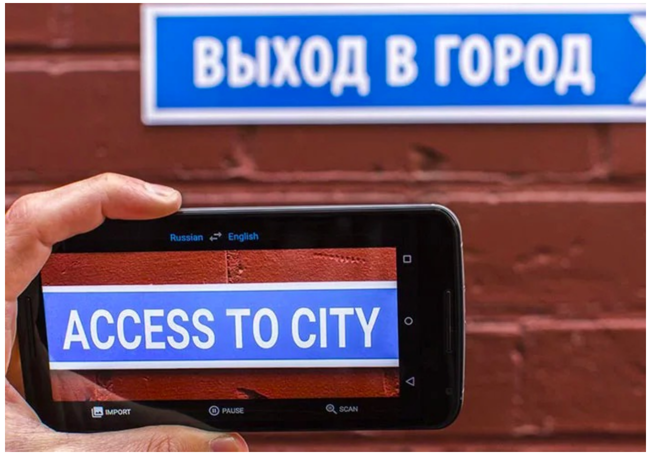
Finding Good Representation for Search and Exploration in RL

Yuandong Tian

Research Scientist

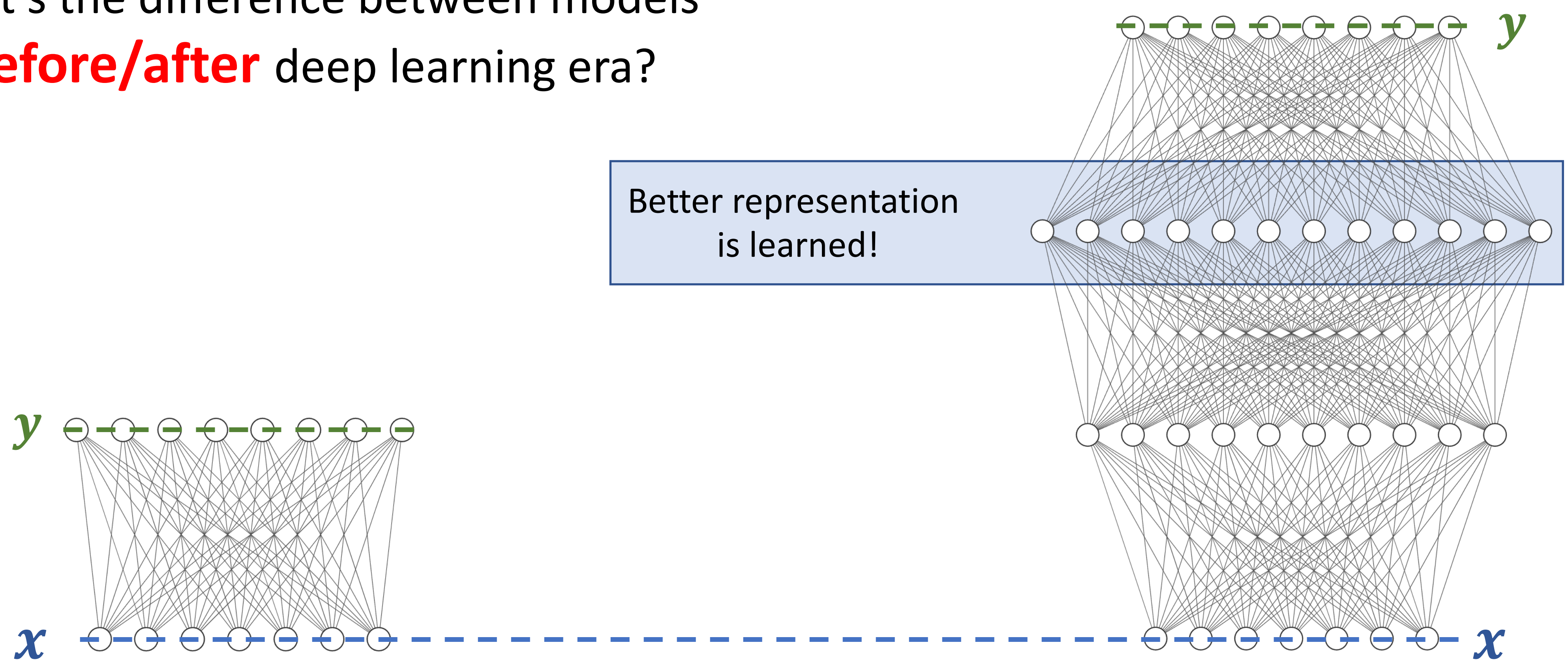
Facebook AI Research

Great Empirical Success of Deep Models



Representation Learning

What’s the difference between models
before/after deep learning era?

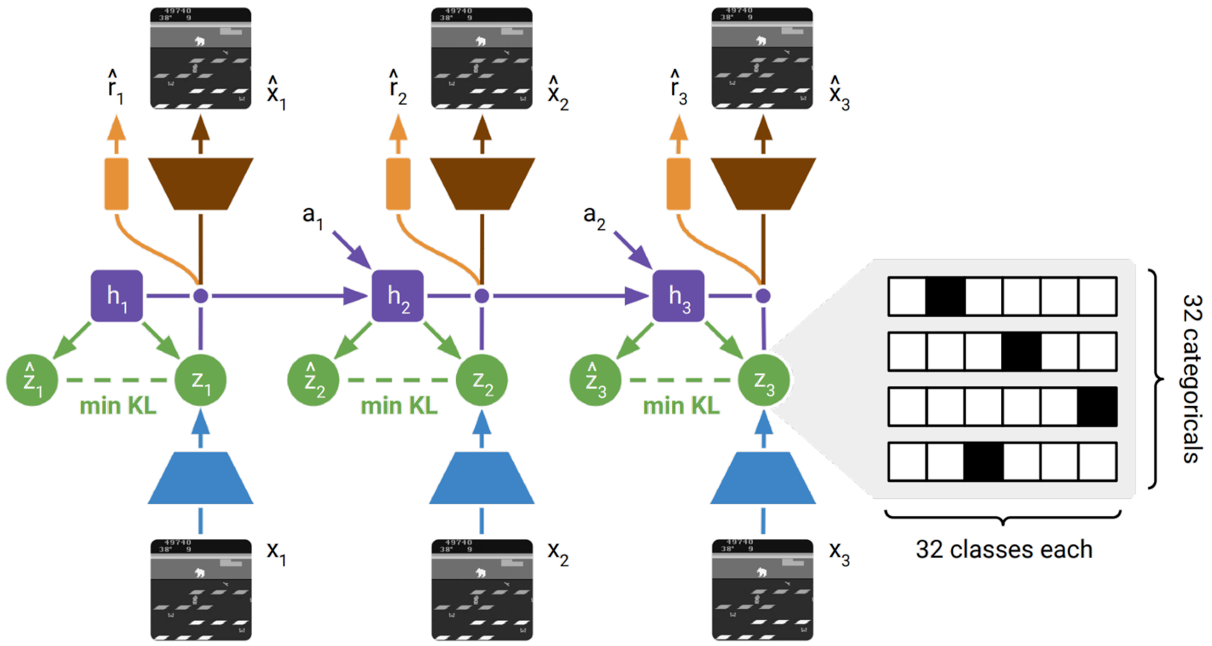


Linear Regression

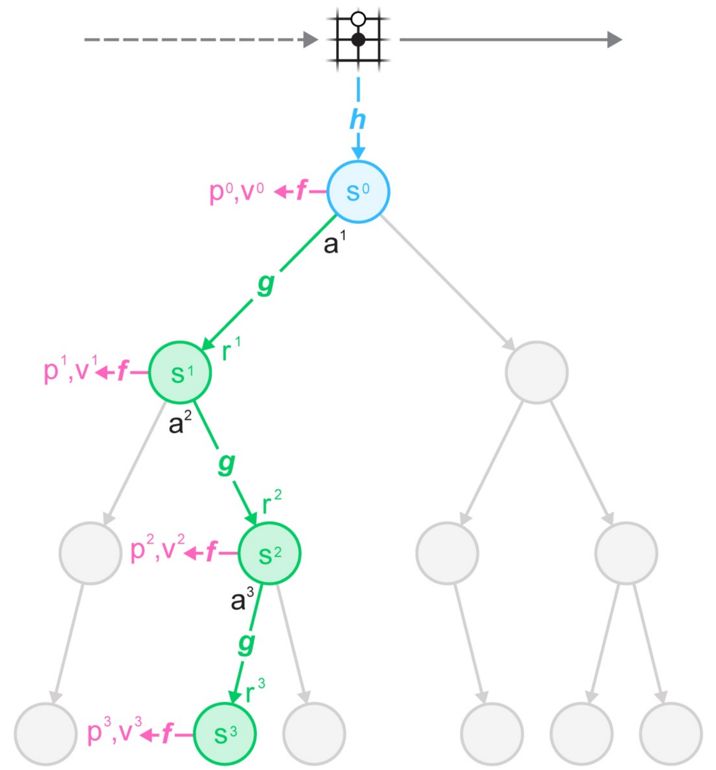
Deep Models

Representation Learning in RL

- State Representation



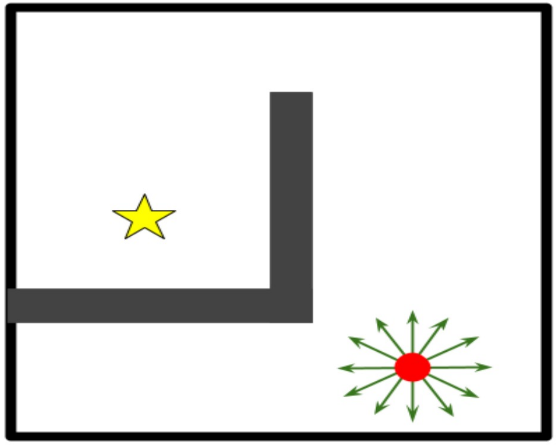
[D. Hafner et al, Mastering Atari with Discrete World Models]

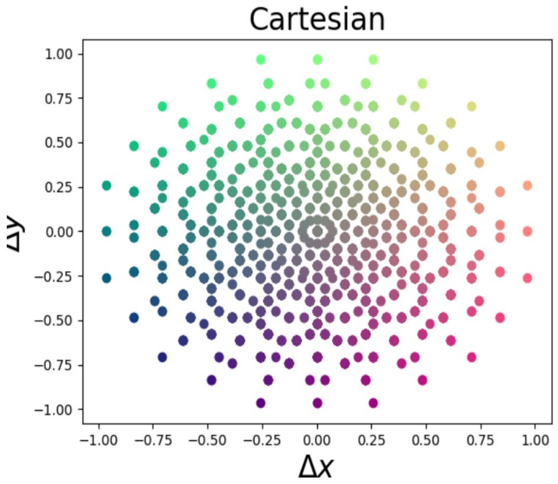


[J. Schrittwieser et al, Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model, Nature]

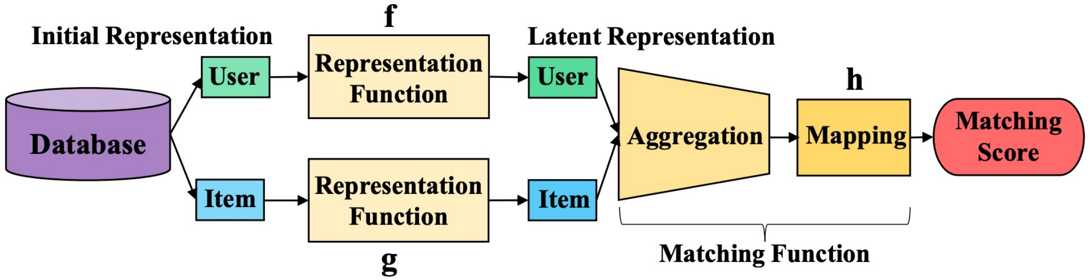
Representation Learning in RL

- Action Representation
  - Action embedding if we have a million of actions to choose from.
  - Action embedding to transfer across different tasks.





[Y. Chandak, Learning Action Representations for Reinforcement Learning, ICML 2019]



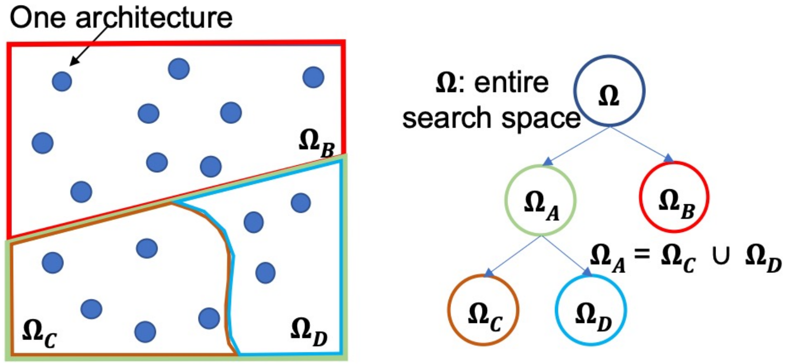
[Z. Deng et al, DeepCF: A Unified Framework of Representation Learning and Matching Function Learning in Recommender System, AAAI 2019]

How about high-level representation?

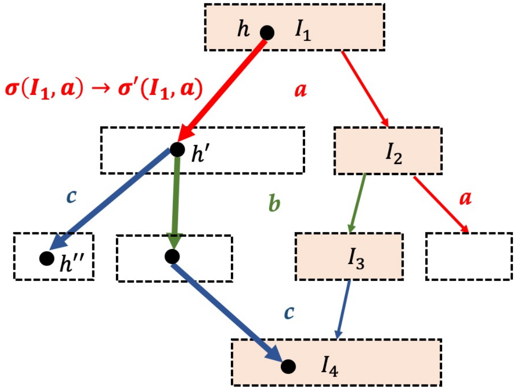
- Representation of the entire policy?
  - [*A. Singh et al, Parrot: Data-Driven Behavioral Priors for Reinforcement Learning, ICLR 2021*]

- Representation of the environment (multi-task learning)?

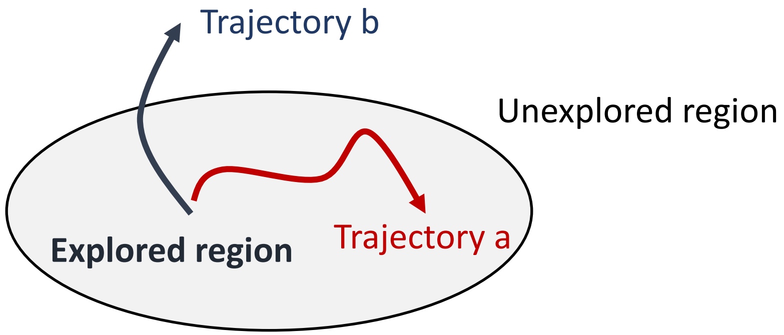
This Talk



Representation of the Action Space



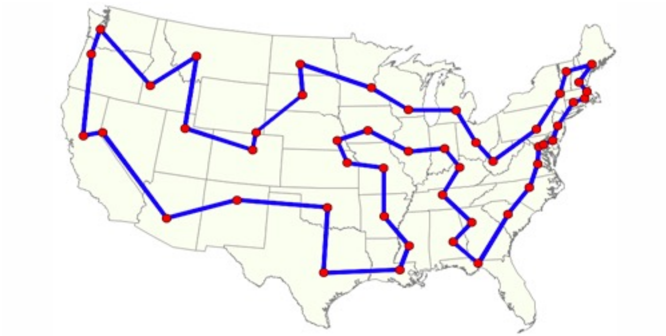
Representation for Easier Search



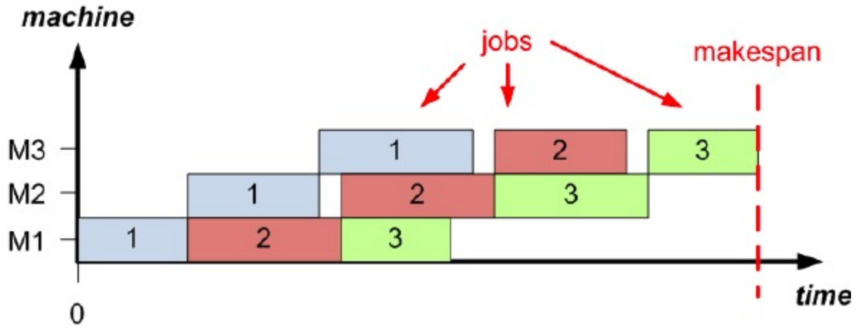
Representation for RL Exploration

Representation of the Action Space

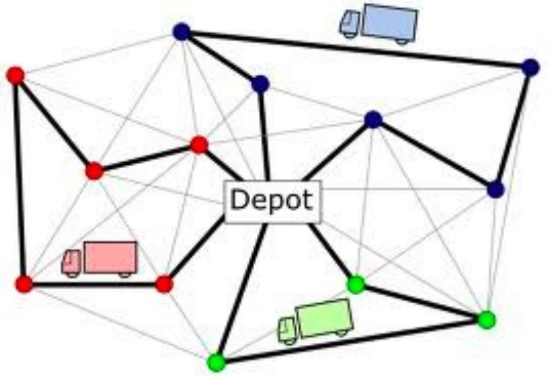
Optimization Problems



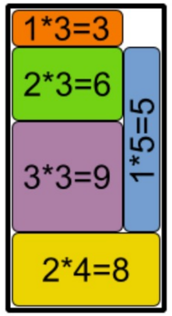
Travel Salesman Problem



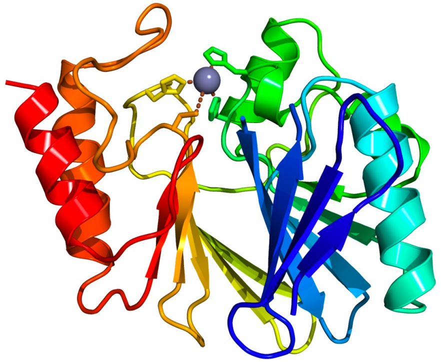
Job Scheduling



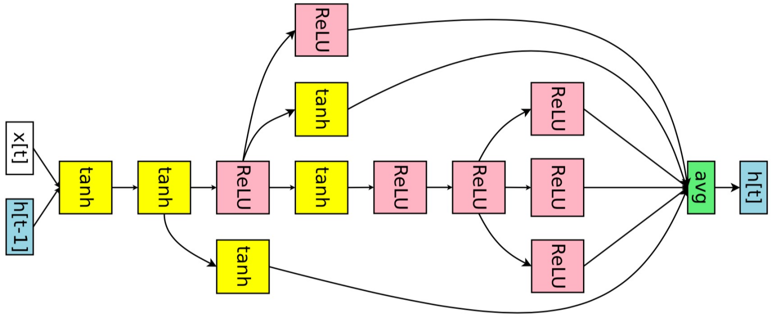
Vehicle Routing



Bin Packing



Protein Folding



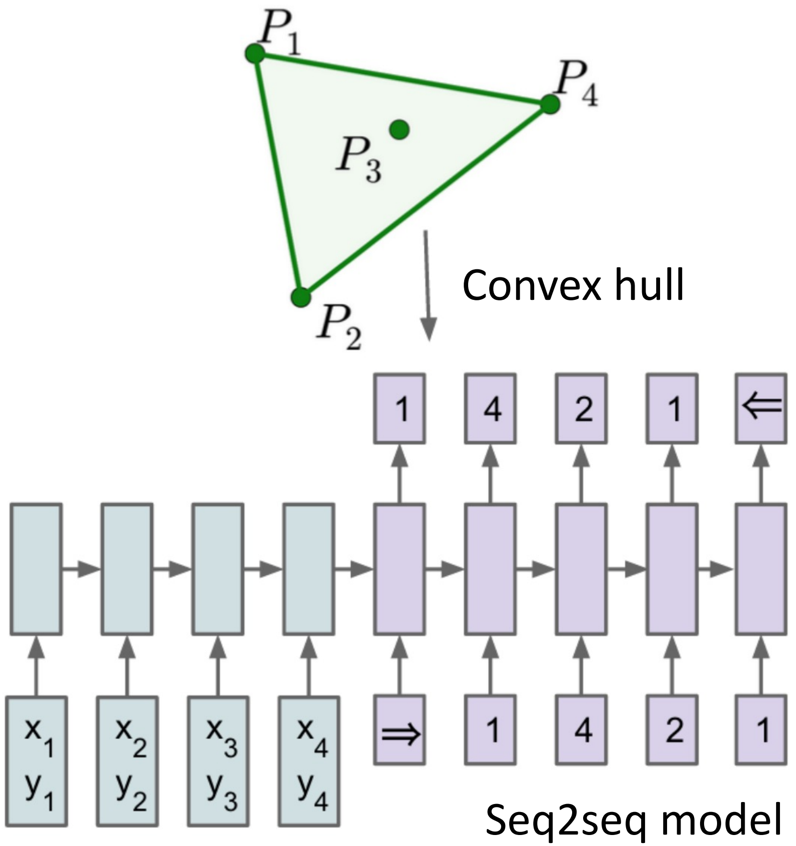
Model-Search

There exists many MDPs for a single optimization!

