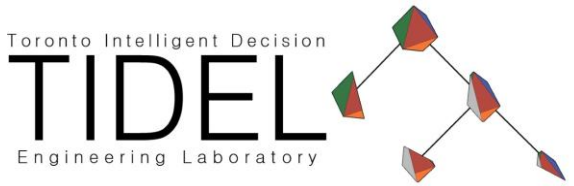
Reinforcement Learning-based Heuristics to Guide Domain-Independent Dynamic Programming

Minori Narita[1], **Ryo Kuroiwa[2]**, J. Christopher Beck[1]

[1]University of Toronto, Canada

[2]National Institute of Informatics, Japan



Domain-Independent Dynamic Programming (DIDP)

(Kuroiwa and Beck [ICAPS2023a, ICAPS2023b])

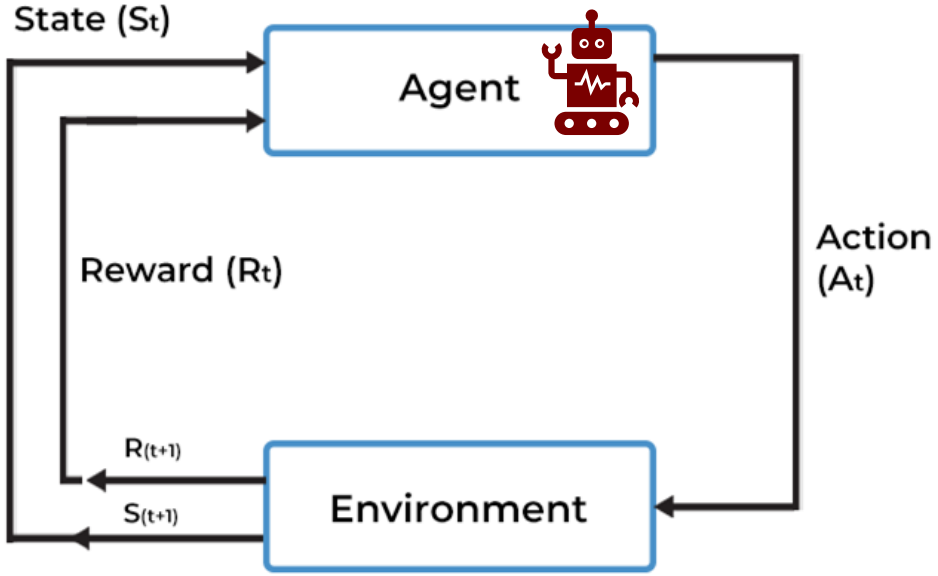
- Model & Solve framework based on dynamic programming (DP) for combinatorial optimization (CO)
- Outperformed MIP and CP across several domains in CO problems

	MIP		CP		CABS	
	c.	m.	c.	m.	c.	m.
TSPTW (340)	224	0	47	0	**259**	0
CVRP (207)	**28**	5	0	0	6	0
m-PDTSP (1178)	940	0	**1049**	0	1035	0
OPTW (144)	16	0	49	0	**64**	0
MDKP (276)	**168**	0	6	0	5	1
Bin Packing (1615)	1160	0	**1234**	0	1167	4
SALBP-1 (2100)	1431	250	1584	0	**1802**	0
$1\|\|\sum w_i T_i$ (375)	107	0	150	0	**288**	0
Talent Scheduling (1000)	0	0	0	0	**239**	0
MOSP (570)	241	14	437	0	**527**	0
Graph-Clear (135)	26	0	4	0	**103**	19

Number of instances proved to optimality (Kuroiwa and Beck, 2025)

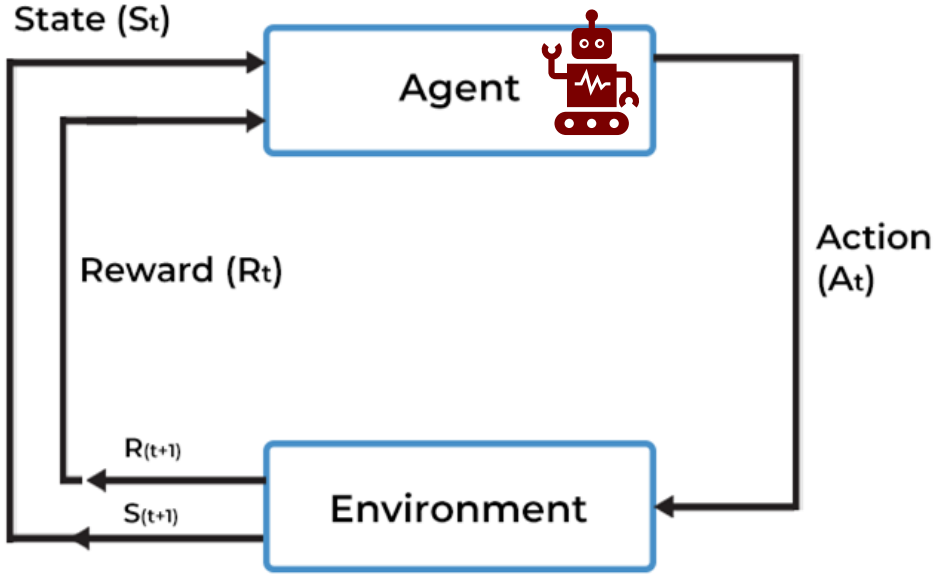
Reinforcement Learning for Combinatorial Optimization

- RL shares several key structures with DP
  - Bellman equation
  - State-based transition systems



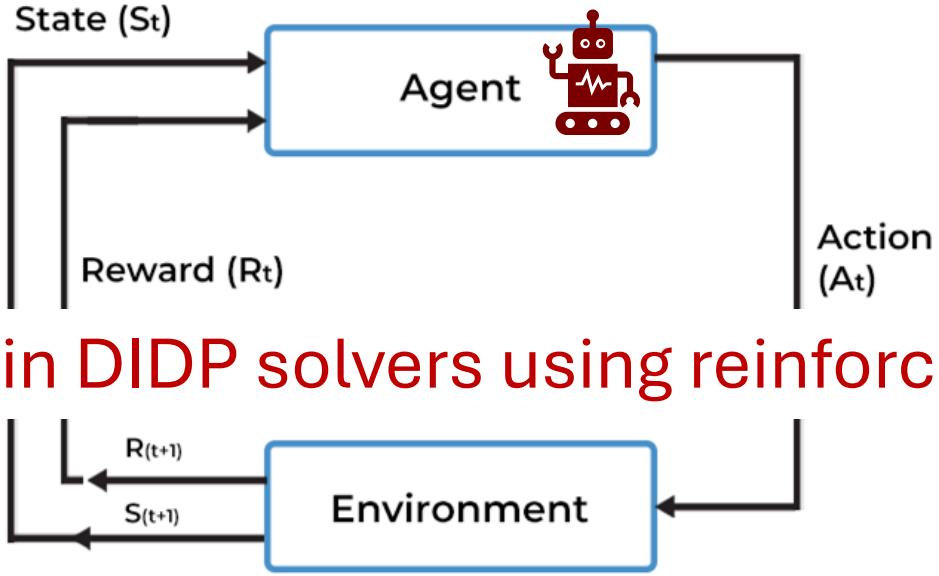
Reinforcement Learning for Combinatorial Optimization

- RL shares several key structures with DP
  - Bellman equation
  - State-based transition systems
- Cappart et al. [AAAI 2021] formulated optimization problems as DP to enable RL-based guidance in CP solvers
  - CP is solving the DP formulations



Reinforcement Learning for Combinatorial Optimization

- RL shares several key structures with DP
  - Bellman equation
  - State-based transition systems
- Cappart et al. [AAAI 2021] formulated optimization problems as DP to enable RL-based guidance in CP solvers
  - CP is solving the DP formulations



Can we guide search in DIDP solvers using reinforcement learning (RL)?

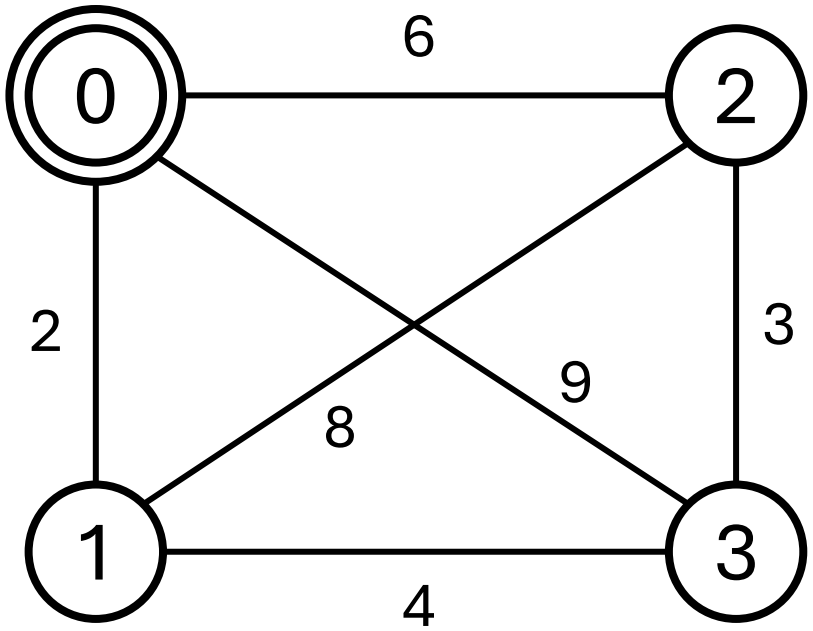
Domain-Independent Dynamic Programming (DIDP)

- The value function $V(s)$ is recursively defined at each state $s \in S$
- Optimal cost is $V(s_0)$, where s_0 is the initial state

Combinatorial Optimization

Traveling Salesperson Problem (TSP)

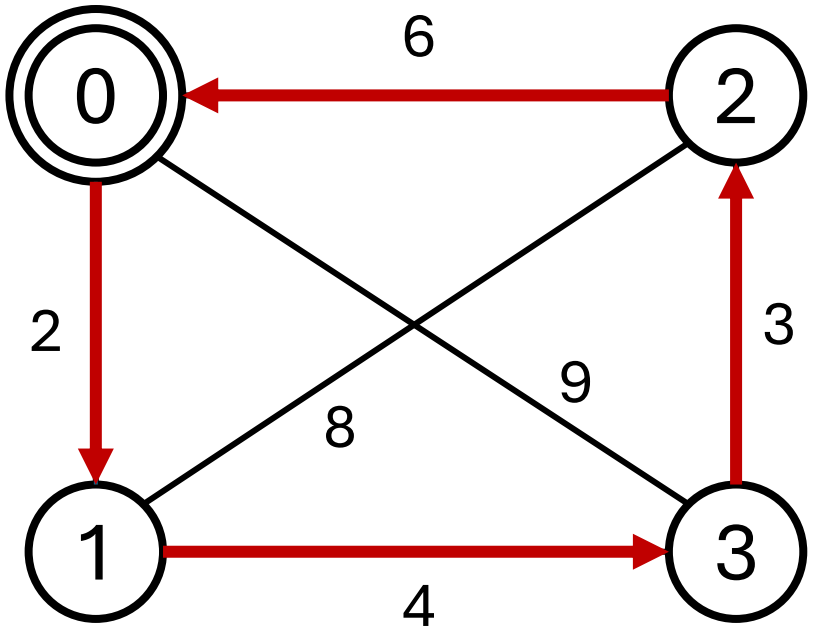
- Minimize the total traveling distance, visiting all customers
- Start at the depot (index 0) and return to the depot



Combinatorial Optimization

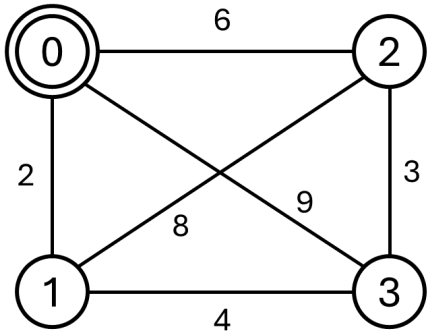
Traveling Salesperson Problem (TSP)

- Minimize the total traveling distance, visiting all customers
- Start at the depot (index 0) and return to the depot



Total distance: **15**

DP Model for TSP



compute $V(N \setminus \{0\}, 0)$

$$V(U,i) = \begin{cases} c_{i0} & \text{If } U = \emptyset \quad \text{Return to depot (Base case)} \\ \min\limits_{j \in U}(c_{ij} + V(U \setminus \{j\}, j)) & \text{If } U \neq \emptyset \quad \text{Visit customer } j \text{ (Transition)} \end{cases}$$

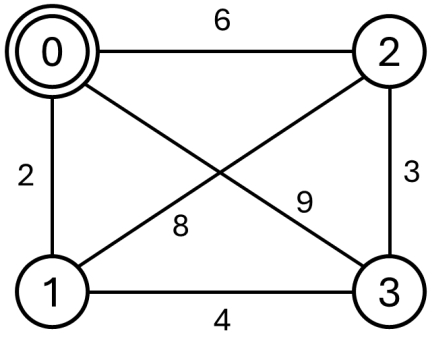
State variables:

- U: unvisited customer set
- i: current location

Constants

- N: set of customers $\{0 \ldots n-1\}$
- c_{ij}: distance from customer i to j

DP Model for TSP



compute $V(N \setminus \{0\}, 0)$

$$V(U,i) = \begin{cases} c_{i0} & \text{If } U = \emptyset \quad \text{Return to depot (Base case)} \\ \min_{j \in U}(c_{ij} + V(U \setminus \{j\}, j)) & \text{If } U \neq \emptyset \quad \text{Visit customer } j \text{ (Transition)} \end{cases}$$

$V(U,i) \geq \sum_{j \in U \cup \{0\}} \min_{k \in N \setminus \{j\}} c_{jk}$ Dual bound function (redundant information)

State variables:

- U: unvisited customer set
- i: current location

Constants

- N: set of customers $\{0 \ldots n-1\}$
- c_{ij}: distance from customer i to j

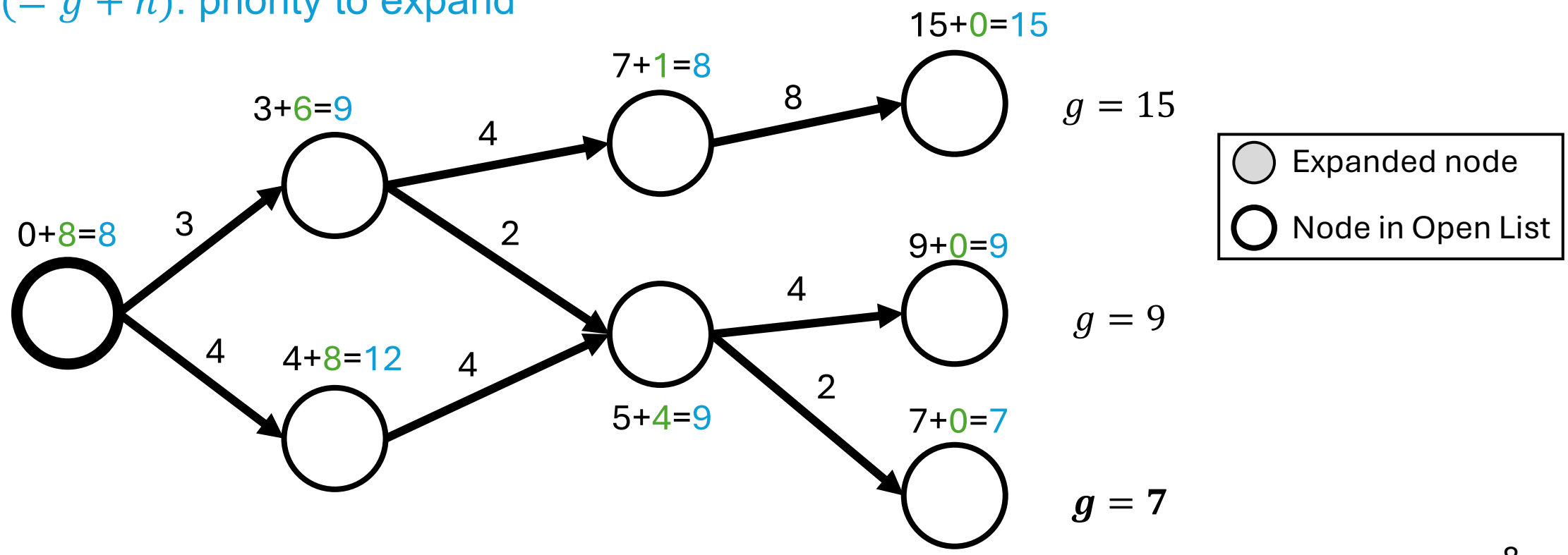
Heuristic State Space Search

Find a path by expanding states in the order of f-values

g-values: actual path cost to a state from the initial state

h-values: estimated path cost to a base case (dual bound in the current DIDP)

f-values: $(= g + h)$: priority to expand



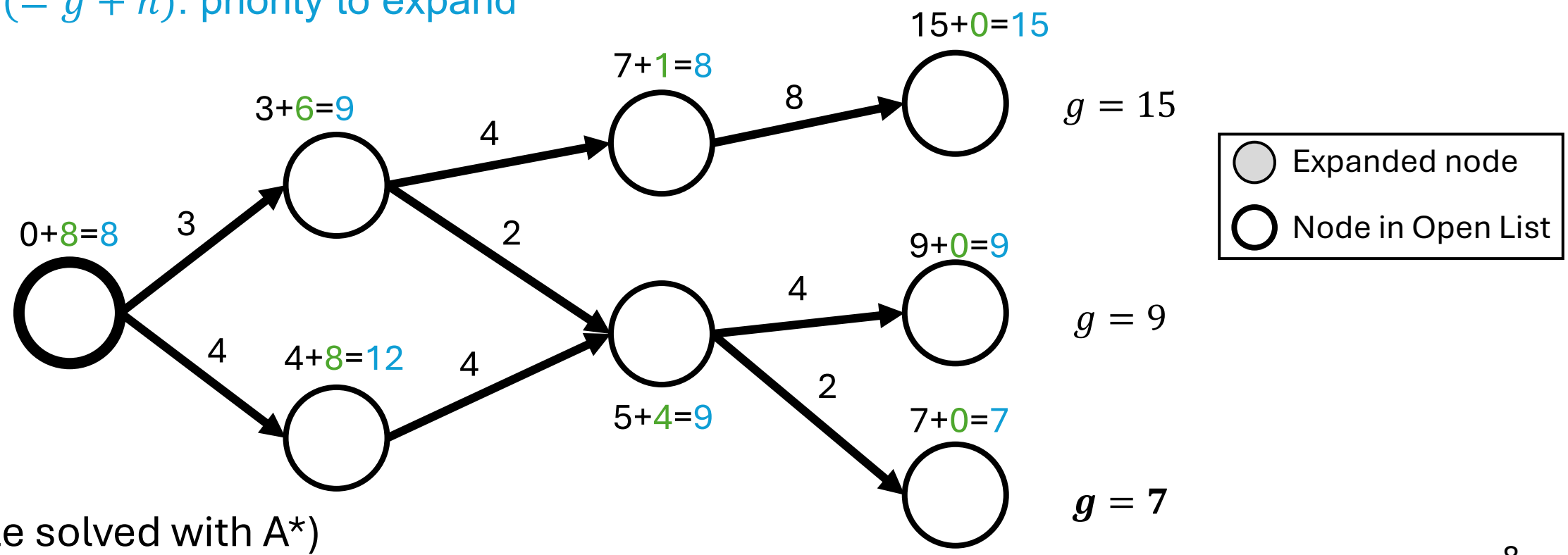
Heuristic State Space Search

Find a path by expanding states in the order of f-values

g-values: actual path cost to a state from the initial state

h-values: estimated path cost to a base case (dual bound in the current DIDP)

f-values: $(= g + h)$: priority to expand



(Example solved with A*)

