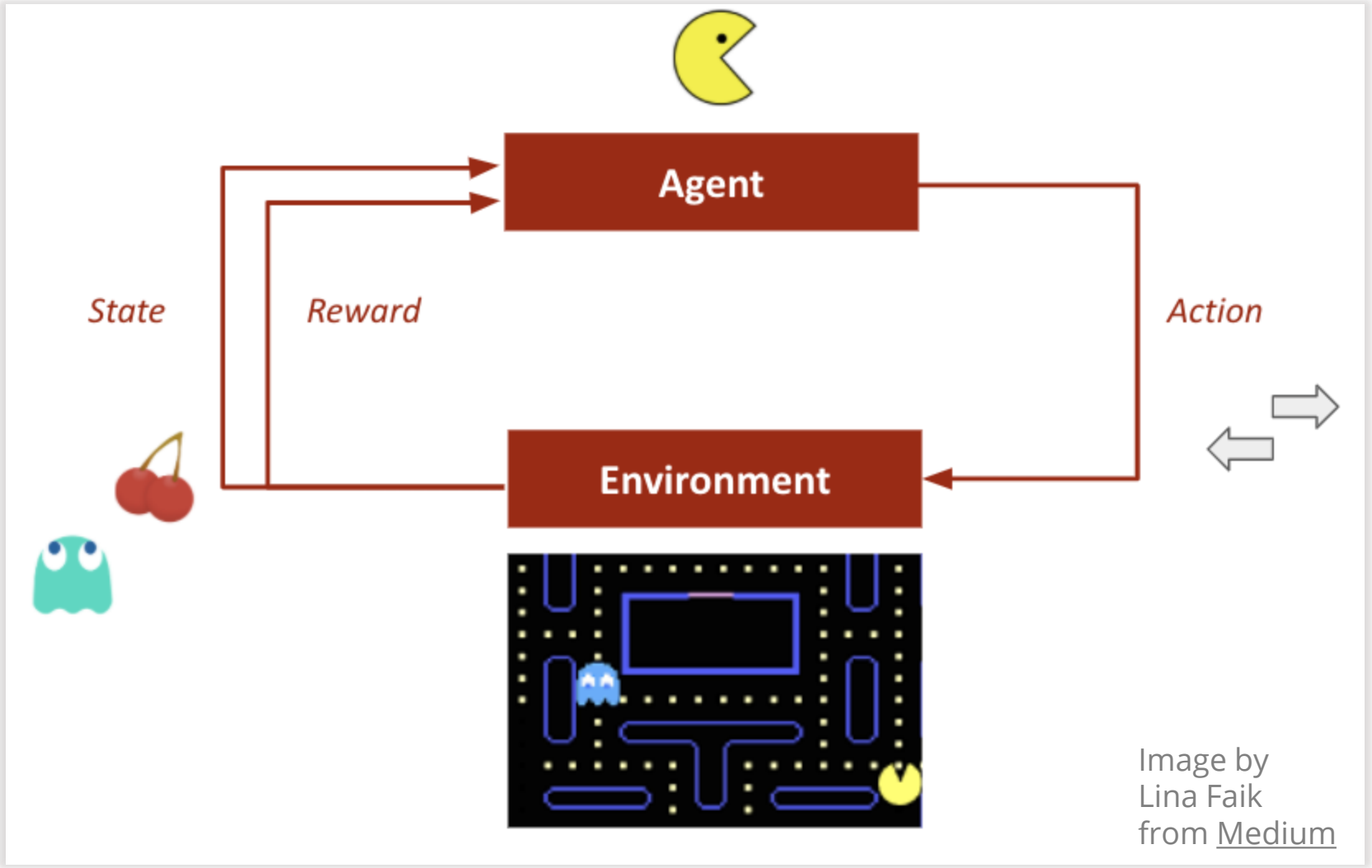


Image by Lina Faik from Medium

Agent

Env.

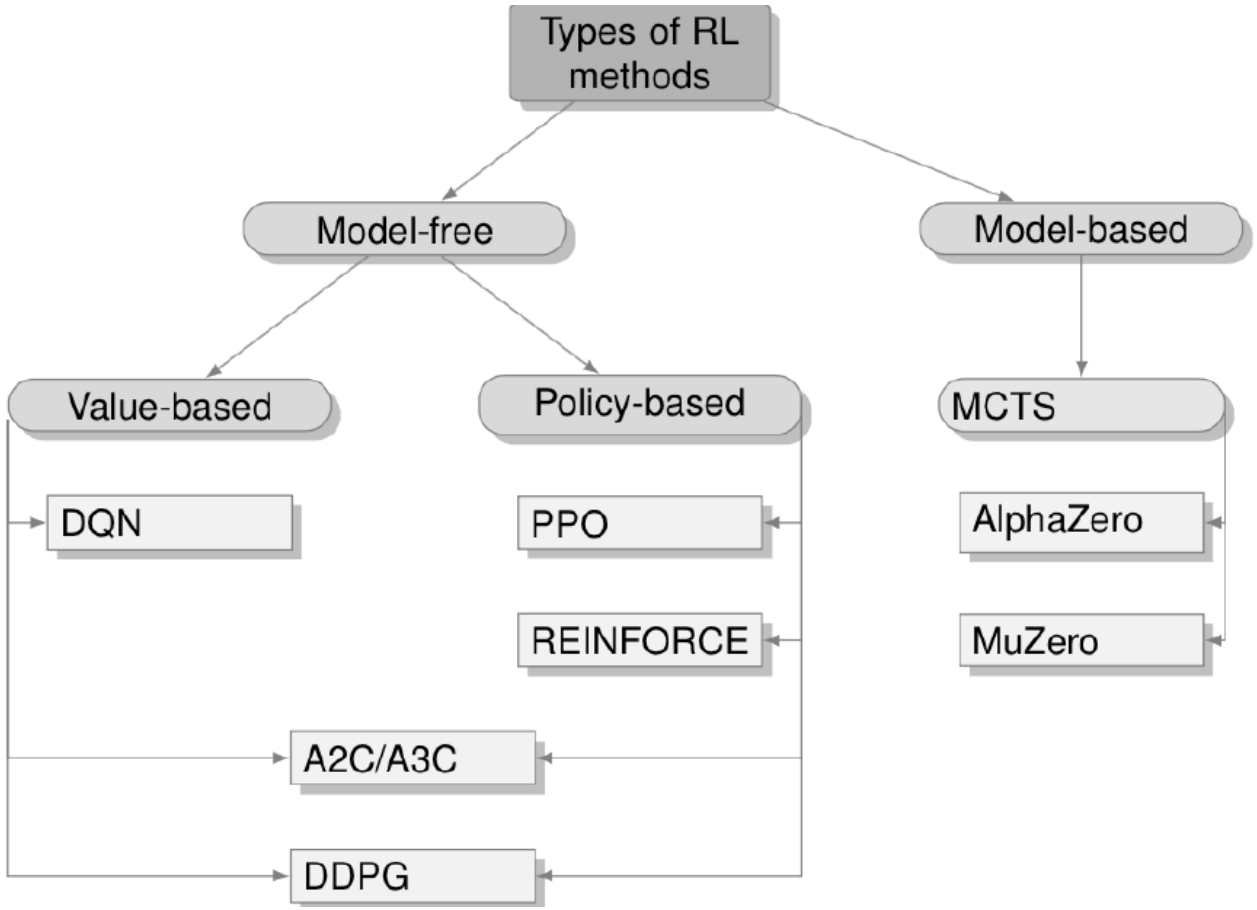
State s_t

Action a_t

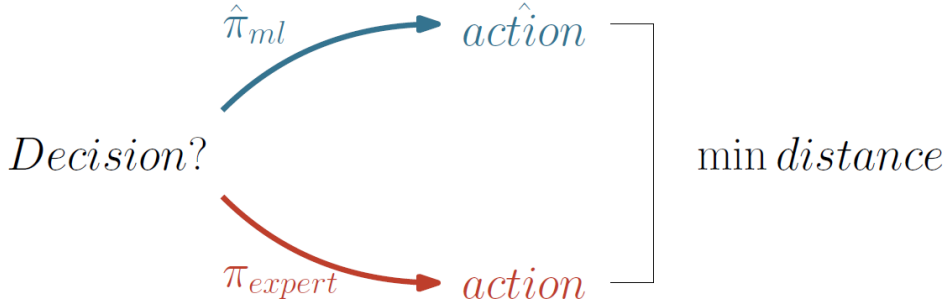
Reward r_t

Policy π

Types of RL Methods



Imitation Learning



Summary of RL

Deep Reinforcement Learning (DRL) Algorithms
Dynamic Programming and Markov Decision Processes (MDPs)

VPG. Vanilla Policy Gradient. 2000
AC. Actor-Critic Methods. 2000

Stochastic Policy
NPG. Natural Policy Gradient. 2002

Trust Region
TRPO. Trust Region Policy Optimization. 2015

GAE. Generalized Advantage Estimation. 2015

Advantage Function
A2C (Advantage Actor-Critic)
A3C. Asynchronous A2C. 2016

Trust Region (approximated, surrogate)
KL Penalty Coefficient
PPO. Proximal Policy Optimization. 2017
Planning (Model-based RL)
MBPO. Model Based PO. 2019
Auxiliary Task (on-policy, off-policy)
PPG. Proximal Policy Gradient. 2020

Energy-Based Policy
SQL. Soft Q-learning. 2017
Maximum Entropy
Automating Entropy Adjustment
SAC. Soft Actor-Critic. 2018

Q-learning. 1992

Q-table → Q net.
Experience Replay
DQN. Deep Q Network. 2014

Q net. → 2 Q net.
Double DQN. 2016
+Advantage Function
Dueling DQN. 2016

Deterministic Policy
DPG. Deterministic Policy Gradient. 2015

Taming the Noise via soft update. 2015

Greedy-Policy → Policy Net.
DDPG. Deep DPG. 2016
D4PG(Distributed Distributional DDPG). 2017

2 Q net. → Twin Critic
Delay Target Update
Policy Smoothing (SPG in DPG)
TD3. Twin Delayed DDPG. 2018

Distributional Perspective
C51 DQN (Categorical 51 grids). 2017

Quantile Regression
QR-DQN. 2017

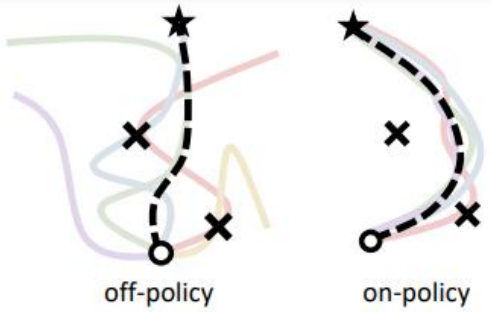
Prioritized sweeping. 1993
PER. Prioritized Experience Replay. 2016
HER. Hindsight Experience Replay. 2017

All DQN Variances
Rainbow DQN. 2017

Use Monte Carlo Tree Search 2006
AlphaGo. 2015
Without human knowledge
AlphaZero. 2017
Planning using Dynamics Model
Value Prediction Network. 2017
+Atari game (continuous state space)
MuZero. 2019 (Model-based RL)

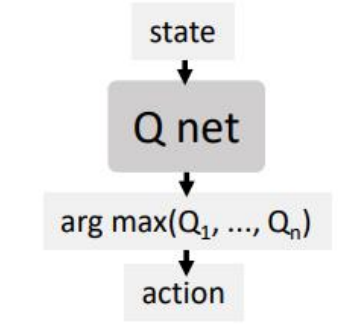
Ape-X DQN. 2018
Ape-X DPG. 2018

IMPALA. 2018
Seed RL. 2020



hybrid action

parameterized action space
Parameterized-DQN. 2018
output a probability
SAC for Discrete Action. 2019
Combination of DQN and DDPG
Hybrid-PPO. 2019



discrete action space
continuous action space

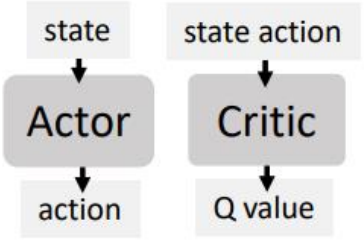
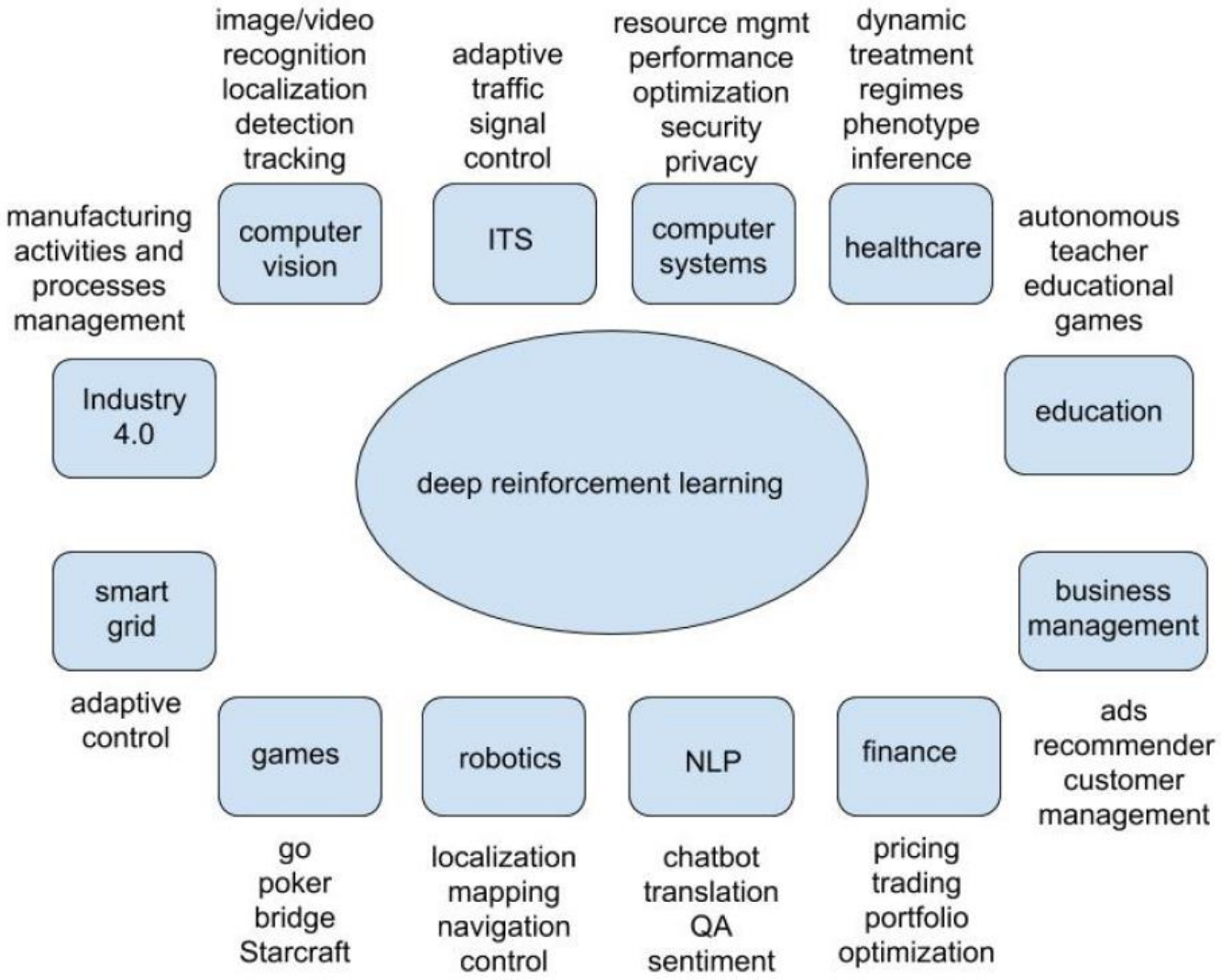