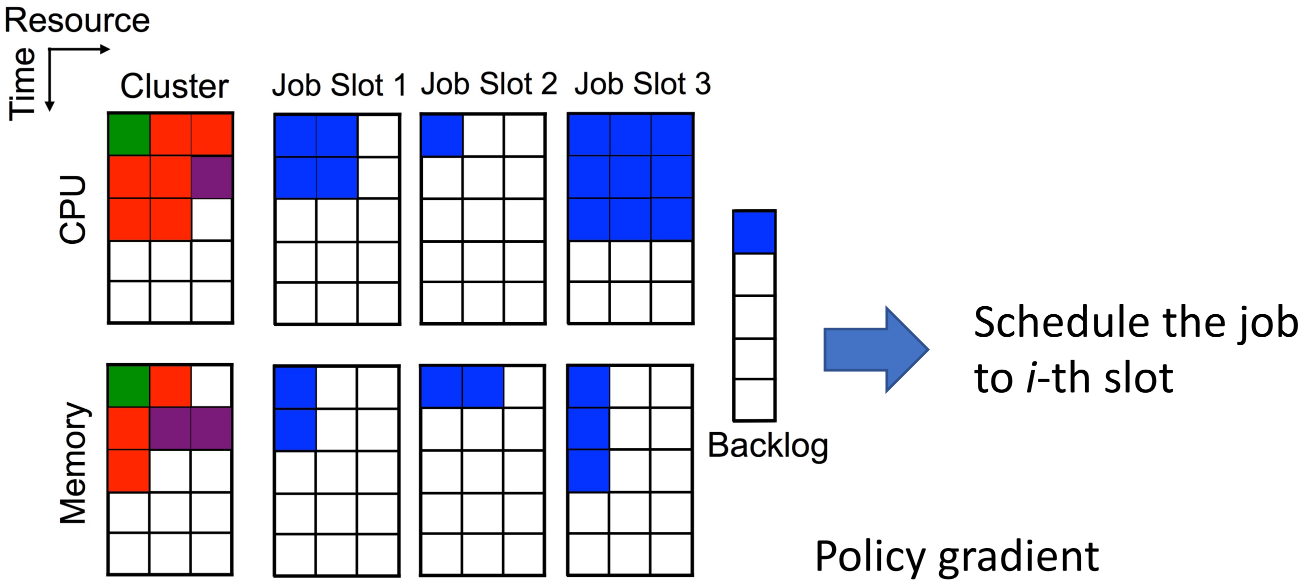
Name	Ways of Parameterization
One-shot Prediction	Spec → Solution
Progressive Prediction	Spec → SolPart1 → SolPart2 → SolPart3
Iterative Refinement	Spec → Sol1 → Sol2 (improved) → Sol3 (Better Improved)
Learned Action Space	Spec → All solution space → Small solution space → …

Representation Matters!

Direct predicting solutions

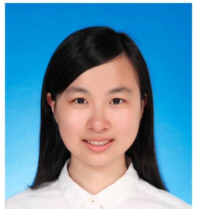


[O. Vinyals. et al, Pointer Networks, NIPS 2015]

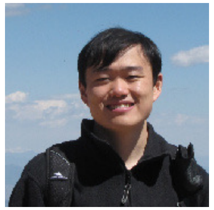


[H. Mao et al, Resource Management with Deep Reinforcement Learning, ACM Workshop on Hot Topics in Networks, 2016]

Local Rewriting Framework

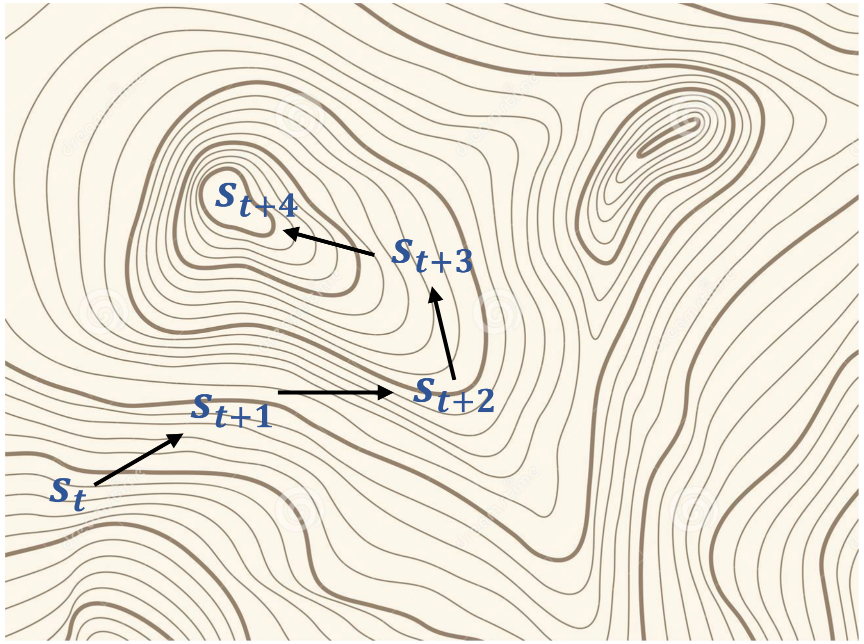


Xinyun Chen

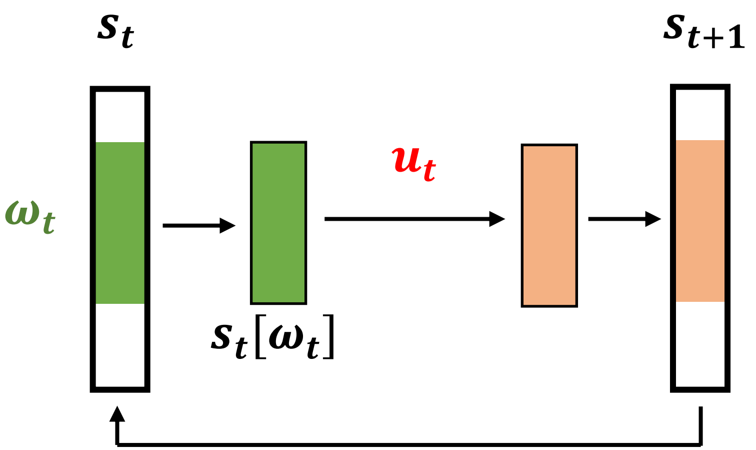


Yuandong Tian

[X. Chen and Y. Tian, ***Learning to Perform Local Rewriting for Combinatorial Optimization***, NeurIPS 2019]



Start from a feasible solution and iteratively converges to a good solution



Current State (i.e. Solution)

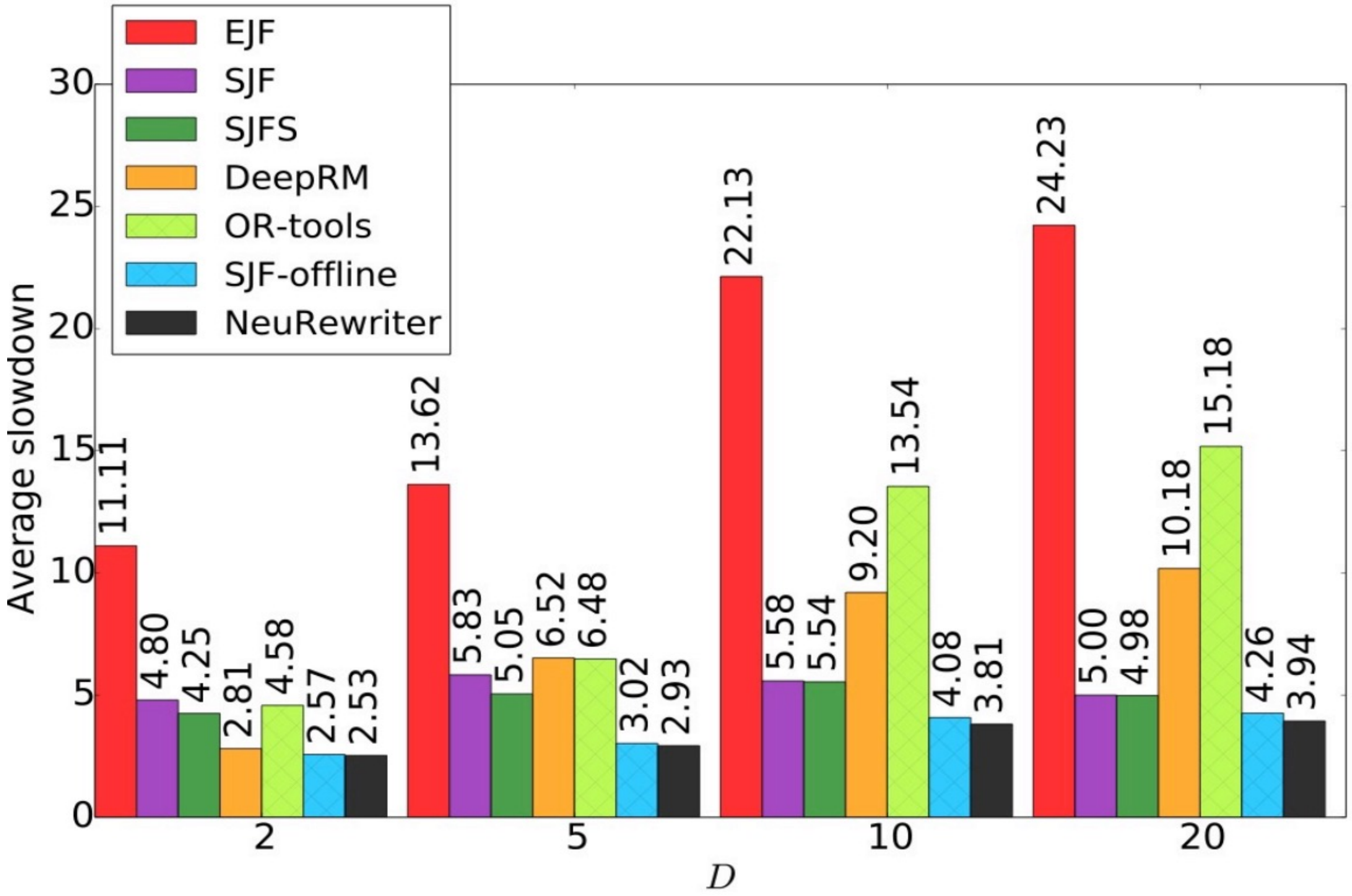
Region-Picker

Rule-Picker

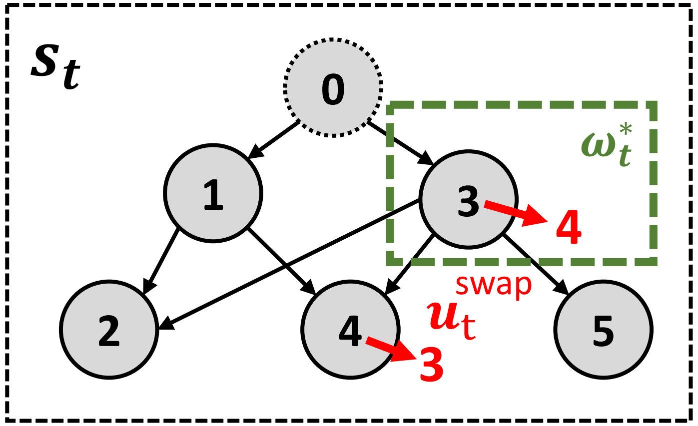
$$s_t \longrightarrow \omega_t \sim \pi_\omega(\cdot\,|s_t) \longrightarrow u_t \sim \pi_u(\cdot\,|s_t\,[\omega_t])$$

$$s_{t+1} = f(s_t, \omega_t, u_t)$$

Online Job Scheduling



D: Number of resources



Baselines:

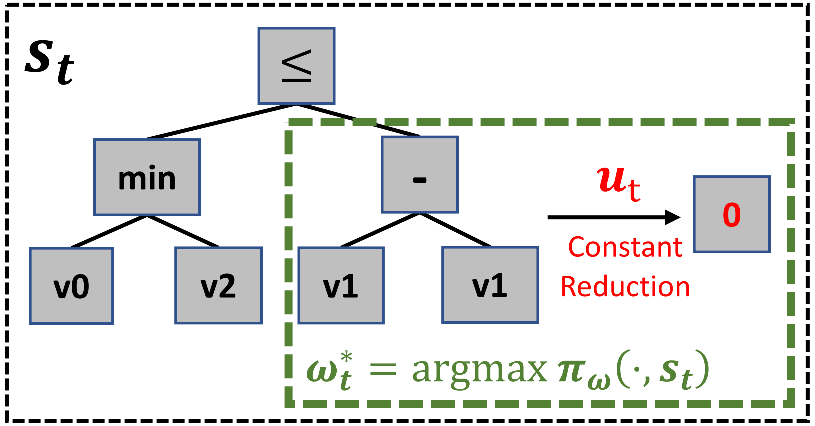
- Earliest Job First (EJF)
- Shortest Job First (SJF)
- Shortest First Search (SJFS)
- DeepRM

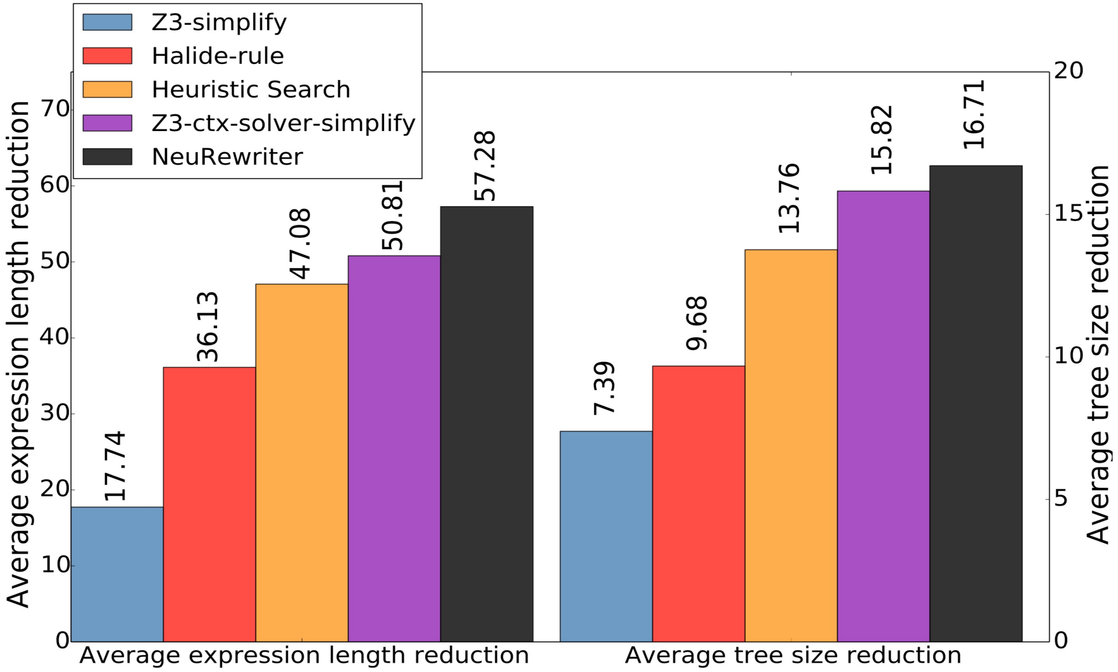
Offline baselines:

- Google OR-tools (OR-tools)
- SJF-offline

	Time (s)
OR-tools	10.0
DeepRM	0.020
NeuRewriter	0.037

Expression Simplification





Baselines:

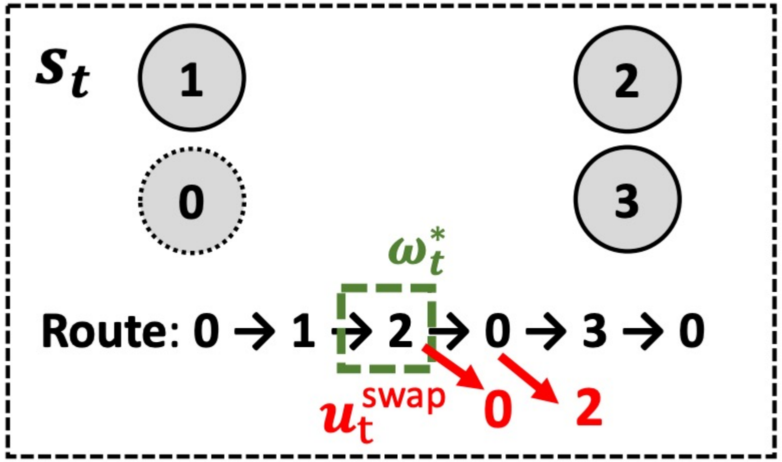
- Z3-simplify
- Z3-ctx-solver-simplify
- Heuristic Search
- Halide rules

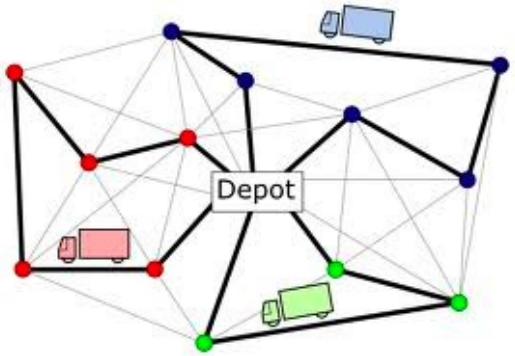
	Time (s)
Z3-solver	1.375
NeuRewriter	0.159

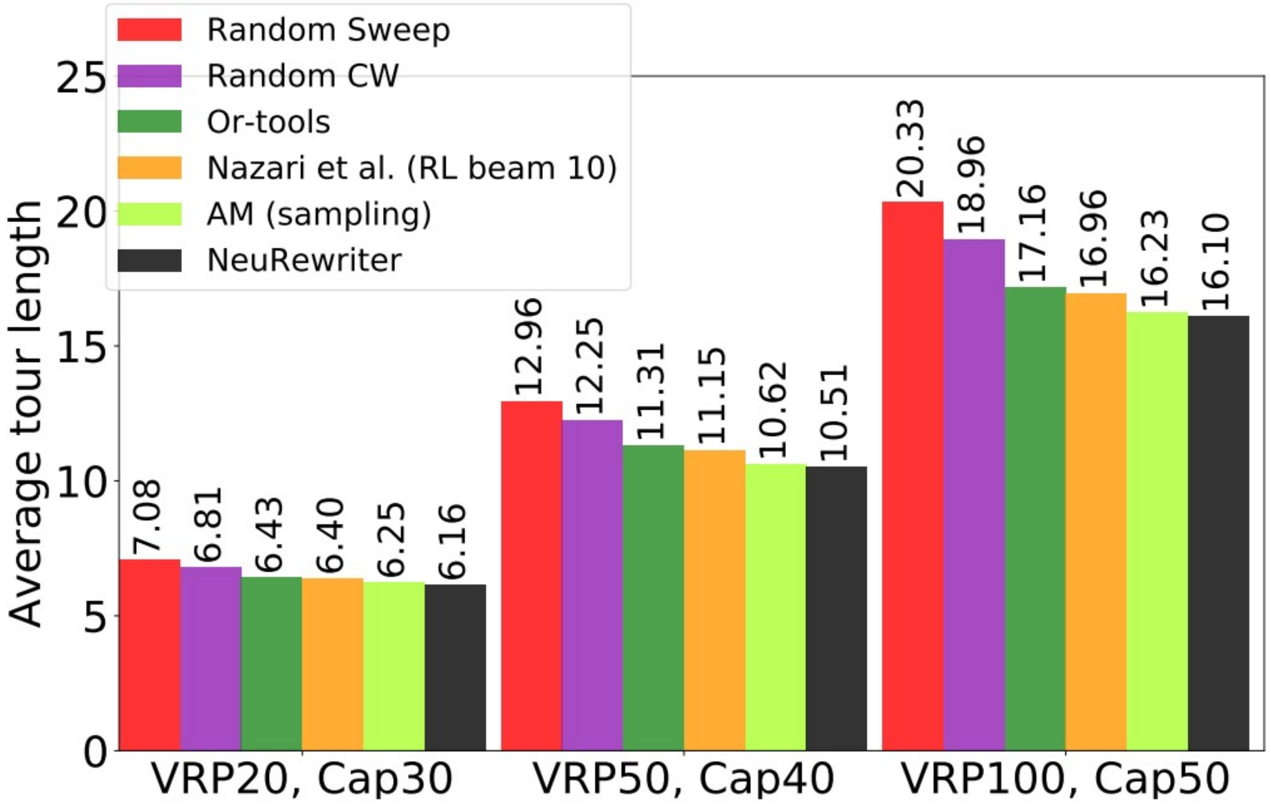
Follow-up work: Getting rid of manually specified rules

[H. Shi et al., Deep Symbolic Superoptimization without Human Knowledge, ICLR 2020]

Capacitated Vehicle Routing

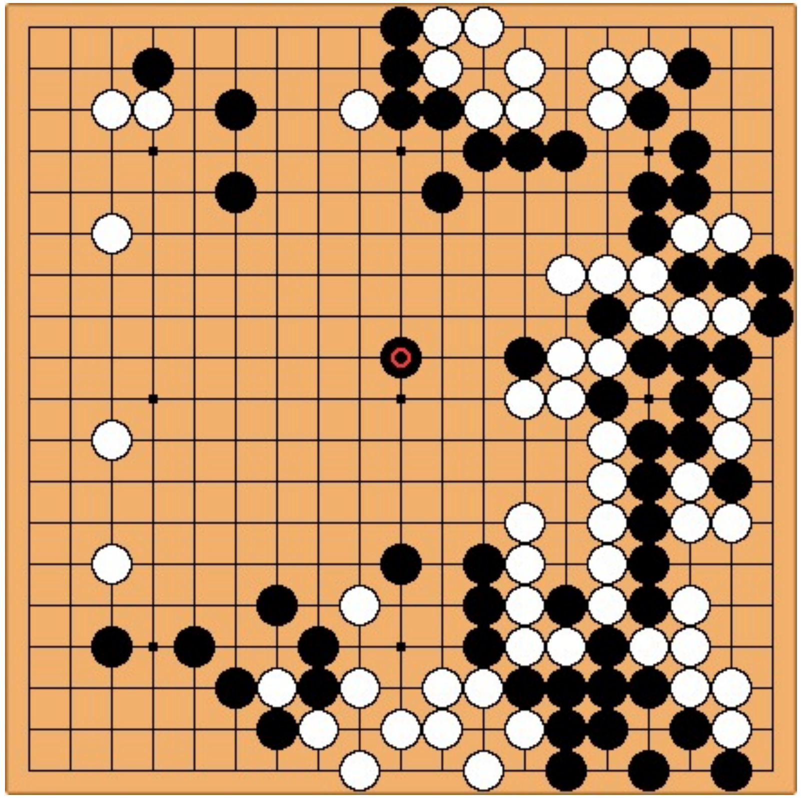




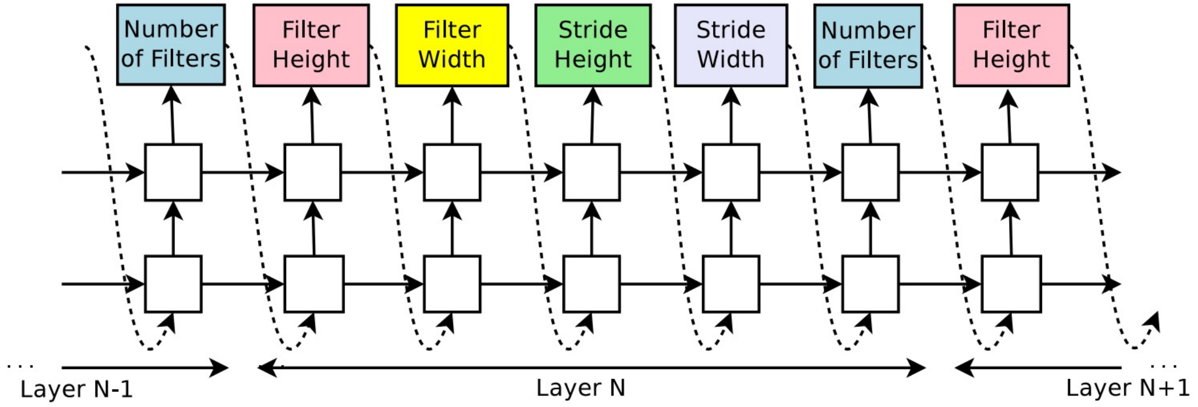


Code is available: https://github.com/facebookresearch/neural-rewriter

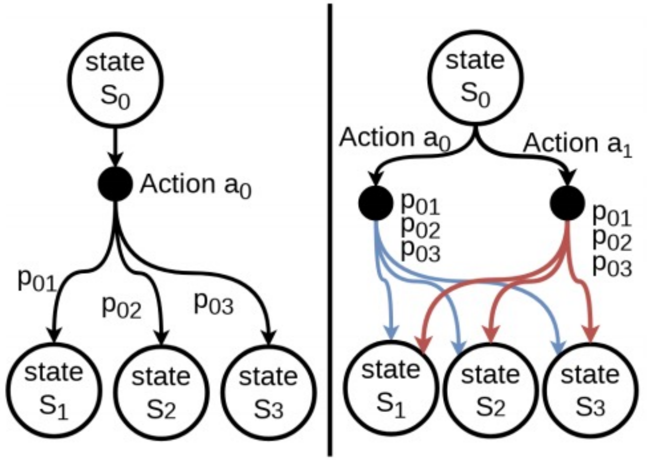
Predefined Action Space



Fixed action space = R^{361}



[B. Zoph and Q. Le, Neural Architecture Search with Reinforcement Learning, 2016]



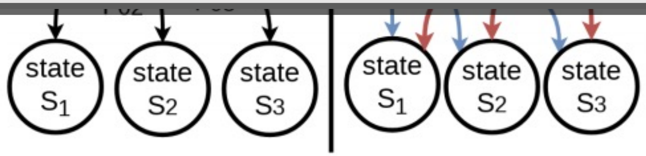
[G. Malazgirt, TauRieL: Targeting Traveling Salesman Problem with a deep reinforcement learning inspired architecture]

Predefined Action Space

Why Predefined Action Space?