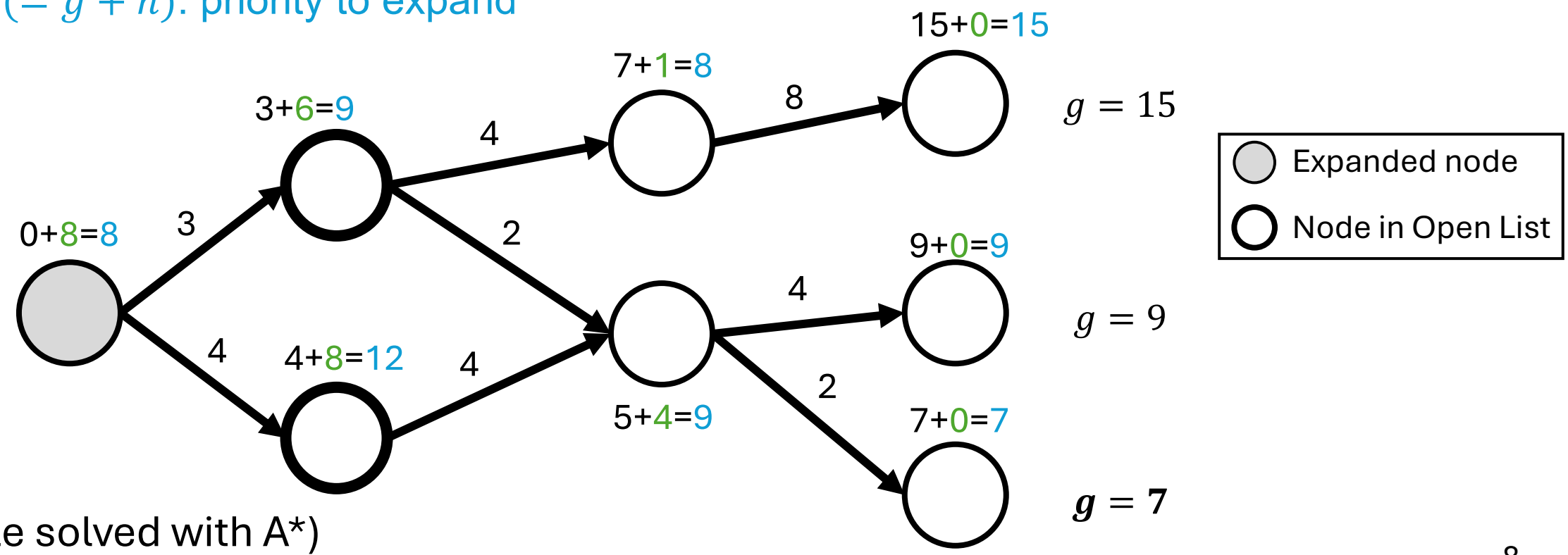
Heuristic State Space Search

Find a path by expanding states in the order of f-values

g-values: actual path cost to a state from the initial state

h-values: estimated path cost to a base case (dual bound in the current DIDP)

f-values: $(= g + h)$: priority to expand



(Example solved with A*)

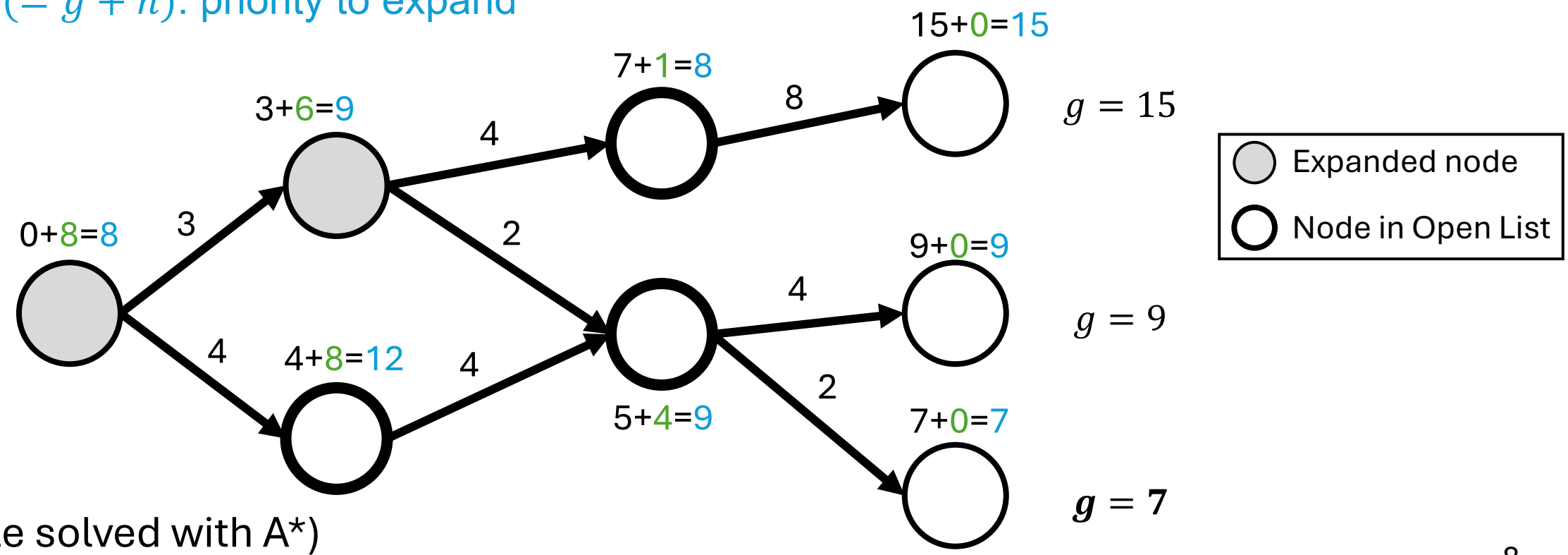
Heuristic State Space Search

Find a path by expanding states in the order of f-values

g-values: actual path cost to a state from the initial state

h-values: estimated path cost to a base case (dual bound in the current DIDP)

f-values: $(= g + h)$: priority to expand



(Example solved with A*)

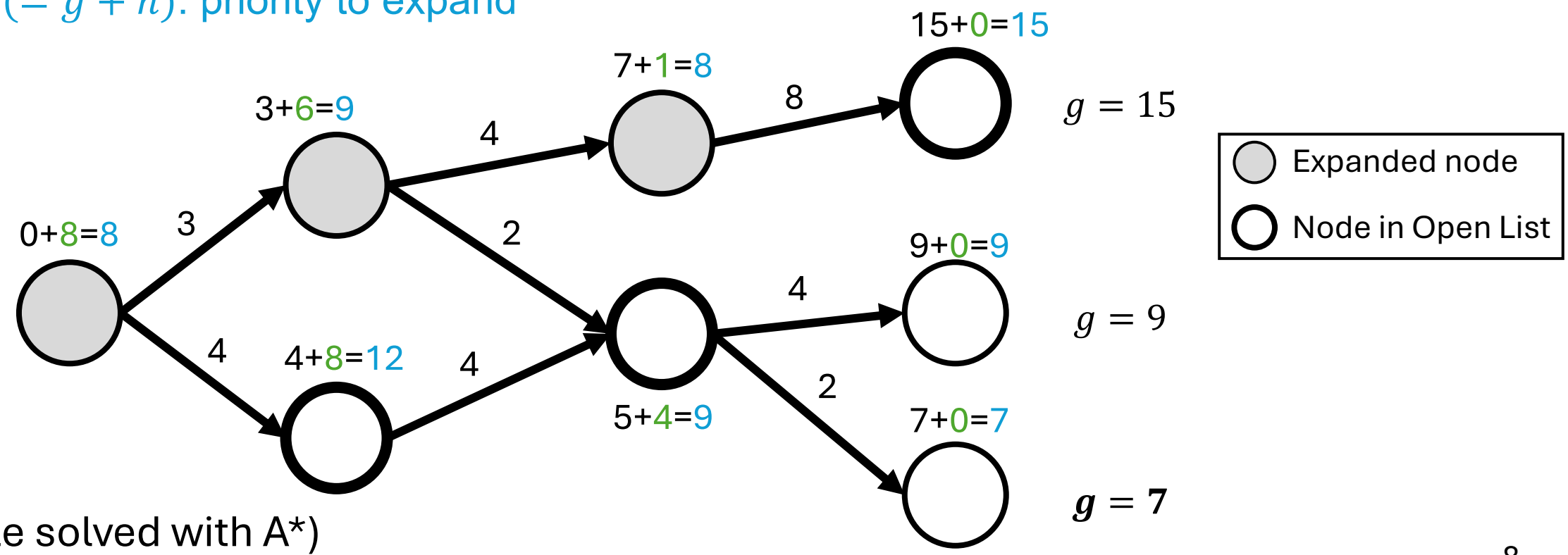
Heuristic State Space Search

Find a path by expanding states in the order of f-values

g-values: actual path cost to a state from the initial state

h-values: estimated path cost to a base case (dual bound in the current DIDP)

f-values: $(= g + h)$: priority to expand



(Example solved with A*)

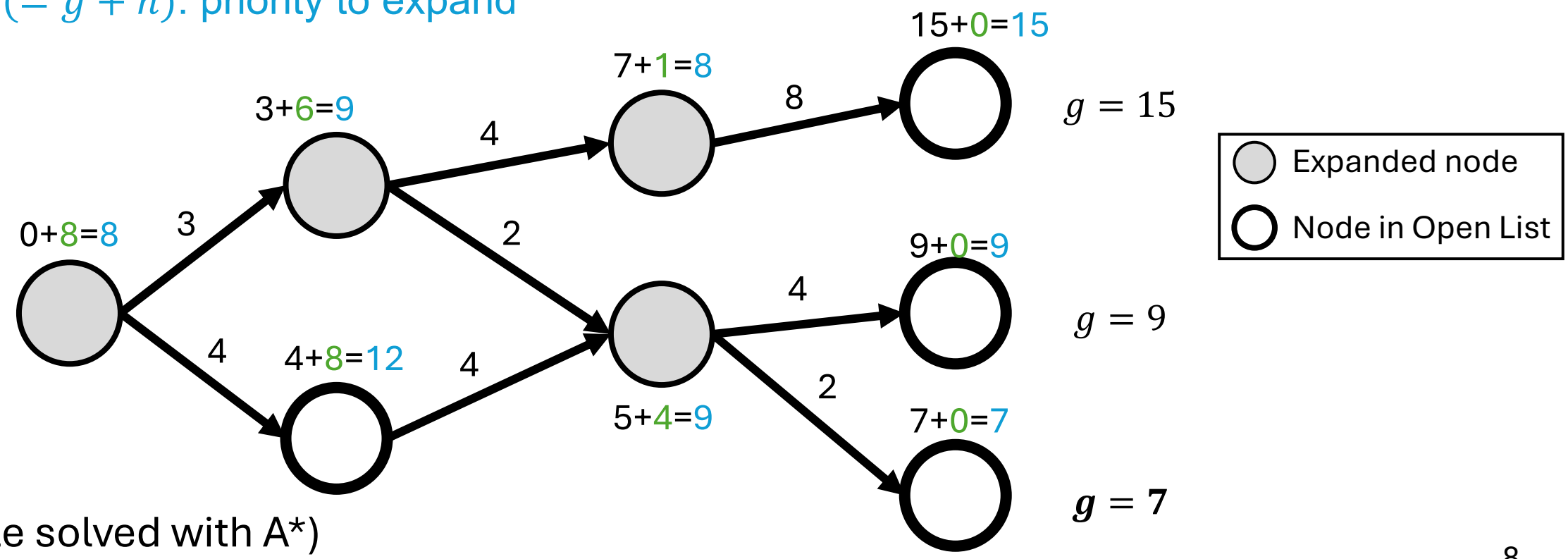
Heuristic State Space Search

Find a path by expanding states in the order of f-values

g-values: actual path cost to a state from the initial state

h-values: estimated path cost to a base case (dual bound in the current DIDP)

f-values: $(= g + h)$: priority to expand



(Example solved with A*)

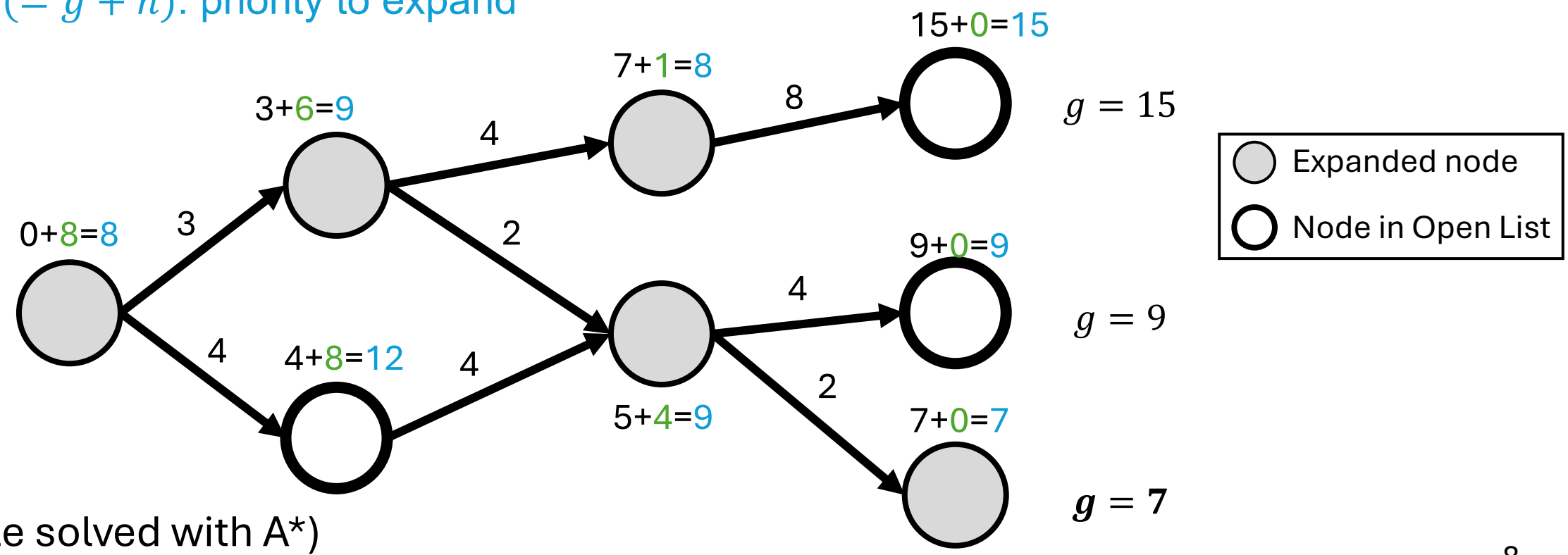
Heuristic State Space Search

Find a path by expanding states in the order of f-values

g-values: actual path cost to a state from the initial state

h-values: estimated path cost to a base case (dual bound in the current DIDP)

f-values: $(= g + h)$: priority to expand



(Example solved with A*)

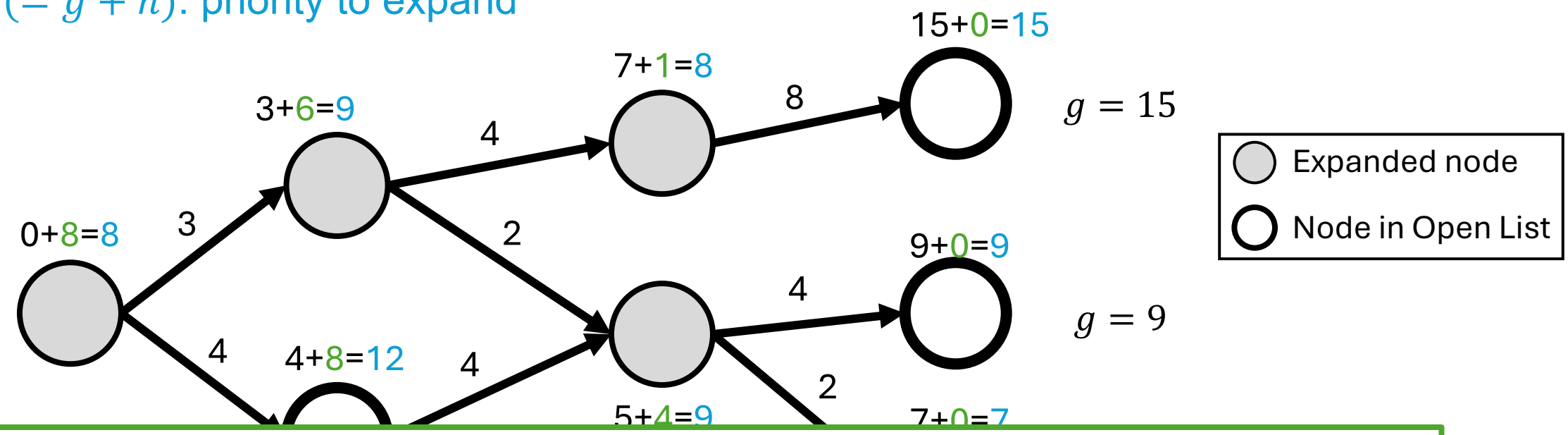
Heuristic State Space Search

Find a path by expanding states in the order of f-values

g-values: actual path cost to a state from the initial state

h-values: estimated path cost to a base case (dual bound in the current DIDP)

f-values: $(= g + h)$: priority to expand



Good h-values will help quickly find good solutions

- A user-defined dual bound function is not necessarily informative

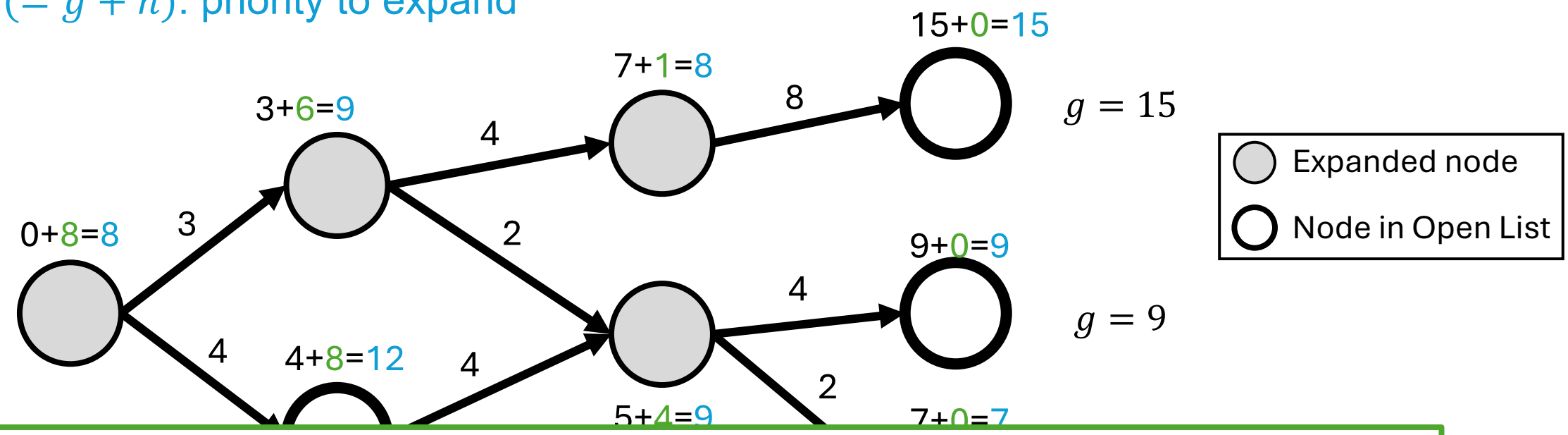
Heuristic State Space Search

Find a path by expanding states in the order of f-values

g-values: actual path cost to a state from the initial state

h-values: estimated path cost to a base case (dual bound in the current DIDP)