Image by 曾伊言 from Github

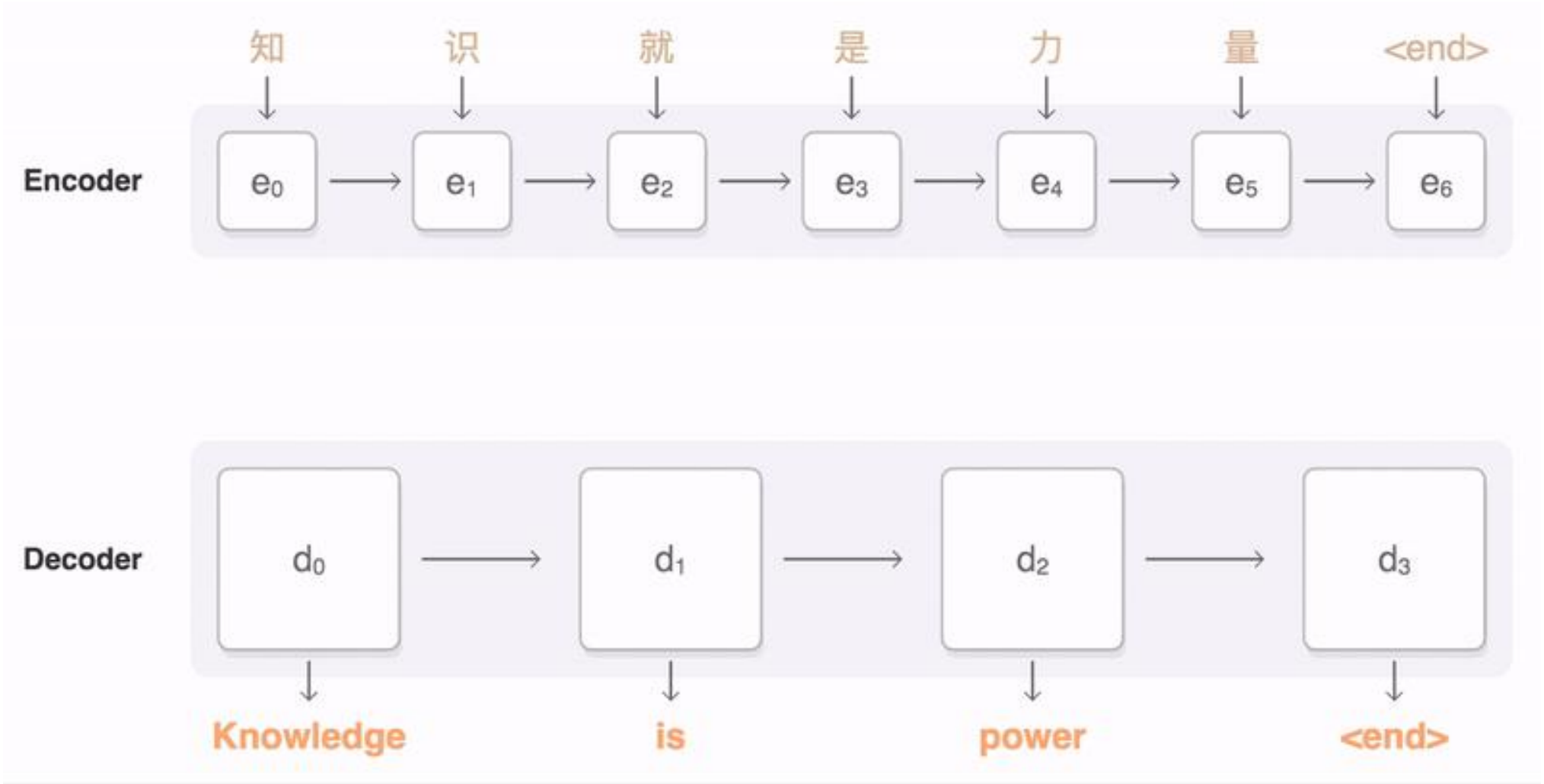
Value-based	Policy-based	Actor-Critic
Q-learning (1992)		
		Actor-Critic (AC, 2000)
	REINFORCE (PG, 2011)	
Deep Q Network (DQN, 2014)		
Nature DQN (2015) Dueling DQN (2015)	Trust Region Policy Optimization (TRPO, 2015)	Deterministic Policy gradient (DPG, 2015)
Double DQN (2016)		Asynchronous Advantage AC (A3C, 2016) Deep DPG (DDPG, 2016)
Rainbow DQN (2017)	Proximal Policy Optimization (PPO, 2017) Distributed PPO (DPPO, 2017)	
Ape-X DQN (2018)		Twin Delayed DDPG (TD3, 2018) Soft Actor-Critic (SAC, 2018)
	Proximal Policy Gradient (PPG, 2020)	

Applications of DRL



Yuxi Li et al. “Deep Reinforcement Learning: An Overview”. arXiv, 2018

Attention Mechanism



Dzmitry Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015

Transformer

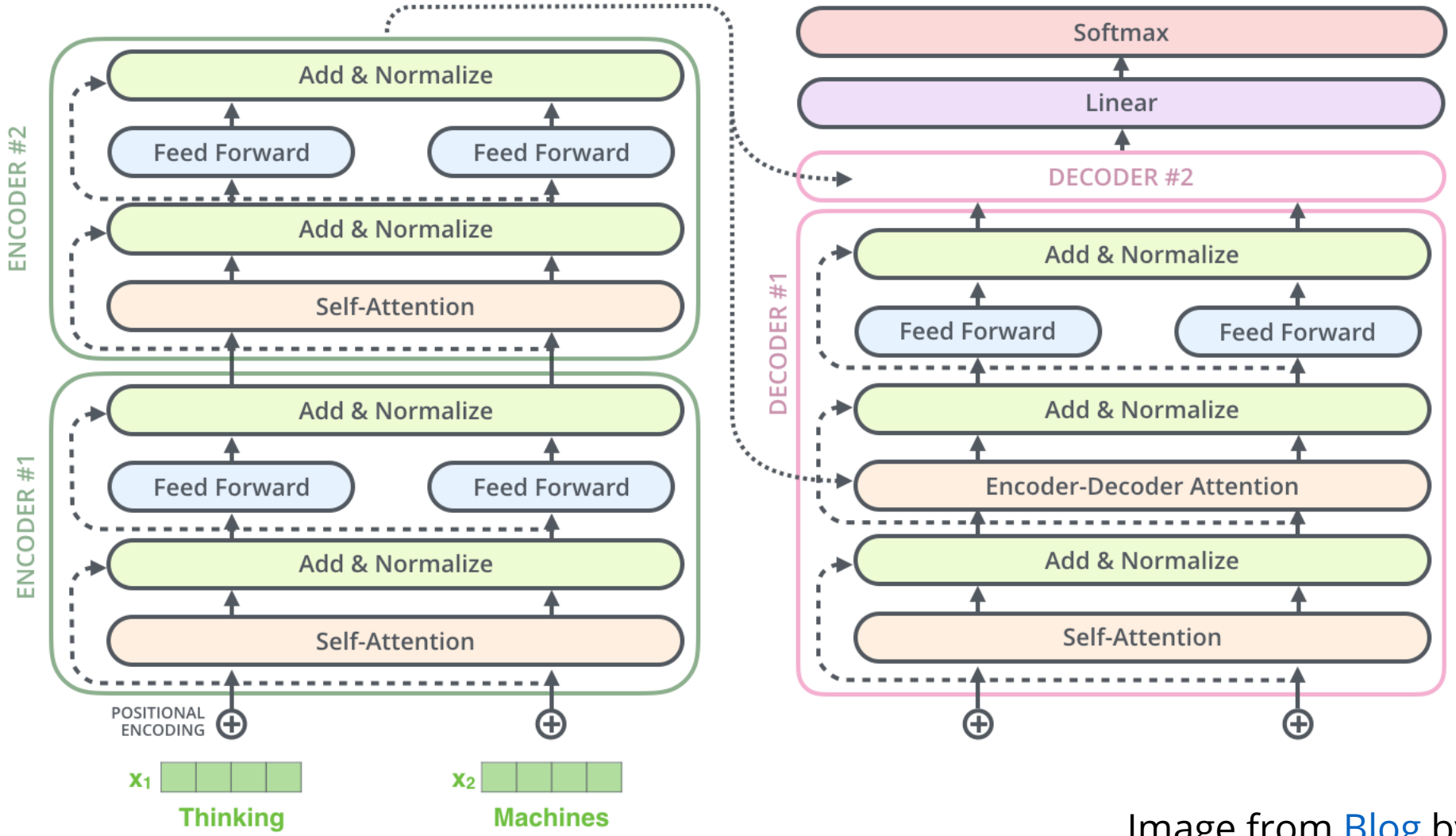
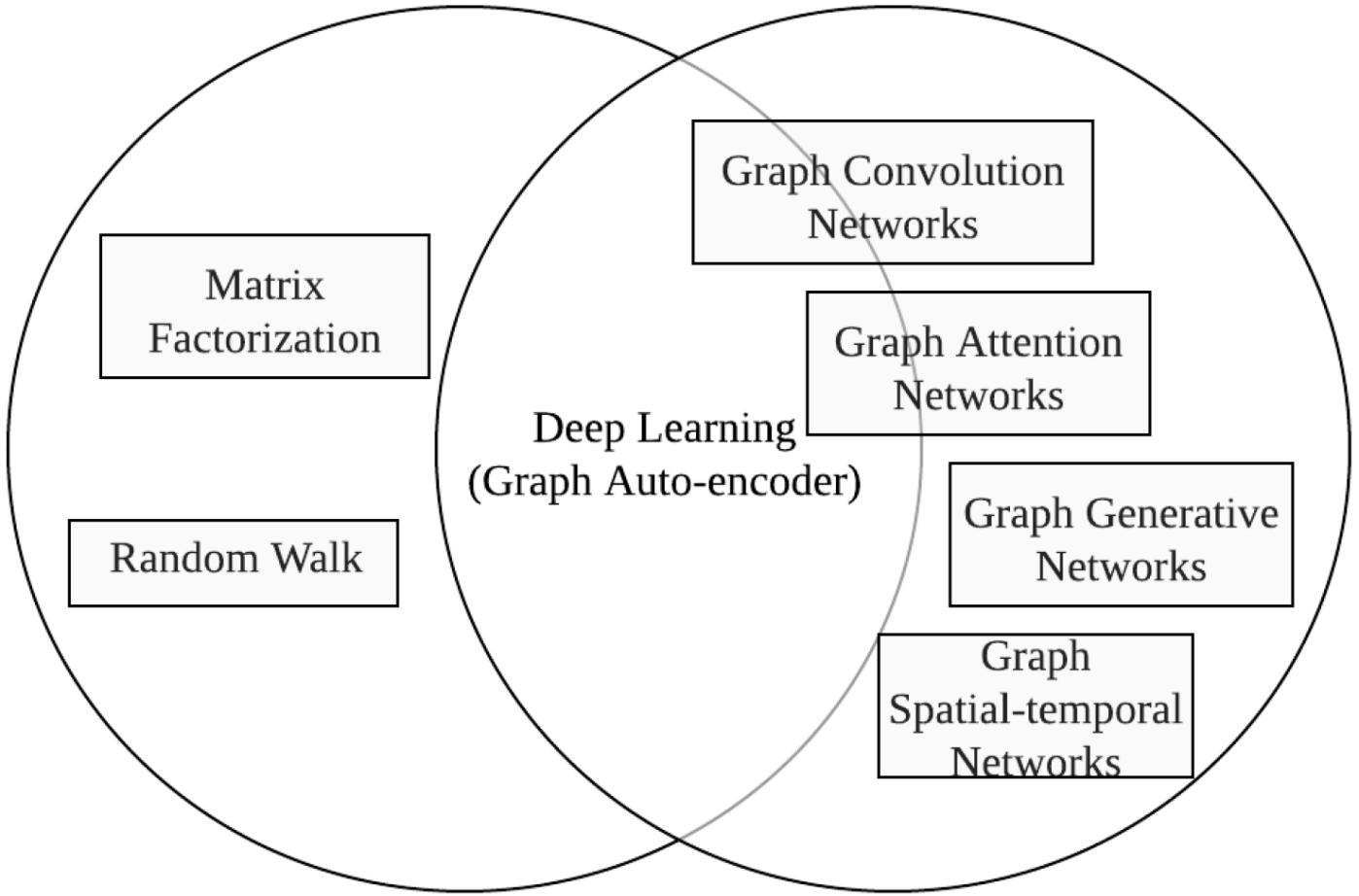


Image from Blog by Jalammar

Graph Learning

Graph Embedding



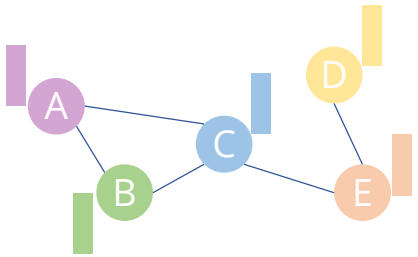
Graph Neural Network

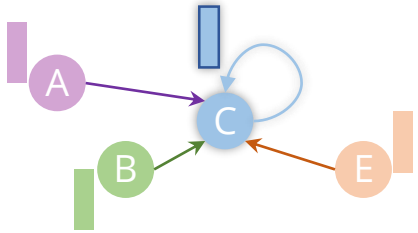
Graph Neural Network

Input

Original Features

$$\boldsymbol{x} = \{\vec{\boldsymbol{x}}_1, \vec{\boldsymbol{x}}_2, \dots, \vec{\boldsymbol{x}}_N\}, \vec{\boldsymbol{x}}_i \in \mathbb{R}^F$$