N+1

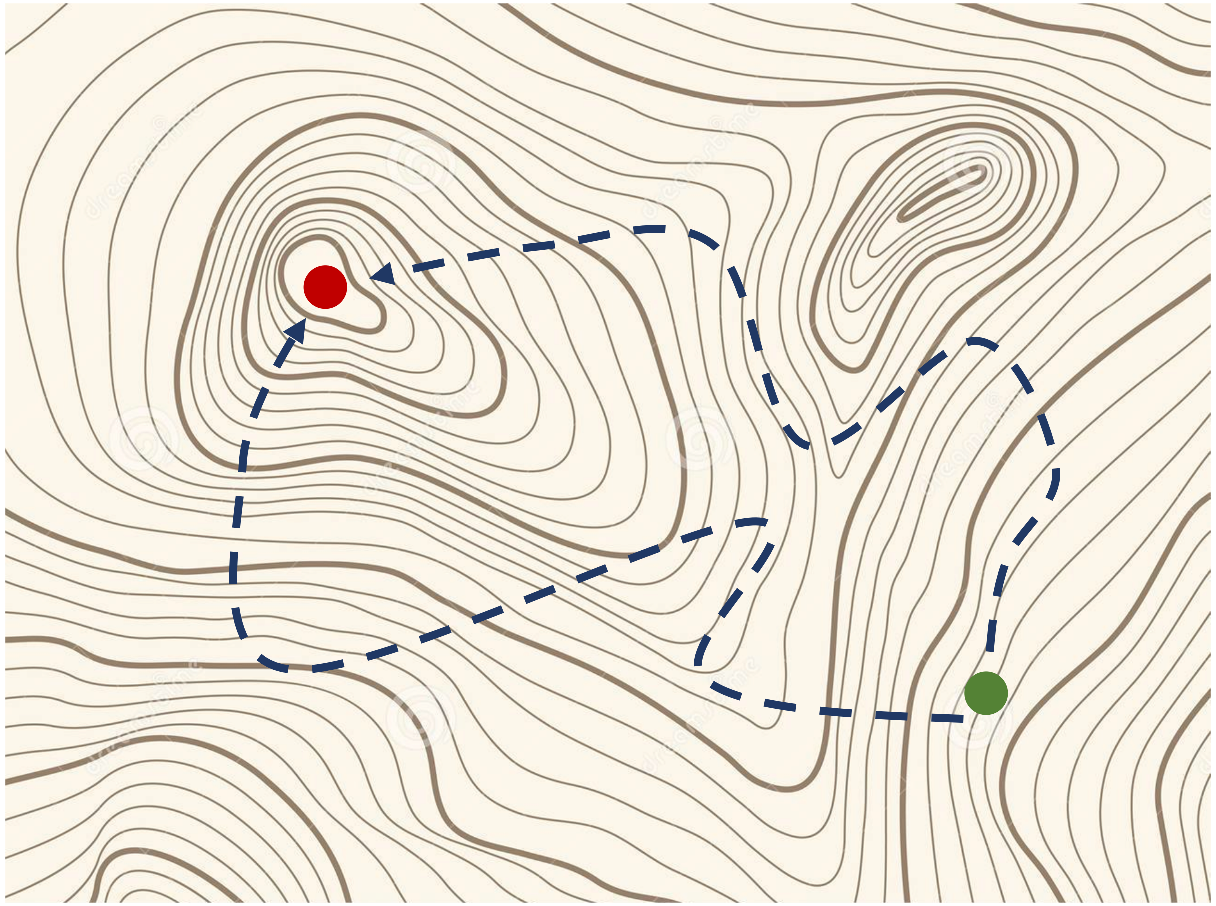
2016]



Fixed action space = R^{361}

[G. Malazgirt, TauRieL: Targeting Traveling Salesman Problem with a deep reinforcement learning inspired architecture]

Why Predefined Action Space?



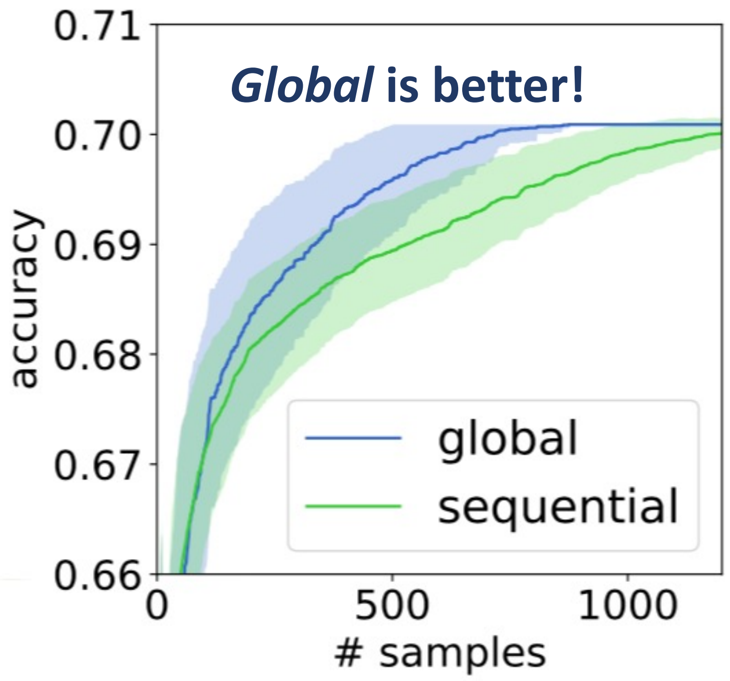
We only care the final solution

We don't care how we get it.

Different Representation matters

Depth = {1, 2, 3, 4, 5}
Channels = {32, 64}
KernelSize = {3x3, 5x5}

1364 networks.

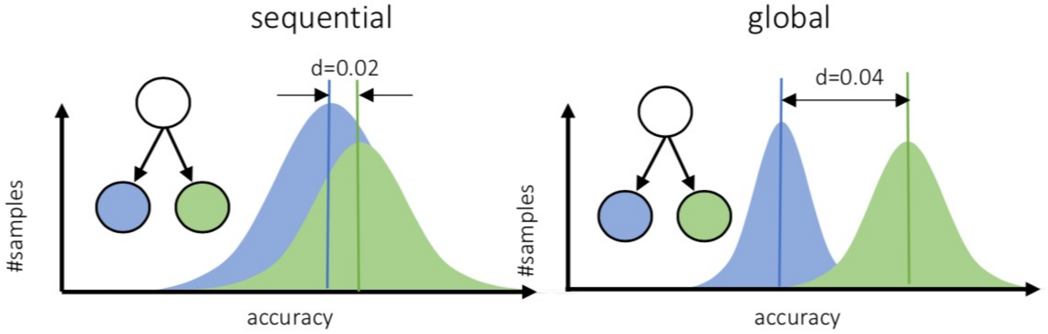


Goal: Find the network with the best accuracy using fewest trials.

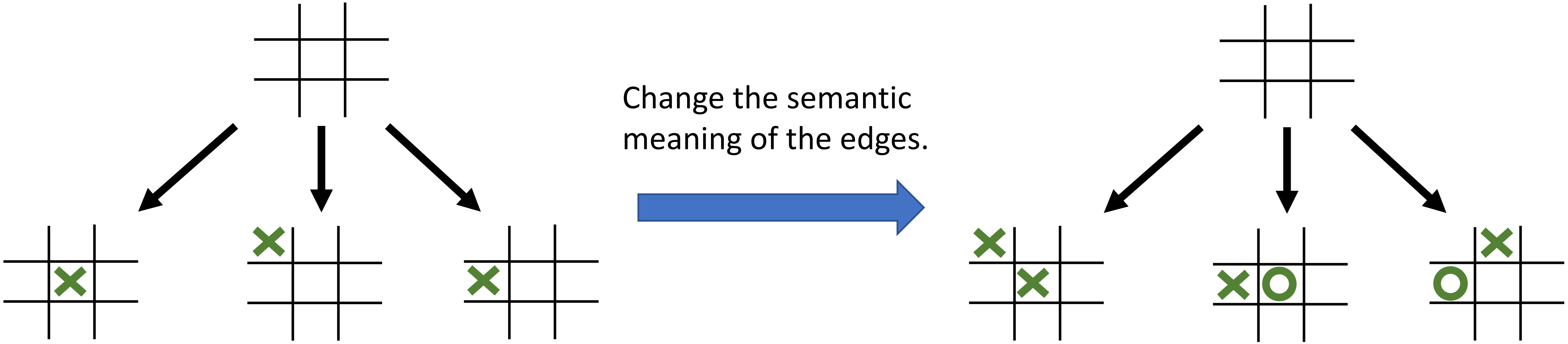
Representation of action space

Sequential = { add a layer, set K, set C }

Global = { Set depth, set all K, set all C }



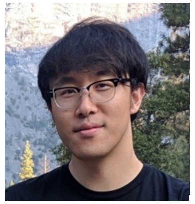
The Meaning of Learning Action Space



Not allowed in games, but doable in optimization.

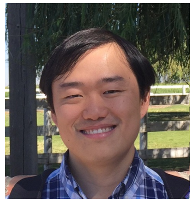
Learning Action Space


Linnan Wang


Saining Xie

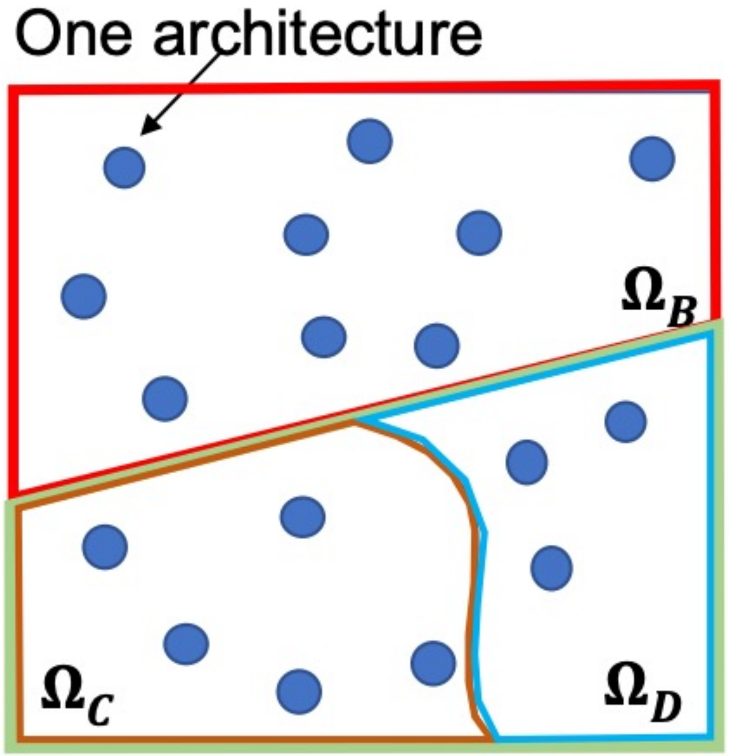

Teng Li

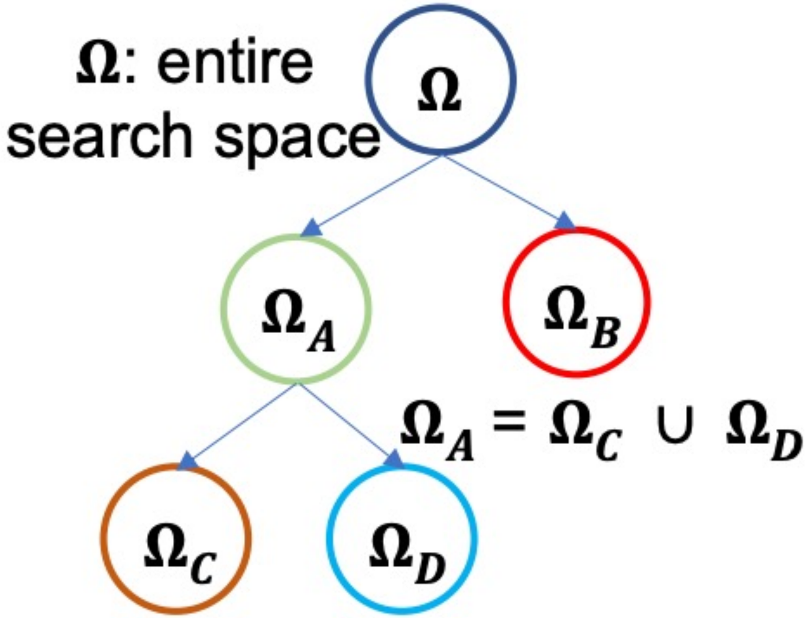

Rodrigo Fonseca


Yuandong Tian

[L. Wang, R. Fonseca, Y. Tian, **Learning Search Space Partition for Black-box Optimization using Monte Carlo Tree Search,** *NeurIPS 2020]*

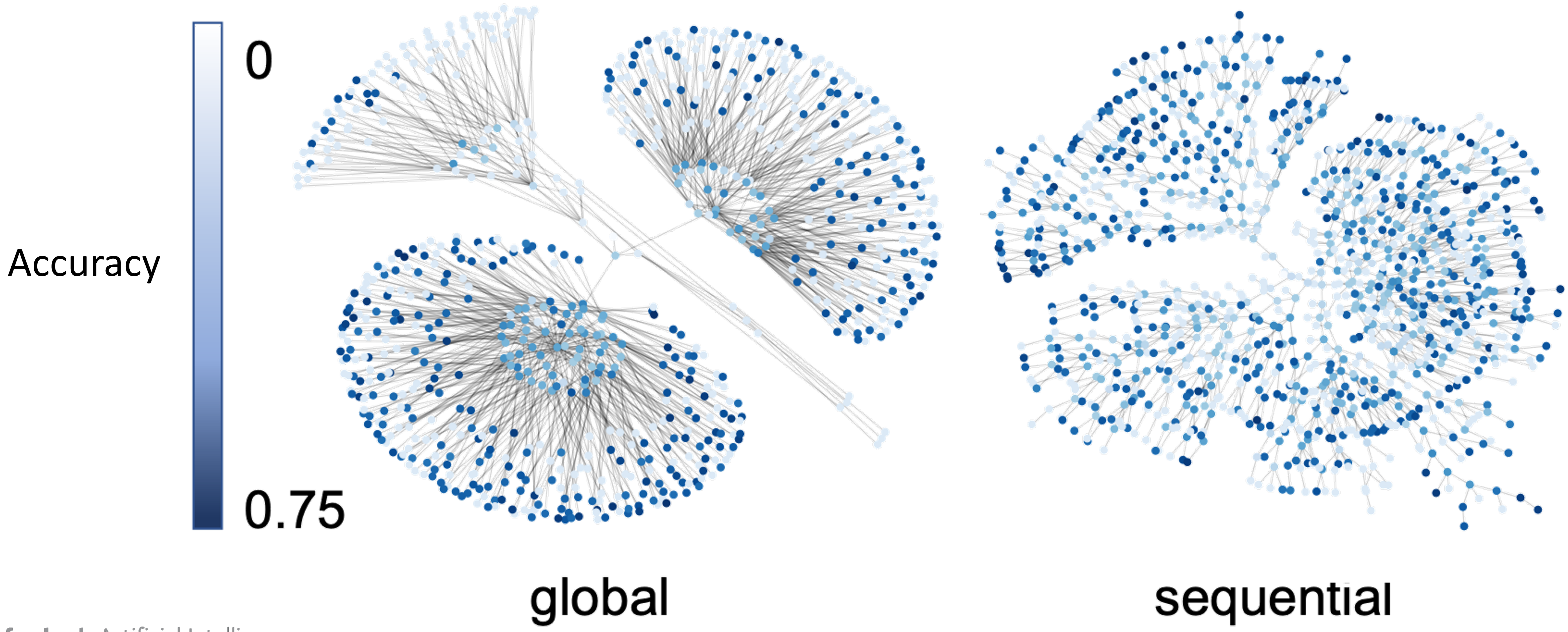
[L. Wang, S. Xie, T. Li, R. Fonseca, Y. Tian, **Sample-Efficient Neural Architecture Search by Learning Action Space,** *TPAMI 2021]*