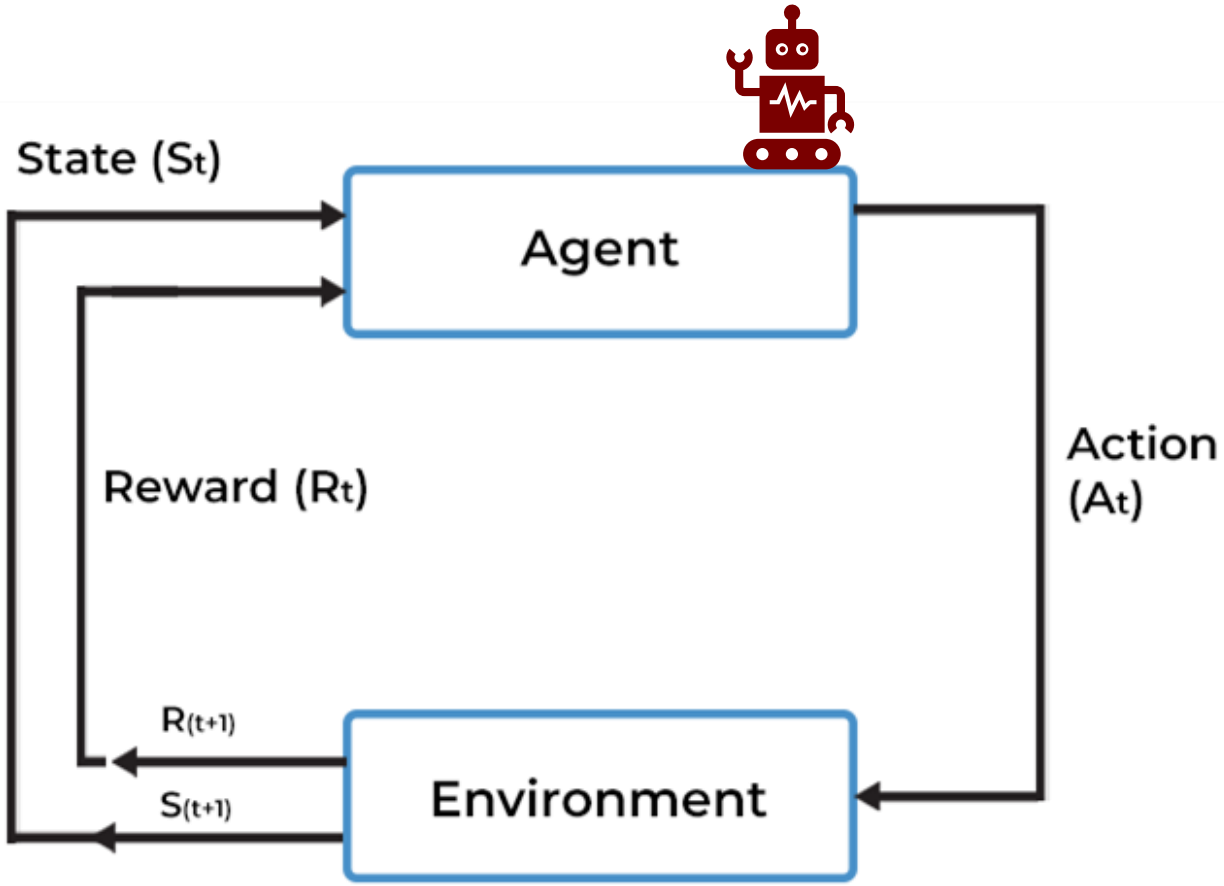
f-values: $(= g + h)$: priority to expand



Good h-values will help quickly find good solutions

- A user-defined dual bound function is not necessarily informative
- **Can we *learn* them through RL?**

Reinforcement Learning (RL)



Mapping from a DIDP Model to an RL Model

DP model

State variables: $\{U, i\}$
- U: unvisited customer set
- i: current location

Transition τ
- Visit customer j ($j \in U$)
  - $U \leftarrow U \setminus \{j\}, i \leftarrow j$

Transition cost: c_{ij}

RL model

State s: $\{U, i\}$

Same as DP

Action a:
Visit customer j

$a \leftarrow \tau$

Transition function $T(s, a)$:
Visit customer j ($j \in U$)

$$T(\{U, i\}, j) = \{U \setminus \{j\}, j\}$$

Effect and precondition of τ

Reward function:

$$R(s, a) = -c_{ia}$$

Negative cost of τ

Mapping from a DIDP Model to an RL Model

DP model

State variables: $\{U, i\}$
- U: unvisited customer set
- i: current location

Transition τ
- Visit customer j ($j \in U$)
  - $U \leftarrow U \setminus \{j\}, i \leftarrow j$

Transition cost: c_{ij}

RL model

State s: $\{U, i\}$

Same as DP

Action a:
Visit customer j

$a \leftarrow \tau$

Transition function $T(s, a)$:
Visit customer j ($j \in U$)

$$T(\{U, i\}, j) = \{U \setminus \{j\}, j\}$$

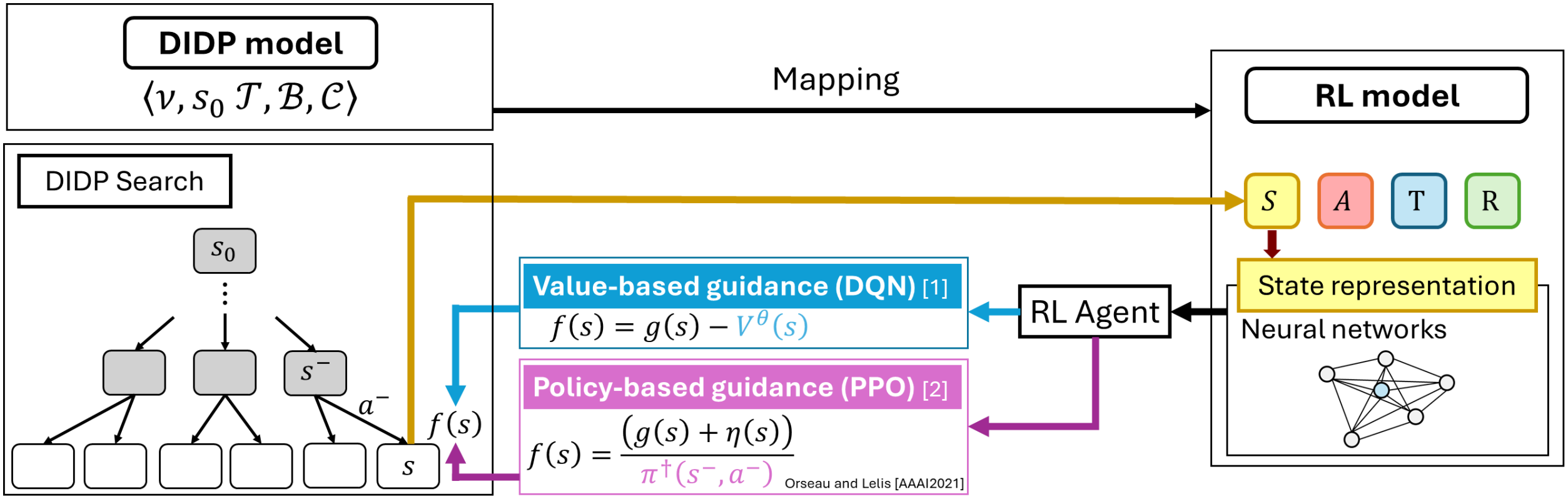
Effect and precondition of τ

Reward function:

$$R(s, a) = -c_{ia}$$

Negative cost of τ

RL-based Guidance

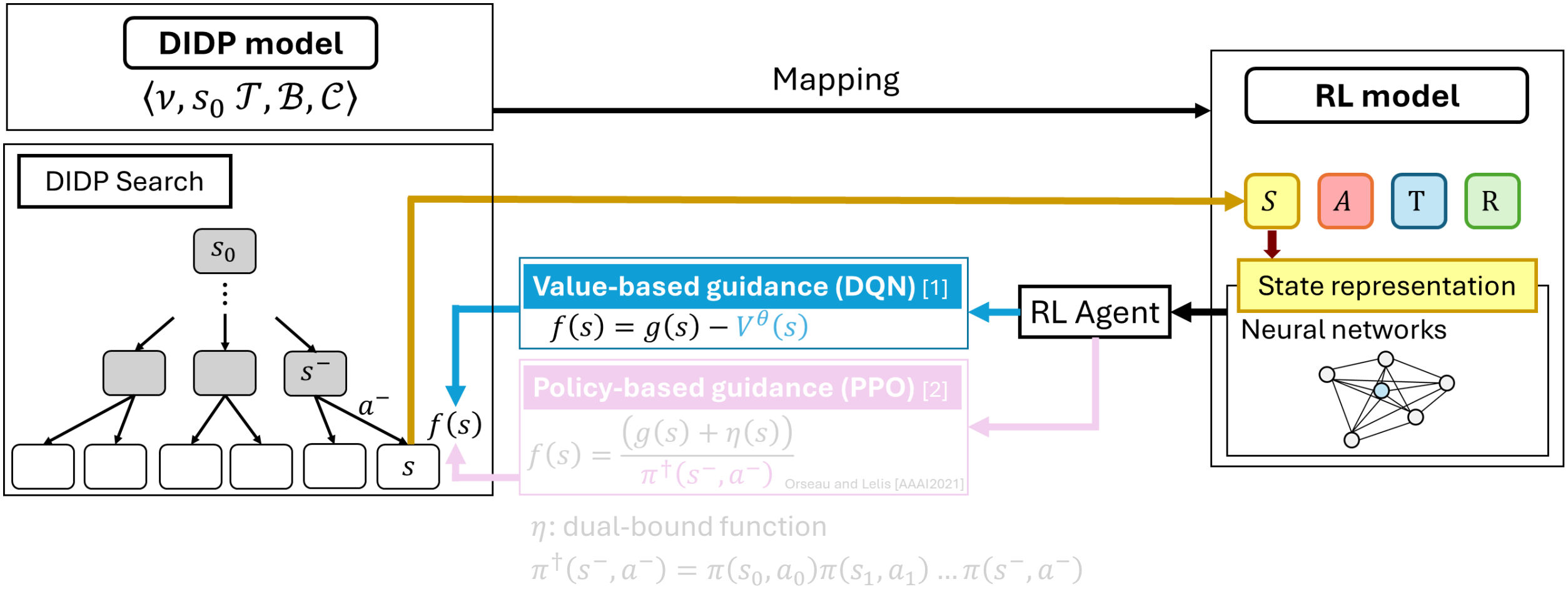


η: dual-bound function

$\pi^\dagger(s^-, a^-) = \pi(s_0, a_0)\pi(s_1, a_1) \ldots \pi(s^-, a^-)$

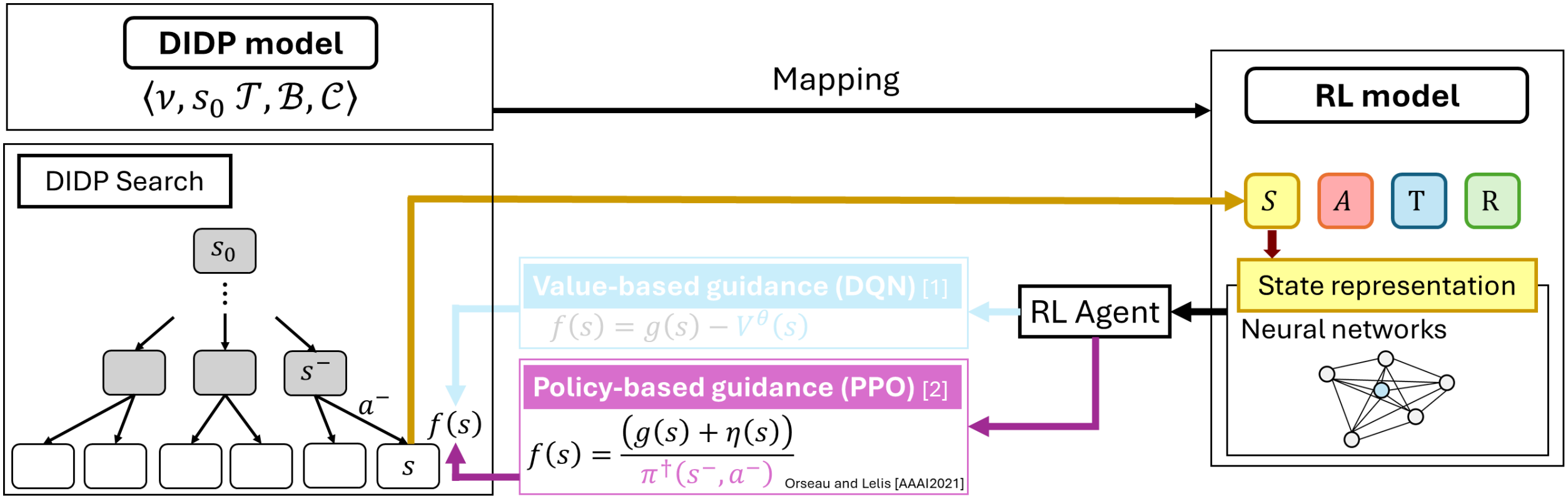
[1] Mnih et al., Nature, 2015.
[2] Schulman et al., ArXiv preprint, 2017.

Value-based Guidance



[1] Mnih et al., Nature, 2015.
[2] Schulman et al., ArXiv preprint, 2017.

Policy-based Guidance



η: dual-bound function

$\pi^{\dagger}(s^-, a^-) = \pi(s_0, a_0)\pi(s_1, a_1) \ldots \pi(s^-, a^-)$

[1] Mnih et al., Nature, 2015.
[2] Schulman et al., ArXiv preprint, 2017.

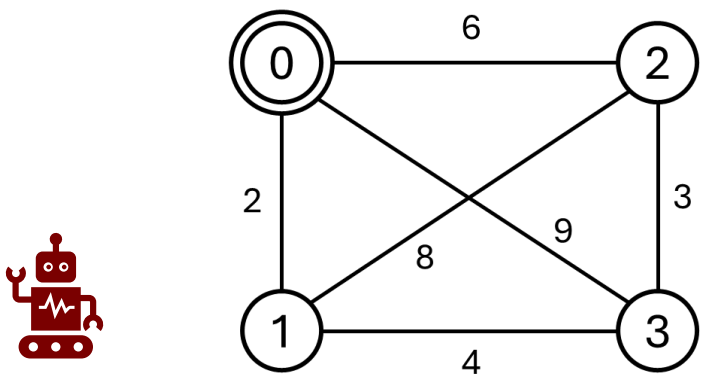
RL Training Process

Instance generation

Randomly generate an instance from a fixed distribution

Solve the instance

Follow the current policy π



Add experiences
$E = \{e_1, e_2, \ldots\}$
s.t. $e = \{s, a, s', r\}$

Experience replay buffer

Sample experiences

Update network parameters

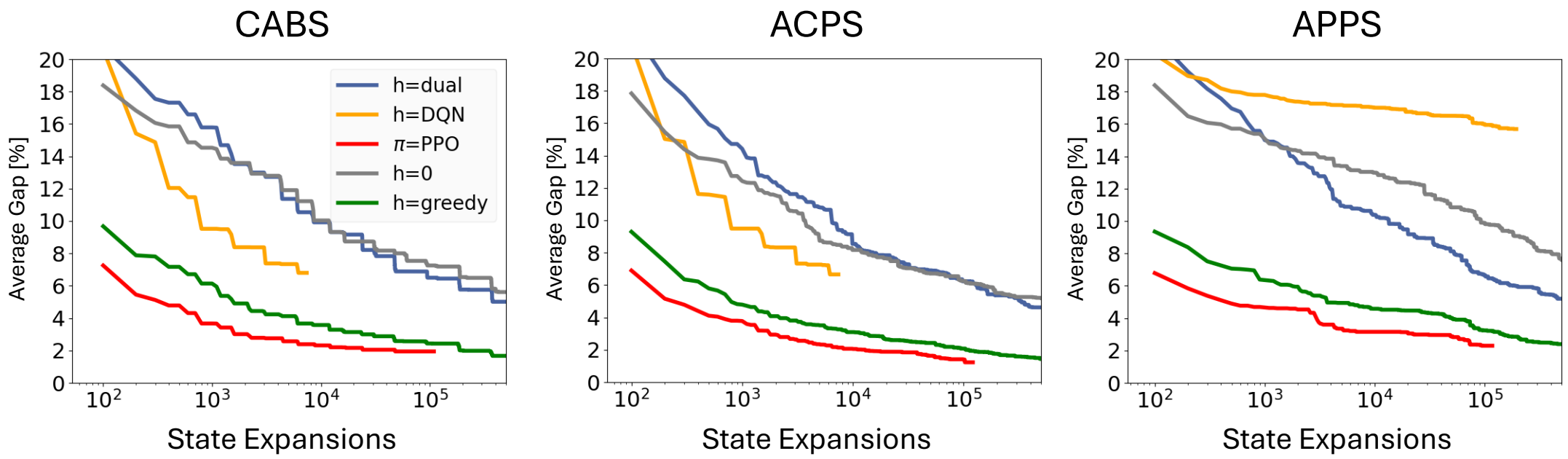
Experiment Settings

- Solvers [Kuroiwa and Beck, ICAPS 2023]
  - Complete Anytime Beam Search (CABS) [Zhang 1995]
  - Anytime Column Progressive Search (ACPS) [Vadlamudi et al. 2012]
  - Anytime Pack Progressive Search (APPS) [Vadlamudi et al. 2016]

Experiment Settings

- Solvers [Kuroiwa and Beck, ICAPS 2023]
  - Complete Anytime Beam Search (CABS) [Zhang 1995]
  - Anytime Column Progressive Search (ACPS) [Vadlamudi et al. 2012]
  - Anytime Pack Progressive Search (APPS) [Vadlamudi et al. 2016]
- Baselines
  - h=0
  - h=dual-bound (default implementation)
  - h=greedy (problem-specific)