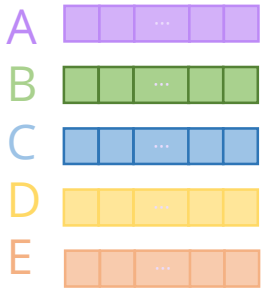


$$\mathbf{h}_i^t = \boldsymbol{\sigma}\Big(\sum_{j \in \mathcal{N}_i} f\big(\mathbf{h}_i^{t-1}, \mathbf{h}_j^{t-1}\big)\Big)$$

Output

New Representation

$$\boldsymbol{h} = \{\vec{\boldsymbol{h}}_1, \vec{\boldsymbol{h}}_2, \dots, \vec{\boldsymbol{h}}_N\}, \vec{\boldsymbol{h}}_i \in \mathbb{R}^{F'}$$



Downstream Tasks

Node Level

$$\boldsymbol{z}_i = f(\mathbf{h}_i)$$

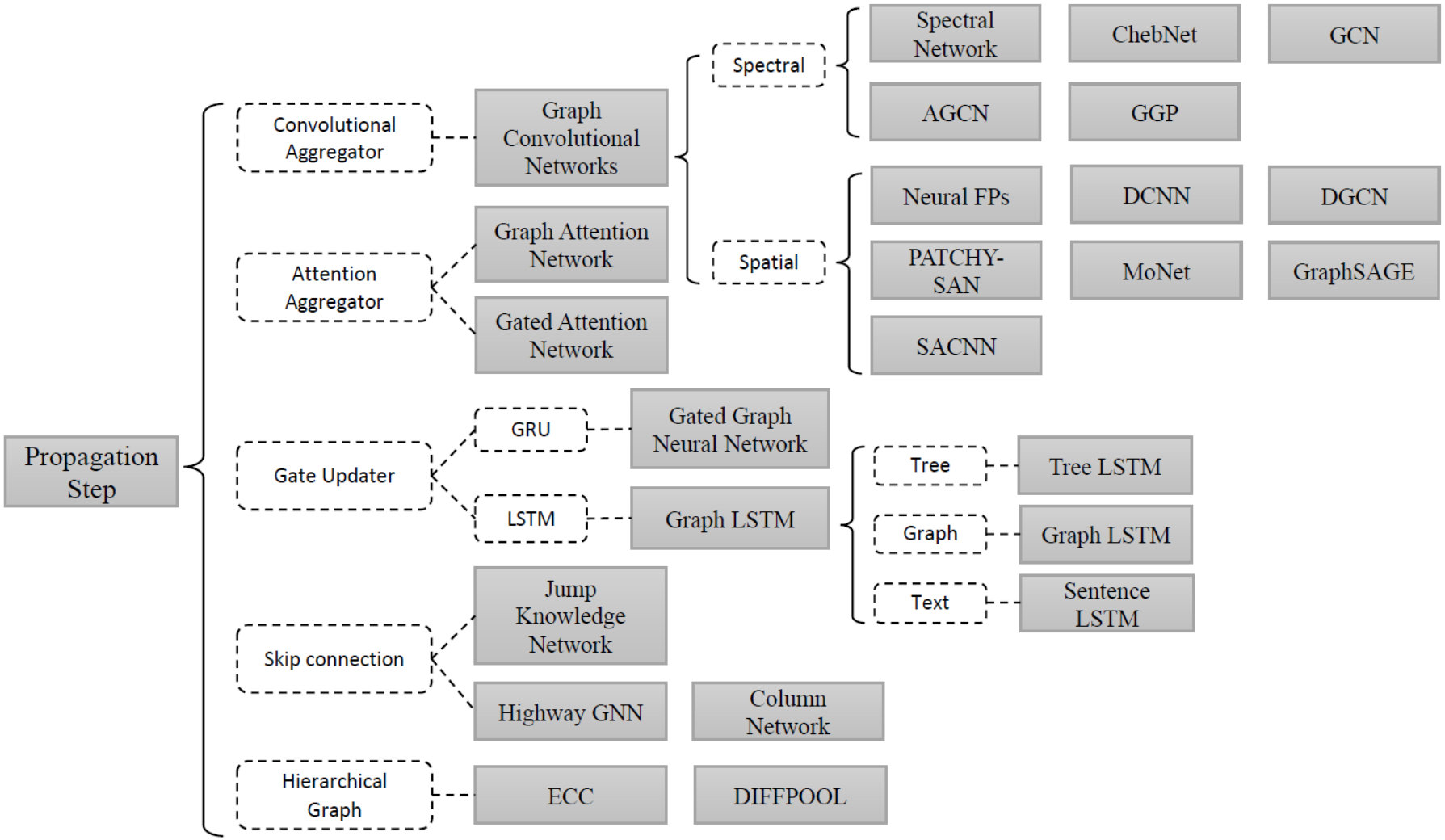
Link Level

$$\boldsymbol{z}_{ij} = f\big(\mathbf{h}_i, \mathbf{h}_i, \mathbf{e}_{ij}\big)$$

Graph Level

$$\boldsymbol{z}_G = f\left(\sum_{i \in \mathcal{V}} \mathbf{h}_i\right)$$

Categories of GNN



ML-based Methods

Learn heuristics during the training process automatically based on ML.

Hyper-heuristics

Strengths

- Neural Network
  - RNN/GNN/...
  - Automatically identify distinct features from training data
  - Capable of tackling different COPs
- Reinforcement Learning
  - Model-free
  - Not overly dependent on problem formulation
  - Able to adapt to dynamic or random environment

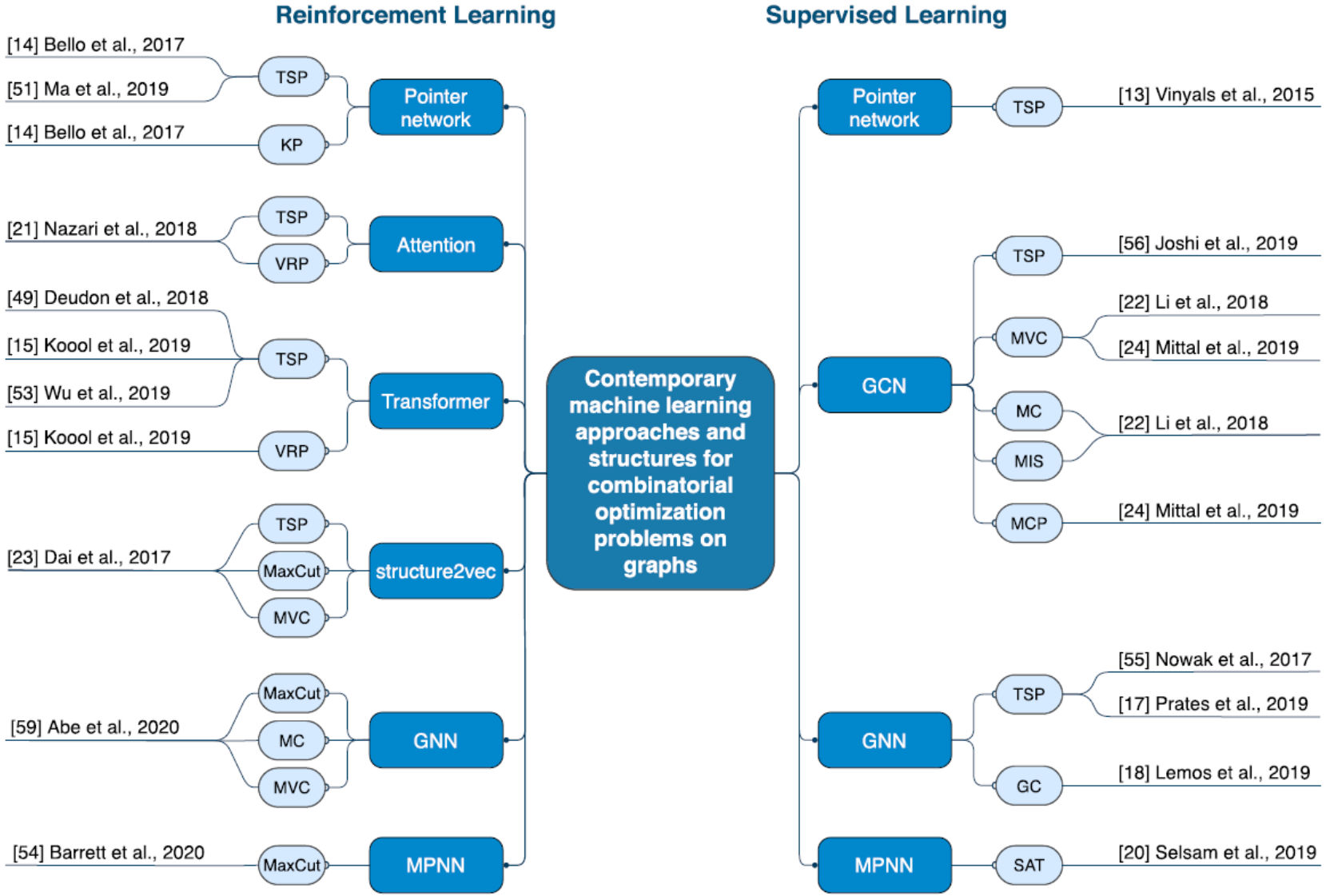
Challenges

- Problem Modeling
  - Not all COPs are easy to model as MDP
- Feature extraction
  - Fully extract the features in the environment
- Additional constraints
  - Reasonably integrate hard constraints into the model
- Multi-objective
  - Design a comprehensive reward function to optimize the policy
- Model generalization
  - Low quality of solutions on large-scale problem

Contents

ML for CO

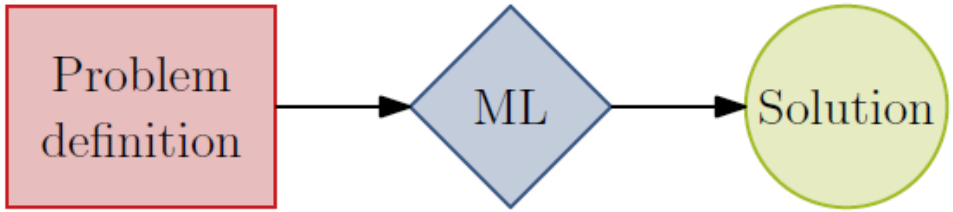
ML4CO over Graph



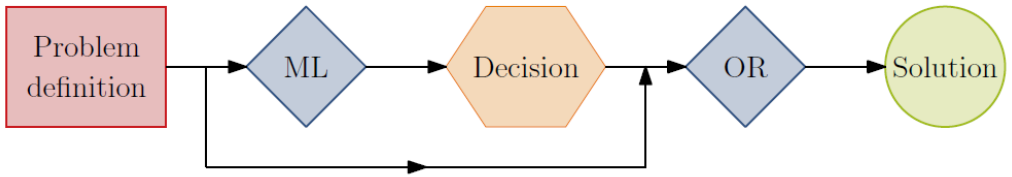
Natalia Vesselinova et al. “Learning Combinatorial Optimization on Graphs: A Survey With Applications to Networking”. IEEE ACCESS, 2020