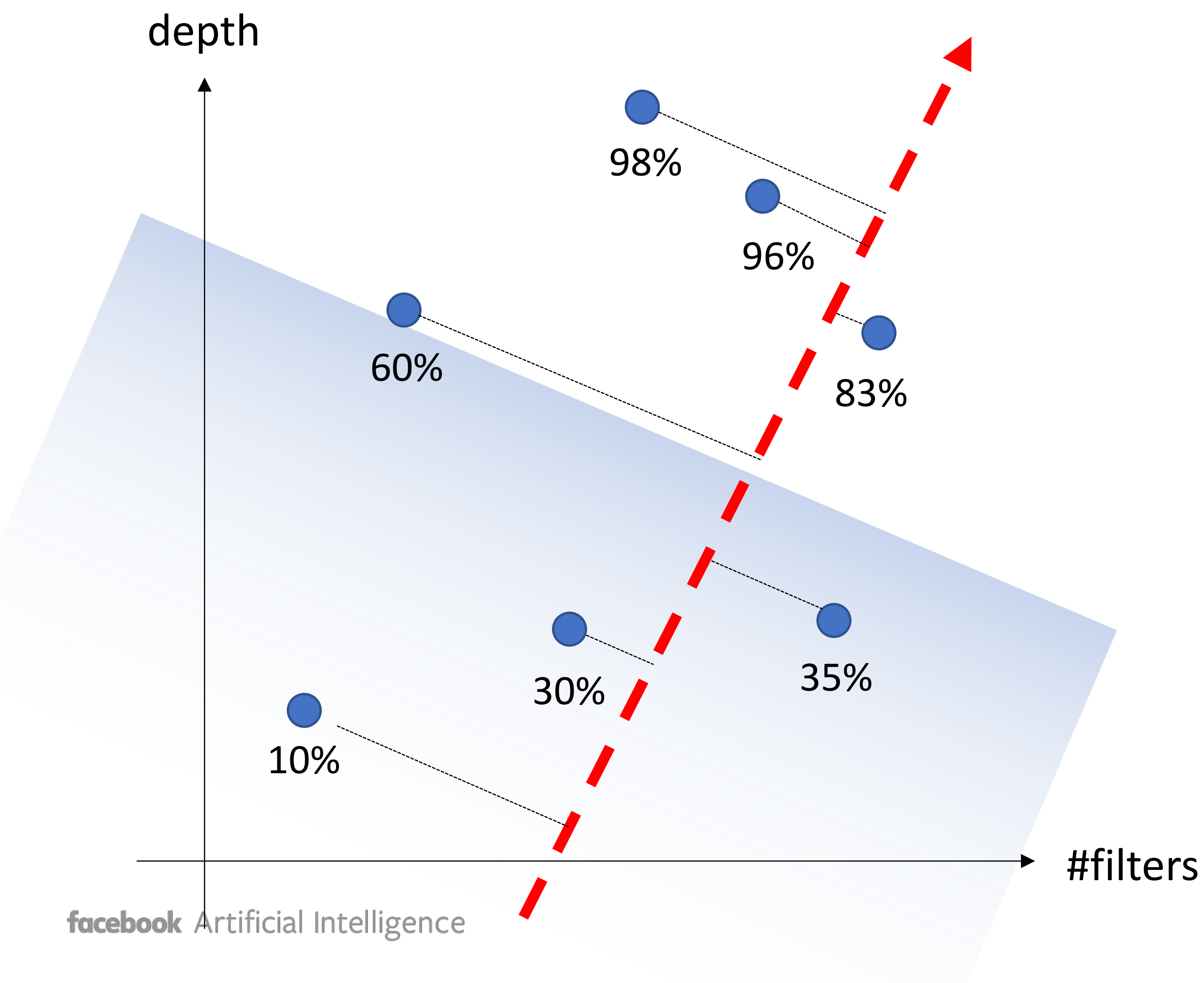


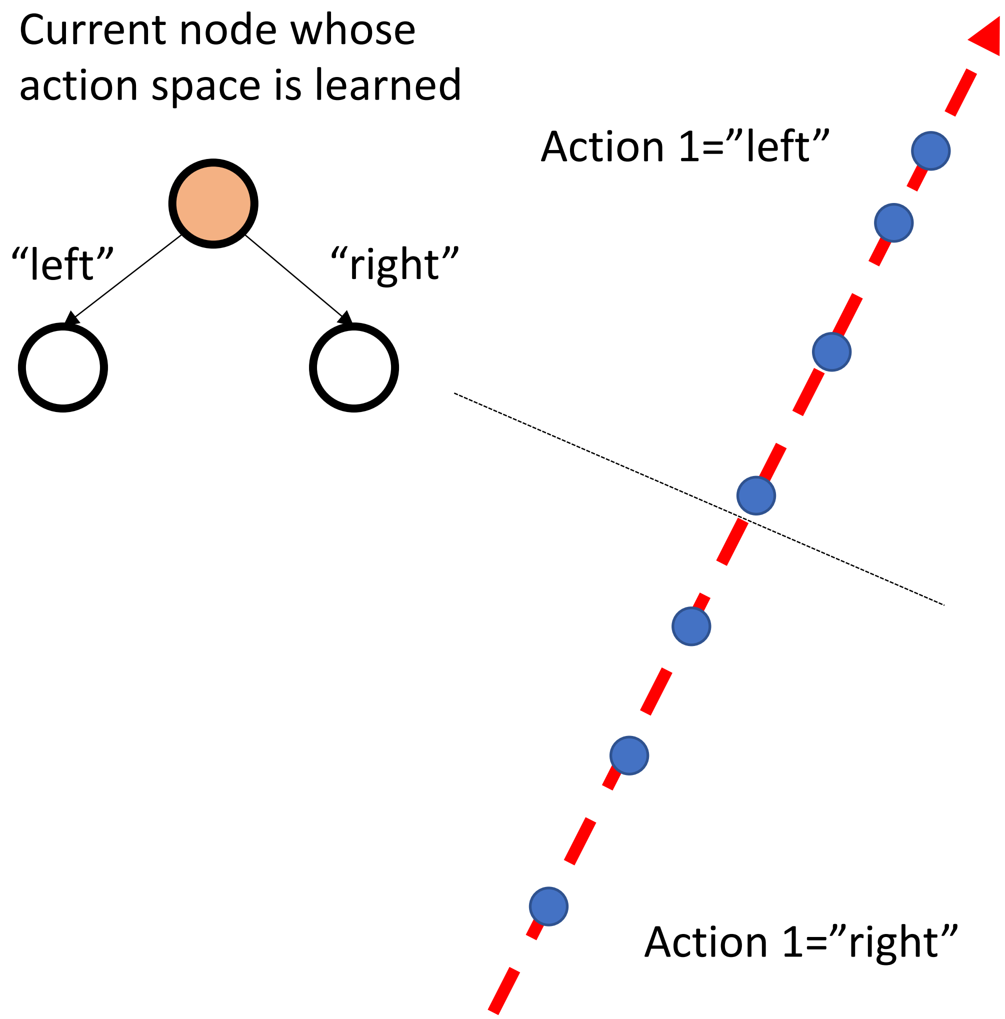
Partition = Action

Different Partition → Different Value Distribution

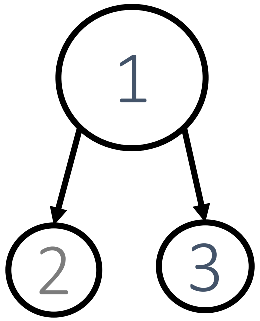


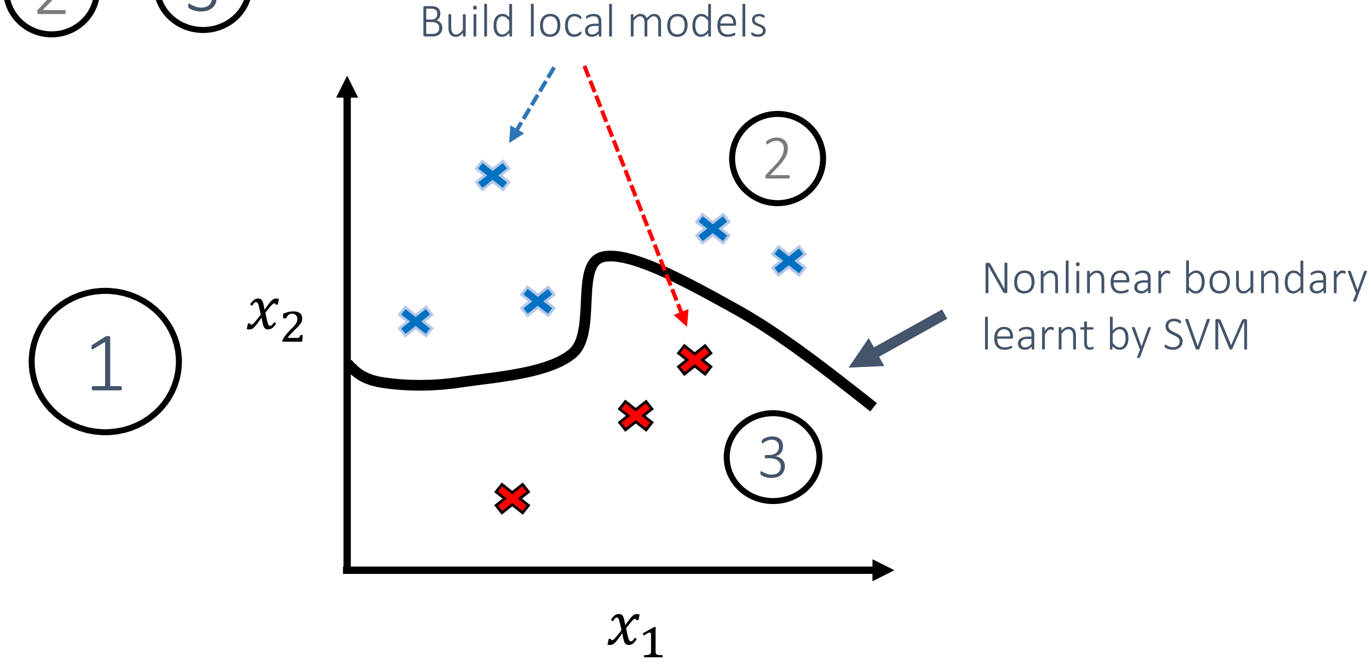
Learn action space

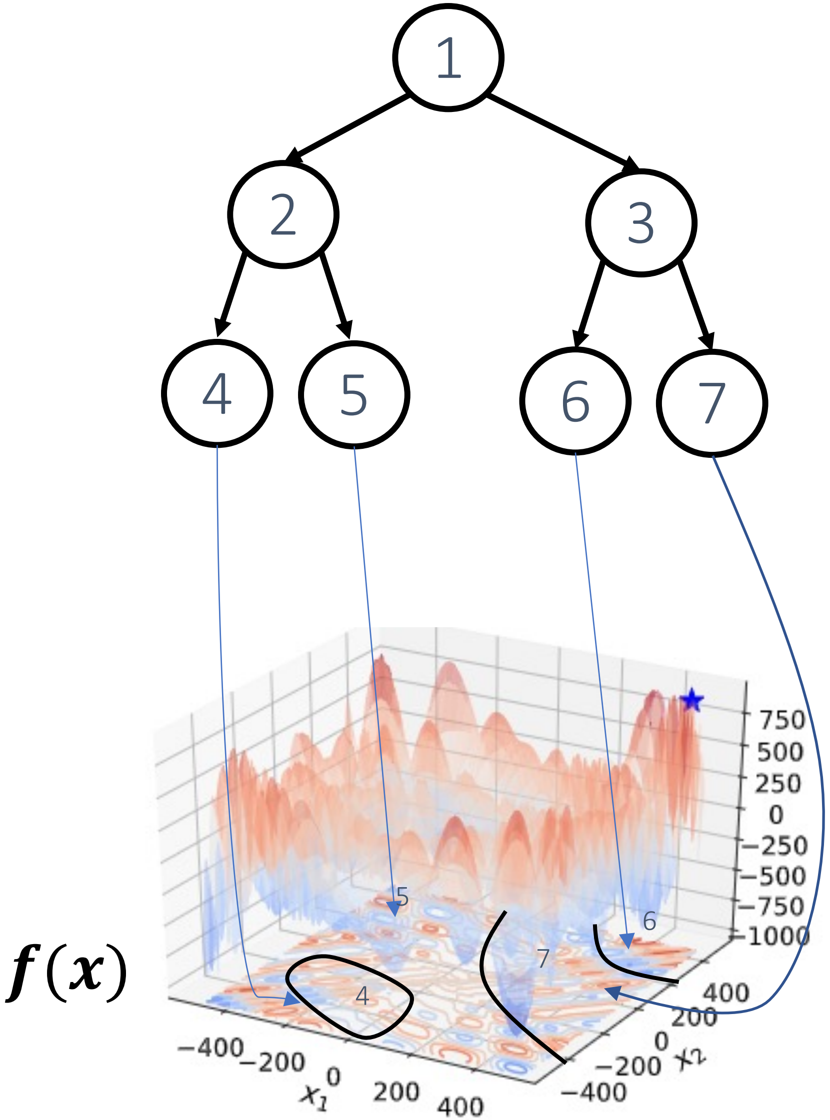




Nonlinear Partition

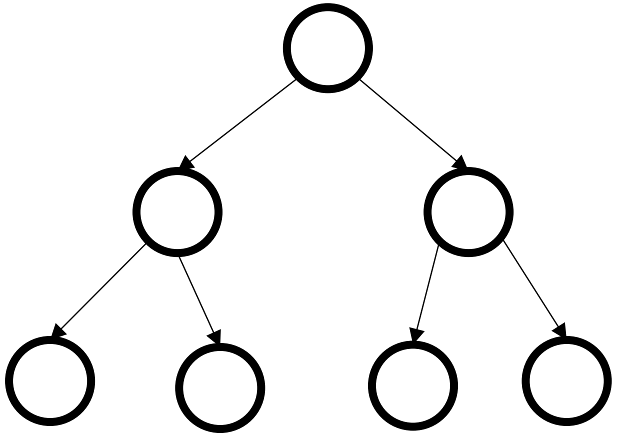






[L. Wang, et al, Learning Search Space Partition for Black-box Optimization using MCTS, NeurIPS 2020]

Approach



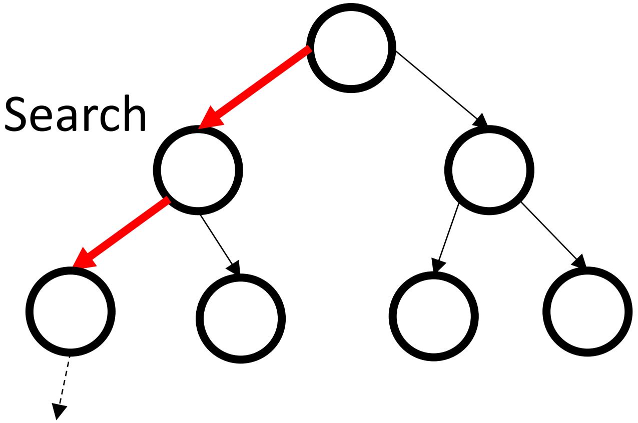
Fixed action branches
(but not action space)

(a) Train the action space.

	Accuracy
(filter=2, depth=5)	85%
(filter=3, depth=7)	92%

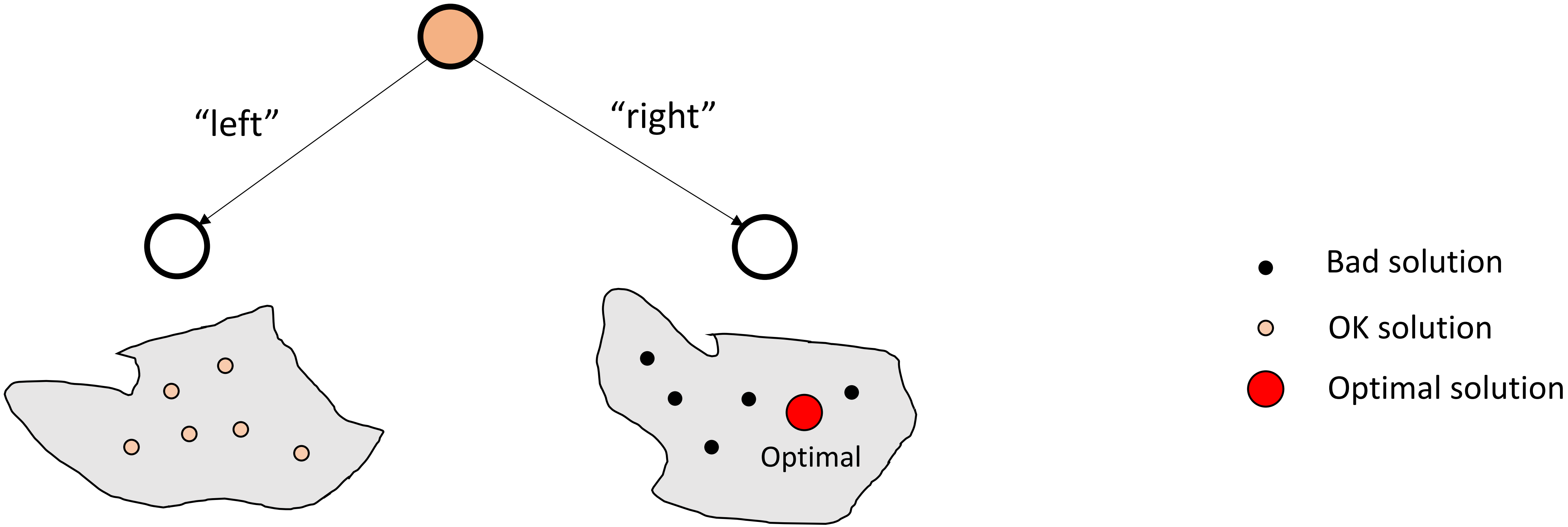
(b) Search using learned action space until a fixed #rollouts are used.

Monte Carlo Tree Search
(MCTS)



Getting the true quality $f(x)$ for the solution x

Why Exploration is Important

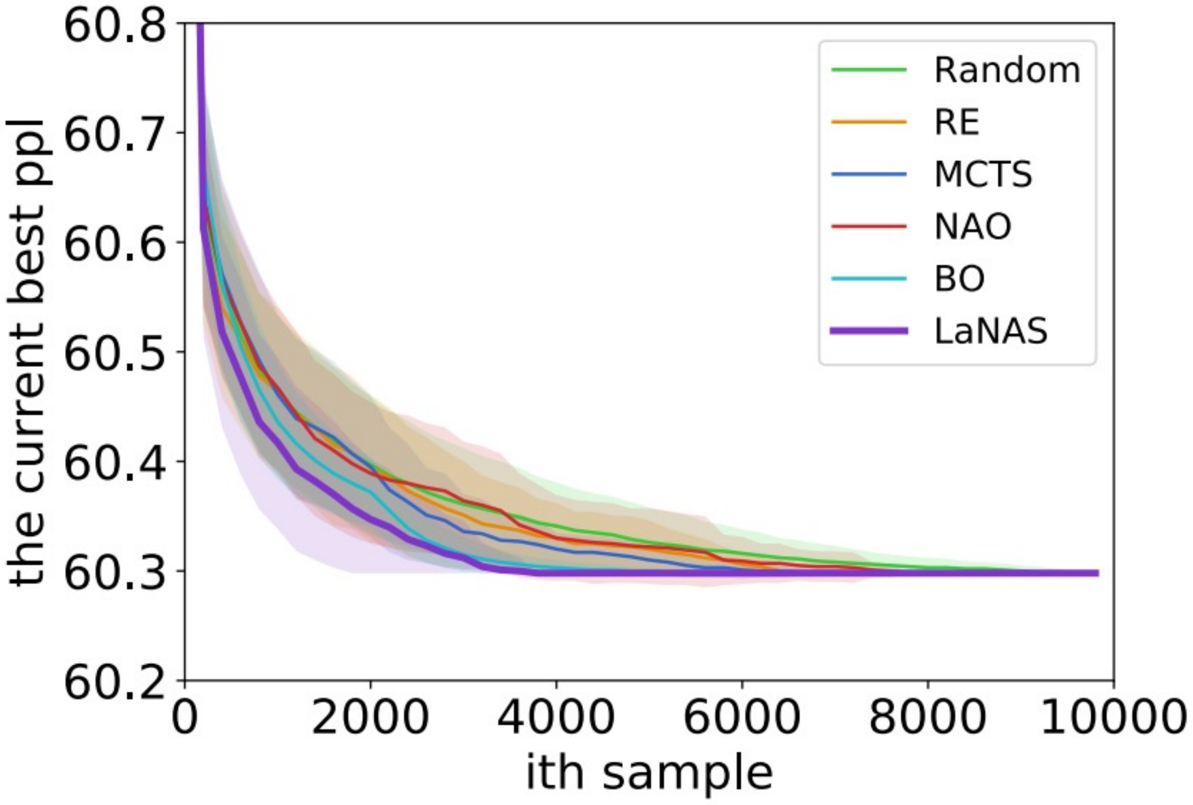


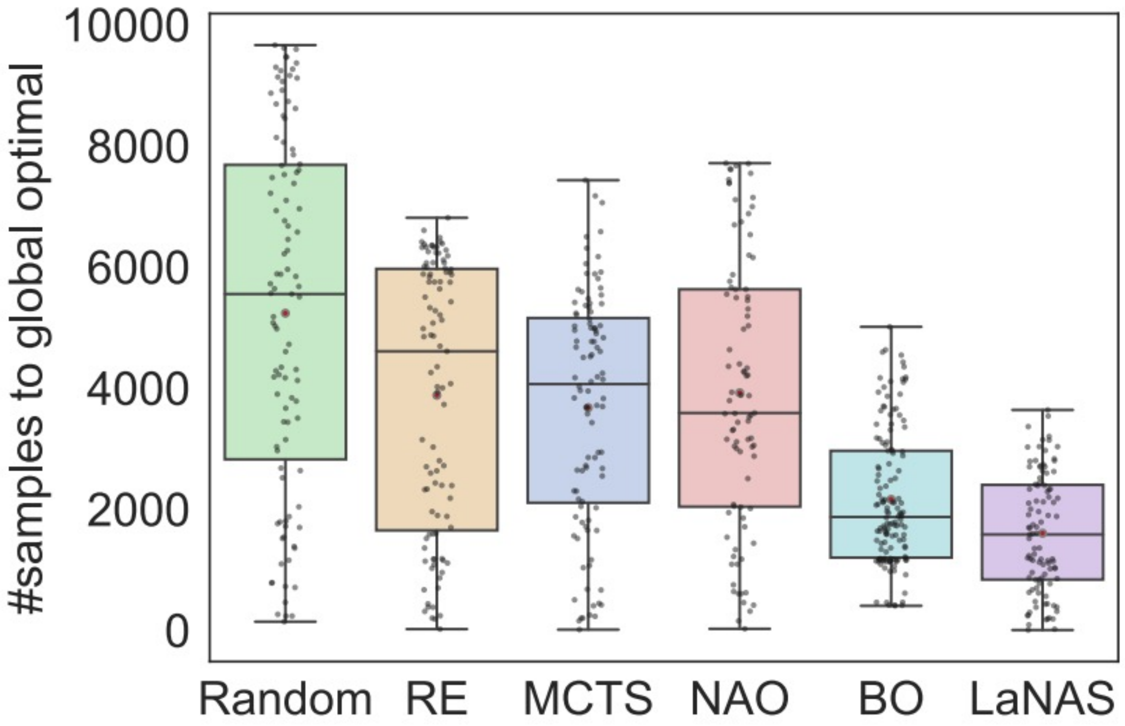
OK solutions
but not optimal

Most solutions are bad but there
exists an optimal one

Performance

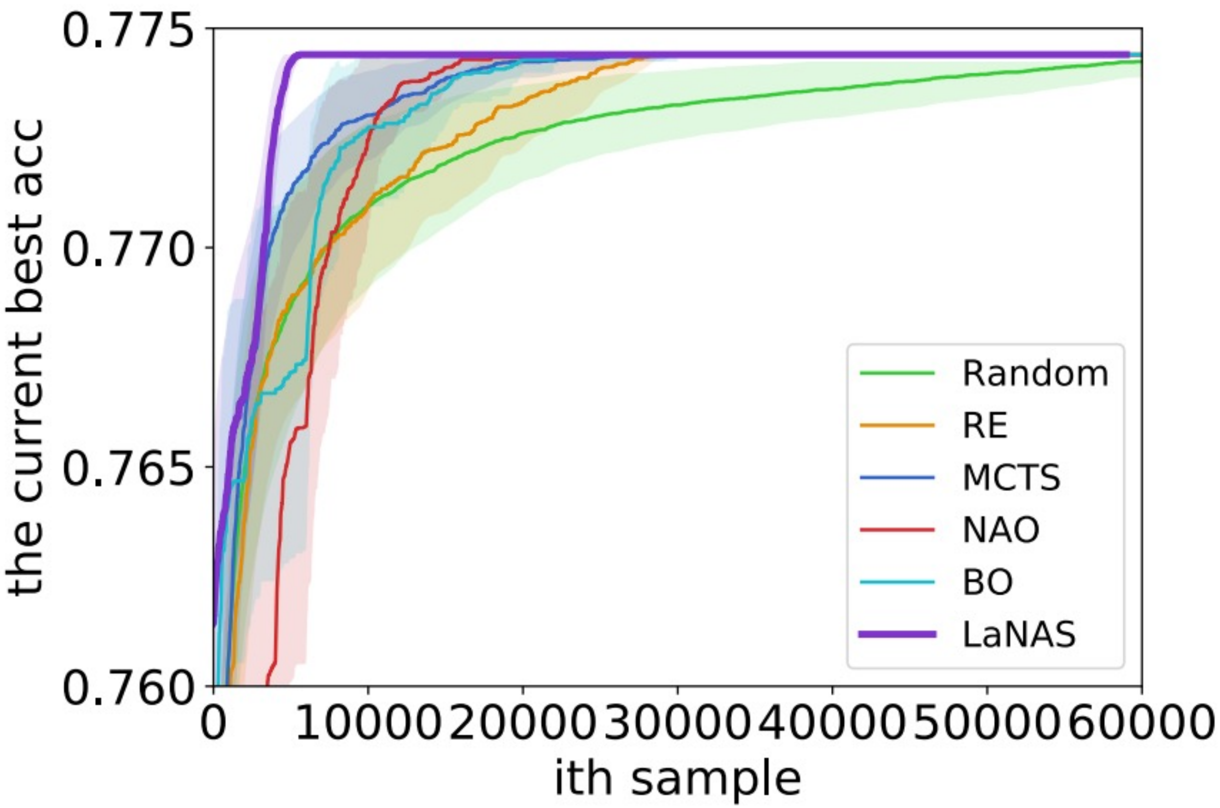
Customized dataset: LSTM-10K (PTB)