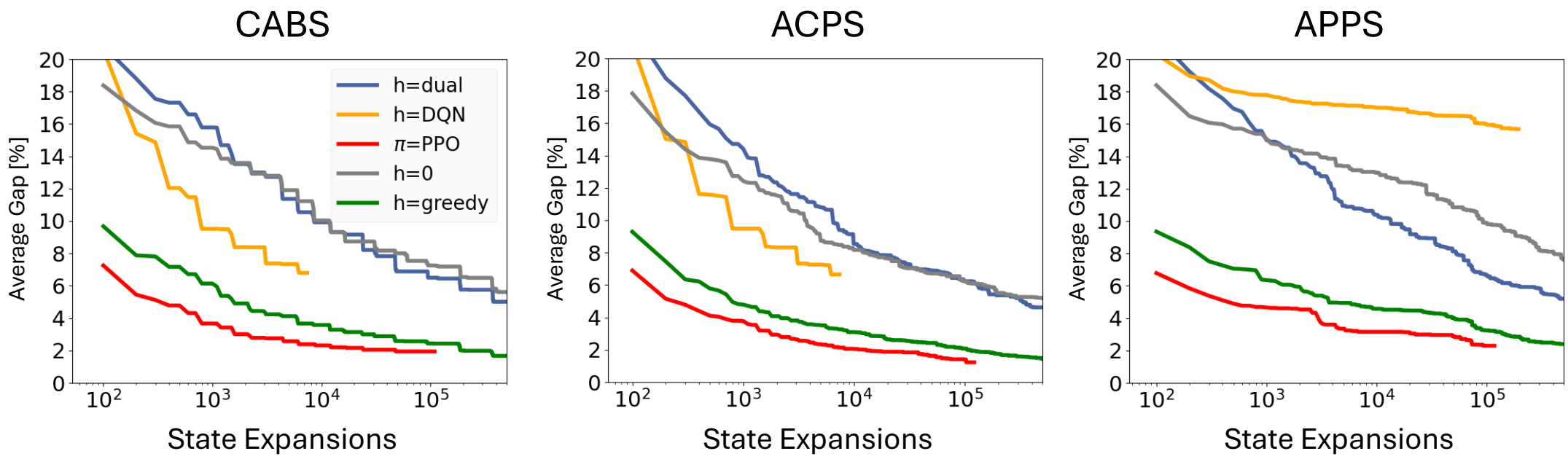
Result (TSP)


CABS
ACPS
APPS
Average Gap [%]
State Expansions
h=dual
h=DQN
π=PPO
h=0
h=greedy
0
2
4
6
8
10
12
14
16
18
20
10^2
10^3
10^4
10^5

Result (TSP)

$$\text{Average Gap [\%]} = \frac{1}{n}\sum_{i \in Instances} \frac{|x(i,m,l) - best(i)|}{best(i)} \times 100$$

- $x(i,m,l)$ is the solution cost of method m for instance i up to the node expansion limit l



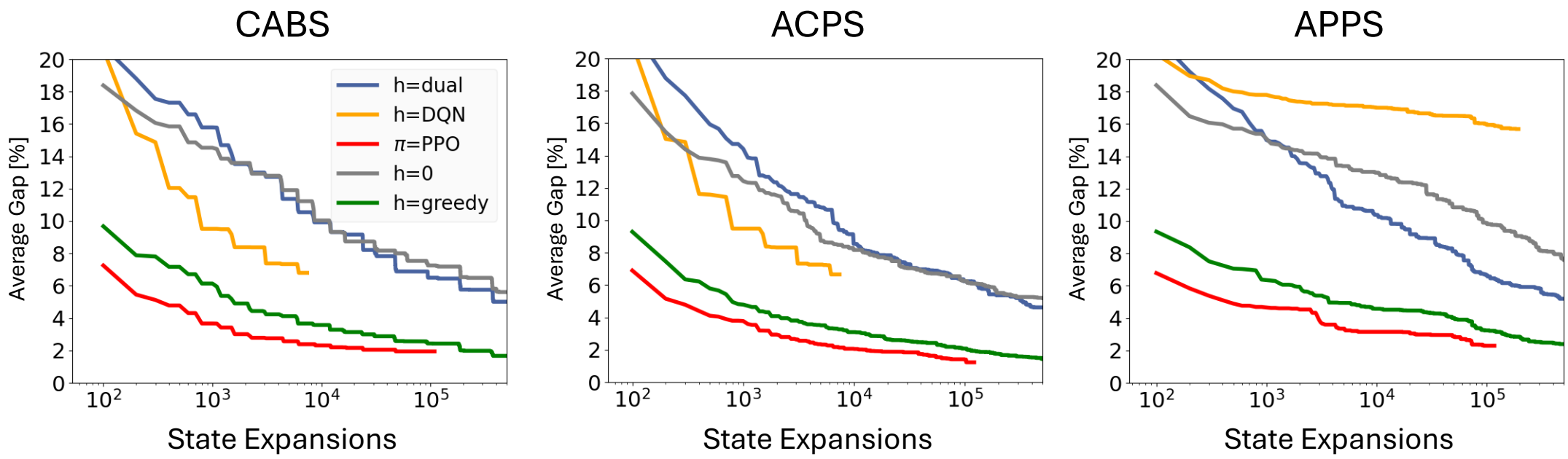
Result (TSP)

Solution cost

Best-known solution for i

$$\text{Average Gap [\%]} = \frac{1}{n}\sum_{i \in Instances} \frac{|x(i,m,l) - best(i)|}{best(i)} \times 100$$

- $x(i,m,l)$ is the solution cost of method m for instance i up to the node expansion limit l

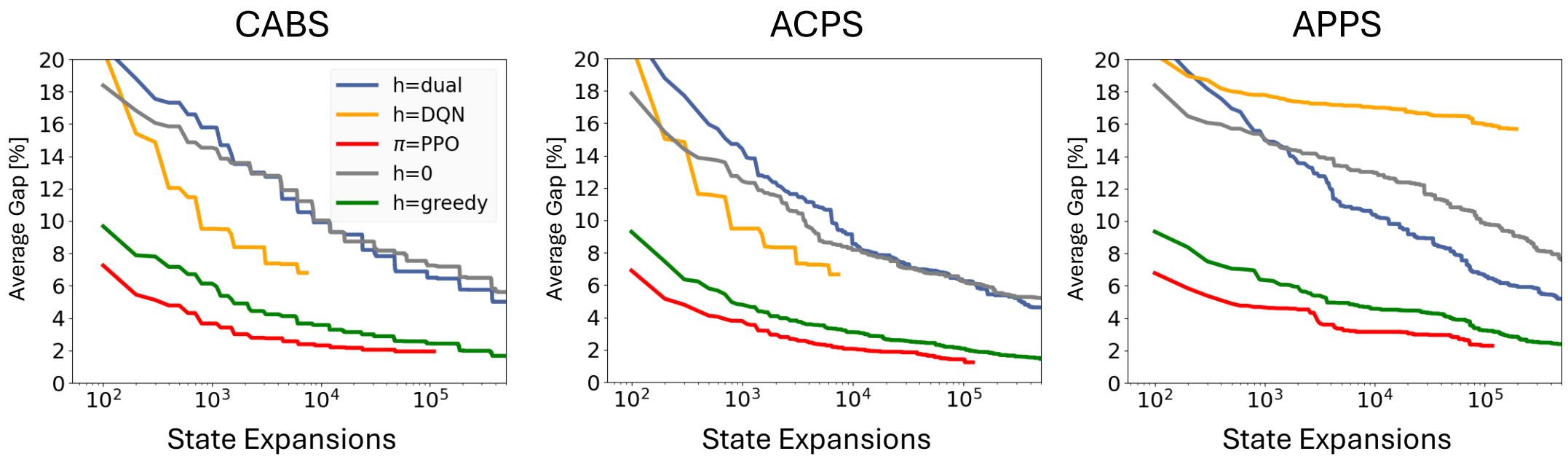


Result (TSP)

Solution cost — Best-known solution for i

$$\text{Average Gap [\%]} = \frac{1}{n}\sum_{i \in Instances} \frac{|x(i,m,l) - best(i)|}{best(i)} \times 100$$

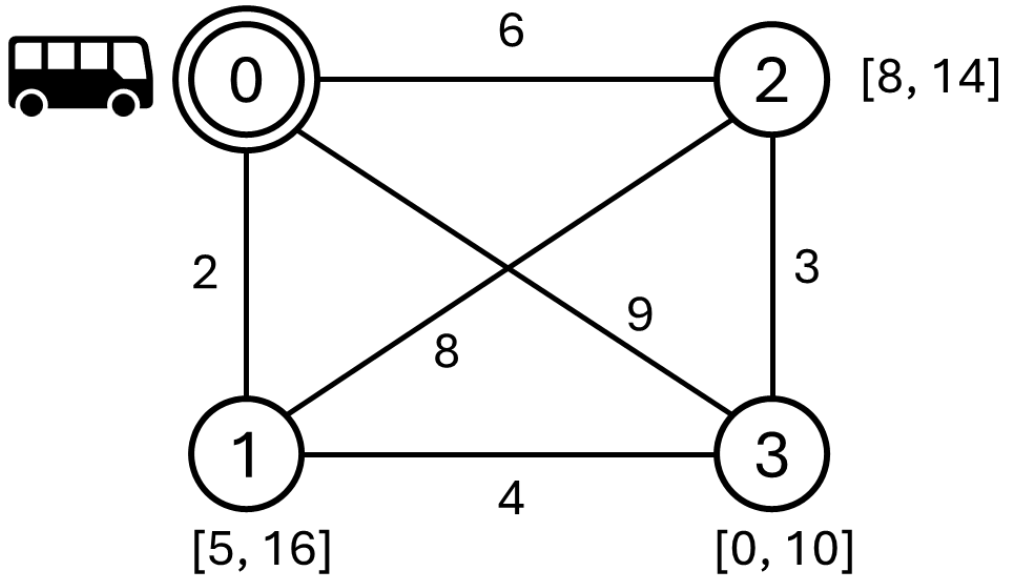
- $x(i,m,l)$ is the solution cost of method m for instance i up to the node expansion limit l



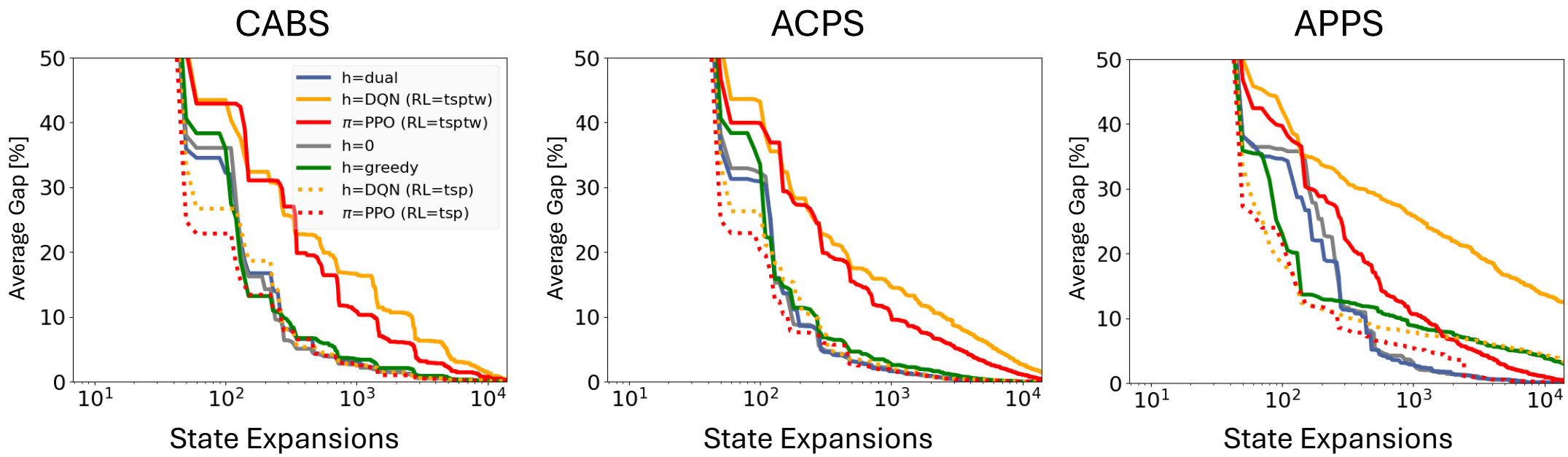
- PPO guidance significantly outperformed other heuristics
- DQN also outperformed the dual-bound guidance except APPS

TSPTW

- Minimize the travel time to visit all customers within time windows
- Many states can be pruned based on time window constraints



Result (TSPTW)

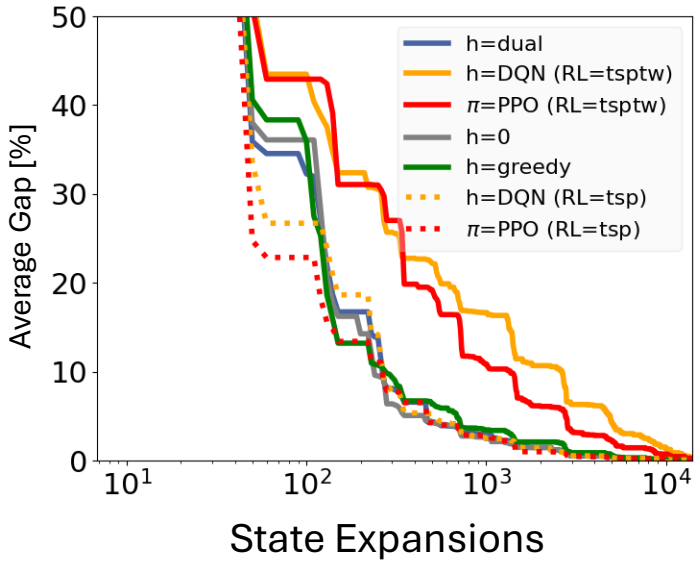


Result (TSPTW)

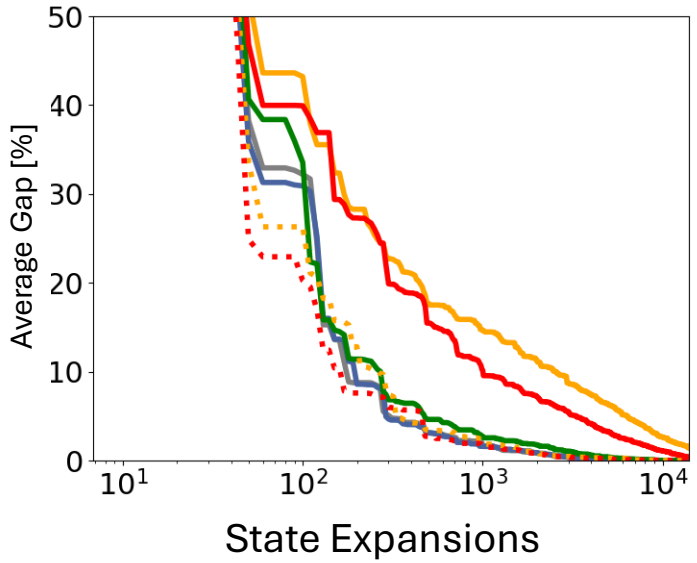
Average gap for pure RL

	TSP $n = 50$ (TSP RL)	TSPTW $n = 50$ (TSPTW RL)
DQN	20.14 [%]	(No feasible sol.)
PPO	3.85 [%]	46.93 [%]

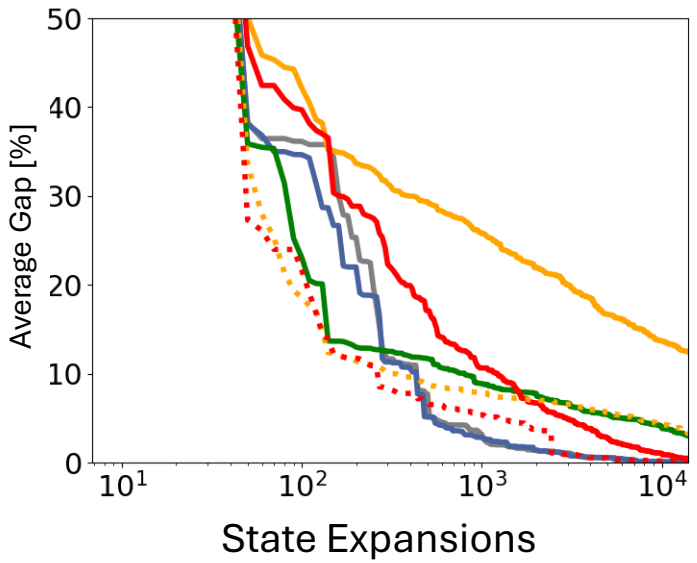
CABS



ACPS



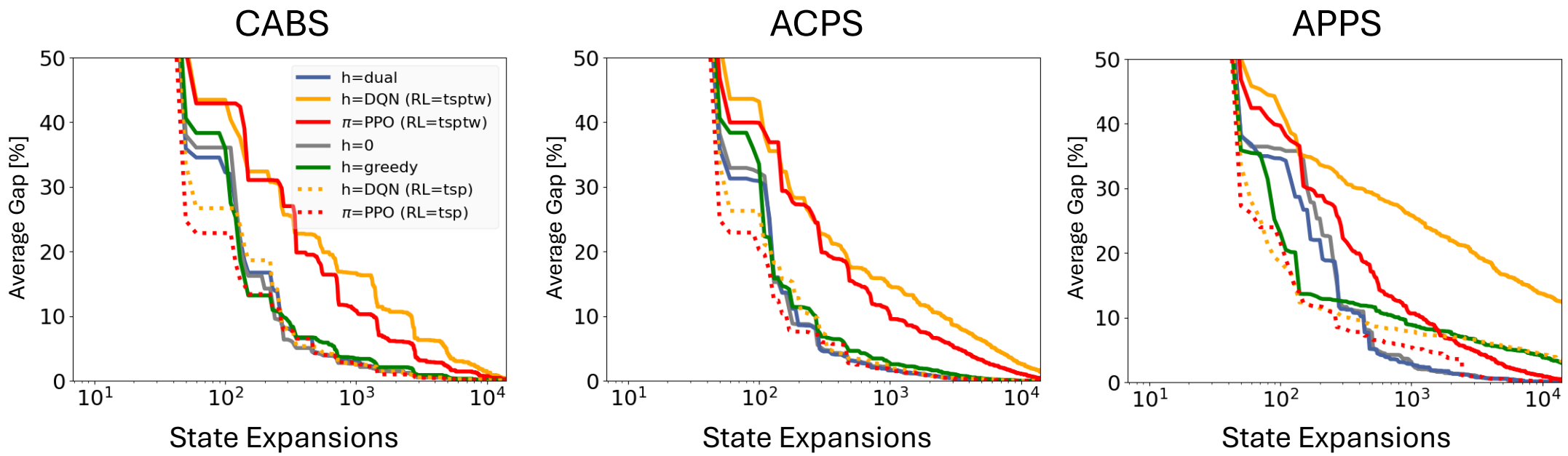
APPS



Result (TSPTW)