ML for CO

End-to-end Learning

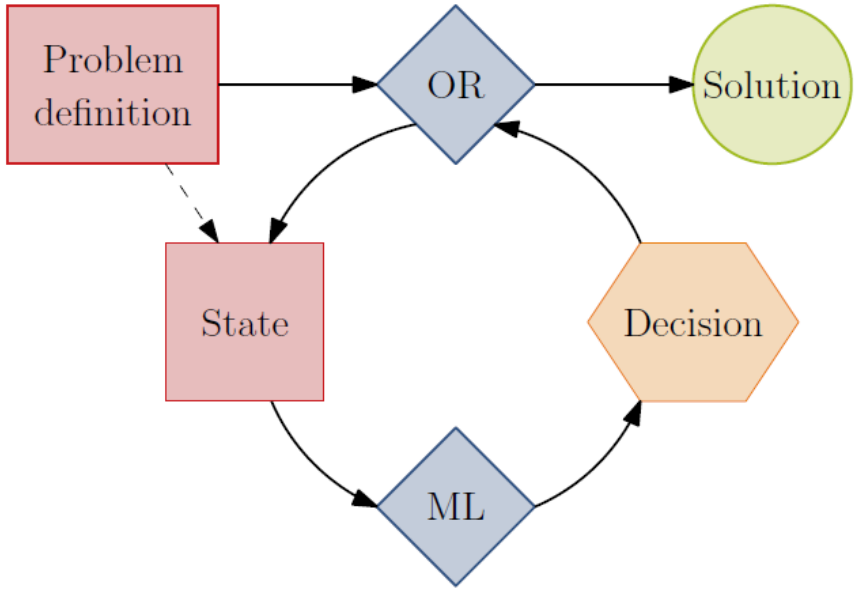


Learning to configure OR



Augment an OR algorithm with valuable pieces of information

ML alongside OR

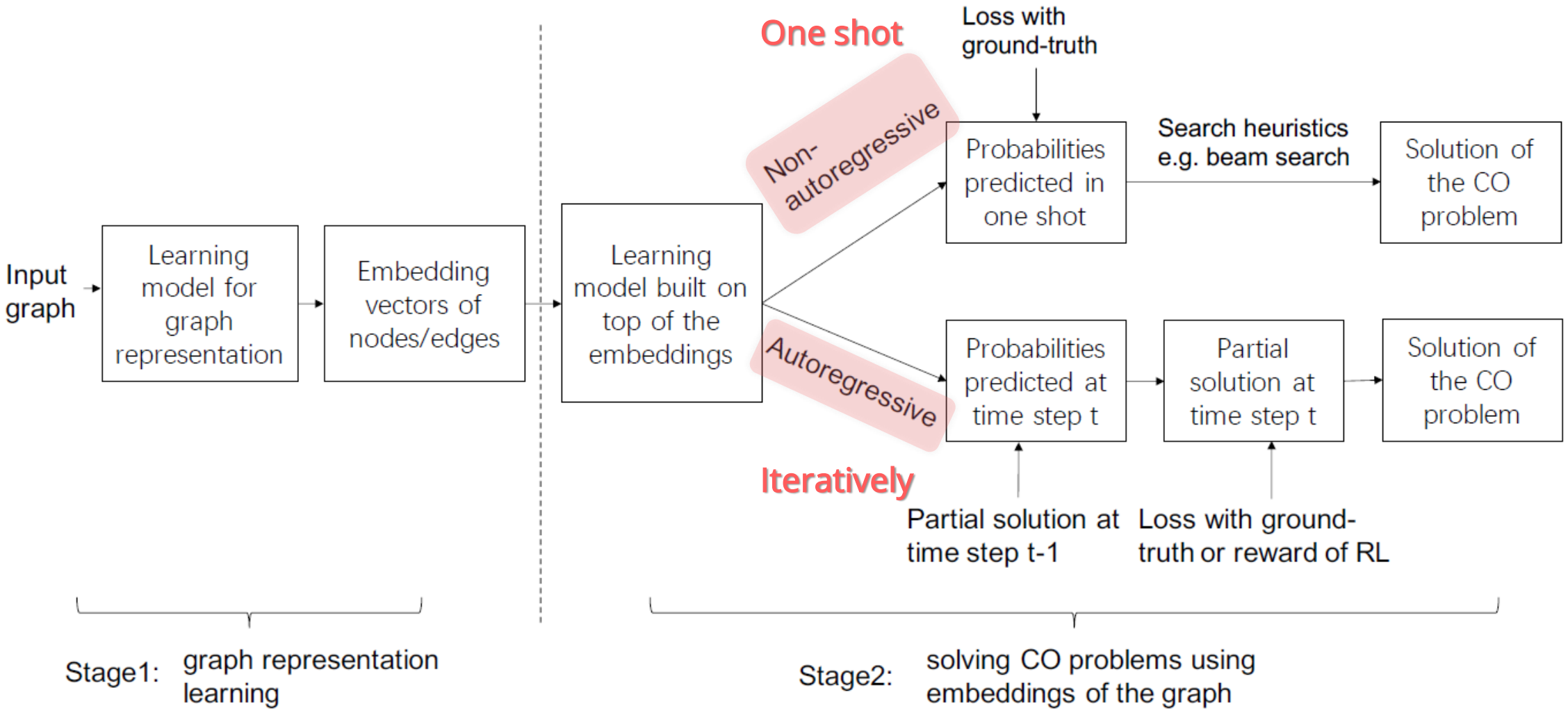


B&B-based | Local Search | Other

Exact | Approximate & Heuristic

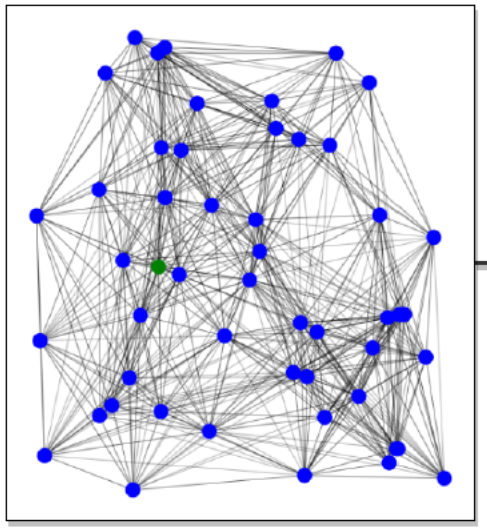
Yoshua Bengio et al. "Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon". arXiv, 2018

Two stages of End-to-End



*Graph ConvNet for TSP

E2E TSP SL GNN



Input 2D graph

Node & edge features

MLP

Node & edge embedding

Graph ConvNet Model

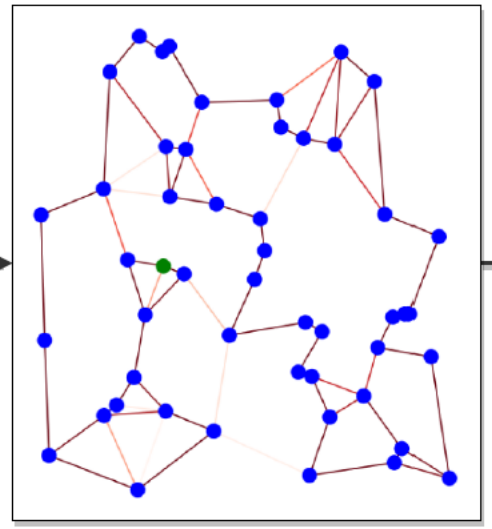
graph

GCN

edge_emb

MLP

edge_prob



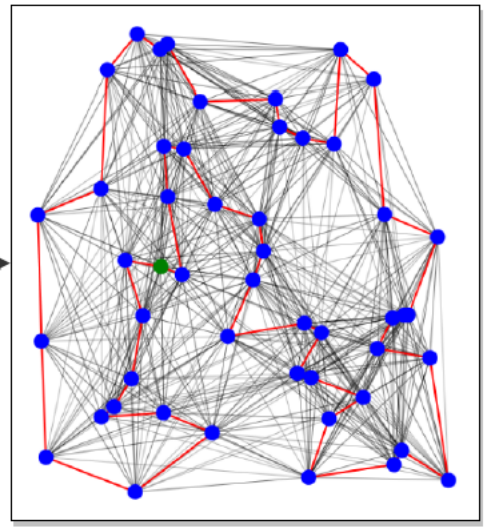
Edge prediction heat-map

An edge adjacency matrix

denoting the probabilities of edges occurring on the TSP tour.

Beam Search Decoder

Beam search & Shortest tour heuristic



Valid TSP tour

Loss function

binary cross-entropy loss

Groun d-truth ⟷ Tour result

Chaitanya K. Joshi et al. “An Efficient Graph Convolutional Network Technique for the Travelling Salesman Problem”. In INFORMS Annual Meeting, 2019

*GNN-TSP Solver

E2E TSP SL GNN

Algorithm 1 Graph Neural Network TSP Solver

procedure GNN-TSP($\mathcal{G} = (\mathcal{V}, \mathcal{E}), C$)

// Compute binary adjacency matrix from edges to source & target vertices

$\mathbf{EV}[i, j] \leftarrow 1$ iff $(\exists v' | e_i = (v_j, v', w)) | \; \forall e_i \in \mathcal{E}, v_j \in \mathcal{V}$

// Compute initial edge embeddings

$\mathbf{E}^{(1)}[i] \leftarrow E_{init}(w, C) \mid \forall e_i = (s, t, w) \in \mathcal{E}$

// Run t_{max} message-passing iterations

for $t = 1 \ldots t_{max}$ **do**

// Refine each vertex embedding with messages received from edges in which it appears either as a source target vertex

$\mathbf{V}_h^{(t+1)}, \mathbf{V}^{(t+1)} \leftarrow V_u(\mathbf{V}_h^{(t)}, \mathbf{EV}^T \times \underset{msg}{E}(\mathbf{E}^{(t)}))$

// Refine each edge embedding with messages received from its source and its target vertex

$\mathbf{E}_h^{(t+1)}, \mathbf{E}^{(t+1)} \leftarrow E_u(\mathbf{E}_h^{(t)}, \mathbf{EV} \times \underset{msg}{V}(\mathbf{V}^{(t)}))$

// Translate edge embeddings into logit probabilities

$\mathbf{E_{logits}} \leftarrow E_{vote}\left(\mathbf{E}^{(t_{max})}\right)$

// Average logits and translate to probability (the operator $\langle\rangle$ indicates arithmetic mean)

prediction $\leftarrow$ sigmoid($\langle \mathbf{E_{logits}} \rangle$)

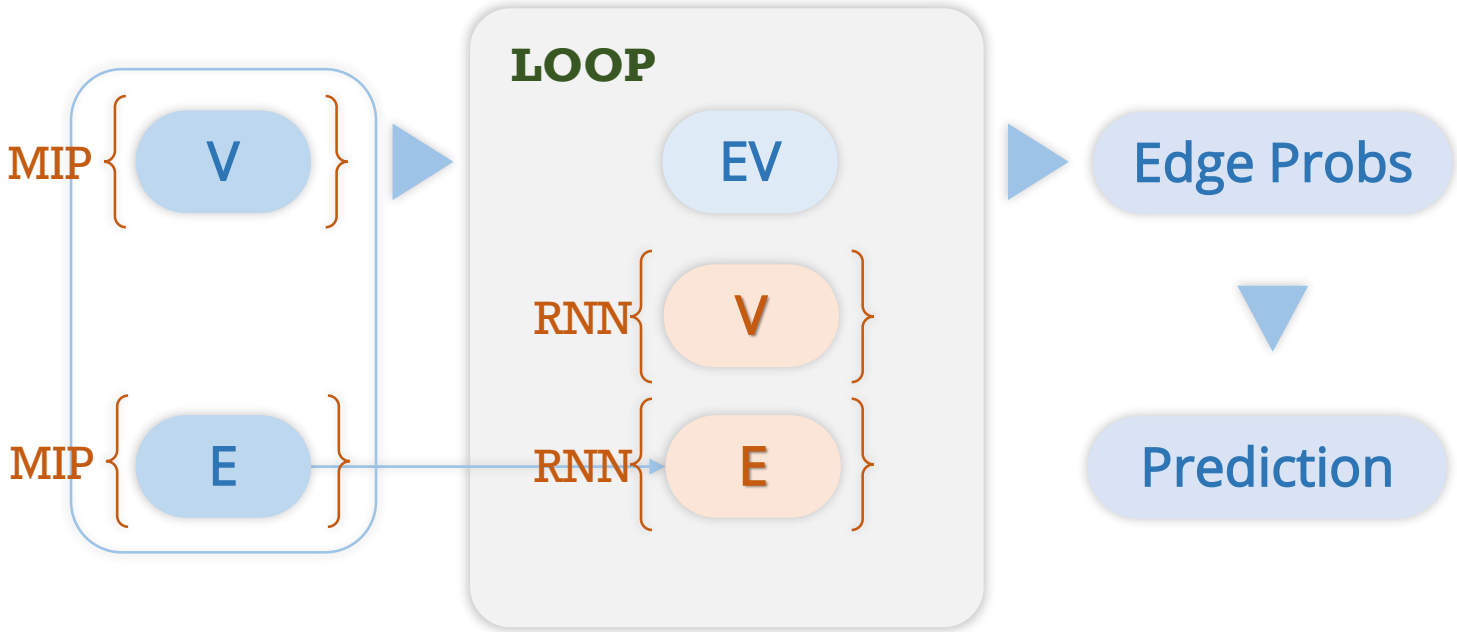
V vertex-to-vertex adjacency matrix

E edge list matrix

EV edge-to-vertex adjacency matrix

$e_i = (s, t, w)$

source vertices
target vertices
weight



NAR vs AR Methods

Non-autoregressive

Predict the probabilities that each node/edge being a part of a solution in one shot. (Fast)

The solution of the CO problem can be found by search heuristics such as beam search.

However

Inherently ignore some sequential characteristics of some CO problems, e.g., the TSP.

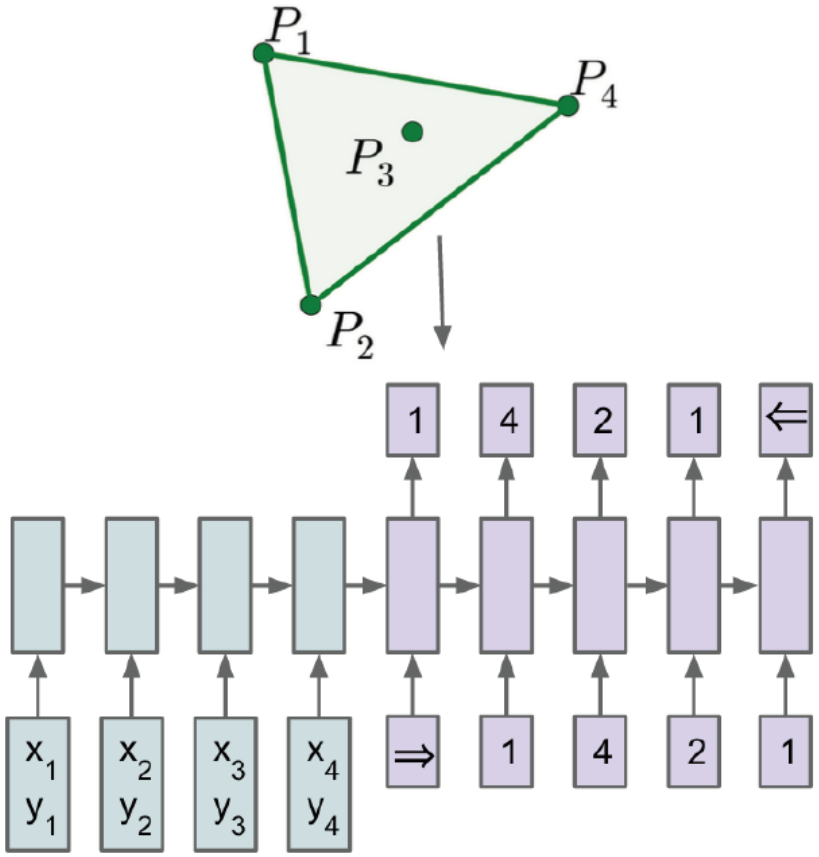
Autoregressive

Predict the decision that which node/edge to add/remove to the partial solution step by step.