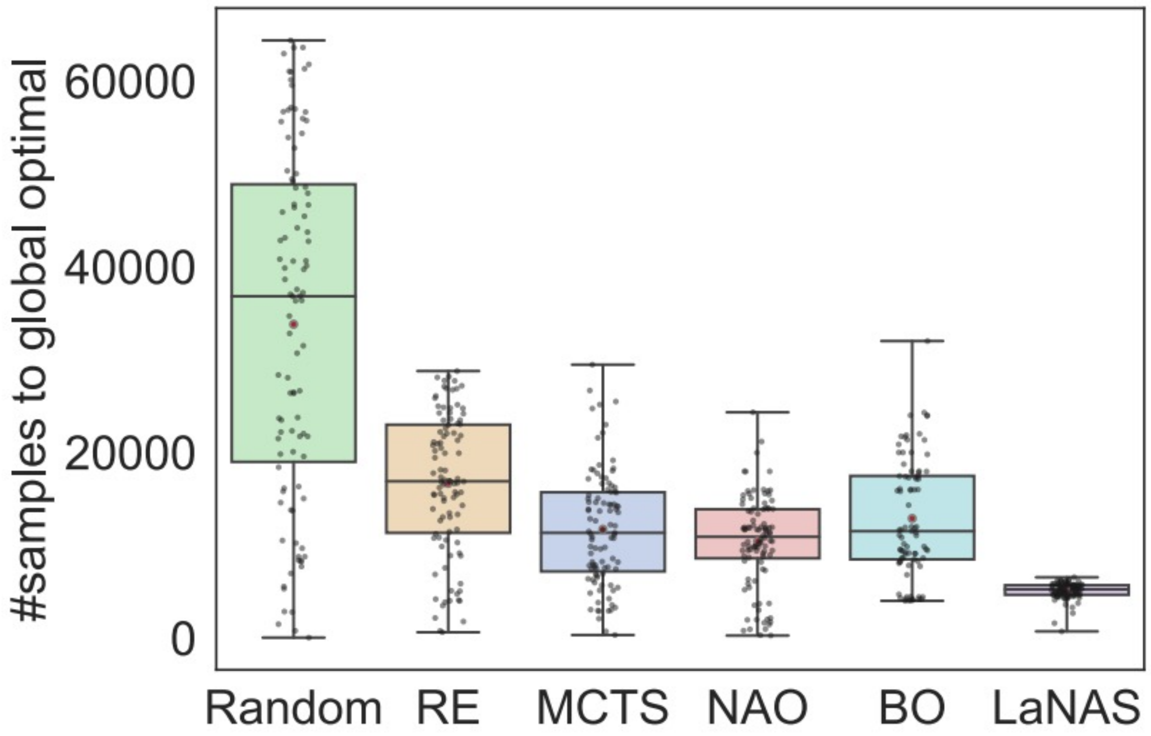




Performance

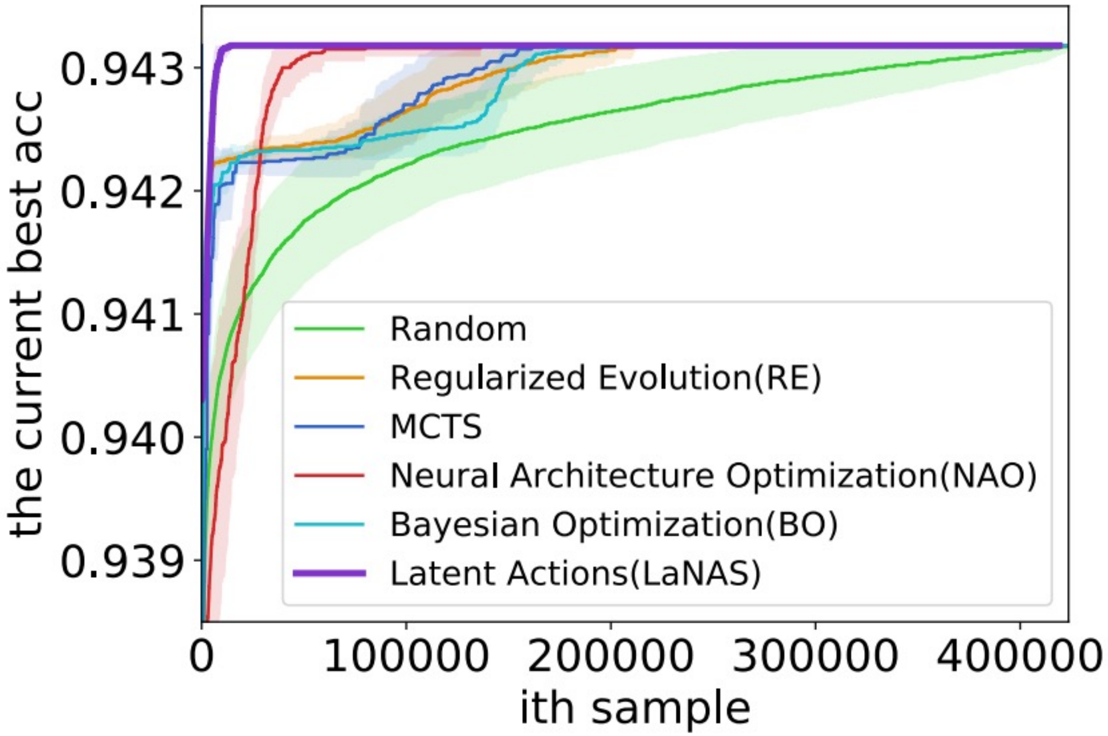
Customized dataset: ConvNet-60K (CIFAR-10, VGG style models)

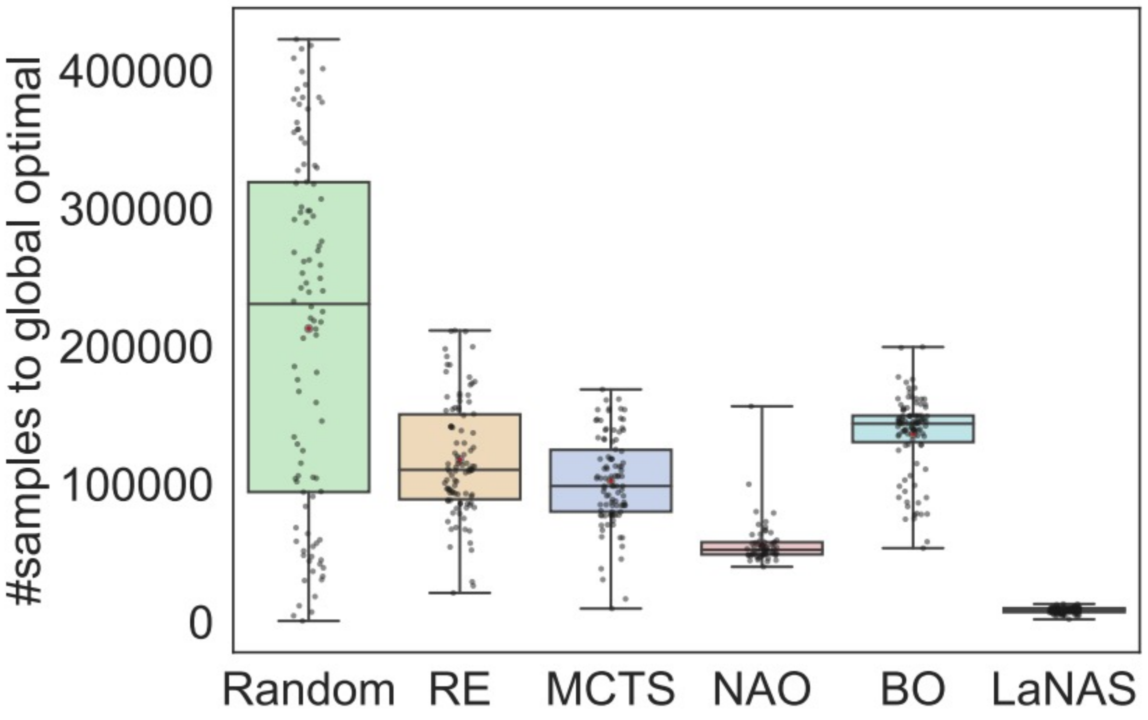




Performance

NASBench-101 (CIFAR-10, 420k models, NASNet Search Space)





Each curve is repeated 100 times. We randomly pick 2k models to initialize.

Open Domain

CIFAR-10 (NASNet style architecture)

Model	Using ImageNet	Params	Top1 err	M	GPU days
		search based methods			
NASNet-A+c/o [22]	X	3.3 M	2.65	20000	2000
AmoebaNet-B+c/o [10]	X	2.8 M	$2.55_{\pm 0.05}$	27000	3150
PNASNet-5 [29]	X	3.2 M	$3.41_{\pm 0.09}$	1160	225
NAO+c/o [30]	X	128.0 M	2.11	1000	200
AmoebaNet-B+c/o	X	34.9 M	$2.13_{\pm 0.04}$	27000	3150
EfficientNet-B7	√	64M	1.01		
BiT-M	√	60M	1.09		
LaNet+c/o	X	3.2 M	$\mathbf{1.63}_{\pm 0.05}$	800	150
LaNet+c/o	X	44.1 M	$\mathbf{0.99}_{\pm 0.02}$	800	150
		one-shot NAS based methods			
ENAS+c/o [18]	X	4.6 M	2.89	-	0.45
DARTS+c/o [20]	X	3.3 M	$2.76_{\pm 0.09}$	-	1.5
BayesNAS+c/o [31]	X	3.4 M	$2.81_{\pm 0.04}$	-	0.2
ASNG-NAS+c/o [32]	X	3.9 M	$2.83_{\pm 0.14}$	-	0.11
XNAS+c/0 [33]	X	3.7 M	1.81		0.3
oneshot-LaNet+c/o	X	3.6 M	$\mathbf{1.68}_{\pm 0.06}$	-	3
oneshot-LaNet+c/o	X	45.3 M	$\mathbf{1.2}_{\pm 0.03}$	-	3

M: number of samples selected.

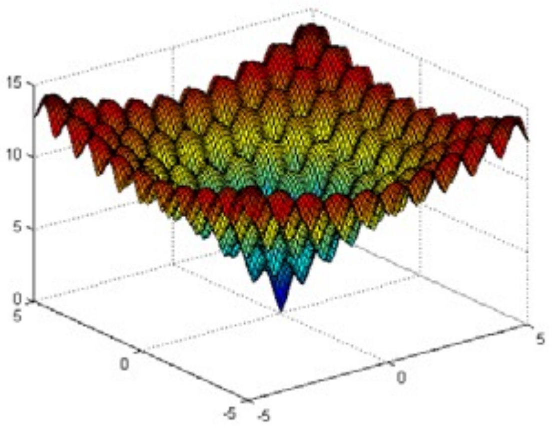
Open Domain

ImageNet
(mobile setting
Flop < 600M)

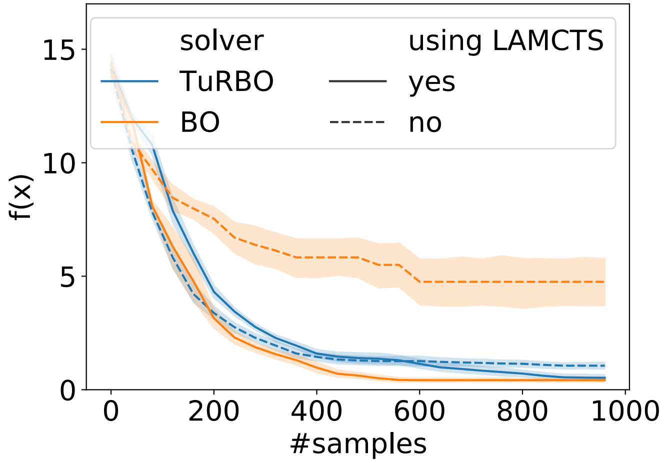
Model	FLOPs	Params	top1 / top5 err
NASNet-A (Zoph et al. (2018))	564M	5.3 M	26.0 / 8.4
NASNet-B (Zoph et al. (2018))	488M	5.3 M	27.2 / 8.7
NASNet-C (Zoph et al. (2018))	558M	4.9 M	27.5 / 9.0
AmoebaNet-A (Real et al. (2018))	555M	5.1 M	25.5 / 8.0
AmoebaNet-B (Real et al. (2018))	555M	5.3 M	26.0 / 8.5
AmoebaNet-C (Real et al. (2018))	570M	6.4 M	**24.3 / 7.6**
PNASNet-5 (Liu et al. (2018a))	588M	5.1 M	25.8 / 8.1
DARTS (Liu et al. (2018b))	574M	4.7 M	26.7 / 8.7
FBNet-C (Wu et al. (2018))	375M	5.5 M	25.1 / -
RandWire-WS (Xie et al. (2019))	583M	5.6 M	25.3 / 7.8
BayesNAS (Zhou et al. (2019))	-	3.9 M	26.5 / 8.9
LaNet	570M	5.1 M	**25.0 / 7.7**

La-MCTS as a meta method

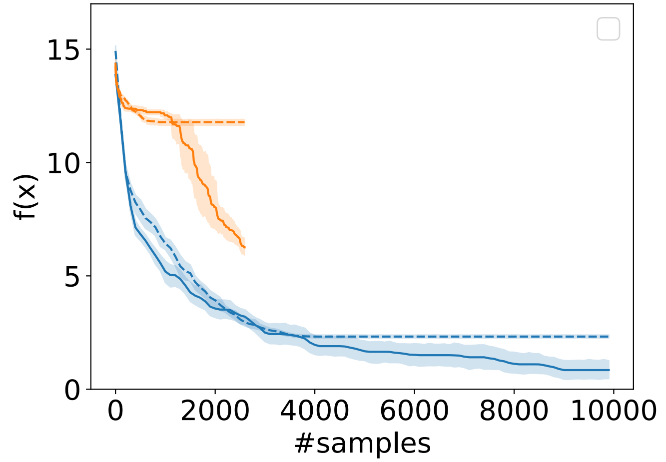
$$x^* = \arg\min_{x \in \Omega} f(x)$$



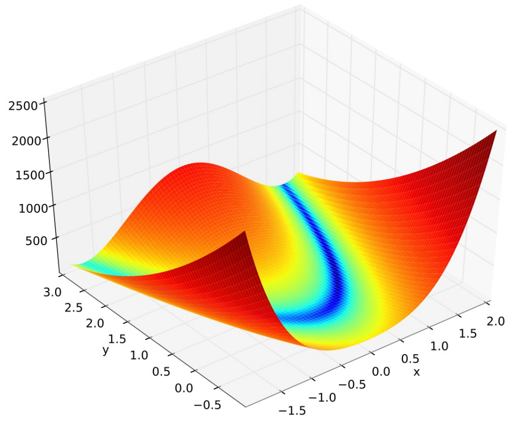
Ackley-2d



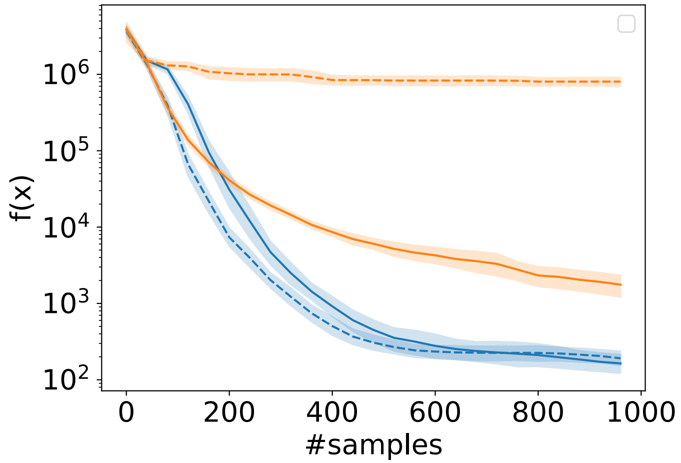
Ackley-20d



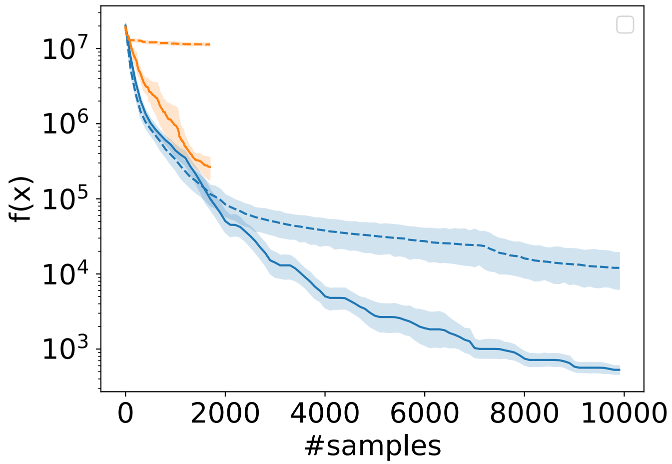
Ackley-100d



Rosenbrock-2d

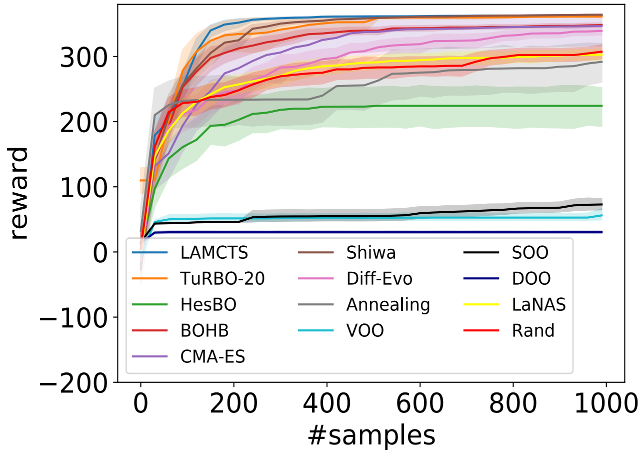


Rosenbrock-20d

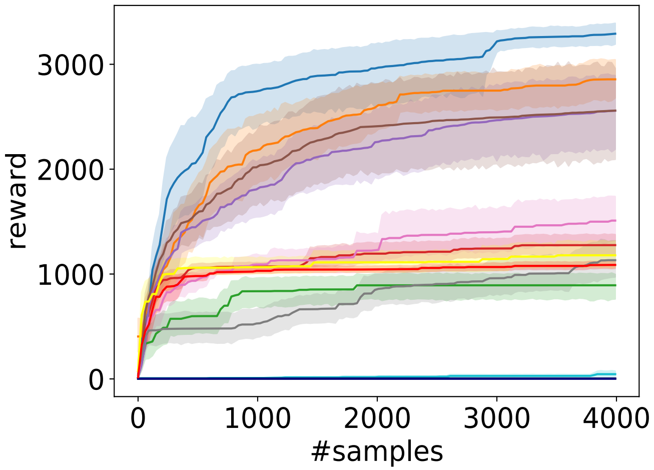


Rosenbrock-100d

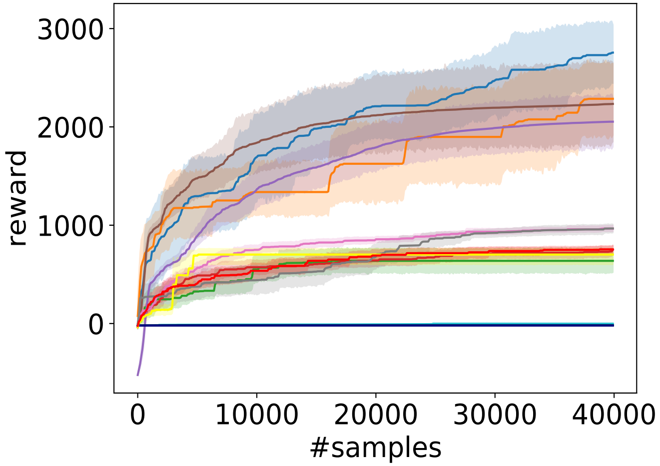
Optimizing linear policy for Mujoco tasks