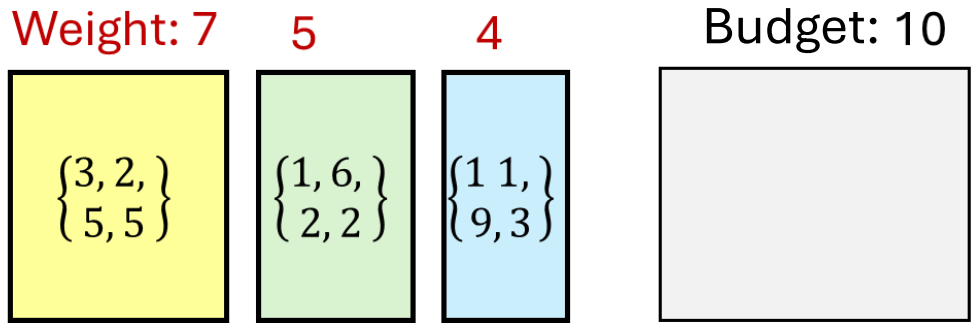
Average gap for pure RL

	TSP $n = 50$ (TSP RL)	TSPTW $n = 50$ (TSPTW RL)
DQN	20.14 [%]	(No feasible sol.)
PPO	3.85 [%]	46.93 [%]



- RL guidance (TSP RL) achieved about the same performance as others including $h = 0$
- Many states are pruned by time windows – unpromising transitions are already pruned

Portfolio



- Maximize profits with budget B
- For each item $i \in \{0 \ldots n-1\}$, determine whether to invest i

Standard deviation

$$\text{maximize} \left(\lambda_1 \sum_{i=1}^{n} \mu_i x_i - \lambda_2 \sqrt[2]{\sum_{i=1}^{n} \sigma_i^2 x_i} + \lambda_3 \sqrt[3]{\sum_{i=1}^{n} \gamma_i^3 x_i} - \lambda_4 \sqrt[4]{\sum_{i=1}^{n} \kappa_i^4 x_i} \right)$$

return skewness

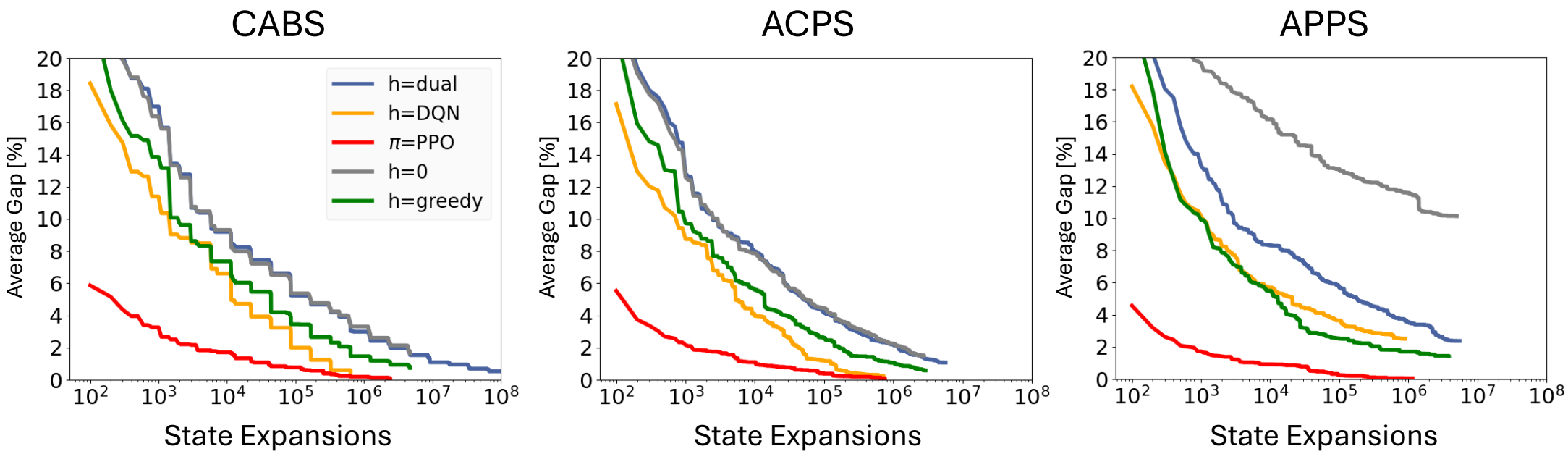
$$\text{subject to} \sum_{i=1}^{n} a_i x_i \leq B$$

cost

(Total cost must be within budget B)

$$x_i \in \{0, 1\} \qquad \forall\, i \in \{1 \ldots . n\}$$

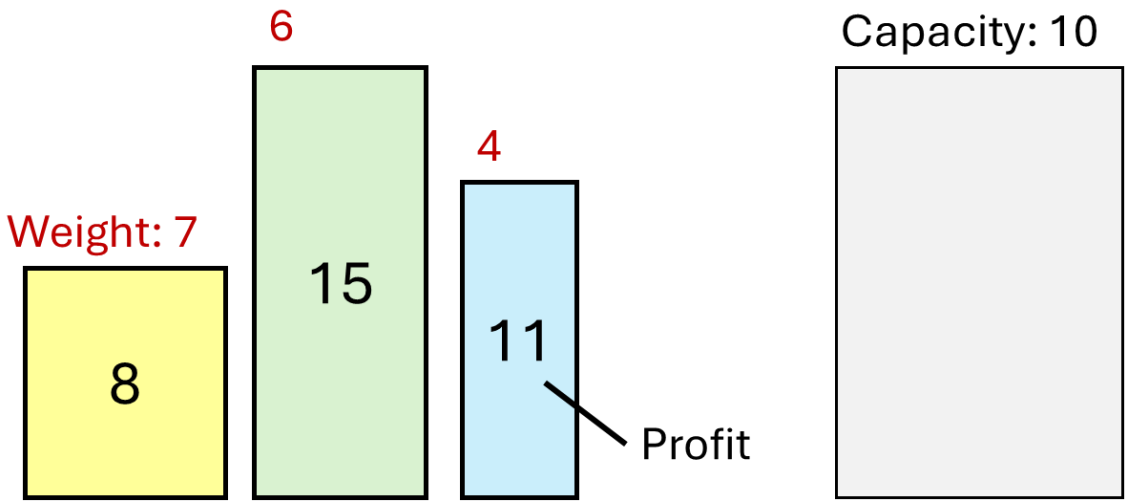
Result (Portfolio)



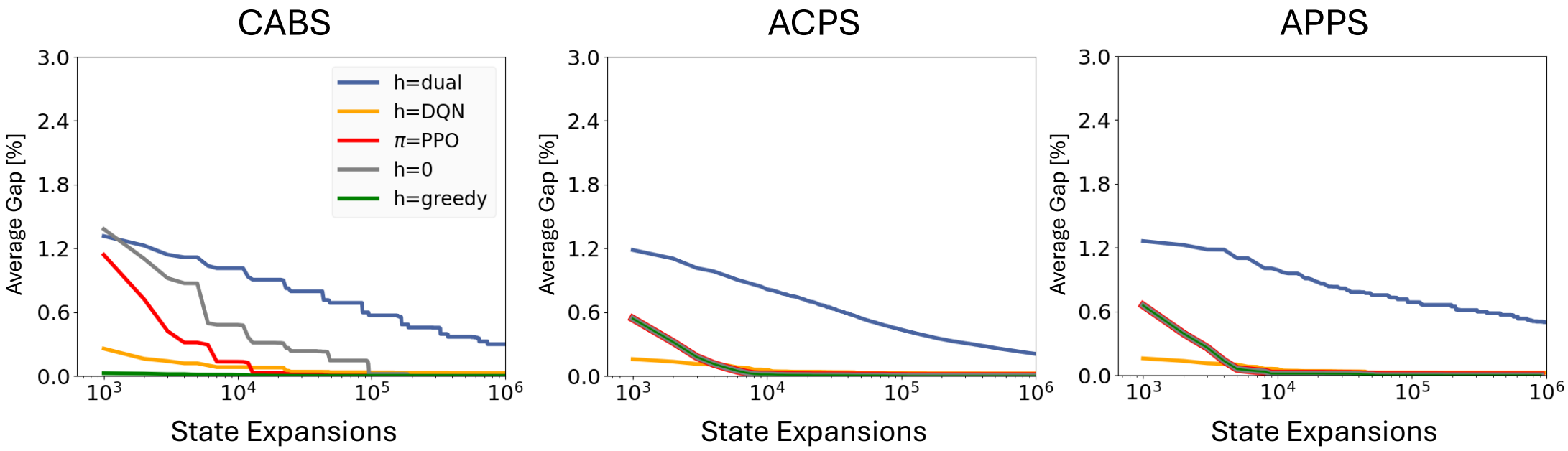
- PPO guidance significantly outperforms problem-specific greedy guidance
- DQN guidance also surpassed the dual-bound guidance

Knapsack

- Maximize profits within capacity C
- For each item $i \in \{0 \dots n-1\}$, determine whether to take i



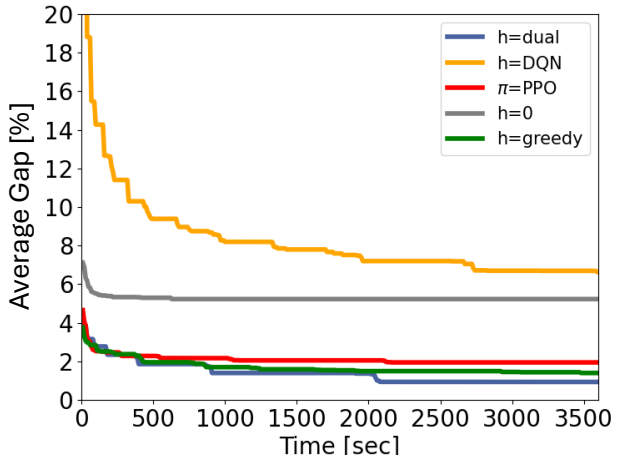
Result (Knapsack)



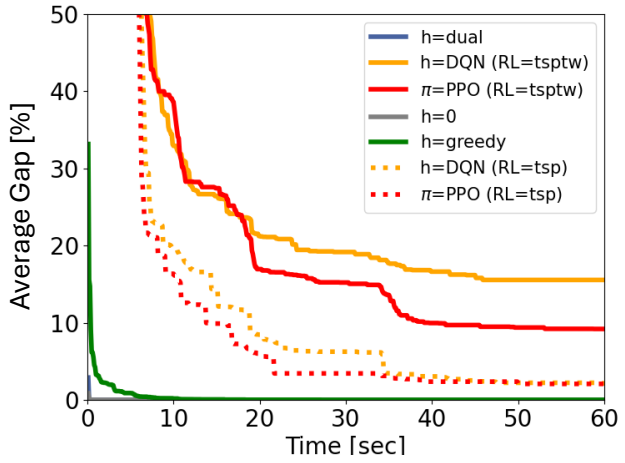
- All the heuristic guidance achieved similar performance
- During training, the policies by DQN and PPO rapidly converged to the best-ratio heuristic (greedy heuristic)

Result (Solve-time Performance, CABS)

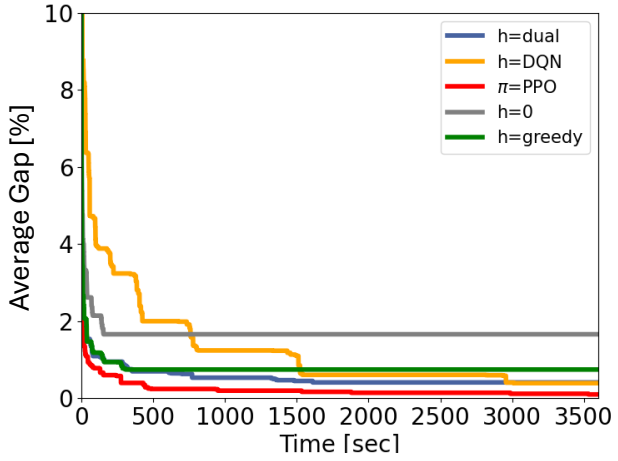
TSP



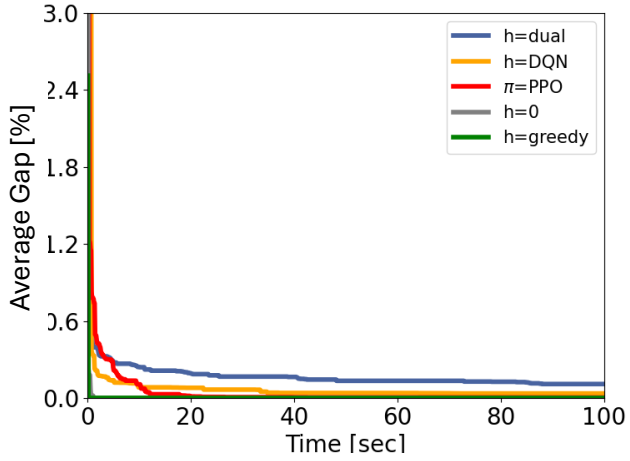
TSPTW



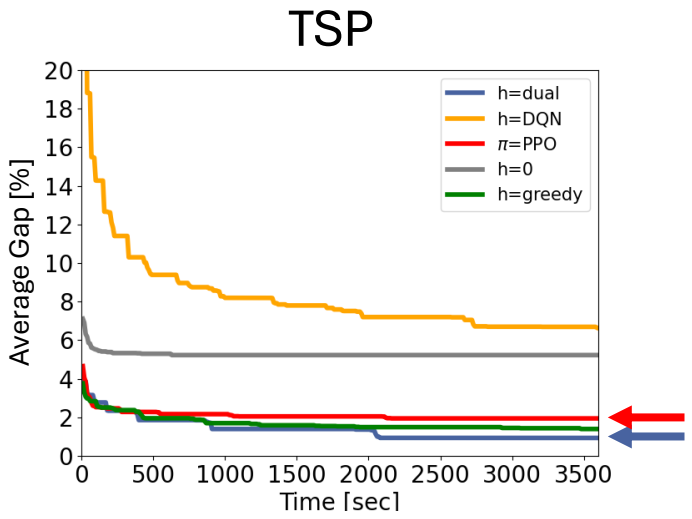
Portfolio

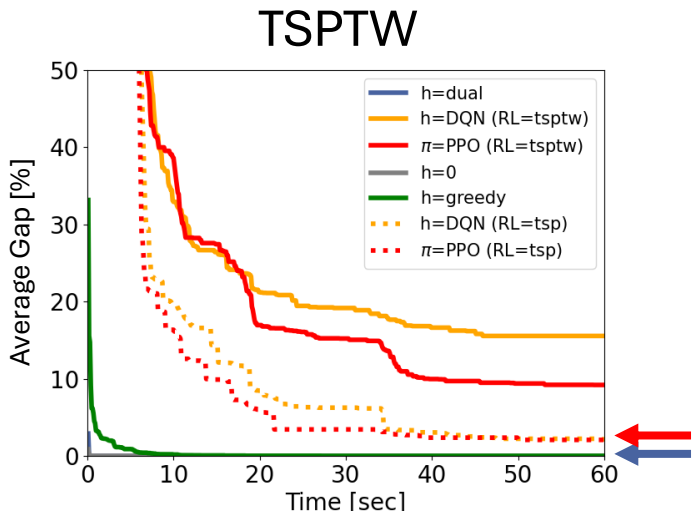


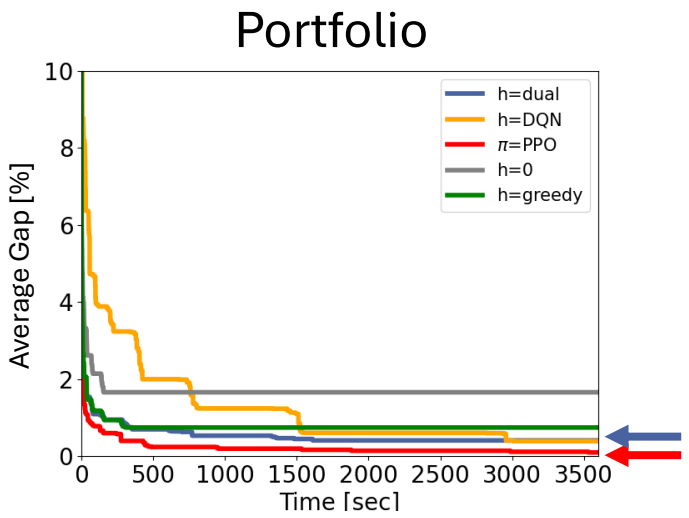
Knapsack

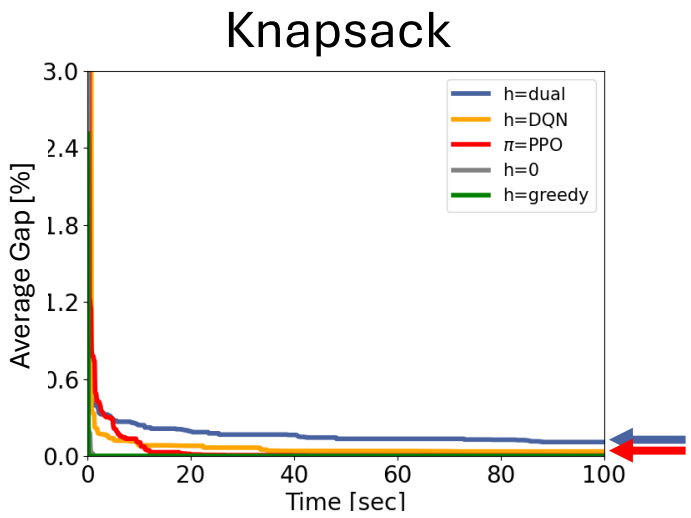


Result (Solve-time Performance, CABS)



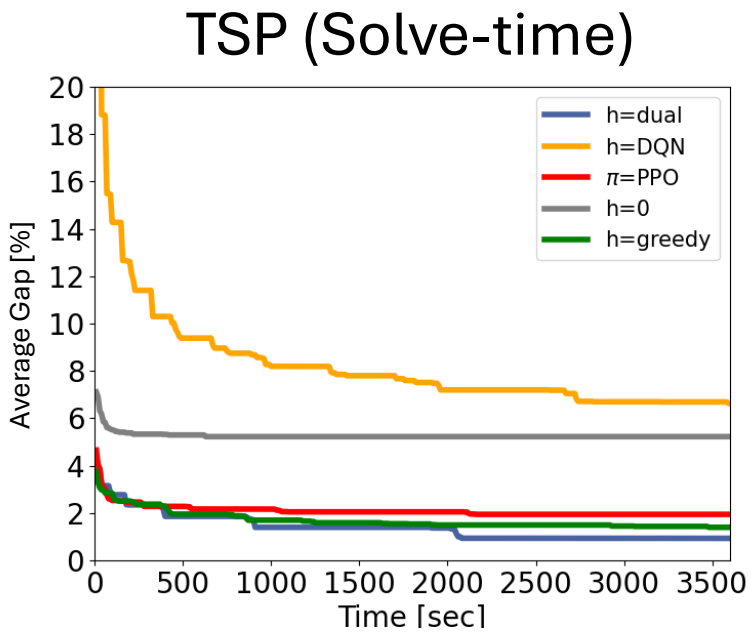






Future Work

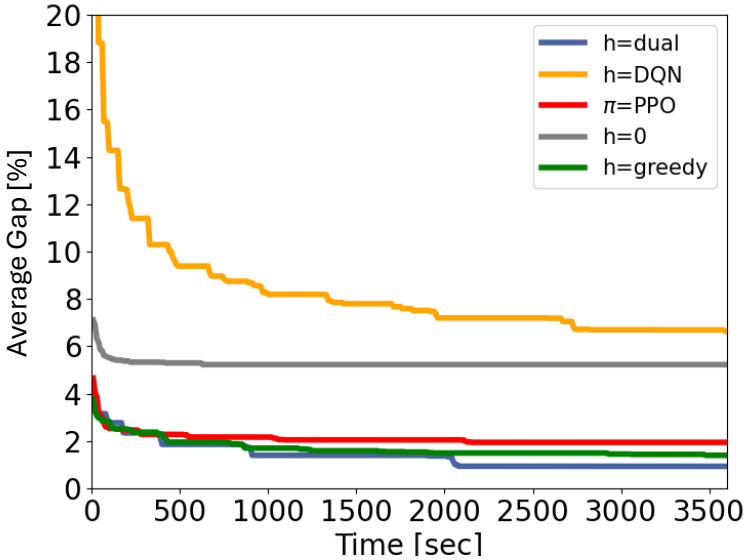
- RL guidance is slower due to longer state evaluation time
- Future work should focus on:
  - Reducing state evaluation time