Can obtain help from RNN and Attention

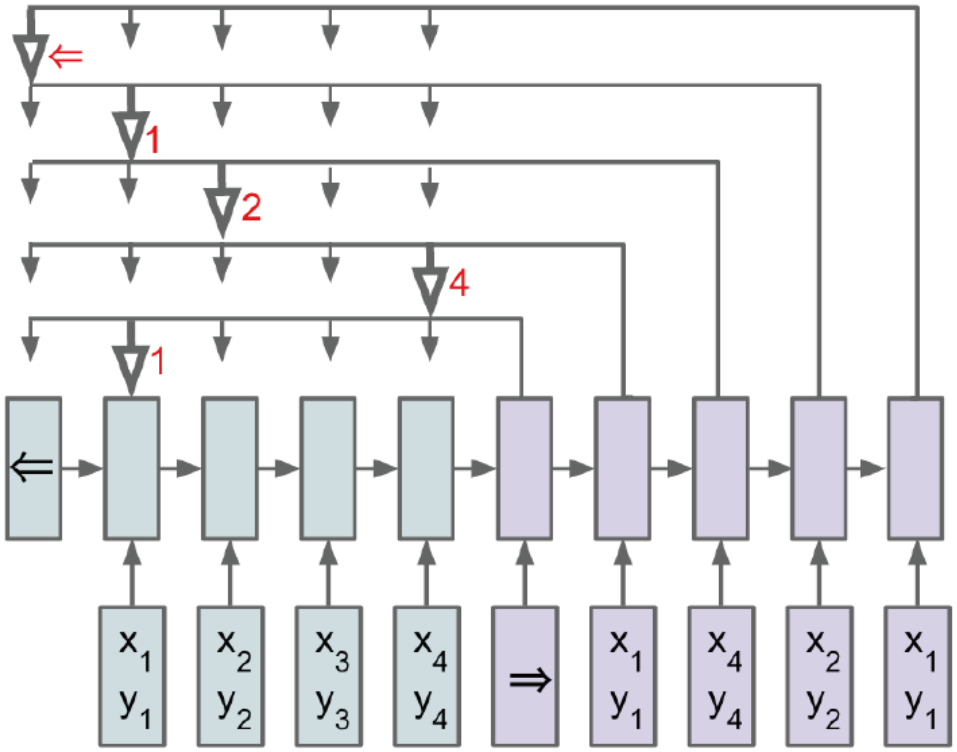
*PTR-NET (Pointer Network)

E2E TSP SL RNN



The output dimensionality is fixed by the dimensionality of the problem

(a) Sequence-to-Sequence



attention mechanism & Softmax

(b) Ptr-Net

Input

Seq2Seq
RNN
Attention

Output

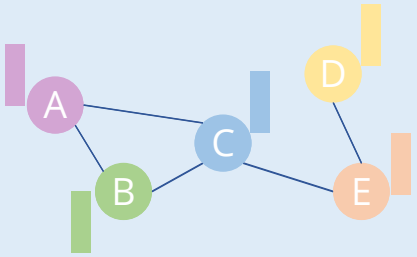
Softmax

Solution

General framework for RL

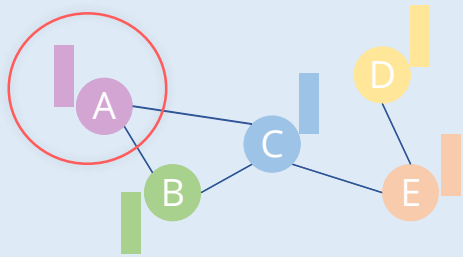
State

The features of Node & Edge



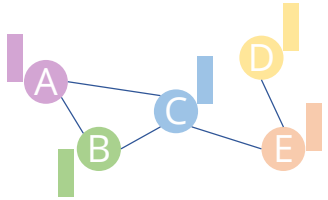
Action

Add or remove a node or edge



Agent

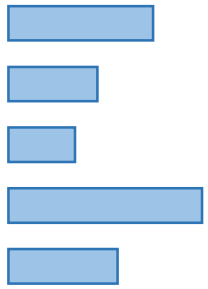
current graph



Neural Network

Softmax

Action Probs



Reward

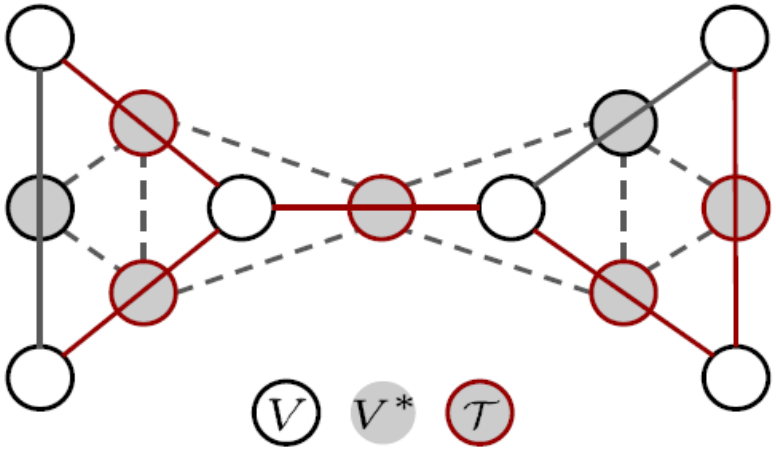
The estimation for solution quality

i.e.
TSP: the length sum of tour
MVC: the weight sum of solution

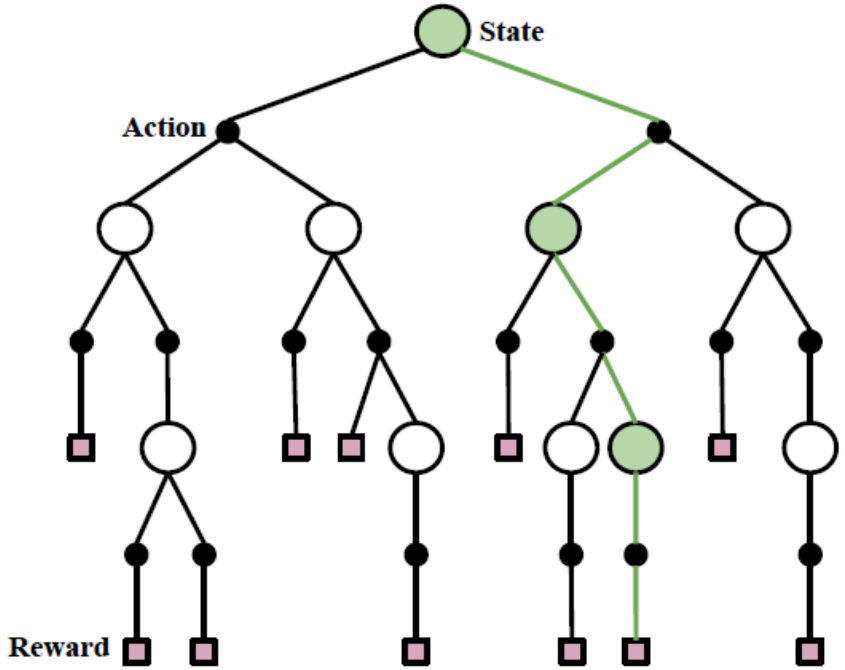
Iddo Drori et al. “Learning to Solve Combinatorial Optimization Problems on Real-World Graphs in Linear Time”. arXiv, 2020

Unified Framework with Line Graph

A



B



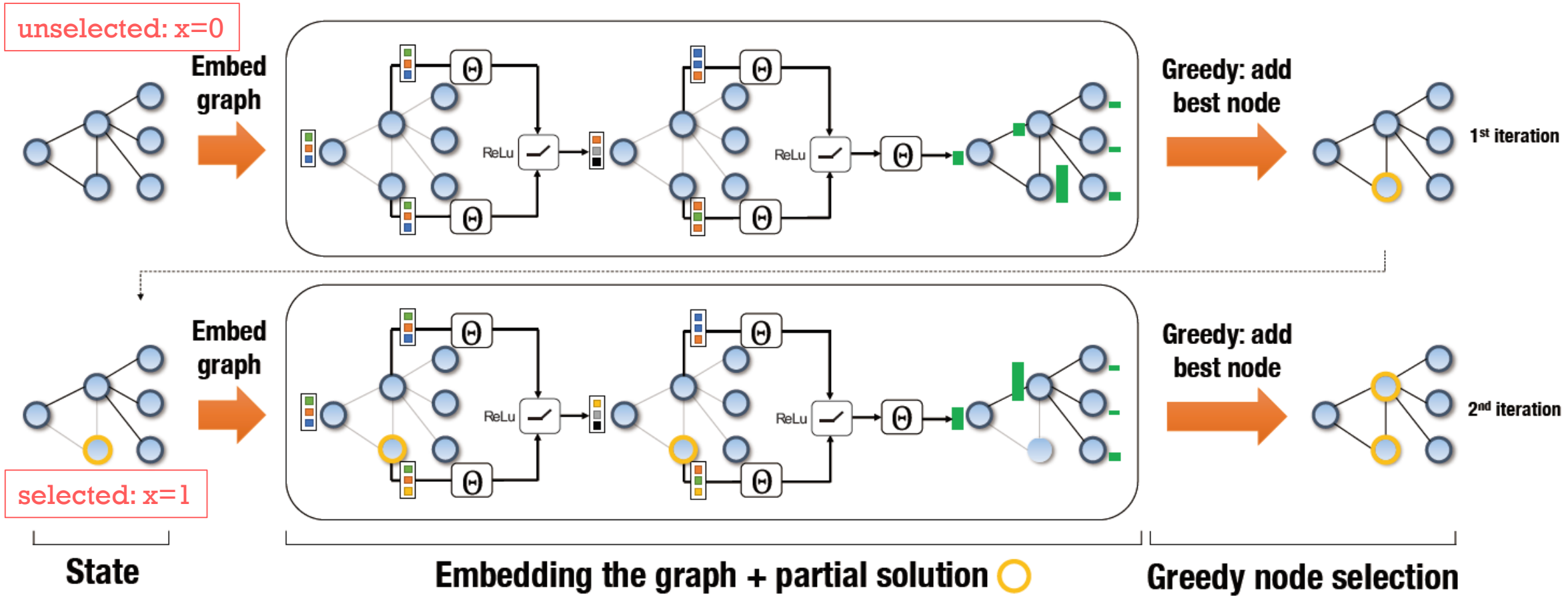
C

Problem	State	Action	Reward
MST	$G = (V, E), G^* = (V^*, E^*), W, \mathcal{T}$	$\mathcal{T} = \mathcal{T} \cup \{e\}$	$-\left(I(\mathcal{T}) + \sum_{e \in E_\pi} W(e)\right)$ (Eq. 4)
SSP	$G = (V, E), G^* = (V^*, E^*), W, \mathcal{Q}_i$	$\mathcal{Q}_i = \mathcal{Q}_i \cup \{e\}$	$-\sum_{i=1}^{\lvert V \rvert}\left(I(\mathcal{Q}_i) + \sum_{e \in \mathcal{Q}_i} W(e)\right)$
TSP	$G = (V, E), \bar{V} = \{\tau(1), .., \tau(i)\}$	$\bar{V} = \bar{V} \cup \{\tau(i+1)\}$	$-\sum_{i=1}^{\lvert V \rvert} \lVert \mathbf{v}_{\tau(i)} - \mathbf{v}_{\tau(i+1)} \rVert_2$ (Eq. 5)
VRP	$G = (V, E), \bar{V}_m = \{\tau(d), \tau_m(2), .., \tau_m(i)\}, M$	$\bar{V}_m = \bar{V}_m \cup \{\tau_m(i+1)\}$	$-\max_{m \in \{1,..,M\}} \left\{ \sum_{i \in V_\tau(m)} \lVert \mathbf{v}_{\tau_m(i)} - \mathbf{v}_{\tau_m(i+1)} \rVert_2 \right\}$

*S2V-DQN for MVC, TSP

E2E | MVC | DQN | GNN

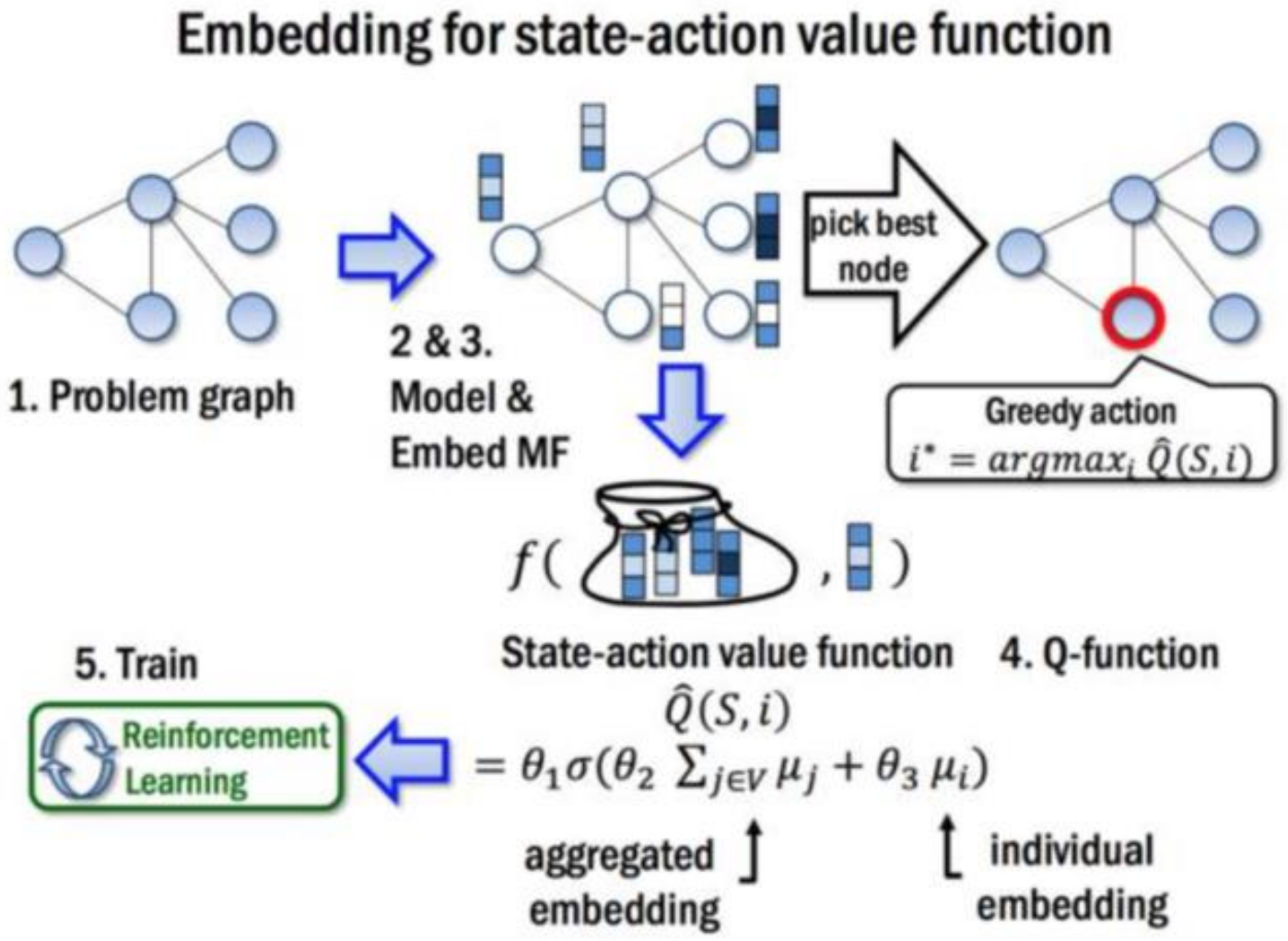
State → Graph Embedding → Greedy Selection



Hanjun Dai et al. “Learning Combinatorial Optimization Algorithms over Graphs”. In NeurIPS, 2017

*S2V-DQN for MVC, TSP

E2E | MVC | DQN | GNN



Hanjun Dai et al. “Learning Combinatorial Optimization Algorithms over Graphs”. In NeurIPS, 2017

*GNN + Attention

E2E | VRP | RF | GNN

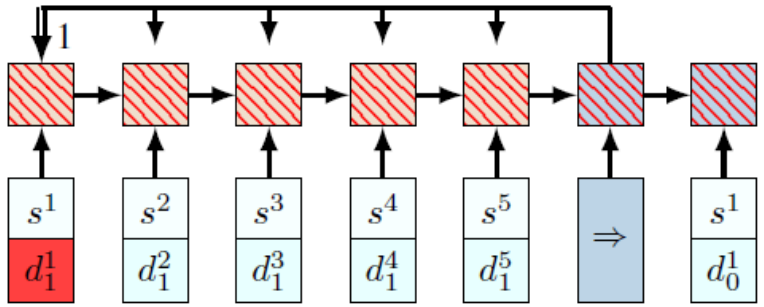


Figure 1: Limitation of the Pointer Network. After a change in dynamic elements (d_1^1 in this example), the whole Pointer Network must be updated to compute the probabilities in the next decision point.