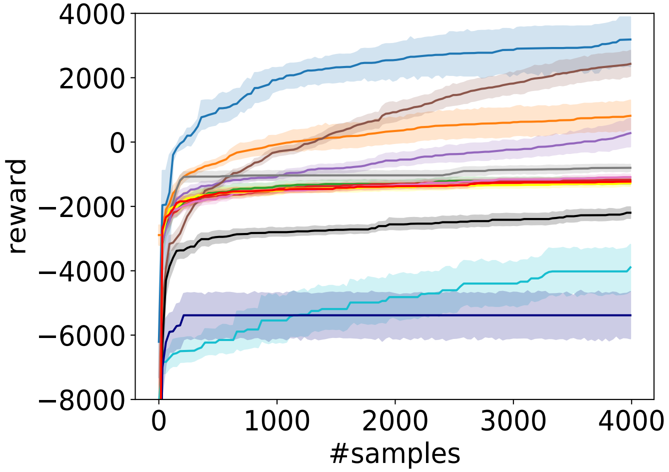
(a) Swimmer, #params = 16



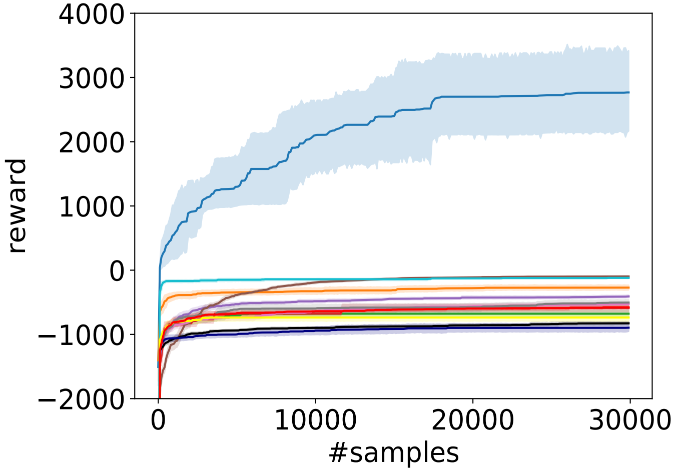
(b) Hopper, #params = 33



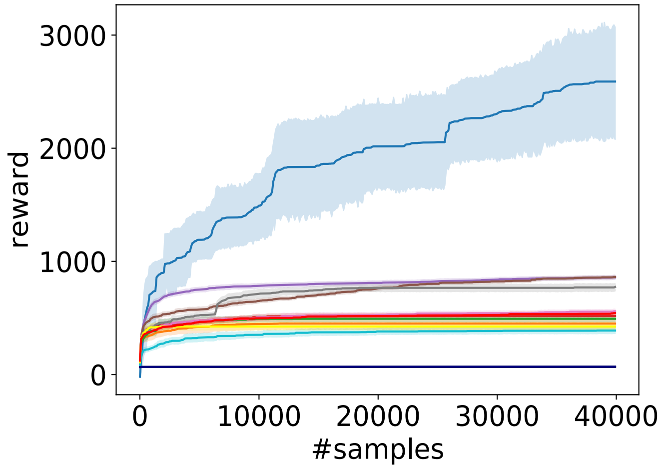
(c) Walker-2d, #params = 102



(d) Half-Cheetah, #params =102

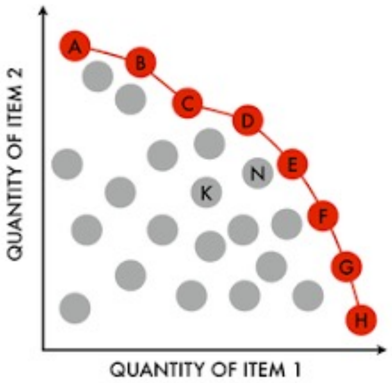


(e) Ant, #params = 888

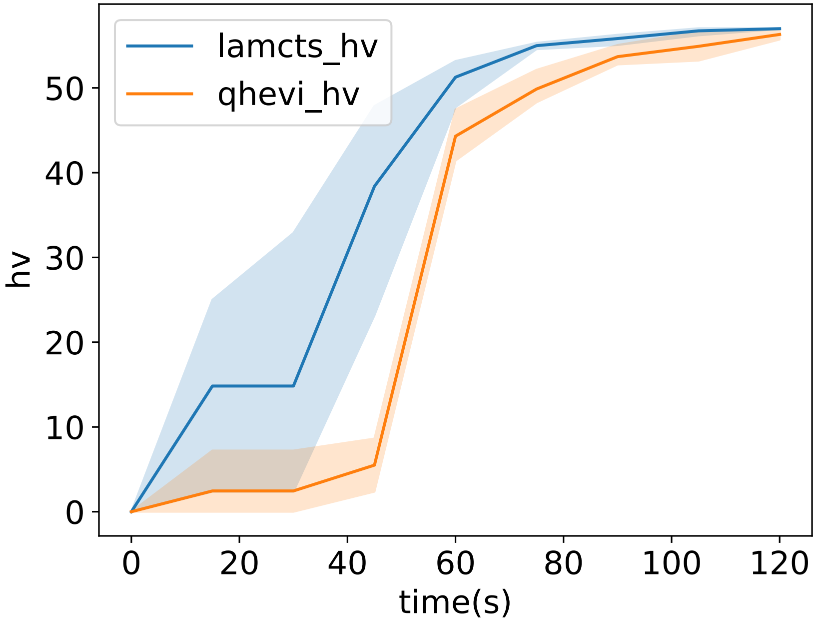


(f) Humanoid, #params = 6392

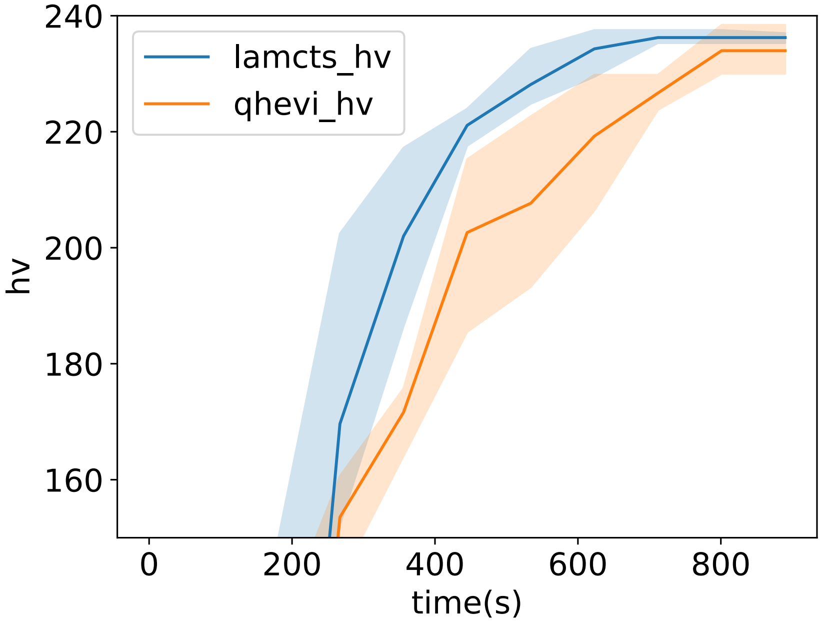
Multi-Objective Optimization



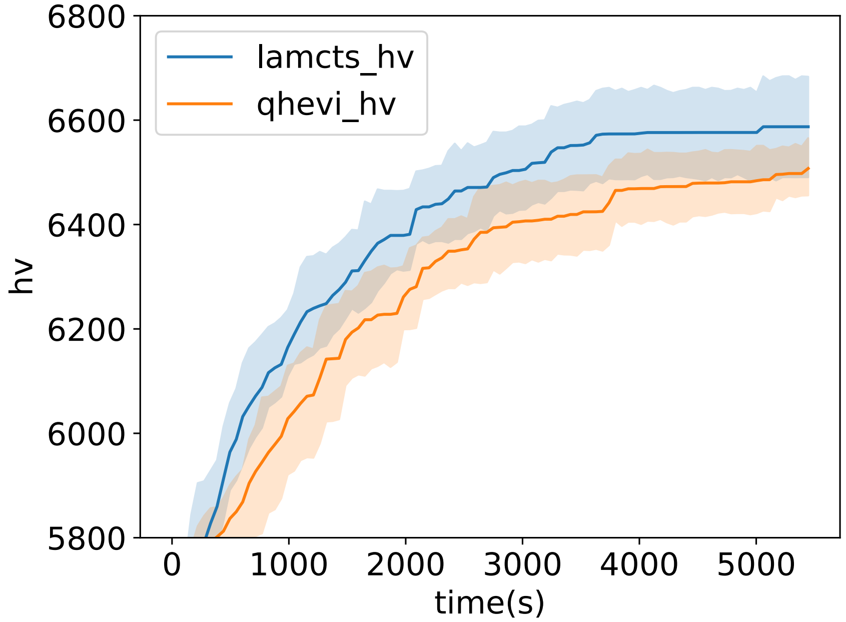
HV: Hyper Volume of the Pareto Frontier



Branin-Currin problem
(2 objective)



Vehicle Safety
(3 objective)



Waveguide

facebook Artificial Intelligence

qEHVI: https://arxiv.org/pdf/2006.05078.pdf

Code is public now!

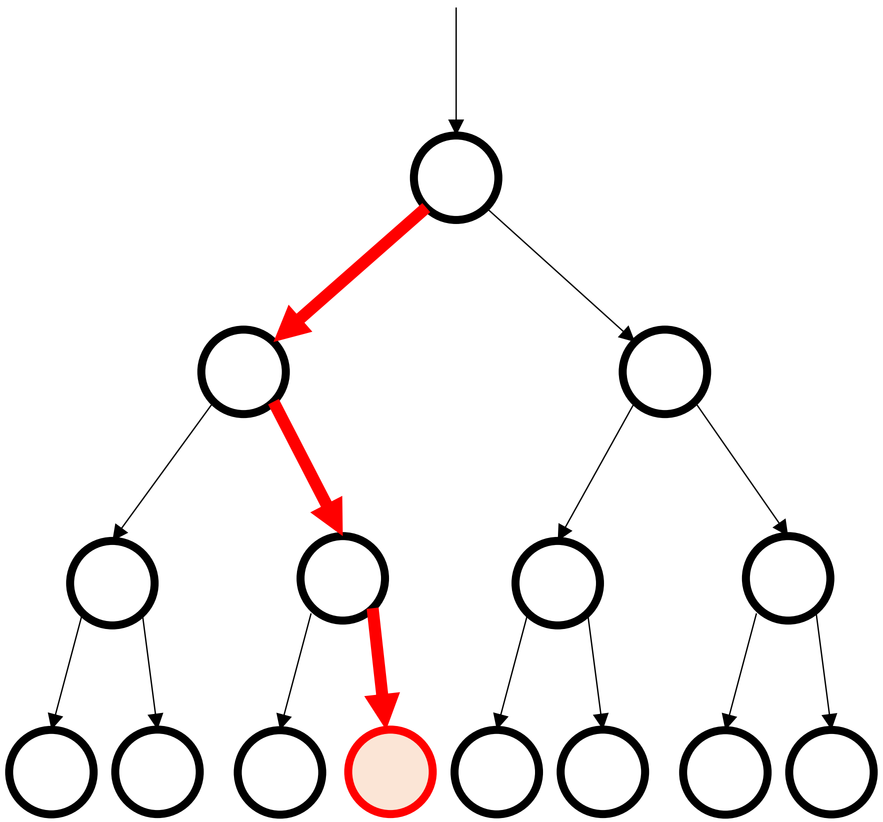
LA-MCTS

https://github.com/facebookresearch/LaMCTS

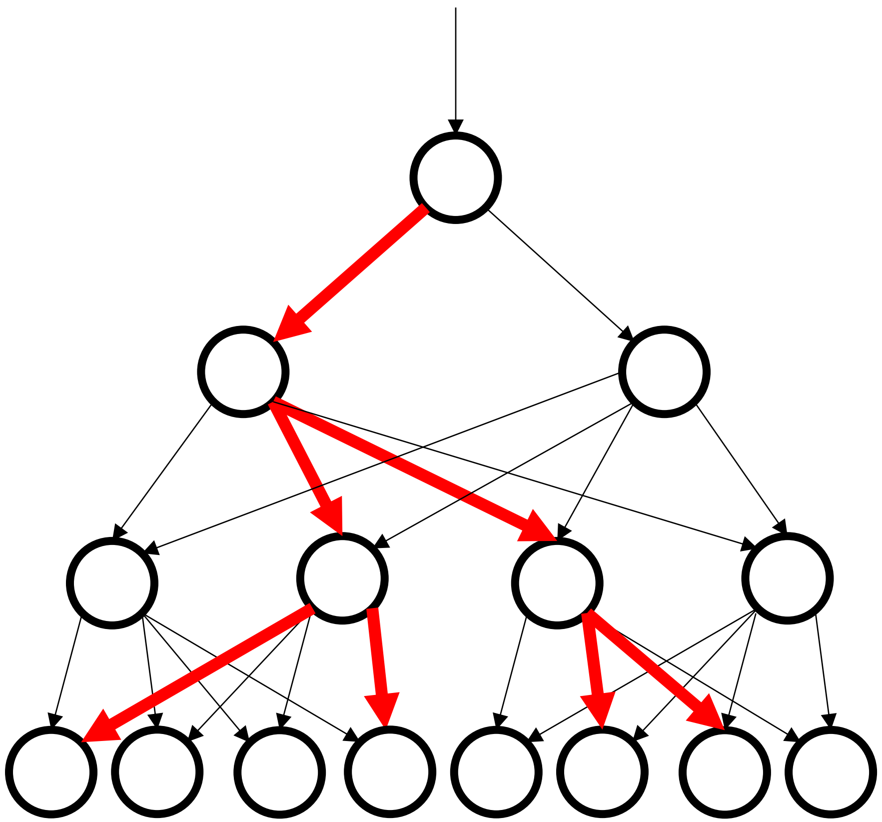
Both 3rd and 8th teams in NeurIPS 2020 Black-box optimization competition use our method!

Representation for Easier Search

Search in Imperfect Information Games



Perfect Information Games (Tree)



Imperfect Information Games (DAG)

Imperfect Information Game 101

Deal one *private* card (0/1) to Player 1
Deal one *private* card (0/1) to Player 2

4 possibilities:

00 01 10 11

Complete information state
or "history" h

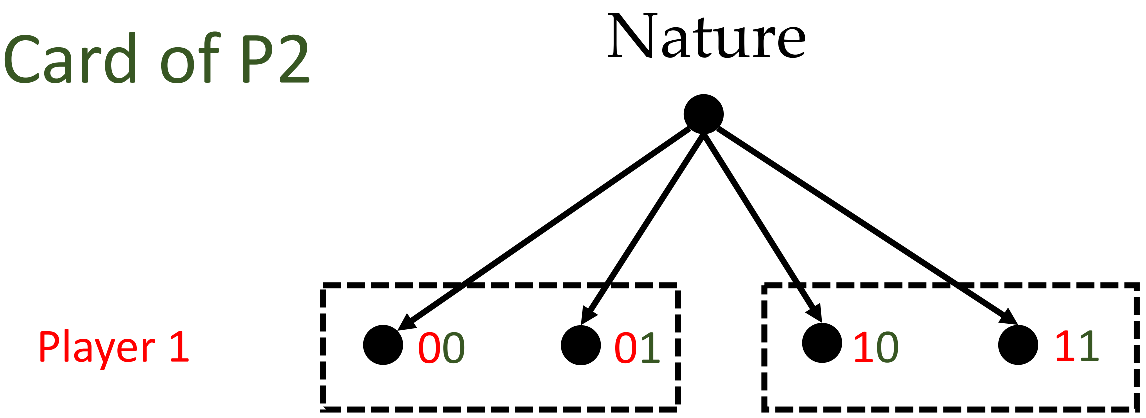
01

Card of P1 Card of P2

Imperfect Information Game 101

01

Card of P1 Card of P2



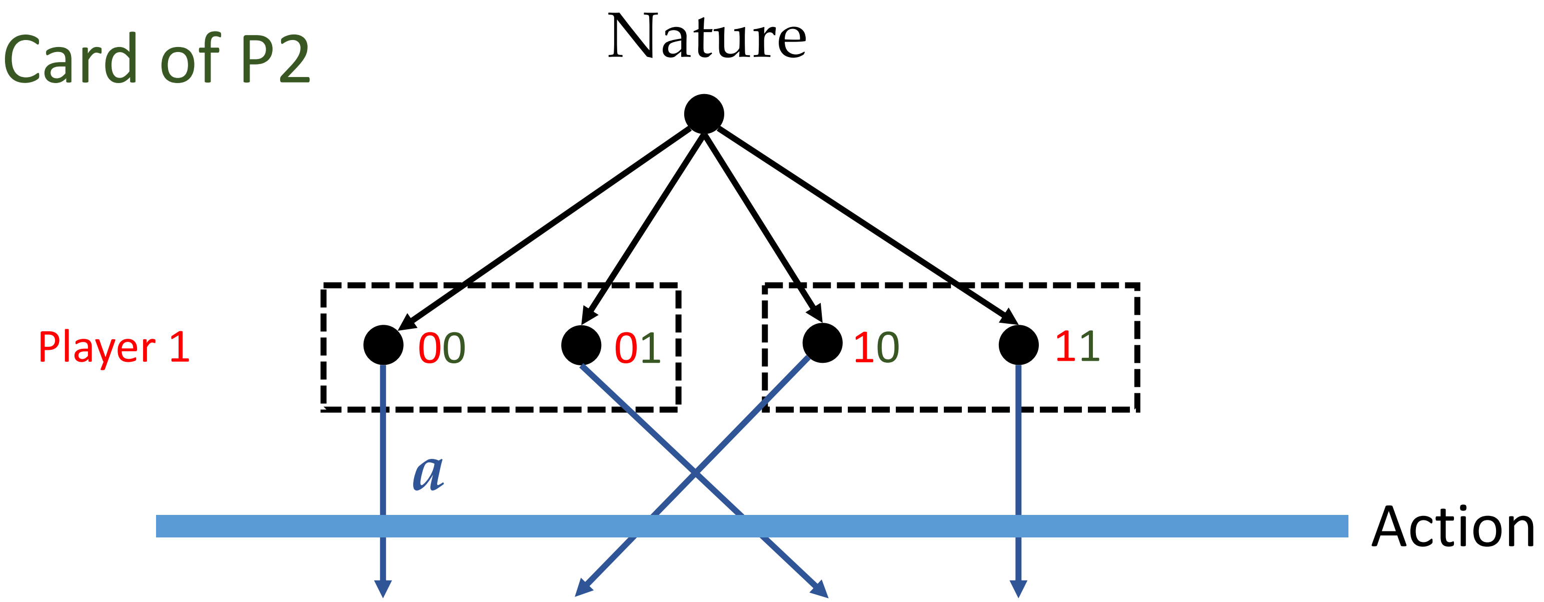
Information Set I
(or "Infoset")

Complete states within the same infosets have the same policy.
(since Player 1 doesn't know the private card of Player 2)