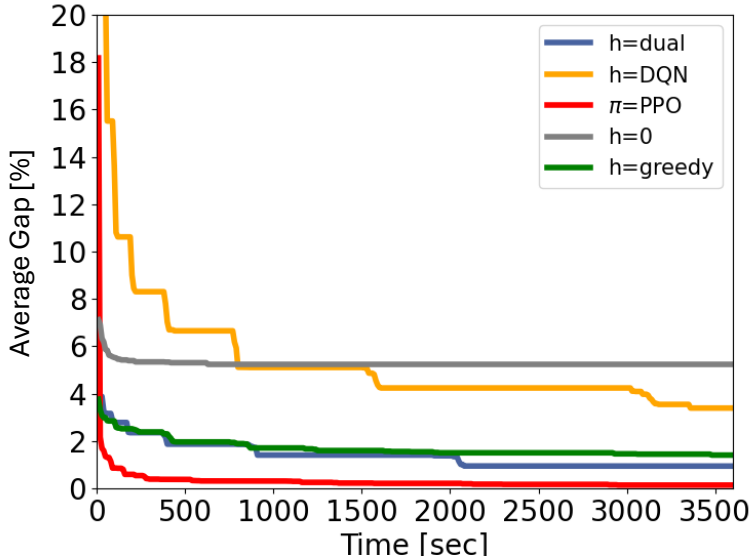
Future Work

- RL guidance is slower due to longer state evaluation time
- Future work should focus on:
  - Reducing state evaluation time

TSP (Solve-time)



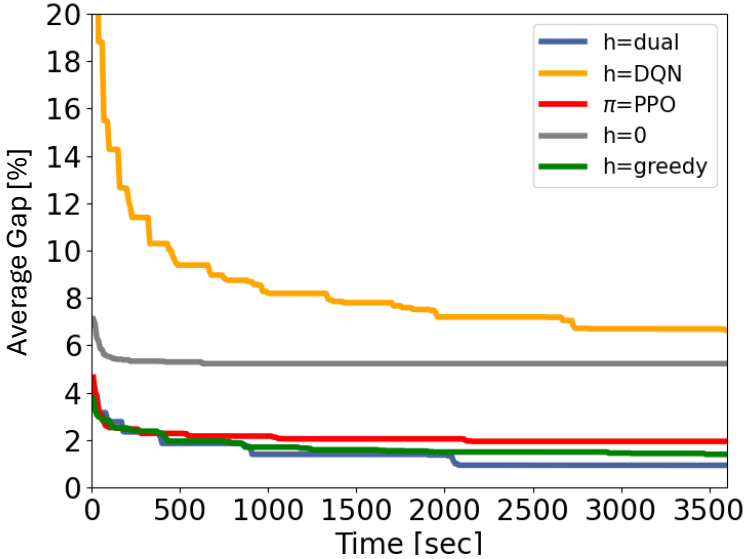
New Result!



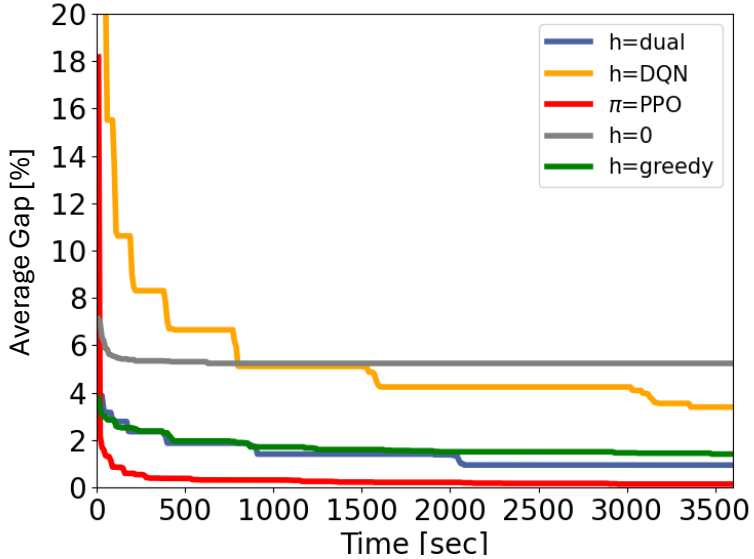
Future Work

- RL guidance is slower due to longer state evaluation time
- Future work should focus on:
  - Reducing state evaluation time
  - Automating RL model building

TSP (Solve-time)



New Result!



Supplemental Slides

Result (Solve Time Performance)

			TSP				TSPTW						0-1 Knapsack				Portfolio			
			n=20		n=50		n=20			n=50			n-50		n=100		n=20		n=50	
Type	Name		Gap	Time	Gap	Time	Feas.	Gap	Time	Feas.	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
DIDP	CABS	h=greedy	0.00	292	2.79	-	20	**0.00***	<1	20	**0.00***	22	0.00	-	0.00	-	**0.00***	15	1.48	-
		h=dual	**0.00***	19	1.86	-	20	**0.00***	<1	20	**0.00***	<1	0.00	-	0.09	-	**0.00***	2	0.81	-
		h=DQN	0.00	154	12.05	t.o.	20	**0.00***	44	20	**0.00***	765	0.00	t.o.	0.05	t.o.	**0.00***	780	0.77	t.o.
		π=PPO	0.00	32	3.89	t.o.	20	**0.00***	40	20	**0.00***	860	0.00	t.o.	0.00	t.o.	**0.00***	477	0.19	t.o.
	ACPS	h=greedy	**0.00***	62	2.45	-	20	**0.00***	<1	20	**0.00***	2	0.00	-	0.00	-	**0.00***	4	1.17	-
		h=dual	**0.00***	8	6.35	-	20	**0.00***	<1	20	**0.00***	<1	0.00	-	0.21	-	**0.00***	<1	2.14	-
		h=DQN	0.00	371	10.09	t.o.	20	**0.00***	11	20	**0.00***	99	0.00	t.o.	0.03	t.o.	**0.00***	180	0.50	t.o.
		π=PPO	0.00	345	2.45	t.o.	20	**0.00***	11	20	**0.00***	123	0.00	t.o.	0.00	t.o.	**0.00***	119	0.14	t.o.
	APPS	h=greedy	**0.00***	66	3.46	-	20	**0.00***	<1	20	**0.00***	4	0.00	-	0.00	-	**0.00***	4	2.86	-
		h=dual	**0.00***	8	8.30	-	20	**0.00***	<1	20	**0.00***	<1	0.00	-	0.58	-	**0.00***	<1	4.74	-
		h=DQN	1.08	t.o.	31.57	t.o.	20	**0.00***	13	20	**0.00***	141	0.00	t.o.	0.03	t.o.	**0.00***	187	5.01	t.o.
		π=PPO	0.20	t.o.	4.17	t.o.	20	**0.00***	12	20	**0.00***	141	0.00	t.o.	0.00	t.o.	**0.00***	119	0.10	t.o.

Bold: lowest average gap for each problem domain and n

*: optimality was proven

Red: the method achieves a better average gap than h=dual-bound using the same solver

Result (Solve Time Performance)

			TSP				TSPTW						0-1 Knapsack				Portfolio			
			n=20		n=50		n=20			n=50			n-50		n=100		n=20		n=50	
Type	Name		Gap	Time	Gap	Time	Feas.	Gap	Time	Feas.	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
DIDP	CABS	h=greedy	0.00	292	2.79	-	20	**0.00***	<1	20	**0.00***	22	0.00	-	0.00	-	**0.00***	15	1.48	-
		h=dual	**0.00***	19	1.86	-	20	**0.00***	<1	20	**0.00***	<1	0.00	-	0.09	-	**0.00***	2	0.81	-
		h=DQN	0.00	154	12.05	t.o.	20	**0.00***	44	20	**0.00***	765	0.00	t.o.	0.05	t.o.	**0.00***	780	0.77	t.o.
		π=PPO	0.00	32	3.89	t.o.	20	**0.00***	40	20	**0.00***	860	0.00	t.o.	0.00	t.o.	**0.00***	477	0.19	t.o.
	ACPS	h=greedy	**0.00***	62	2.45	-	20	**0.00***	<1	20	**0.00***	2	0.00	-	0.00	-	**0.00***	4	1.17	-
		h=dual	**0.00***	8	6.35	-	20	**0.00***	<1	20	**0.00***	<1	0.00	-	0.21	-	**0.00***	<1	2.14	-
		h=DQN	0.00	371	10.09	t.o.	20	**0.00***	11	20	**0.00***	99	0.00	t.o.	0.03	t.o.	**0.00***	180	0.50	t.o.
		π=PPO	0.00	345	2.45	t.o.	20	**0.00***	11	20	**0.00***	123	0.00	t.o.	0.00	t.o.	**0.00***	119	0.14	t.o.
	APPS	h=greedy	**0.00***	66	3.46	-	20	**0.00***	<1	20	**0.00***	4	0.00	-	0.00	-	**0.00***	4	2.86	-
		h=dual	**0.00***	8	8.30	-	20	**0.00***	<1	20	**0.00***	<1	0.00	-	0.58	-	**0.00***	<1	4.74	-
		h=DQN	1.08	t.o.	31.57	t.o.	20	**0.00***	13	20	**0.00***	141	0.00	t.o.	0.03	t.o.	**0.00***	187	5.01	t.o.
		π=PPO	0.20	t.o.	4.17	t.o.	20	**0.00***	12	20	**0.00***	141	0.00	t.o.	0.00	t.o.	**0.00***	119	0.10	t.o.

Bold: lowest average gap for each problem domain and n

*: optimality was proven

Red: the method achieves a better average gap than h=dual-bound using the same solver

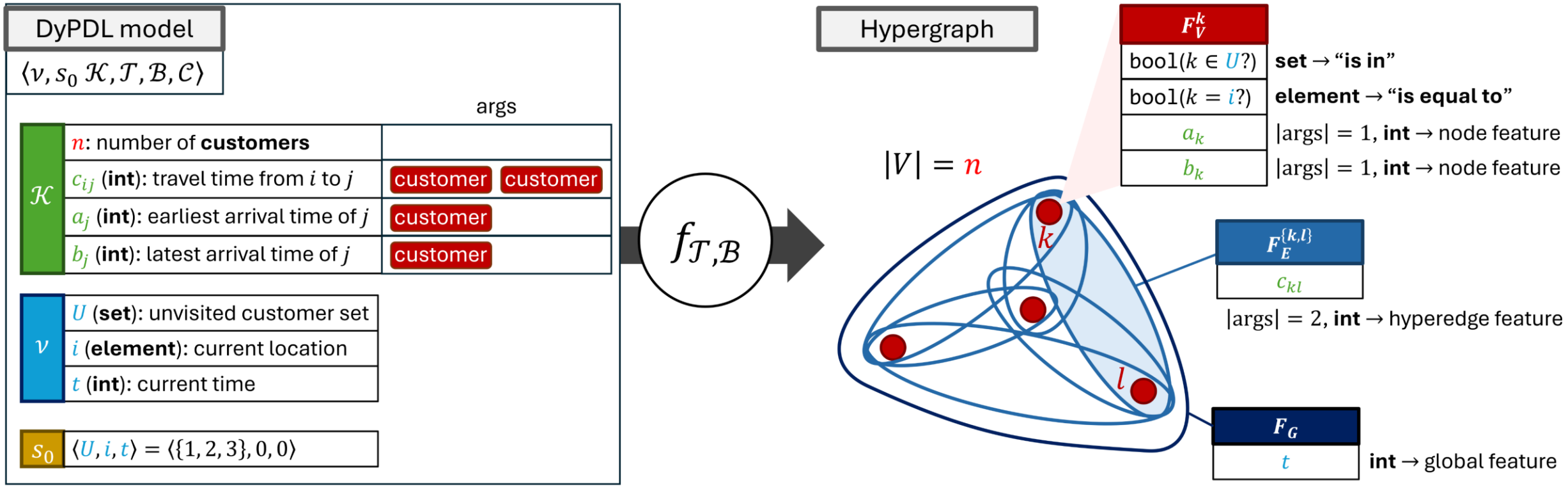
Result (Solve Time Performance, full-result)

			TSP				TSPTW						0-1 Knapsack				Portfolio			
			n=20		n=50		n=20			n=50			n-50		n=100		n=20		n=50	
Type	Name		Gap	Time	Gap	Time	Feas.	Gap	Time	Feas.	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
CP	CP Optimizer		0.00	25	2.75	t.o.	20	12.04	t.o.	20	32.32	t.o.	0.00	t.o.	0.02	t.o.	**0.00***	<1	**0.00**	t.o.
	BaB-DQN		3.77	t.o.	18.46	t.o.	20	**0.00***	216	20	40.83	t.o.	0.00	t.o.	0.00	t.o.	**0.00***	1270	10.18	t.o.
	RBS-PPO		0.12	t.o.	1.17	t.o.	20	0.65	t.o.	20	33.82	t.o.	0.00	t.o.	0.00	t.o.	**0.00***	487	0.50	t.o.
MIP	Gurobi		**0.00***	3	**0.03**	t.o.	20	**0.00***	<1	20	**0.00***	119	**0.00***	<1	**0.00***	<1	-	-	-	-
Heuristics	DQN		2.29	1251	(20.14)	(18239)	16	35.12	1405	0	-	-	0.02	44	0.04	121	3.63	36	8.09	132
	PPO		0.26	249	3.85	1783	20	25.64	423	20	46.93	2050	0.04	51	0.13	132	3.35	38	0.75	111
	Dual-bounds		2.91	4	9.58	1424	0	-	-	0	-	-	0.07	8	0.37	44	4.19	1	18.46	4
	Greedy		5.30	11	8.48	176	0	-	-	0	-	-	0.03	3	0.01	8	5.22	2	6.84	7
DIDP	CABS	h=greedy	0.00	292	2.79	-	20	**0.00***	<1	20	**0.00***	22	0.00	-	0.00	-	**0.00***	15	1.48	-
		h=dual	**0.00***	19	1.86	-	20	**0.00***	<1	20	**0.00***	<1	0.00	-	0.09	-	**0.00***	2	0.81	-
		h=DQN	0.00	154	12.05	t.o.	20	**0.00***	44	20	**0.00***	765	0.00	t.o.	0.05	t.o.	**0.00***	780	0.77	t.o.
		π=PPO	0.00	32	3.89	t.o.	20	**0.00***	40	20	**0.00***	860	0.00	t.o.	0.00	t.o.	**0.00***	477	0.19	t.o.
	ACPS	h=greedy	**0.00***	62	2.45	-	20	**0.00***	<1	20	**0.00***	2	0.00	-	0.00	-	**0.00***	4	1.17	-
		h=dual	**0.00***	8	6.35	-	20	**0.00***	<1	20	**0.00***	<1	0.00	-	0.21	-	**0.00***	<1	2.14	-
		h=DQN	0.00	371	10.09	t.o.	20	**0.00***	11	20	**0.00***	99	0.00	t.o.	0.03	t.o.	**0.00***	180	0.50	t.o.
		π=PPO	0.00	345	2.45	t.o.	20	**0.00***	11	20	**0.00***	123	0.00	t.o.	0.00	t.o.	**0.00***	119	0.14	t.o.
	APPS	h=greedy	**0.00***	66	3.46	-	20	**0.00***	<1	20	**0.00***	4	0.00	-	0.00	-	**0.00***	4	2.86	-
		h=dual	**0.00***	8	8.30	-	20	**0.00***	<1	20	**0.00***	<1	0.00	-	0.58	-	**0.00***	<1	4.74	-
		h=DQN	1.08	t.o.	31.57	t.o.	20	**0.00***	13	20	**0.00***	141	0.00	t.o.	0.03	t.o.	**0.00***	187	5.01	t.o.
		π=PPO	0.20	t.o.	4.17	t.o.	20	**0.00***	12	20	**0.00***	141	0.00	t.o.	0.00	t.o.	**0.00***	119	0.10	t.o.