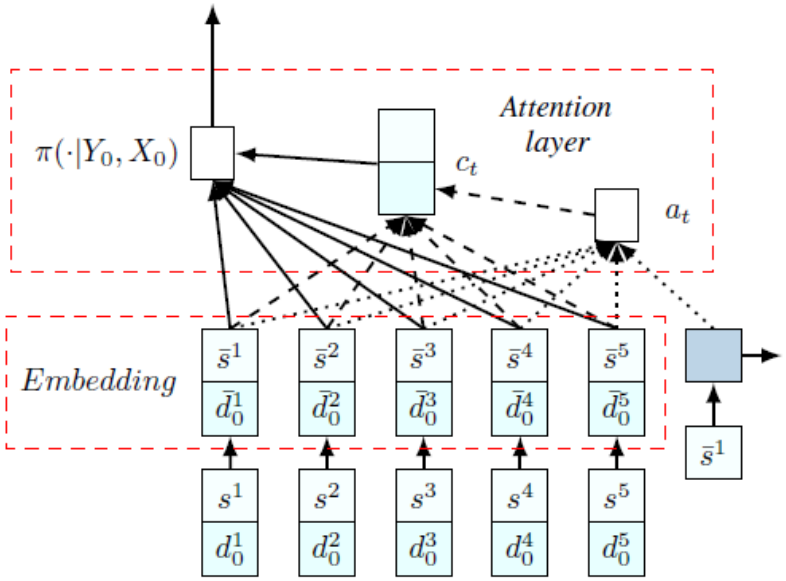


Figure 2: Our proposed model. The embedding layer maps the inputs to a high-dimensional vector space. On the right, an RNN decoder stores the information of the decoded sequence. Then, the RNN hidden state and embedded input produce a probability distribution over the next input using the attention mechanism.

Every step

Softmax

Attention

Graph Embedding

*Multi-head Attention

E2E VRP RF GNN

Encoder Attention Decoder

Node Embeddings

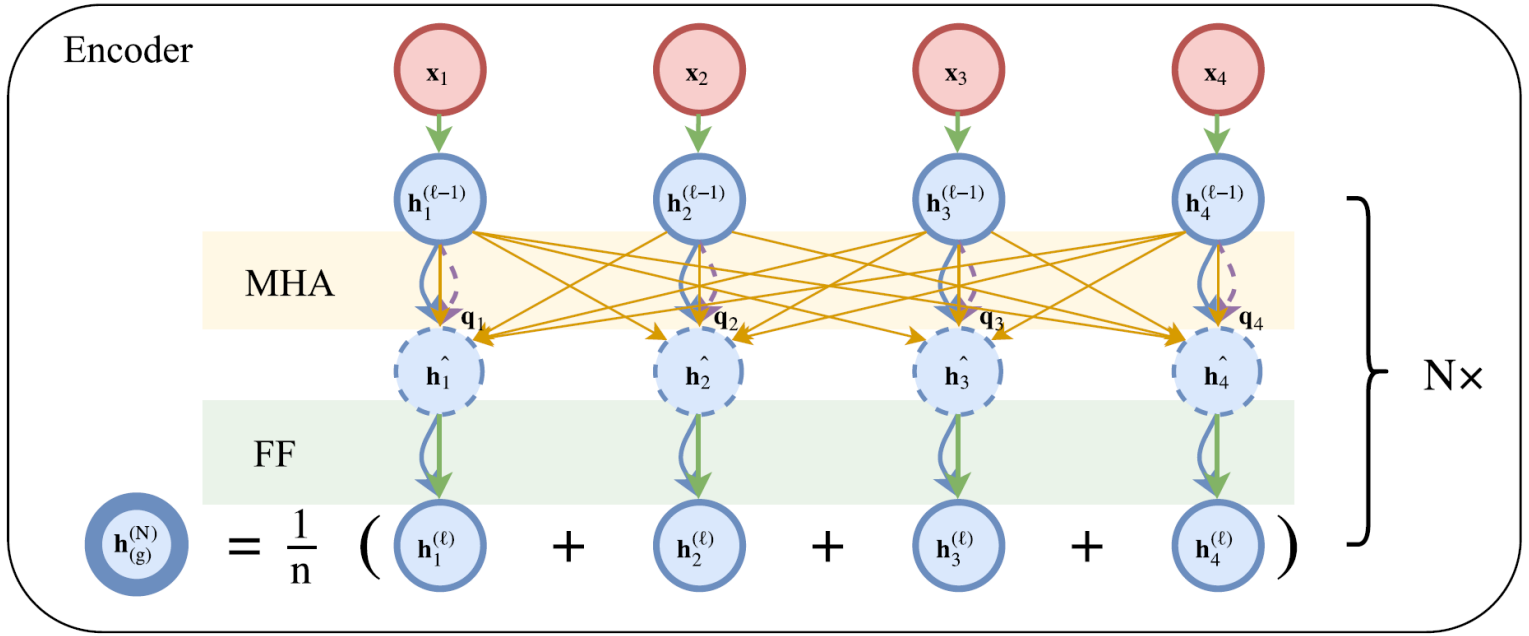
$$\mathbf{h}_i^{(0)} = W^{\mathrm{x}}\mathbf{x}_i + \mathbf{b}^{\mathrm{x}}$$

Graph Embedding

$$\bar{\mathbf{h}}^{(N)} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{h}_i^{(N)}$$

$$\hat{\mathbf{h}}_i = \mathrm{BN}^{\ell}\left(\mathbf{h}_i^{(\ell-1)} + \mathrm{MHA}_i^{\ell}\left(\mathbf{h}_1^{(\ell-1)}, \ldots, \mathbf{h}_n^{(\ell-1)}\right)\right)$$

$$\mathbf{h}_i^{(\ell)} = \mathrm{BN}^{\ell}\left(\hat{\mathbf{h}}_i + \mathrm{FF}^{\ell}(\hat{\mathbf{h}}_i)\right)$$ Skip Connection



Layer $\ell \in \{1, .., N\}$

MHA
multi-head attention layer

FF
fully connected feed-forward layer

BN
batch normalization

*Multi-head Attention

E2E VRP RF GNN

Encoder Attention **Decoder**

Context Vector

$$\mathbf{h}_{(c)}^{(N)} = \begin{cases} \left[\bar{\mathbf{h}}^{(N)}, \mathbf{h}_{\pi_{t-1}}^{(N)}, \mathbf{h}_{\pi_1}^{(N)}\right] & t > 1 \\ \left[\bar{\mathbf{h}}^{(N)}, \mathbf{v}^1, \mathbf{v}^f\right] & t = 1 \end{cases}$$

$\mathbf{v}^1, \mathbf{v}^f$ is learnable

$$\mathbf{q}_{(c)} = W^Q \mathbf{h}_{(c)}$$

$$\mathbf{k}_i = W^K \mathbf{h}_i$$

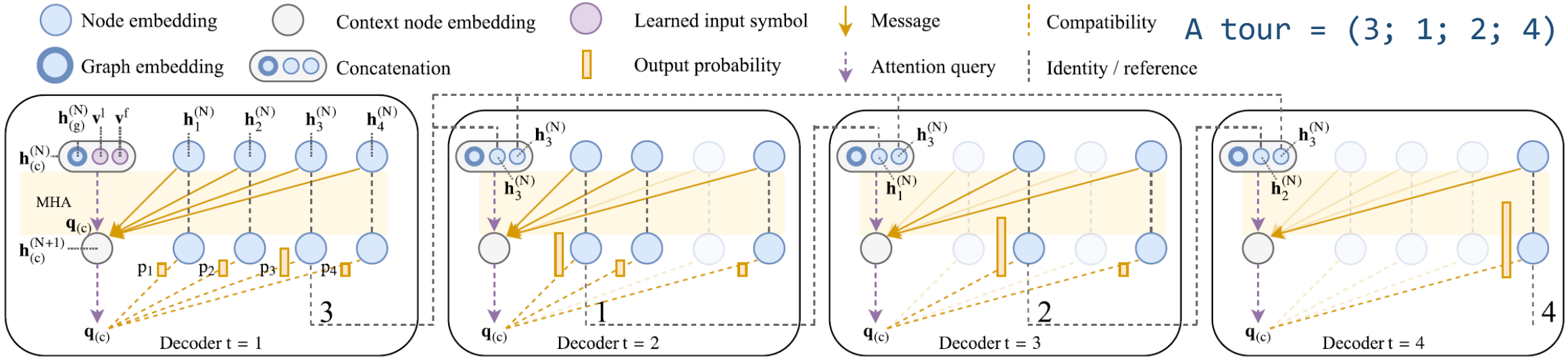
$$\mathbf{v}_i = W^V \mathbf{h}_i$$

$$u_{(c)j} = \begin{cases} C \cdot \tanh\left(\frac{\mathbf{q}_{(c)}^T \mathbf{k}_j}{\sqrt{d_k}}\right) & if\, j \neq \pi_t, \forall t' < t \\ -\infty & otherwise. \end{cases}$$

log-probabilities

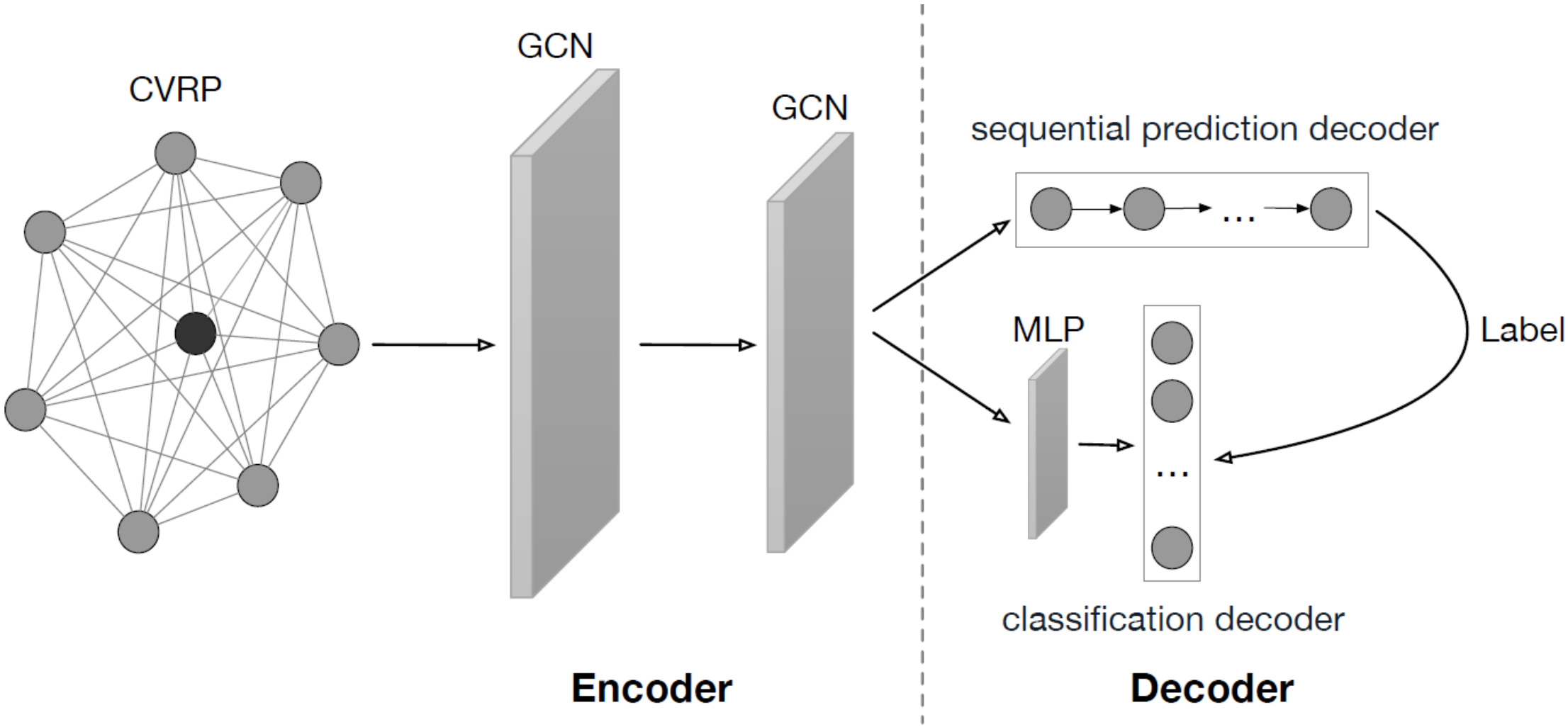
mask

$$p_i = p_{\boldsymbol{\theta}}(\pi_t = i \mid s, \boldsymbol{\pi}_{1:t-1}) = \frac{e^{u_{(c)i}}}{\sum_j e^{u_{(c)j}}}$$ SoftMax -> Output