Imperfect Information Game 101

01

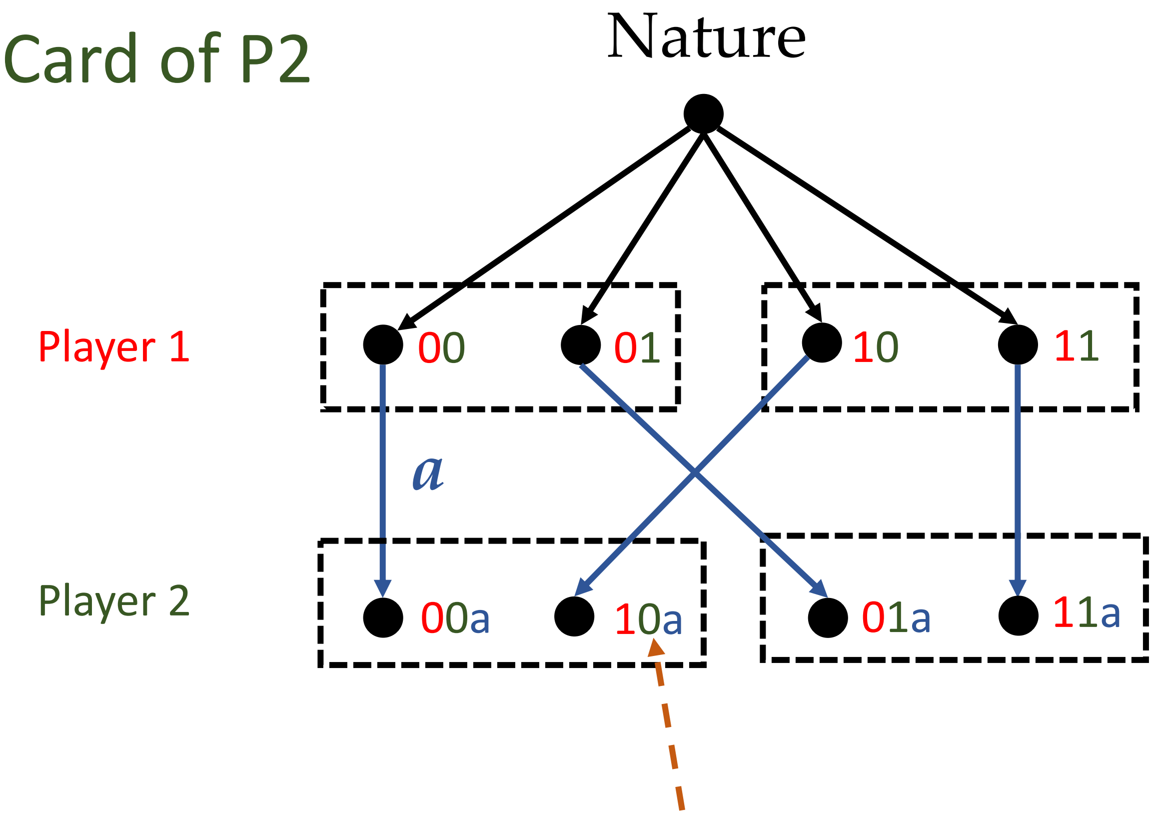
Card of P1 Card of P2



Imperfect Information Game 101

01

Card of P1 Card of P2

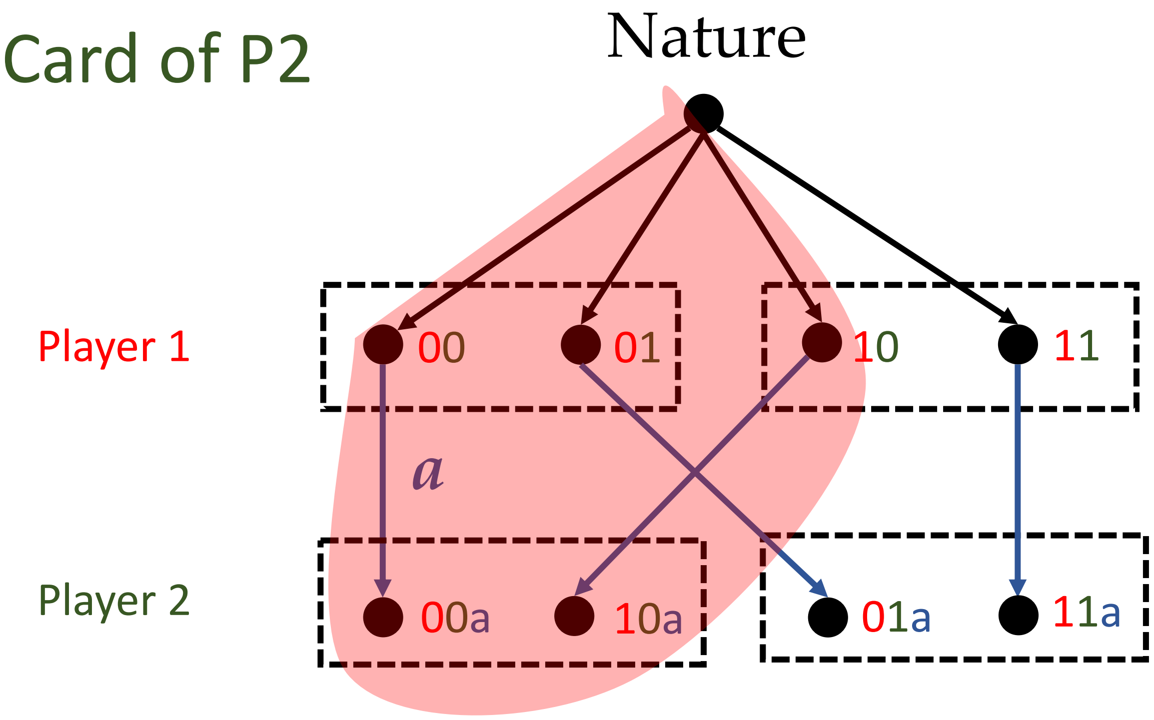


The action is public, so it is in the history of game

Imperfect Information Game 101

01

Card of P1 Card of P2

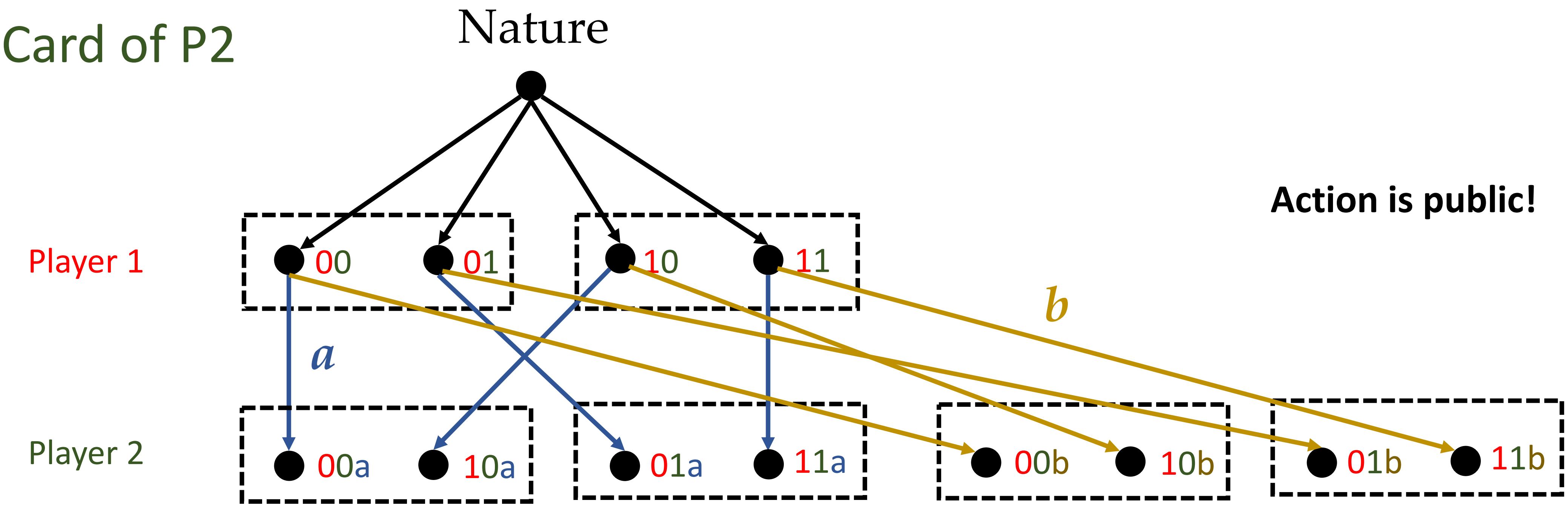


The DAG circle is here.

Imperfect Information Game 101

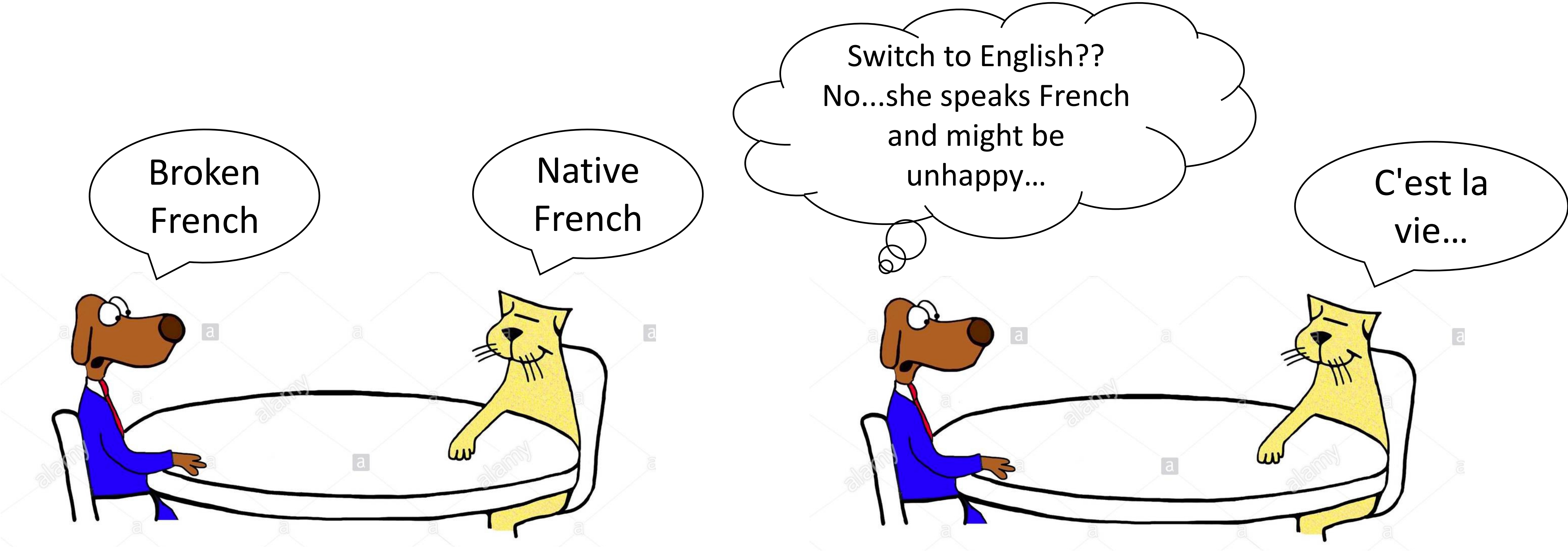
01

Card of P1 Card of P2



Action is public!

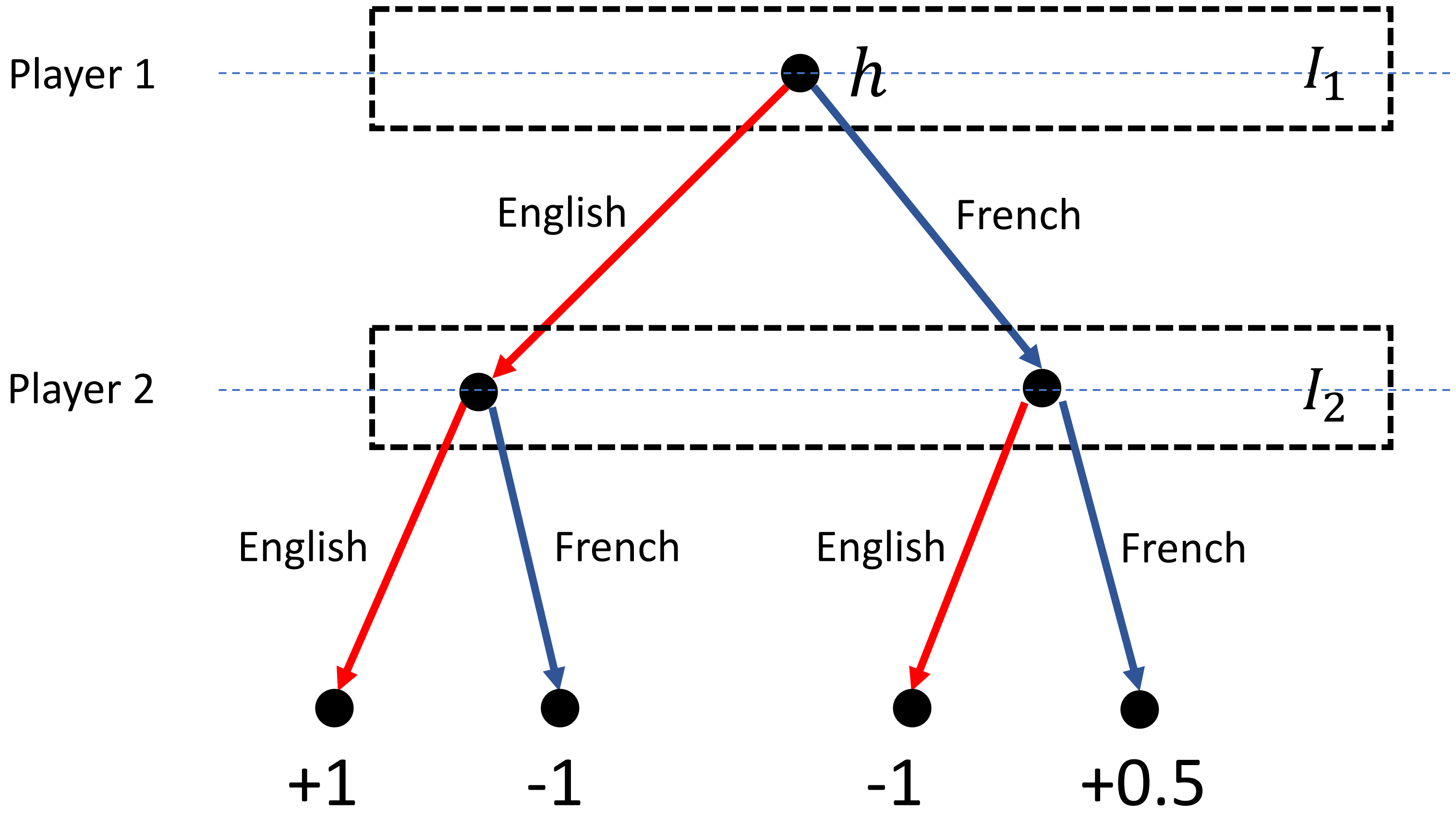
Optimizing one node at a time → Local optimum!



A **unilateral** change of policy doesn’t improve co-operative communication

(many single-agent DRL approach improves by unilateral changes of agent policy)

Optimizing one node at a time → Local optimum!



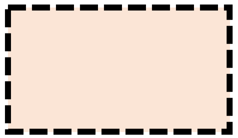
Player 2 makes the decision without knowing player 1's action.

(French, French):
local Nash Equilibrium +0.5

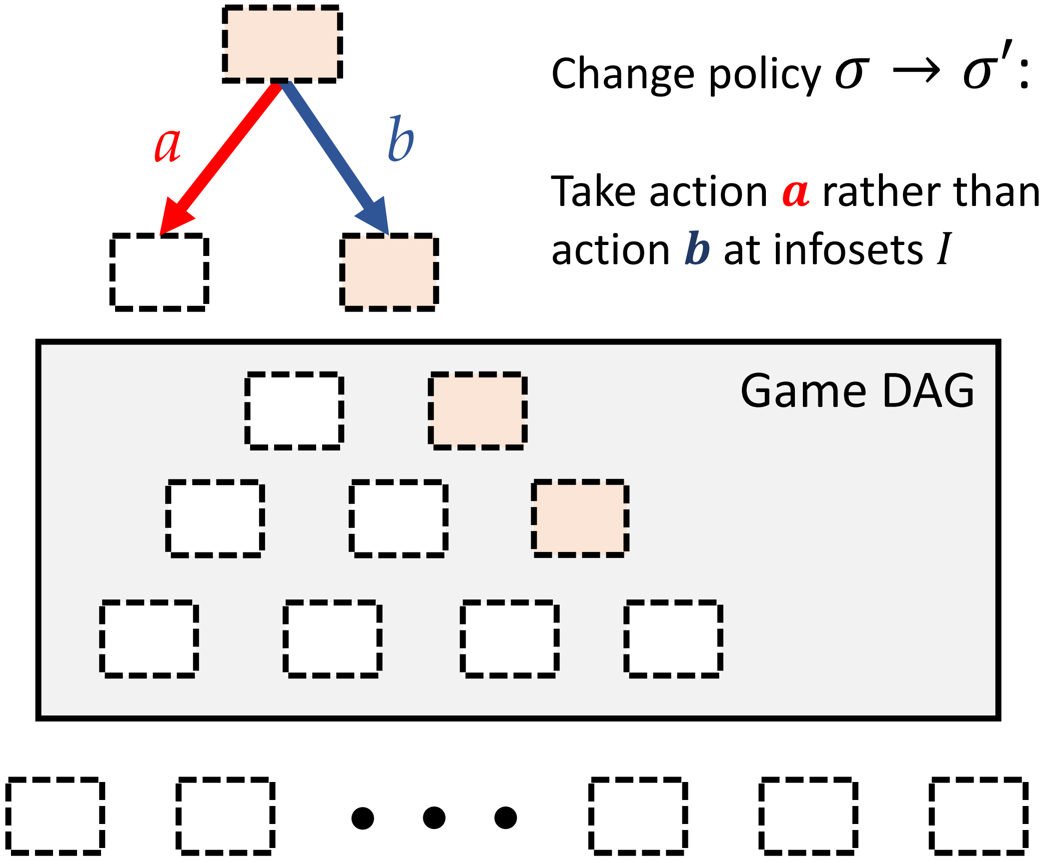
(English, English):
global Nash Equilibrium +1.0

A ***joint optimization*** of policy $\sigma(I_1)$ and $\sigma(I_2)$ *yields optimal solution*

Naïve Formulation



active infosets
$\sigma \to \sigma'$



Change policy $\sigma \to \sigma'$:

Take action $\boldsymbol{a}$ rather than action $\boldsymbol{b}$ at infosets I

$h \in Z$ terminal states when the reward is revealed

Reachability **Value at the terminal**

$$\bar{v}^{\sigma} = \sum_{h \in \boxed{Z}} \boxed{\pi^{\sigma}}(h)\boxed{v}(h)$$

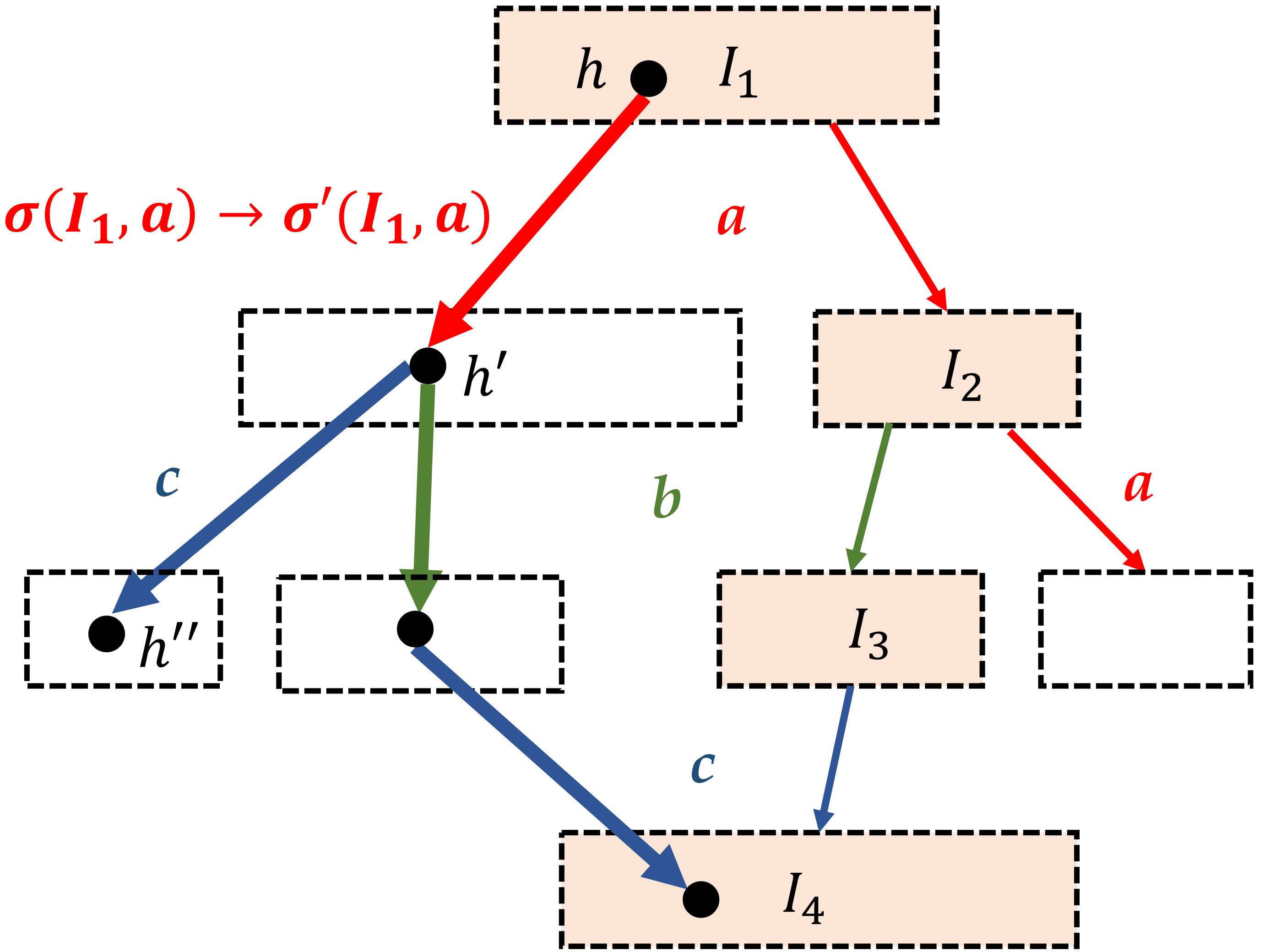
Game value **Terminal states**

Idea:

1. Pick a **subtree** and do *local* improvement
2. Multiple infosets need to be picked for *joint policy search*

But, things are complicated!

Dependency between policies



A change of $\sigma(I_1, a)$ affects **all** the reachability of down-stream states and/or infosets, no matter they are *active* or not.

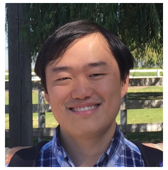
A trajectory could re-enter into another active set and leave and re-enter again.

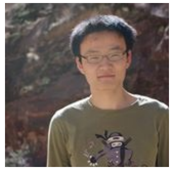
The value of an inactive infoset I_3 will change since the reachability to I_3 changes.

An infoset might contain both affected states and unaffected states.

Do we need to consider all infoSets?

Policy-change Density


Yuandong Tian


Qucheng Gong


Tina Jiang

[*Y. Tian et al*, **Joint Policy Search for Multi-agent Collaboration with Imperfect Information**, *NeurIPS 2020*]

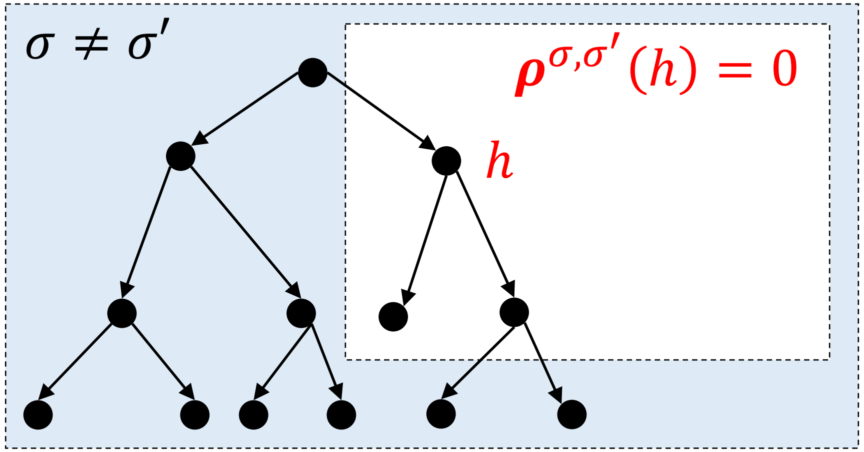
$$\rho^{\sigma,\sigma'}(h) = \pi^{\sigma'}(h)\left[\sum_{a\in A(I)} \sigma'(I,a)v^{\sigma}(ha) - v^{\sigma}(h)\right]$$

<u>Two key properties:</u>

(a) Its summation yields overall value changes

$$\bar{v}^{\sigma'} - \bar{v}^{\sigma} = \sum_{h\notin Z} \rho^{\sigma,\sigma'}(h)$$

(b) For regions whose policy doesn't change, it vanishes even if policy changes at downstream/upstream states.