Future Idea: General Mapping from DIDP to RL

- DyPDL model has a natural graph representation
- Given DyPDL model may be represented as a heterogeneous hypergraph

TSPTW



Mapping from a DP Model to an RL Model

DIDP model $\langle v, s_0\ \mathcal{T}, \mathcal{B}, \mathcal{C} \rangle$

State: s Transition: $\tau = \langle \text{eff}_\tau, \text{cost}_\tau, \text{pre}_\tau, \text{forced}_\tau \rangle$ Base case: $\langle \mathcal{C}_B, \text{cost}_B \rangle$ Target state: s_0

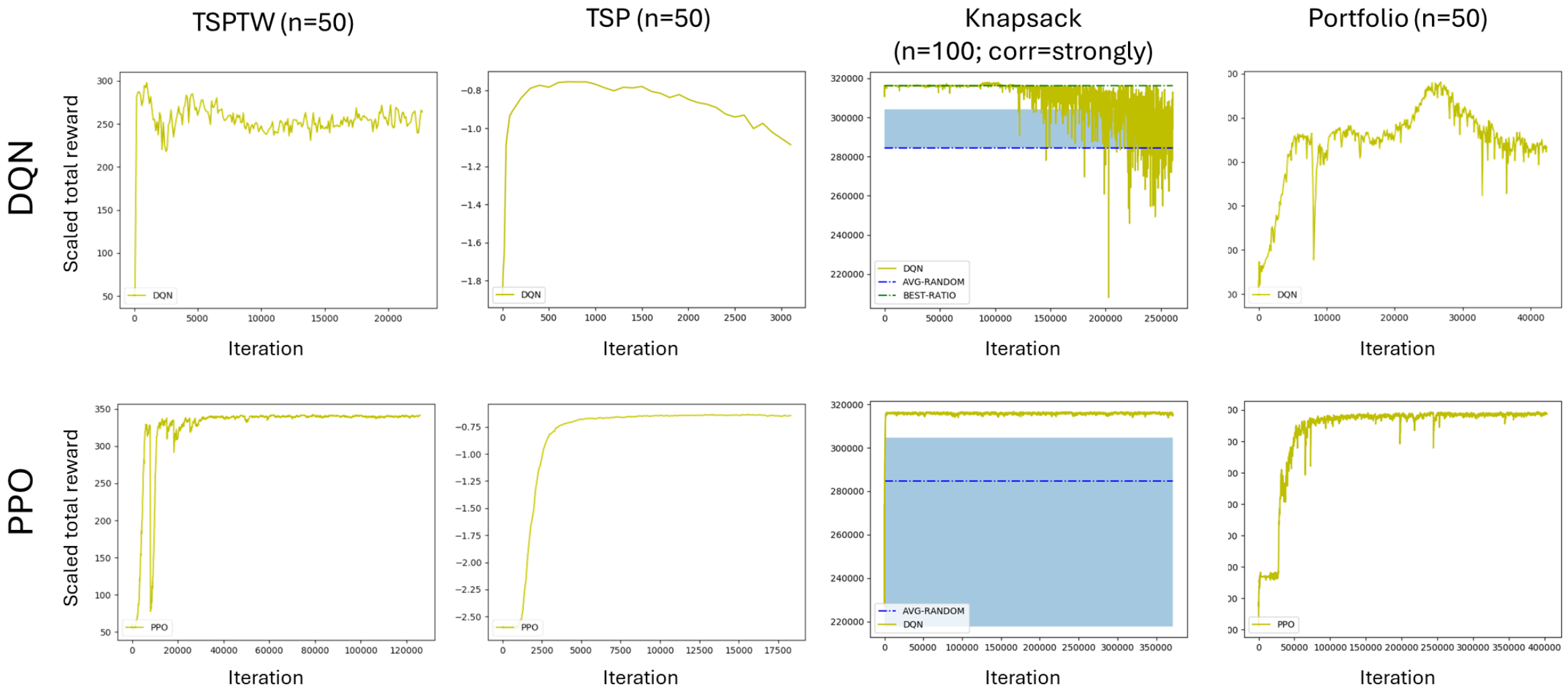
RL model

State s	Action a	Transition $\mathrm{T}(s, a)$	Reward $\mathrm{R}(s, a)$
DIDP state = RL state	$a \leftarrow \tau$	Let $\mathrm{T}(s, a) = \text{eff}_\tau[v_1, \ldots, v_n](s)$ The transition is not applicable if $s \not\models \text{pre}_\tau$	$\mathrm{R}(s, a) = \beta \cdot \text{cost}_\tau(s)$ if $\exists\ B \in \mathcal{B}$ s.t. $s \models \mathcal{C}_B$: $\mathrm{R}(s, \cdot) = \beta \cdot \max_{\{B \mid B \in \mathcal{B}; s \models \mathcal{C}_B\}} \text{cost}_B(s)$

Terminal state: s is a terminal state iff $\exists \mathrm{B} \in \mathcal{B}$ s.t. $s \models \mathcal{C}_B$ or $\nexists a$ s.t. $s \models \text{pre}_\tau$

Initial state: s_0

Learning Curve



TSP (DP model)

$$\text{compute } V(N \setminus \{0\}, 0)$$

$$V(U,i) = \begin{cases} c_{i0} & \text{if } U = \emptyset \\ \min_{j \in U} c_{ij} + V(U \setminus \{j\}, j) & \text{if } U \neq \emptyset \end{cases}$$

$$V(U,i) \geq \max \left\{ \sum_{j \in U \cup \{0\}} c_j^{\text{in}}, \sum_{j \in U \cup \{i\}} c_j^{\text{out}} \right\}$$

State variables:

- U: unvisited customer set
- i: current location

Constants

- N: set of customers $\{0 \dots n-1\}$
- c_{ij}: distance from customer i to j
- c_j^{in}: minimum travel time to j
- c_j^{out}: minimum travel time from j

TSPTW (DP model)

$$\text{compute } V(N \setminus \{0\}, 0, 0)$$

$$V(U,i,t) = \begin{cases} c_{i0} & \text{if } U = \emptyset \\ \min_{j \in U'} c_{ij} + V(U \setminus \{j\}, j, \max(t + c_{ij}, a_j)) & \text{if } U \neq \emptyset \end{cases}$$

$$V(U,i,t) = \infty \qquad \text{if } \exists j \in U, t + c^*_{ij} > b_j$$

$$V(U,i,t) \leq V(U,i,t') \qquad \text{if } t \leq t'$$

$$V(U,i,t) \geq \max\left\{ \sum_{j \in U \cup \{0\}} c^{\text{in}}_j, \sum_{j \in U \cup \{i\}} c^{\text{out}}_j \right\}$$

State variables:

- U: unvisited customer set
- i: current location
- t: current time

Constants

- N: set of customers $\{0 \dots n-1\}$
- c_{ij}: distance from customer i to j
- $[a_j, b_j]$: time window of j
- c_j^{in}, c_j^{out} : minimum travel time to/from j
- c_{ij}^*: shortest path from customer i to j

0-1 Knapsack (DP model)

$$\text{compute } V(0,0)$$

$$V(x,i) = \begin{cases} \max\{p_i + V(x+w_i, i+1), V(x, i+1)\} & \text{if } i < n \wedge x + w_i \leq B \\ V(x, i+1) & \text{if } i < n \wedge x + w_i > B \\ 0 & \text{otherwise.} \end{cases}$$

$$V(x,i) \leq \min \left\{ \sum_{j=i}^{n-1} p_j, \max_{j \in \{i..n-1\}} \left(\frac{p_j}{w_j} \right) \cdot (B - x) \right\}$$

State variables:

- x: current weight
- i: current item index

Constants

- B: budget
- w_i: weight of item i
- p_i: profit of item i

Portfolio (DP model)

compute $V(0, 0, \emptyset)$

$$V(x, i, Y) = \begin{cases} \max(\nu(Y \cup \{i\}) - \nu(Y) + V(x + w_i, i + 1, Y \cup \{i\}), \\ \qquad V(x, i + 1, Y)) & \text{if } i < n \wedge x + w_i \leq B \\ V(x, i + 1, Y) & \text{if } i < n \wedge x + w_i > B \\ 0 & \text{otherwise.} \end{cases}$$

$$V(x, i, Y) \leq \min \left(\lambda_1 \sum_{j=i}^{n-1} \mu_j + \lambda_3 \sqrt[3]{\sum_{j=i}^{n-1} \gamma_j^3}, \max_{j \in \{i..n-1\}} \left(\frac{\lambda_1 \mu_j + \lambda_3 \sqrt[3]{\gamma_j^3}}{w_j} \right) \cdot (B - x) \right)$$

State variables:

- x: current weight
- i: current item index
- Y: set of investments up to i

Constants

- B: budget
- w_i: weight of item i
- $\mu_i, \sigma_i, \gamma_i, \kappa_i$: mean, SD, skewness, kurtosis of item i
- $\lambda_1, \lambda_2, \lambda_3, \lambda_4$: weights for financial characteristic

Cappart et al. [AAAI 2021]

Constraints The constraints of our encoding have two purposes. Firstly, they must ensure the consistency of the DP formulation. It is done (1) by setting the initial state to a value (e.g., ϵ), (2) by linking the state of each stage to the previous one through the transition function (T), and finally (3) by enforcing each transition to be *valid*, in the sense that they can only generate a feasible state of the system. Secondly, other constraints are added in order to remove dominated actions and the subsequent states. In the CP terminology, such constraints are called *redundant constraint*, they do not change the semantic of the model, but speed-up the search. The constraints inferred by our encoding are as follows, where `validityCondition` and `dominanceCondition` are both Boolean functions detecting non-valid transitions and dominated actions, respectively.

Each variable corresponds to each state in DP

$$\mathbf{x}_1^s = \epsilon \tag{4}$$

$$\forall i \in \{1, \ldots, n\} : \mathbf{x}_{i+1}^s = T(\mathbf{x}_i^s, \mathbf{x}_i^a) \tag{5}$$

$$\forall i \in \{1, \ldots, n\} : \texttt{validityCondition}(\mathbf{x}_i^s, \mathbf{x}_i^a) \tag{6}$$

$$\forall i \in \{1, \ldots, n\} : \texttt{dominanceCondition}(\mathbf{x}_i^s, \mathbf{x}_i^a) \tag{7}$$

Setting the initial state is done in Eq. (4), enforcing the transition function in Eq. (5), keeping only the valid transitions in Eq. (6), and pruning the dominated states in Eq. (7)

Variable representing the action taken at stage i

Updated Results

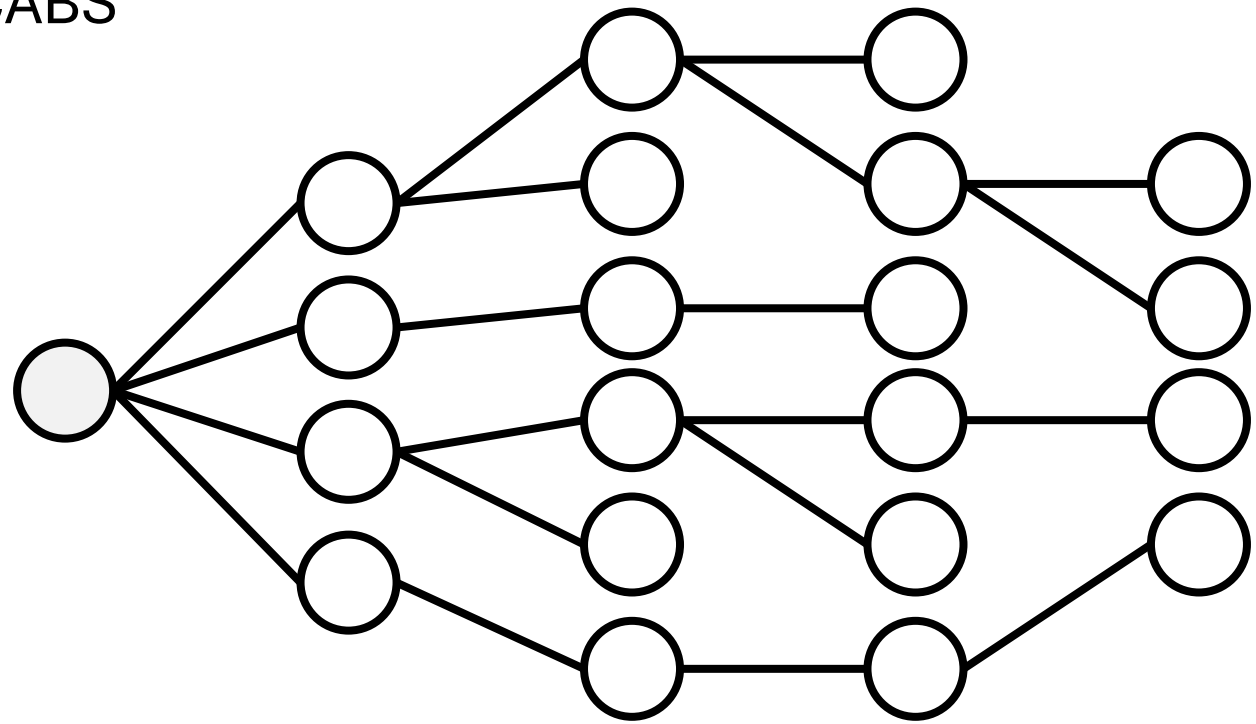
Batched State Evaluation

- RL prediction is costly due to the large number of neural network parameters and communication overheads between Python and Rust
- The prediction is called every time a new node is expanded

Batched State Evaluation

- To improve the solve-time efficiency, a batch of states are evaluated in a single call, utilizing the GPU's parallel computation ability

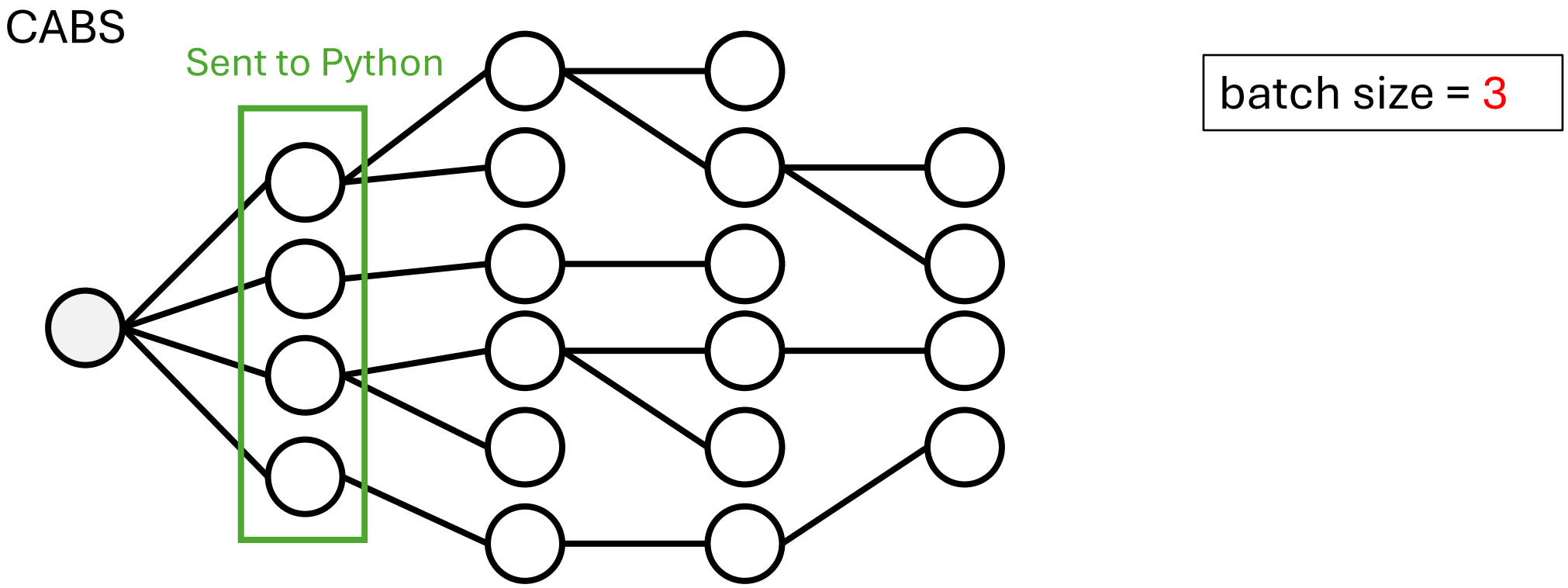
CABS



batch size = 3

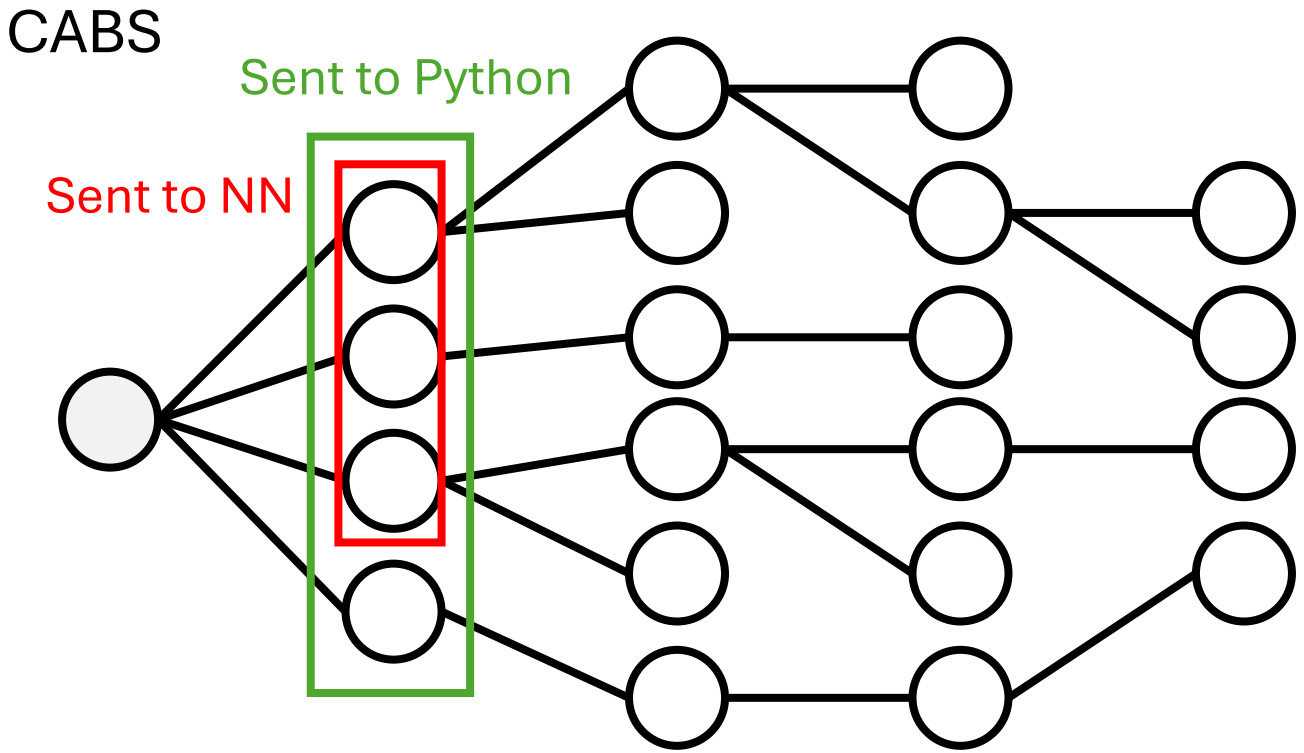
Batched State Evaluation

- To improve the solve-time efficiency, a batch of states are evaluated in a single call, utilizing the GPU's parallel computation ability