Wouter Kool. “Attention, Learn To Solve Routing Problems!”. In ICLR 2019

*GCN-NPEC

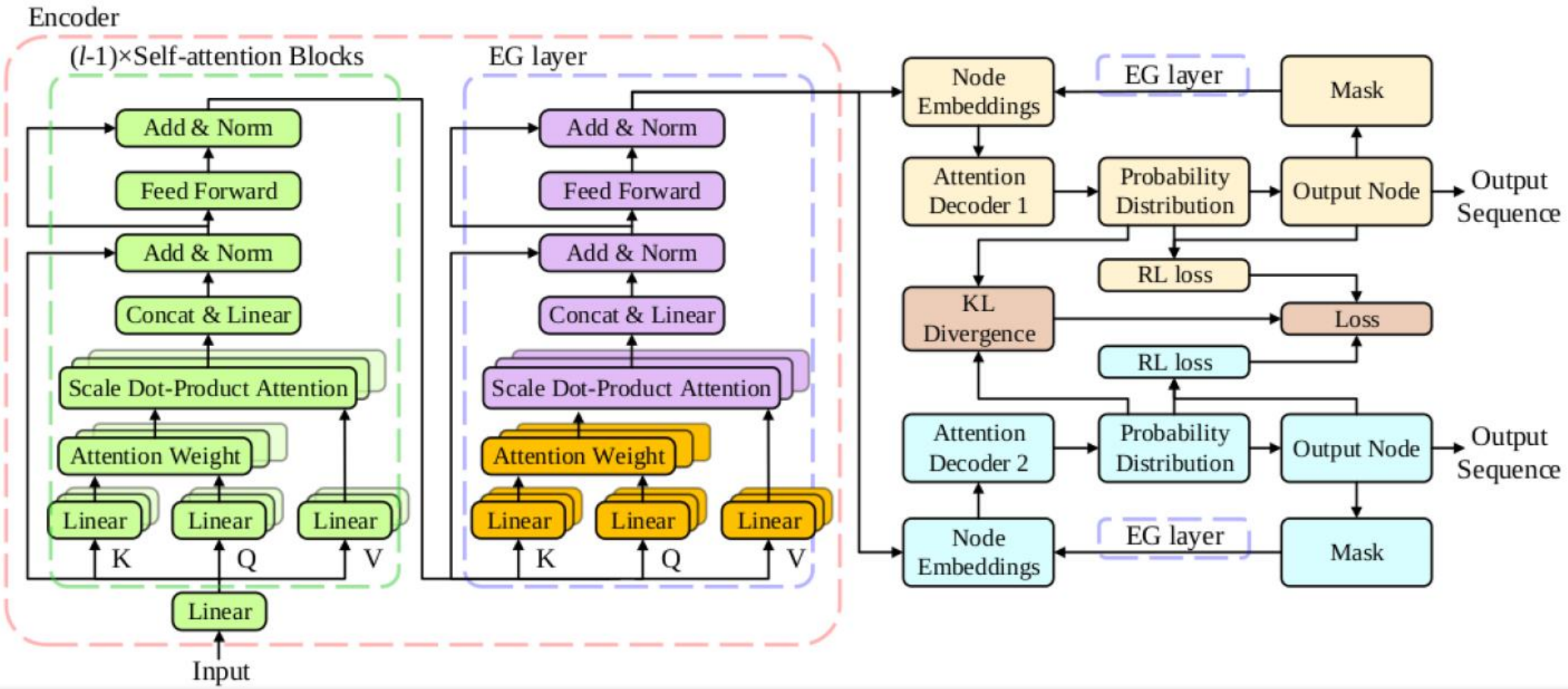


Lu Duan et al. “Efficiently Solving the Practical Vehicle Routing Problem: A Novel Joint Learning Approach”. In KDD, 2020

*MDAM for VRP

E2E | VRP | RF | TF

Multi-Decoder Attention Model ▶ Train multiple diverse policies ▶ beam search strategy ▶ The best solution



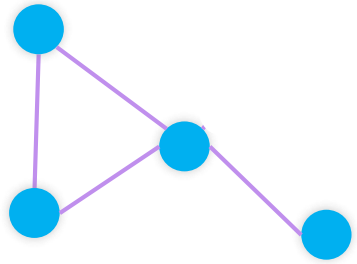
Liang Xin et al. "Multi-Decoder Attention Model with Embedding Glimpse for Solving Vehicle Routing Problems ". In AAAI, 2021

*GNN + Tree Search

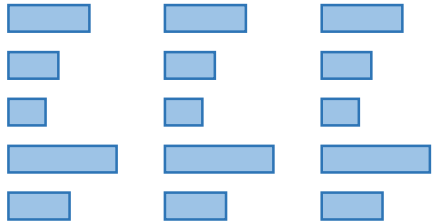
LS | MIS | SL | TF

Improve the diversity of solution with multiple outputs and local search

Graph Reduction



Multiple Outputs

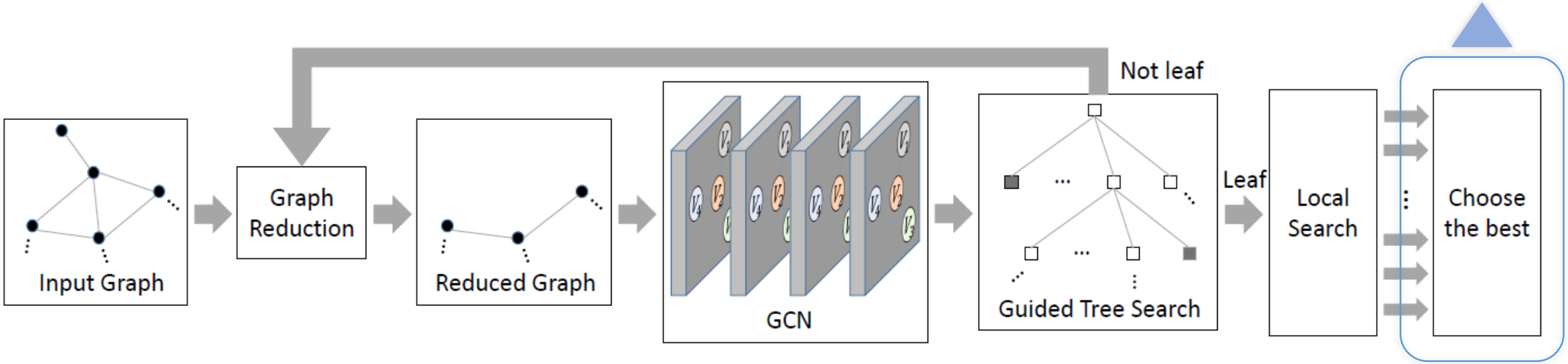


Local Searh

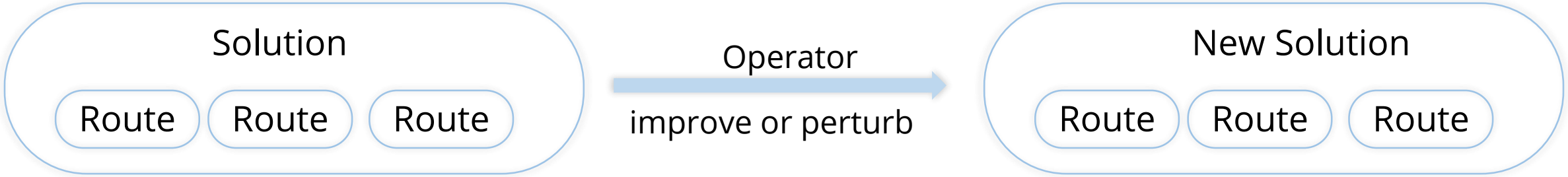
inserting, deleting, and swapping nodes in current solution

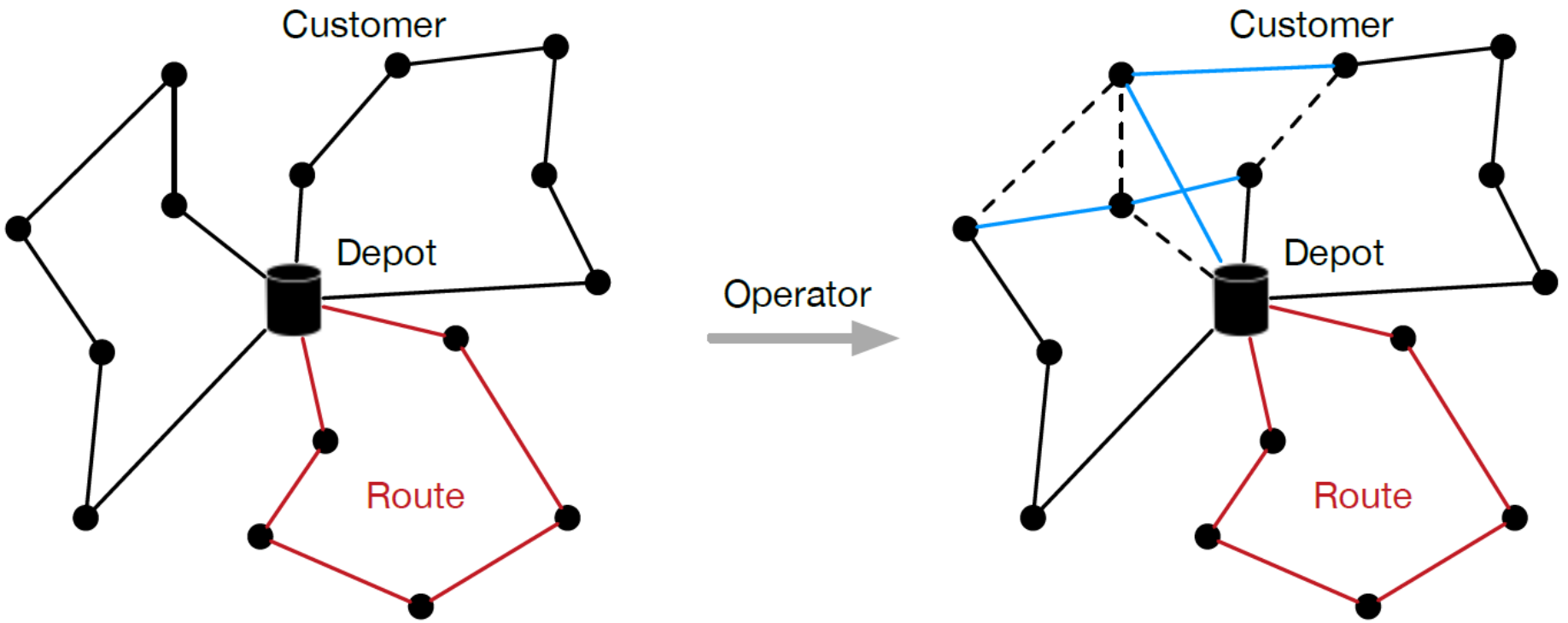
Training Lost

Only minimize the loss between the best and ground-true



*Iterative Method



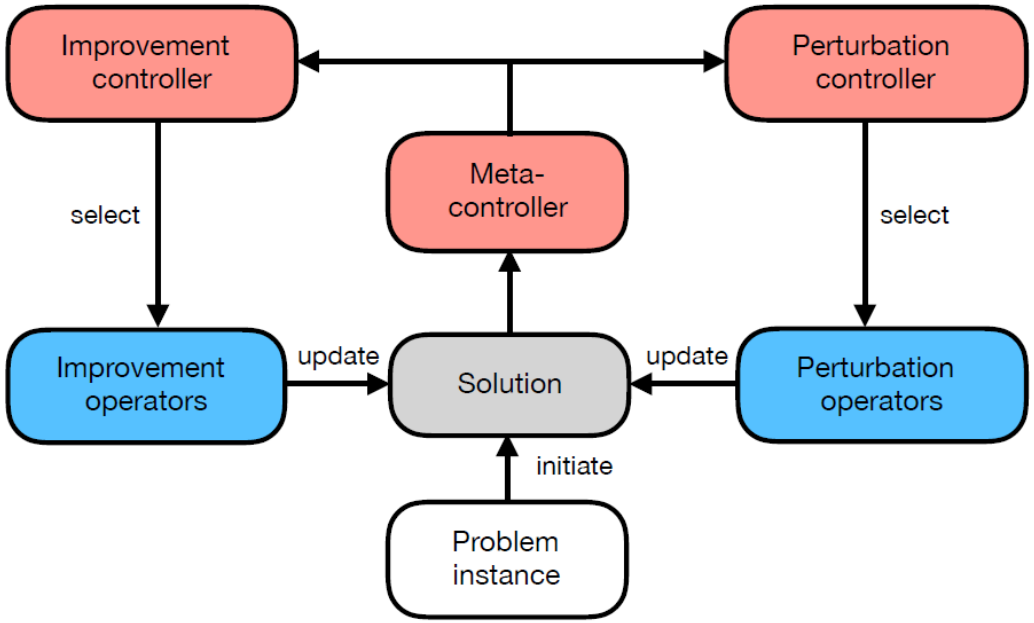


- intra-route
- inter-route

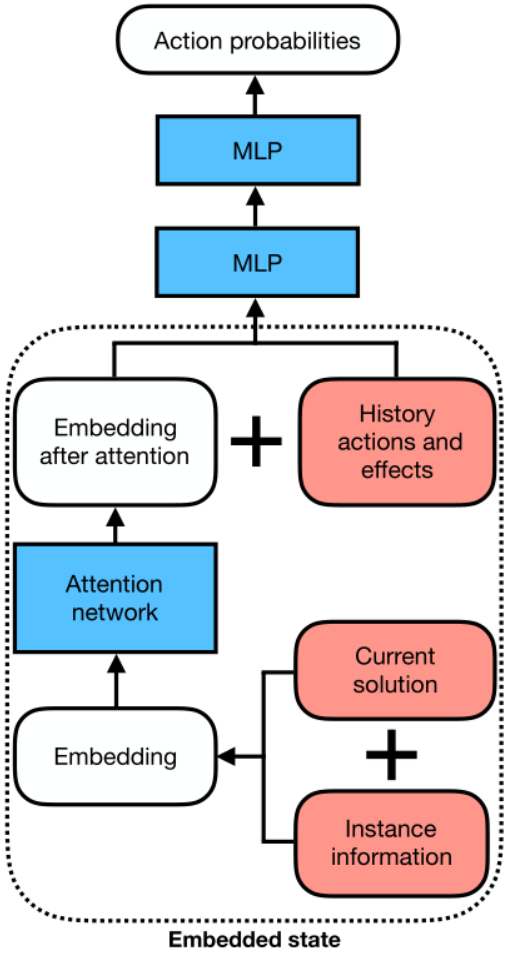
*Iterative Method

Hierarchy framework

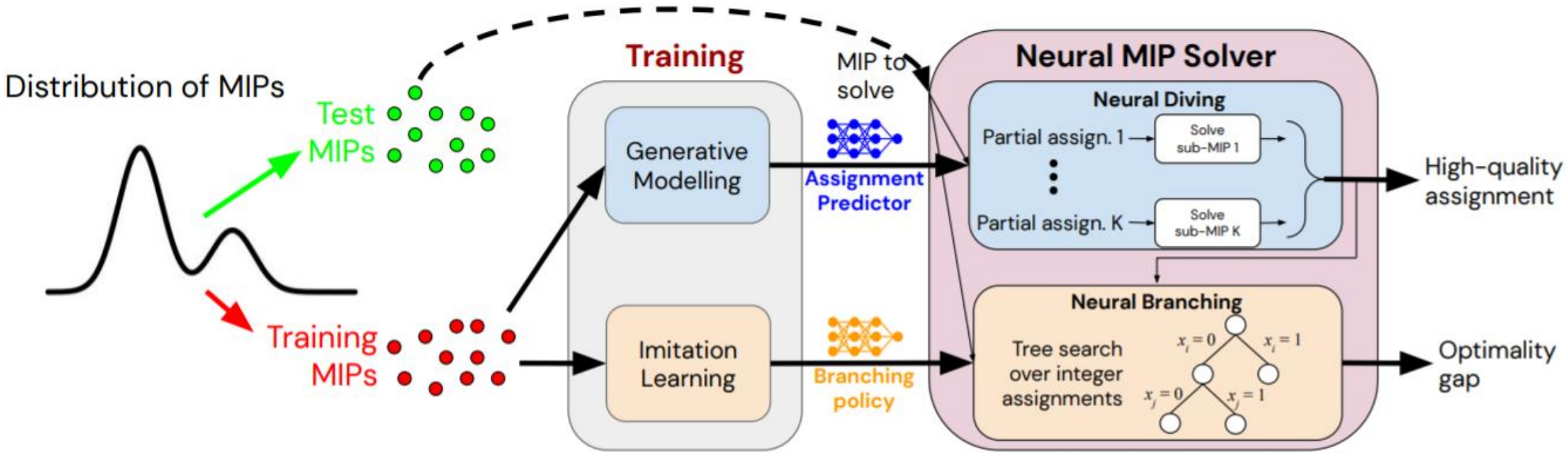
1. generates a feasible solution.
2. iteratively updates the solution
   - with an improvement operator selected by an RL-based controller
   - with a perturbation operator chosen by a rule-based controller.
3. choose the best one among all visited solutions.