Batched State Evaluation

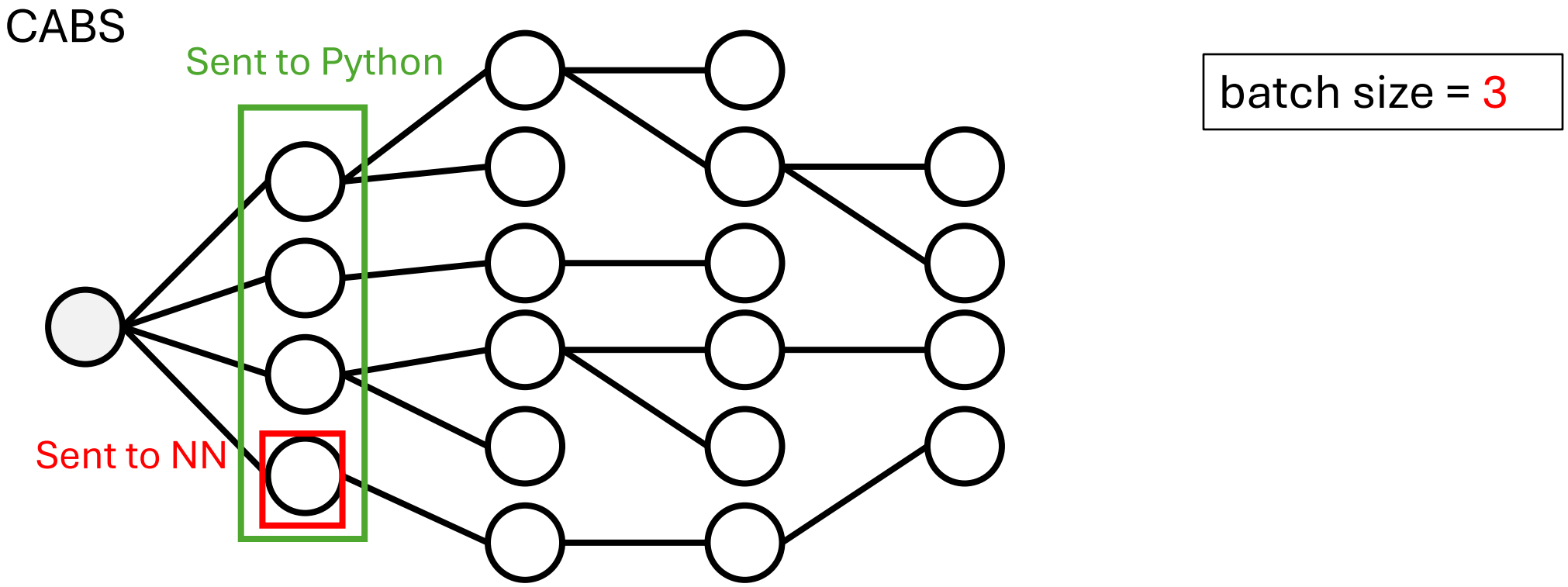
- To improve the solve-time efficiency, a batch of states are evaluated in a single call, utilizing the GPU's parallel computation ability



batch size = 3

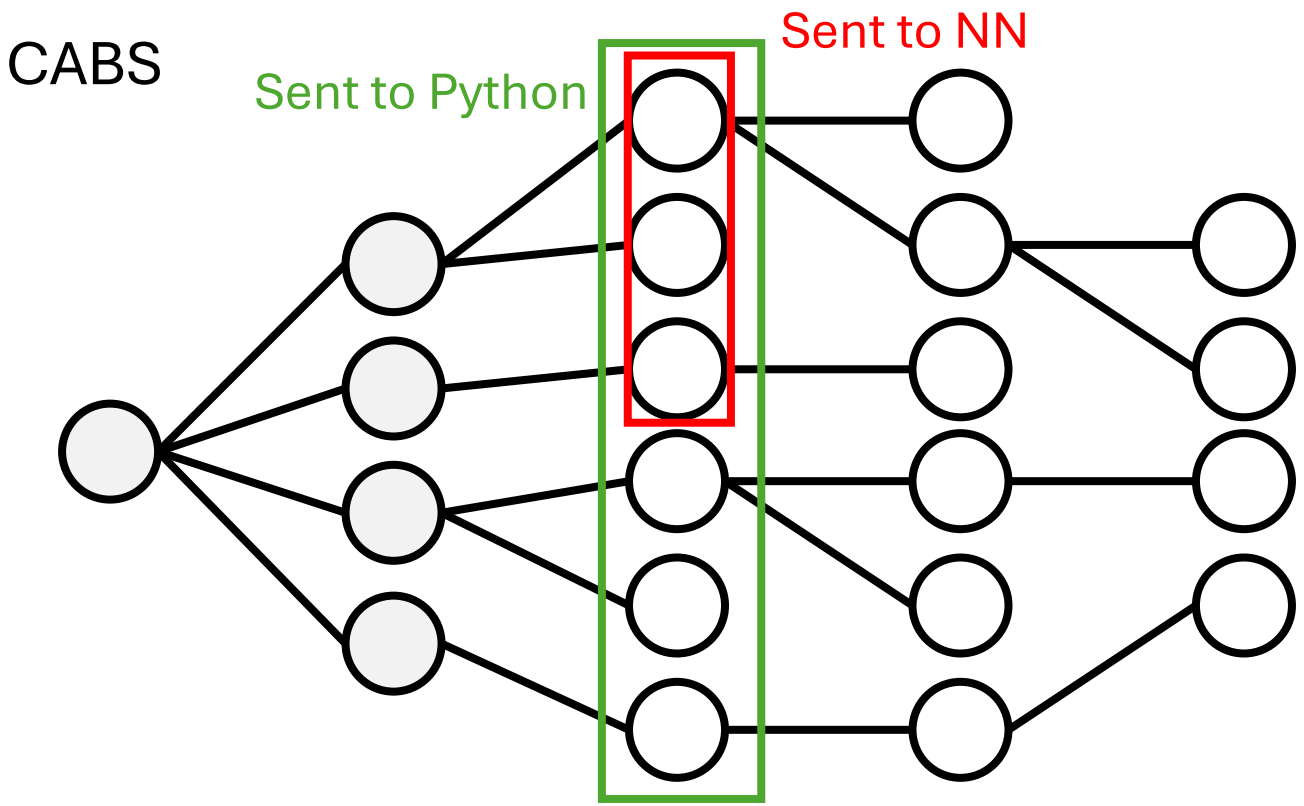
Batched State Evaluation

- To improve the solve-time efficiency, a batch of states are evaluated in a single call, utilizing the GPU's parallel computation ability



Batched State Evaluation

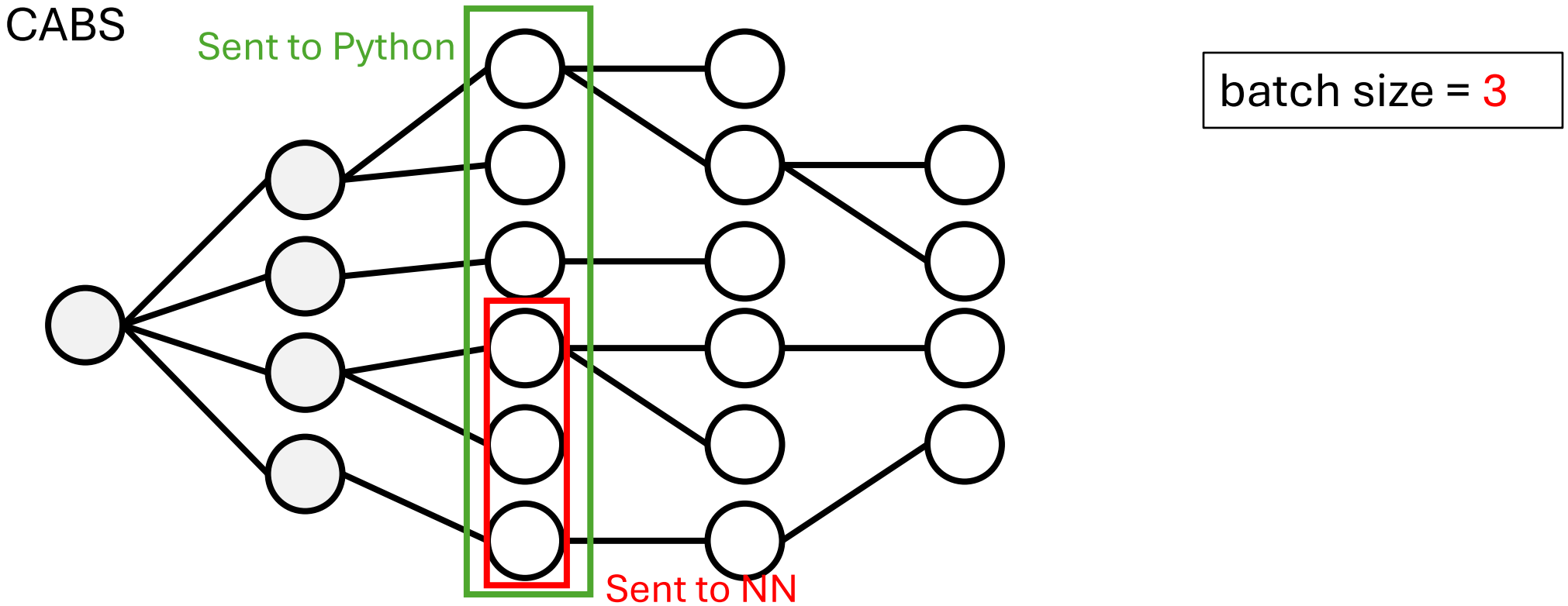
- To improve the solve-time efficiency, a batch of states are evaluated in a single call, utilizing the GPU's parallel computation ability



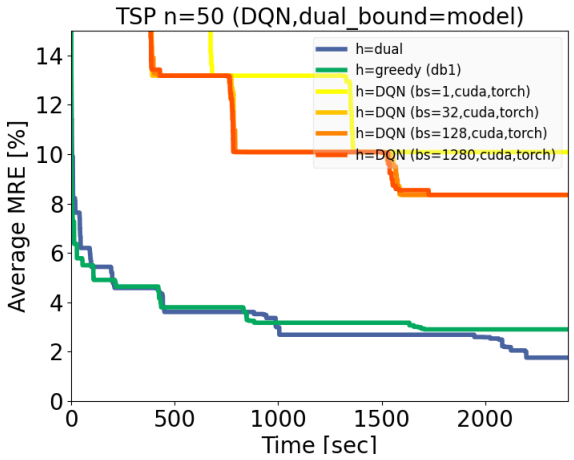
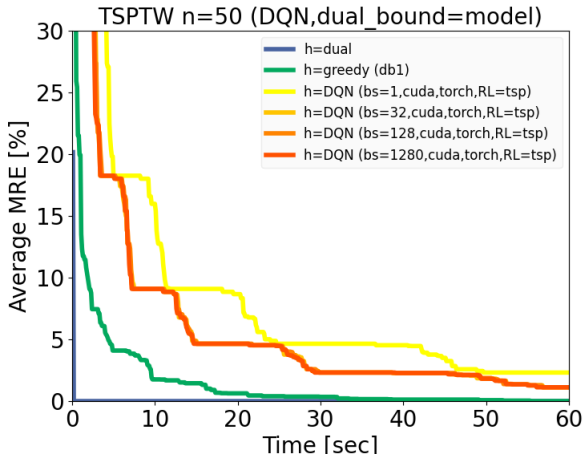
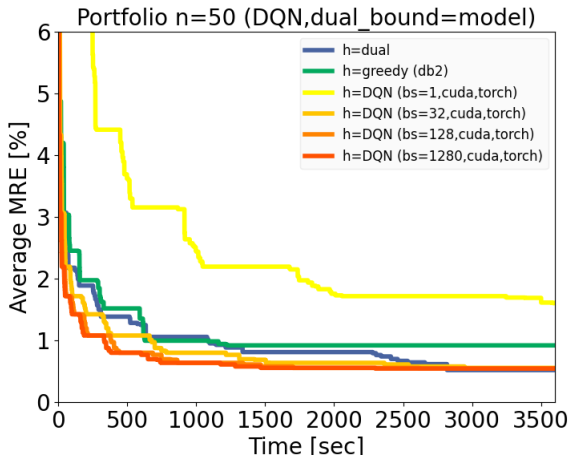
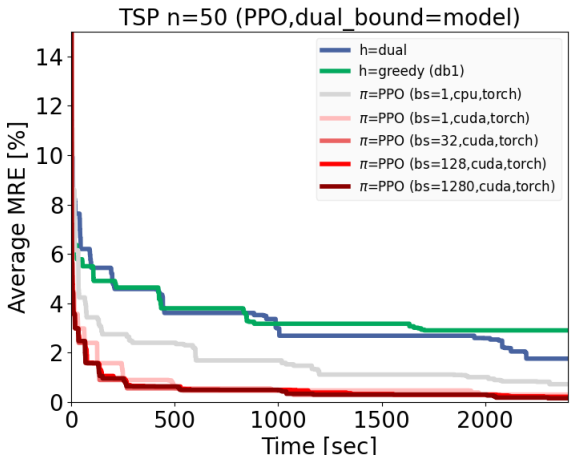
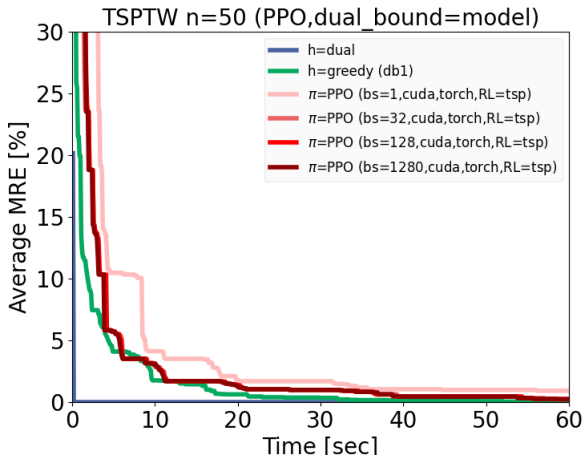
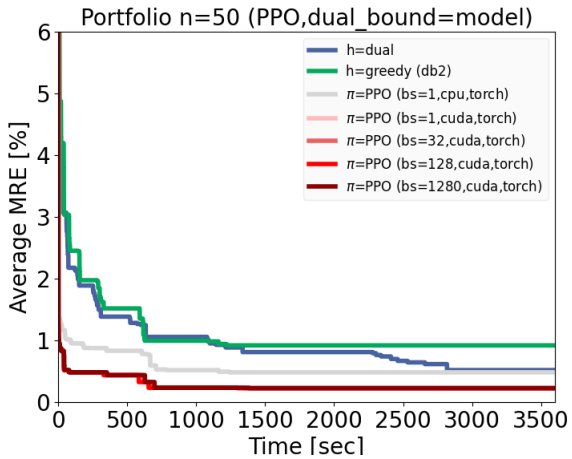
batch size = 3

Batched State Evaluation

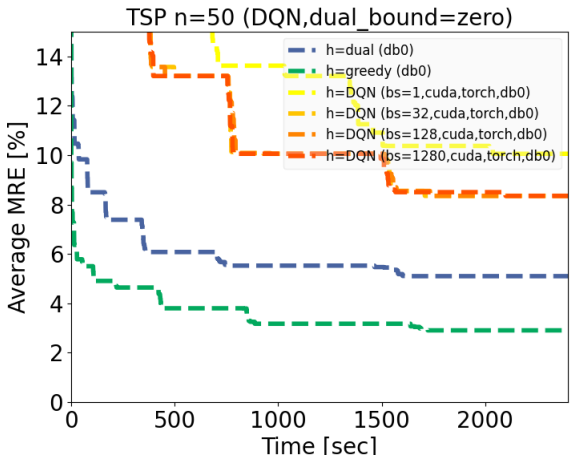
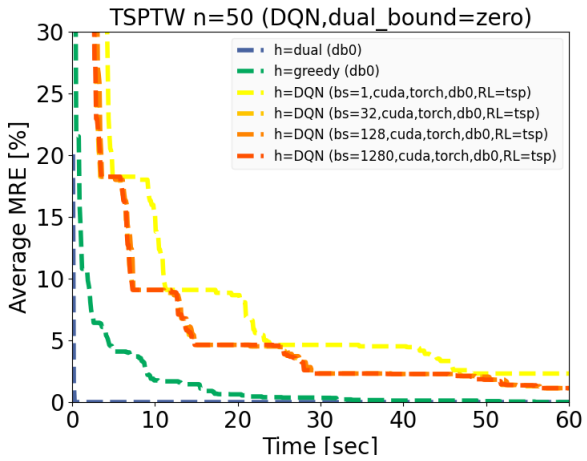
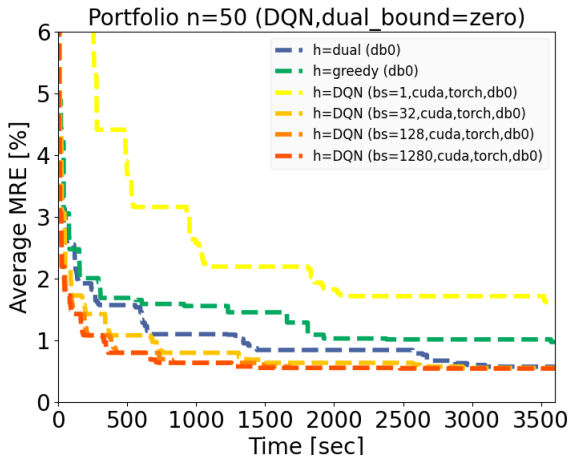
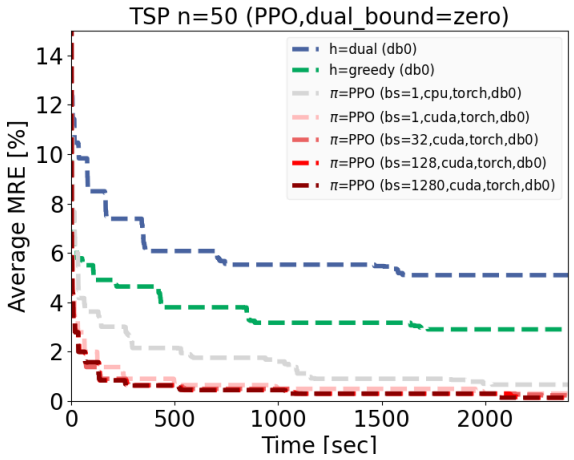
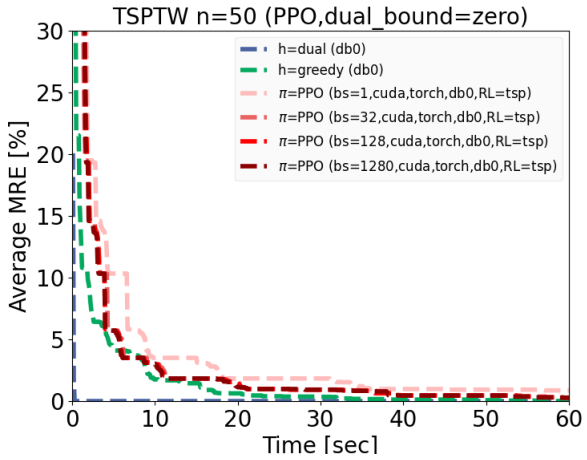
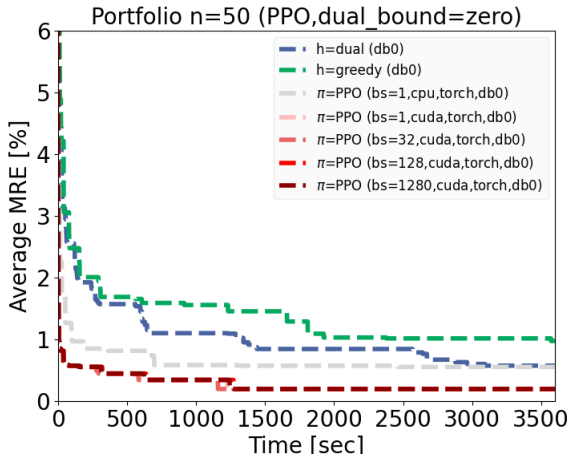
- To improve the solve-time efficiency, a batch of states are evaluated in a single call, utilizing the GPU's parallel computation ability



Results (CABS with the model dual bound)

	TSP	TSPTW	Portfolio
DQN			
PPO			

Results (CABS with zero dual bound)

	TSP	TSPTW	Portfolio
DQN			
PPO			

Discussion

- Batch evaluation significantly improves the solve-time efficiency across different domains
- PPO guidance outperformed other heuristics even without using dual-bounds in TSP and Portfolio
  - Dual-bounds may not provide strong guidance in directing search towards better solutions

Next Steps

- RL guidance is most impactful when heuristic quality plays a critical role in determining solution quality
- To test our hypothesis, pick two additional domains, one with a strong dual bound and one with a weak dual bound
  - Compare the advantage of RL guidance across these contrasting scenarios
- Evaluate RL guidance on TSPTW instances with varying time-window ranges
  - Time-window range controls the number of states pruned
  - Examine how the impact of strong guidance varies with the importance of heuristic quality in achieving good solutions