Policy network



*NN + Google MIP Solver



Neural Diving — High quality joint variable assignments ▶ smaller 'sub-MIPs'

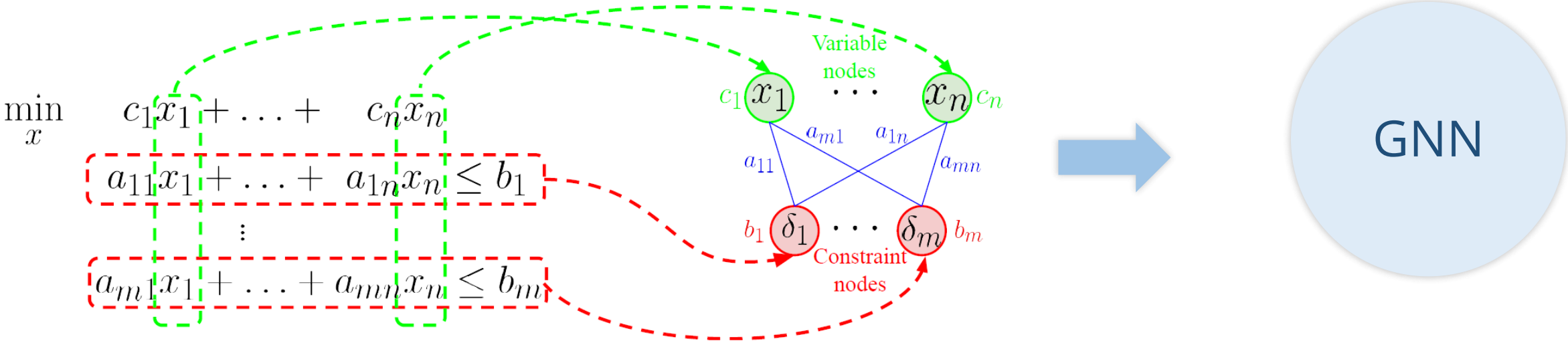
Neural Branching — Bound the gap between the objective value of the best assignment a an optimal one

Vinod Nair et al. "Solving Mixed Integer Programs Using Neural Networks". arXiv, 2021

*MIP -> Bi-Graph -> GNN

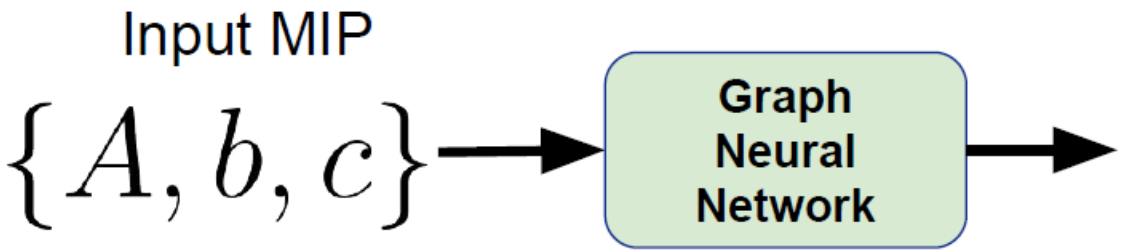
Convert a mixed integer program into a bipartite graph

Use graphNets for learning

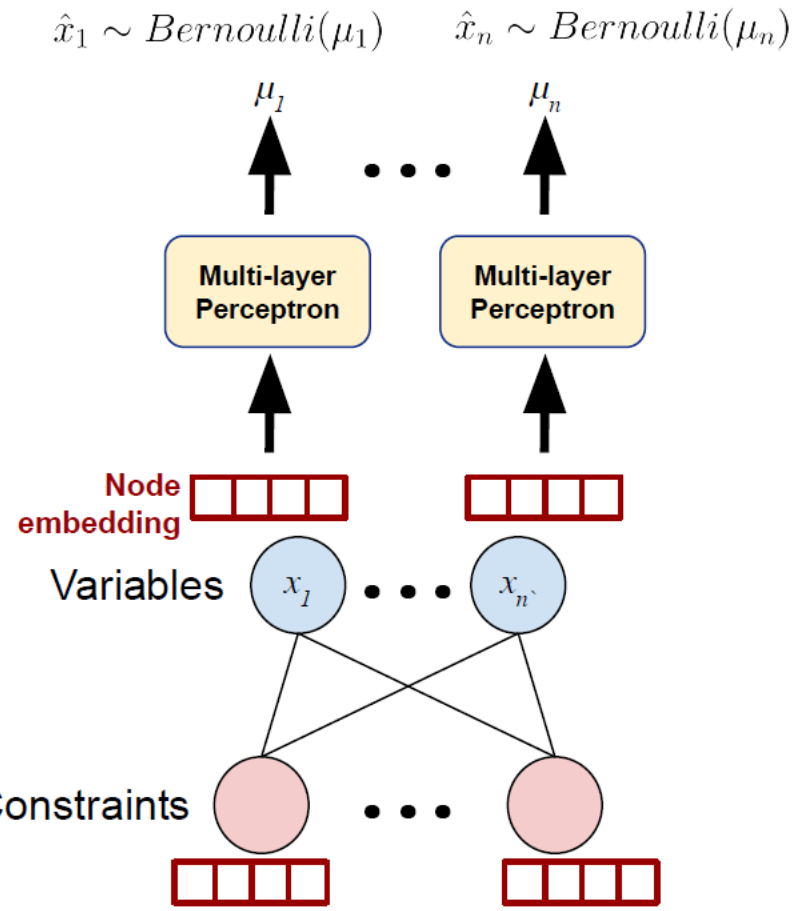


*Neural Diving

Learn a generative model of feasible assignments of discrete variables x given a MIP $G = \{A, b, c\}$



Generative Model

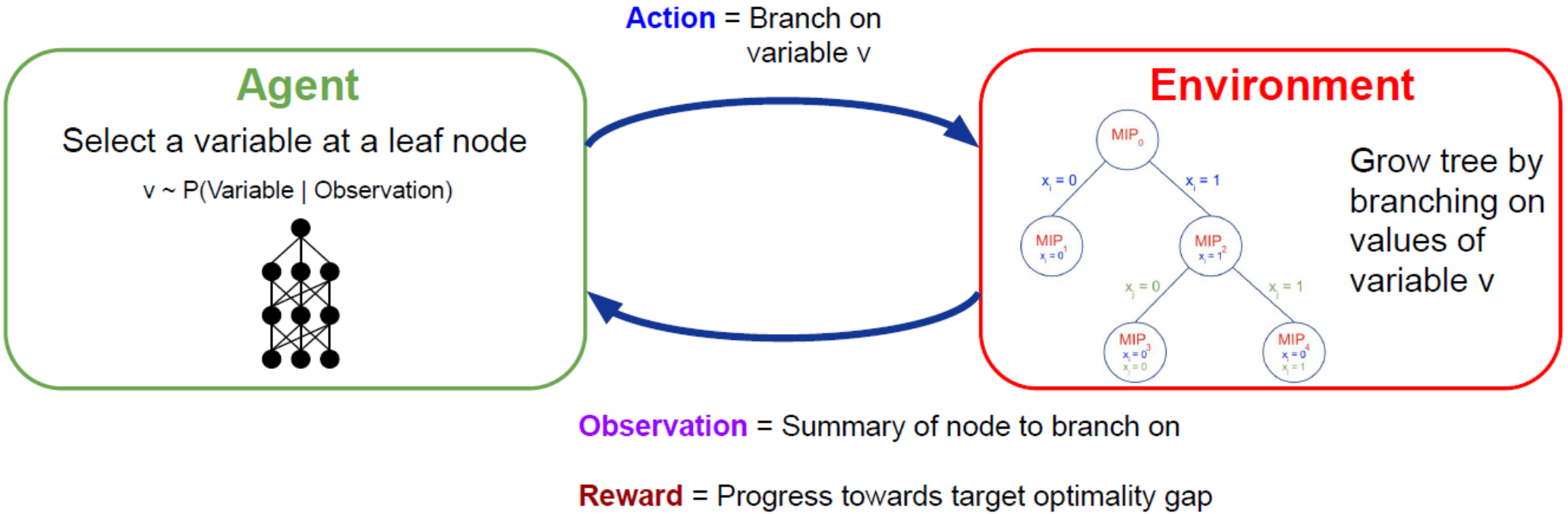


Supervised Learning

Train the model on (MIP, assignment) pairs generated using an existing solver

*Neural Branching

Branching as a sequential decision problem



Imitation Learning

Expert: Full strong branching + Alternating Direction Method of Multipliers

Methods: Behavioral Cloning + Dataset Aggregation

Contents

ML for CO

Development of ML4CO

Neural Network Architecture

NeurIPS 2015	NeurIPS 2017	NeurIPS 2018	ICLR 2019	AAAI 2021
PTR-NET	S2V-DQN	Ptr+GCN+AM	AM	MDAM
+ Seq2Seq	+ Graph Embedding	+ GNN & Attention	+ Multi-head Attention	+ Transformer

Reinforcement Learning methods

NeurIPS 2014	ICLR 2017	arXiv 2018	AAAI 2020 (Workshop)	NeurIPS 2020
L2B	Google-Bello	R2	GPT	MIT-Delarue
+ IL	+ RL	+ MCTS	+ Hierarchical RL	+ Combinatorial Actions

ML alongside Traditional Algorithms

NeurIPS 2014	EJOR 2016	IJCAI 2017	ICML 2020	arXiv 2021
L2B	DASH	Gatech-Khalil	CMU-Tang	Google
+ B&B node selection	+ B&B variable selection	+ Primal heuristic selection	+ B&B cut plane	+ MIP Solver

Papers Accepted by Top Conferences recently

Ayya Alieva et al. “Learning to Make Decisions via Submodular Regularization”. In ICLR, 2021

Tianshu Yu et al. “Deep Latent Graph Matching”. In ICML, 2021.

Quentin Cappart et al. “Combining Reinforcement Learning and Constraint Programming for Combinatorial Optimization”. In AAAI, 2021.

Liang Xin et al. “Multi-Decoder Attention Model with Embedding Glimpse for Solving Vehicle Routing Problems”. In AAAI, 2021.

Hang Zhao et al. “Online 3D Bin Packing with Constrained Deep Reinforcement Learning”. In AAAI, 2021.

Cong Zhang et al. “Learning to Dispatch for Job Shop Scheduling via Deep Reinforcement Learning”. In NeurIPS, 2020.

Sahil Manchanda et al. “ GCOMB: Learning Budget-constrained Combinatorial Algorithms over Billion-sized Graphs”. In NeurIPS, 2020.

Arthur Delarue et al. “Reinforcement Learning with Combinatorial Actions: An Application to Vehicle Routing”. In NeurIPS, 2020.

Marin Vlastelica Pogančić et al. “Differentiation of Blackbox Combinatorial Solver”. In ICLR, 2020.
Hao Lu et al. “A Learning-based Iterative Method for Solving Vehicle Routing Problems ”. In ICLR, 2020.

Papers Rejected by NIPS 20, ICLR 21, ICML 21

Solving NP-Hard Problems on Graphs with Extended AlphaGo Zero

Embedding a random graph via GNN: mean-field inference theory and RL applications to NP-Hard multi-robot/machine scheduling

A Graph Neural Network Assisted Monte Carlo Tree Search Approach to Traveling Salesman Problem

Learning to Search for Fast Maximum Common Subgraph Detection

ScheduleNet: Learn to Solve MinMax mTSP Using Reinforcement Learning with Delayed Reward

Neural Subgraph Matching

Rethinking Graph Neural Networks for Graph Coloring

Rewriting by Generating: Learn Heuristics for Large-scale Vehicle Routing Problems

Learning a Transferable Scheduling Policy for Various Vehicle Routing Problems based on Graph-centric Representation Learning

Learning Latent Topology for Graph Matching

Summary of Reviewers' comments

Novelty

Performance	Solution quality	Inference speed
Scalability	Large-scale	Training efficiency
Generalizability	Various COPs	Different constraints
Analyticity	Model architecture	Training method

Conclusion

Challenges	Possible Research Directions
Problem Formulation	MDP; Dynamic and augmenting state input; Action output
Feature extraction	Powerful NN (Attention/ GNN/ Transformer/ GAN?/...)
Additional constraints	Relaxation; Task-specific; Multi-agent RL
Multi-objective	Reward design; Hierarchical RL; Multi-task Learning
Model generalization	Integrate traditional methods; Divide-and-conquer; Iteratively

Thanks

Machine Learning for Combinational Optimization

Tianfu Wang

2021/09/03