Value Changes w.r.t Localized Policy Change

<u>Main Theorem</u> (New representation of value change)

$$\underbrace{\bar{v}^{\sigma'} - \bar{v}^{\sigma}}_{\text{Overall value changes}} = \sum_{I \in \mathcal{I}} \sum_{h \in I} \rho^{\sigma,\sigma'}(h)$$

All active Infosets where $\sigma' \neq \sigma$

Only depend on σ' via reachability $\pi^{\sigma'}(h)$

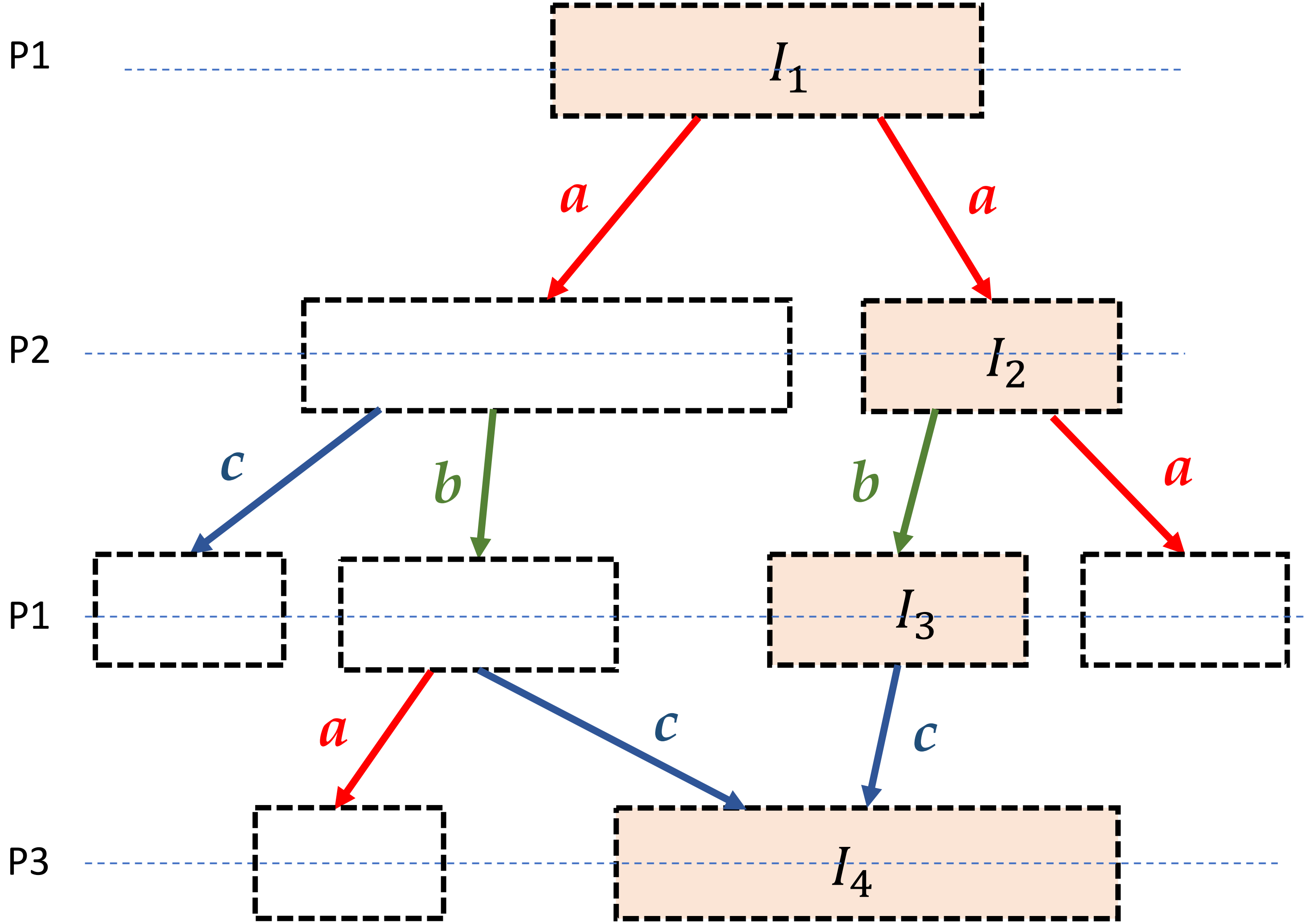
Inactive Infosets doesn't matter!!

JPS (Joint Policy Search)

1. Initial infosets $I_{\text{cand}} = \{I_1\}$
2. Pick $I \in I_{\text{cand}}$
3. Pick an action a
4. Set $\sigma'(I, b) = \delta(a = b)$
5. Compute $\rho^{\sigma,\sigma'}$
6. Set $I_{\text{cand}} = \text{Succ}(I, a)$

Repeat until maximal depth D is reached.

Backtrace
(depth-first search)



Performance

	Comm (Def. 1)				Mini-Hanabi	Simple Bidding (Def. 2)			2SuitBridge (Def. 3)		
	$L=3$	$L=5$	$L=6$	$L=7$	[15]	$N=4$	$N=8$	$N=16$	$N=3$	$N=4$	$N=5$
CFR1k [43]	0.89*	0.85	0.85	0.85	9.11*	2.18*	4.96*	10.47	1.01*	1.62*	2.60
CFR1k+JPS	**1.00***	**1.00***	**1.00***	**1.00***	**9.50***	2.20*	**5.00***	**10.56***	**1.07***	**1.71***	**2.74***
A2C [26]	0.60*	0.57	0.51	0.02	8.20*	2.19	4.79	9.97	0.66	1.03	1.71
BAD [15]	**1.00***	0.88	0.50	0.29	9.47*	**2.23***	4.99*	9.81	0.53	0.98	1.31
Best Known	1.00	1.00	1.00	1.00	10	2.25	5.06	10.75	1.13	1.84	2.89
#States	633	34785	270273	2129793	53	241	1985	16129	4081	25576	147421
#Infosets	129	2049	8193	32769	45	61	249	1009	1021	5116	24571

JPS can improve existing policies, and help it jump out of local optima

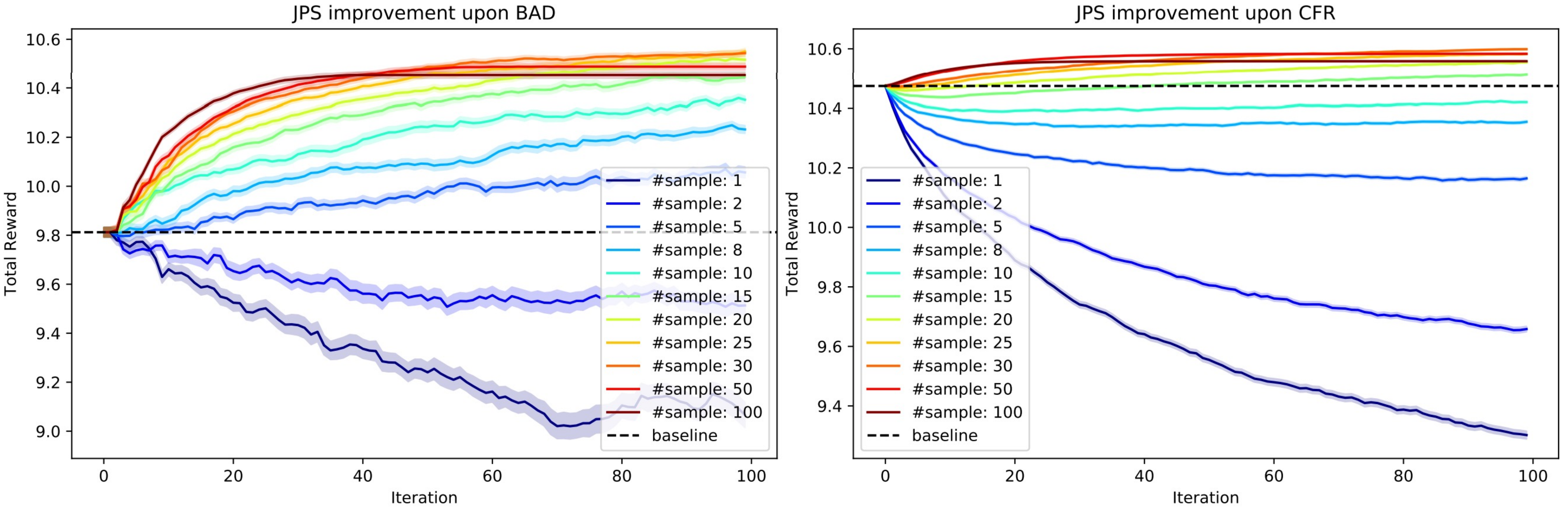
Sample-based JPS

	Initialization	All states	#Samples per infoset							
			1	2	5	8	15	20	25	30
Mini-Hanabi [15]	CFR1k [43]	9.50	**10.00**	9.99	9.95	9.75	9.51	9.51	9.51	9.51
SimpleBidding ($N = 16$)	CFR1k [43]	10.56	10.47	10.47	10.49	10.52	10.58	10.60	10.61	**10.61**
SimpleBidding ($N = 16$)	BAD [15]	10.47	9.91	9.95	10.22	10.34	10.50	10.55	**10.57**	10.55
2-suited Bridge ($N = 3$)	BAD [15]	1.12	0.89	1.12	**1.13**	1.13	1.12	1.12	1.12	1.12
2-suited Bridge ($N = 4$)	BAD [15]	1.71	1.23	1.63	1.71	**1.71**	1.68	1.67	1.68	1.69
2-suited Bridge ($N = 5$)	BAD [15]	2.77	2.12	2.51	2.74	**2.79**	2.79	2.76	2.77	2.78

$$\bar{v}^{\sigma'} - \bar{v}^{\sigma} = \sum_{I \in \mathcal{I}} \sum_{h \in I} \rho^{\sigma,\sigma'}(h)$$

Sample h in each I

Sample-based JPS



Simple Biddings (*N=16*)

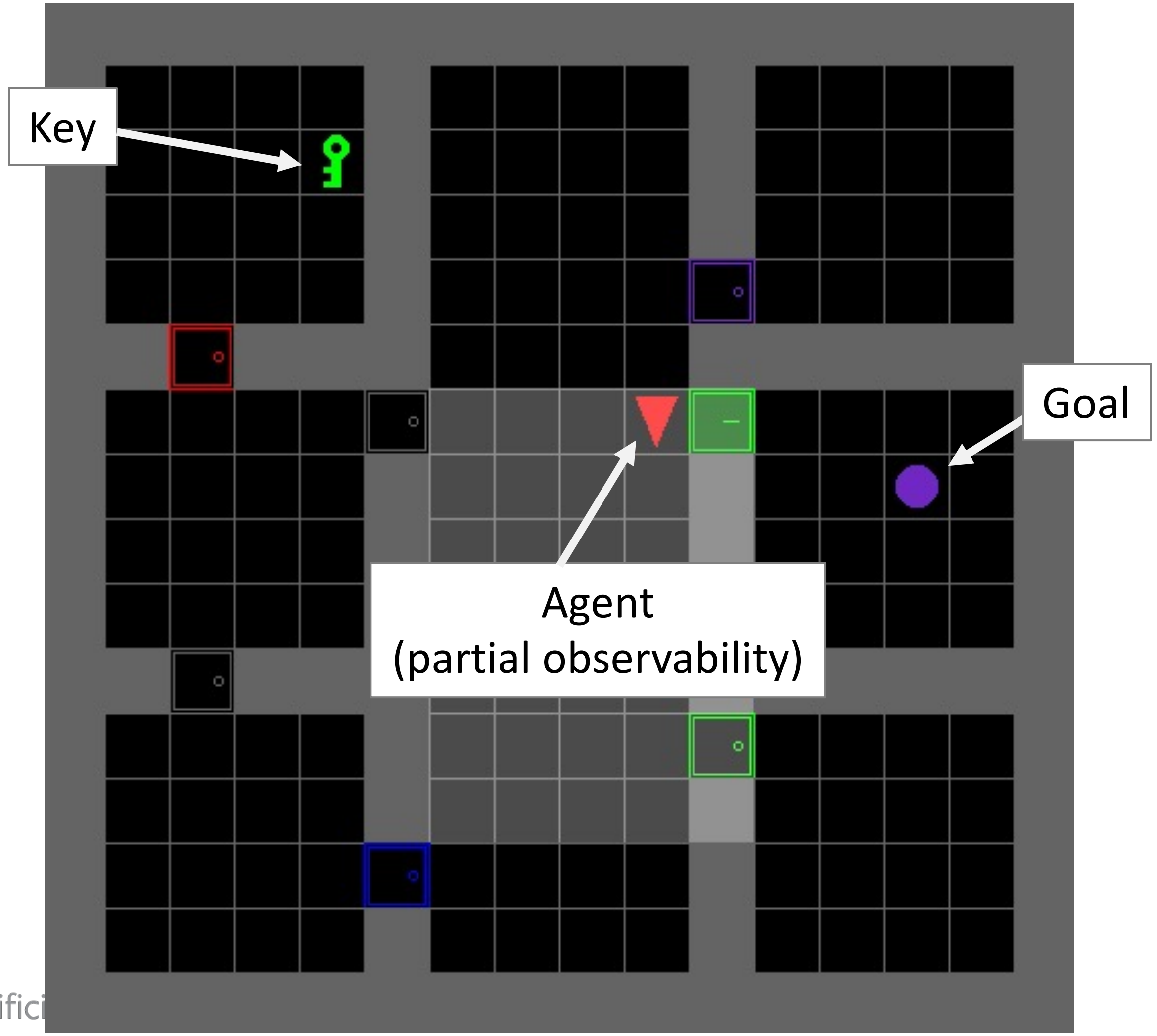
Contract Bridge Bidding

Methods	**Vs. WBridge5 (1000 games)** (IMPs/board)
Previous SoTA (Rong et al, 2019)	+ 0.25 (on 64 games)
Our A2C baseline	+ 0.29 ± 0.22
1% JPS (2 days)	+ 0.44 ± 0.20
5% JPS (2 days)	+ 0.37 ± 0.19
1% JPS (14 days)	**+ 0.63 ± 0.20**

WBridge5: Champions of computer bridge tournament in 2005, 2007, 2008, 2016-2018

Representation for RL Exploration

Exploration in Environment with Sparse Reward

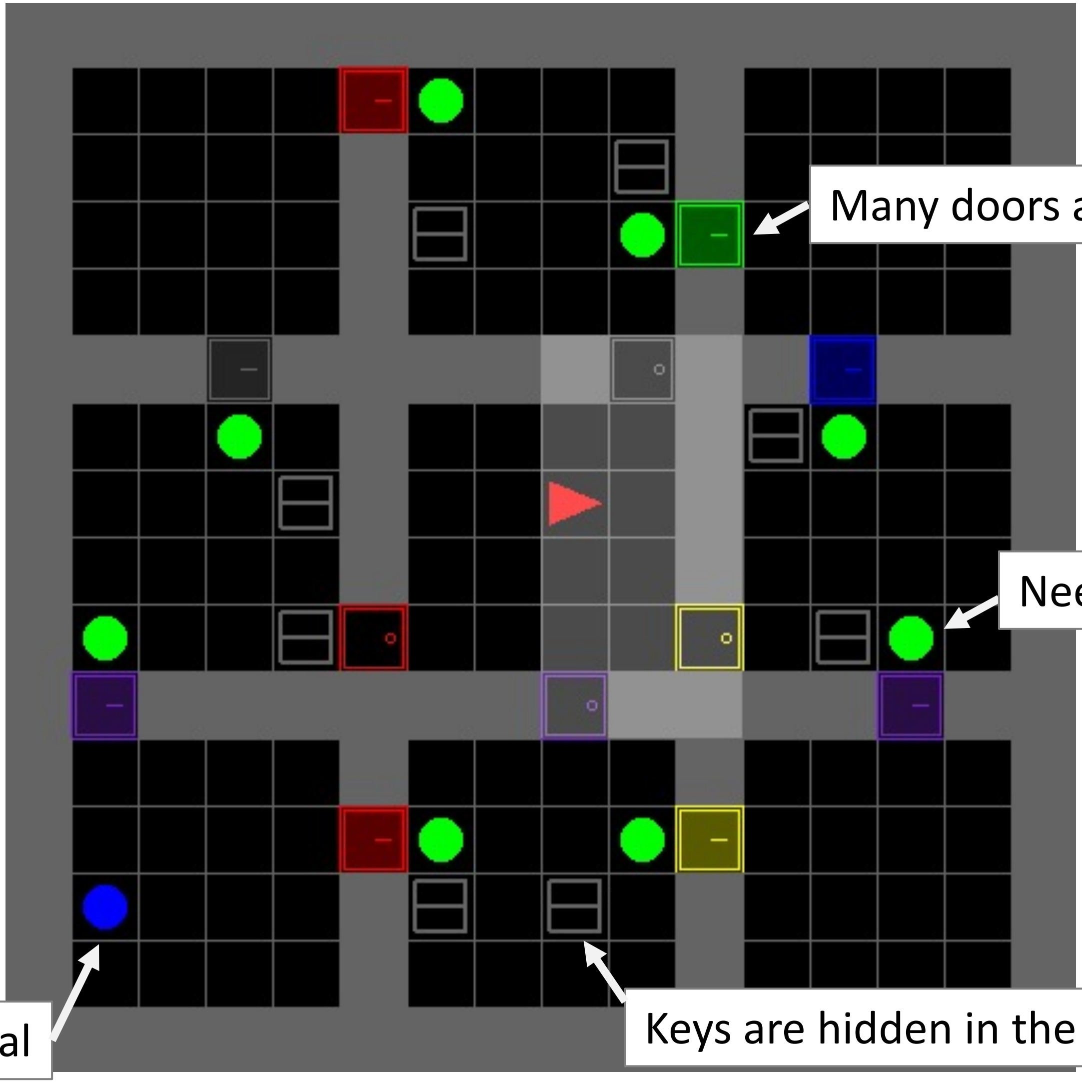


No external reward

when agent wonders around.
when agent picks the key
when agent opens all doors
when agent opens the locked door
…

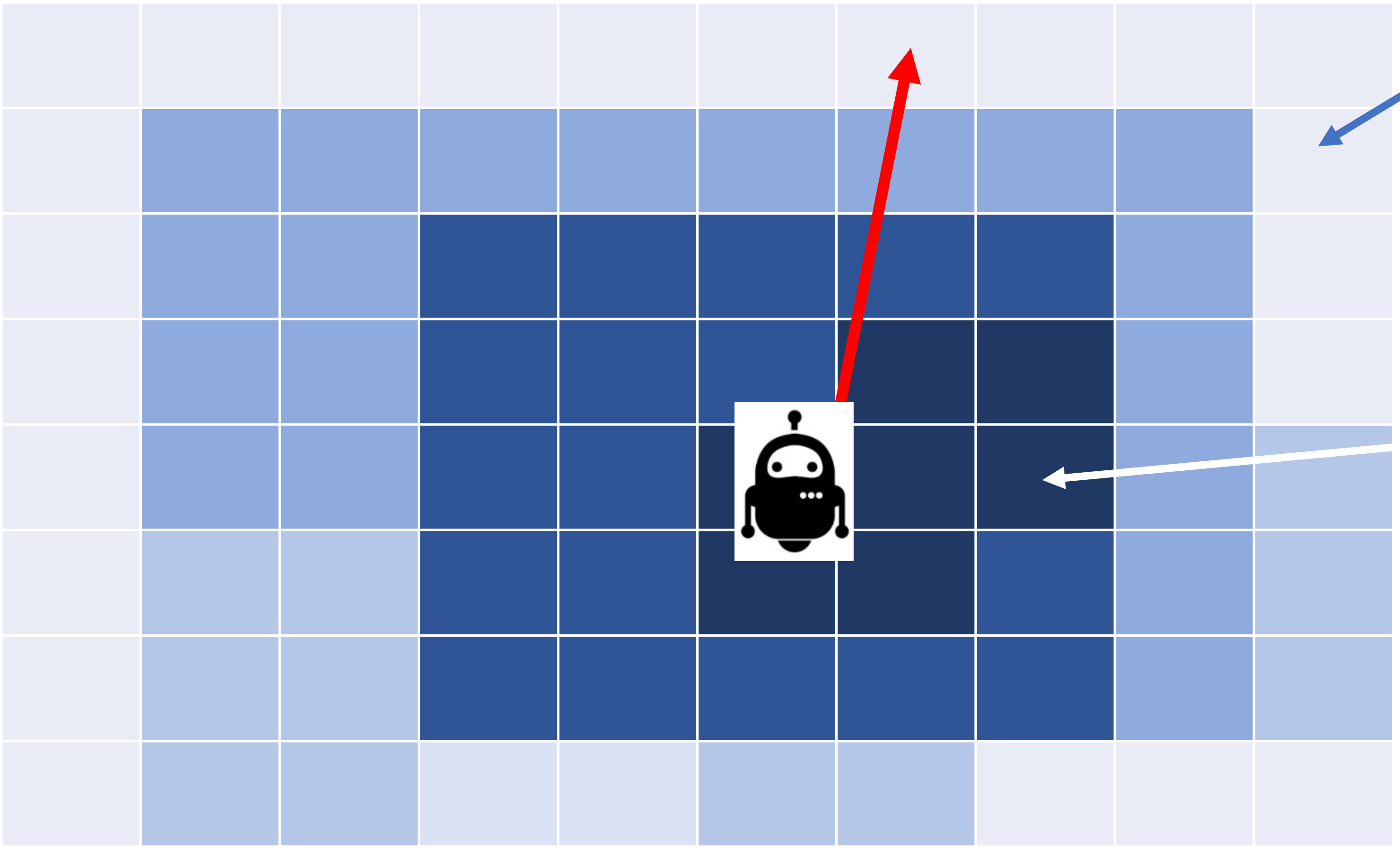
until the agent reaches the goal

And more complicated situations…



Count-based Exploration

Low visitation counts $N(s)$
High intrinsic rewards



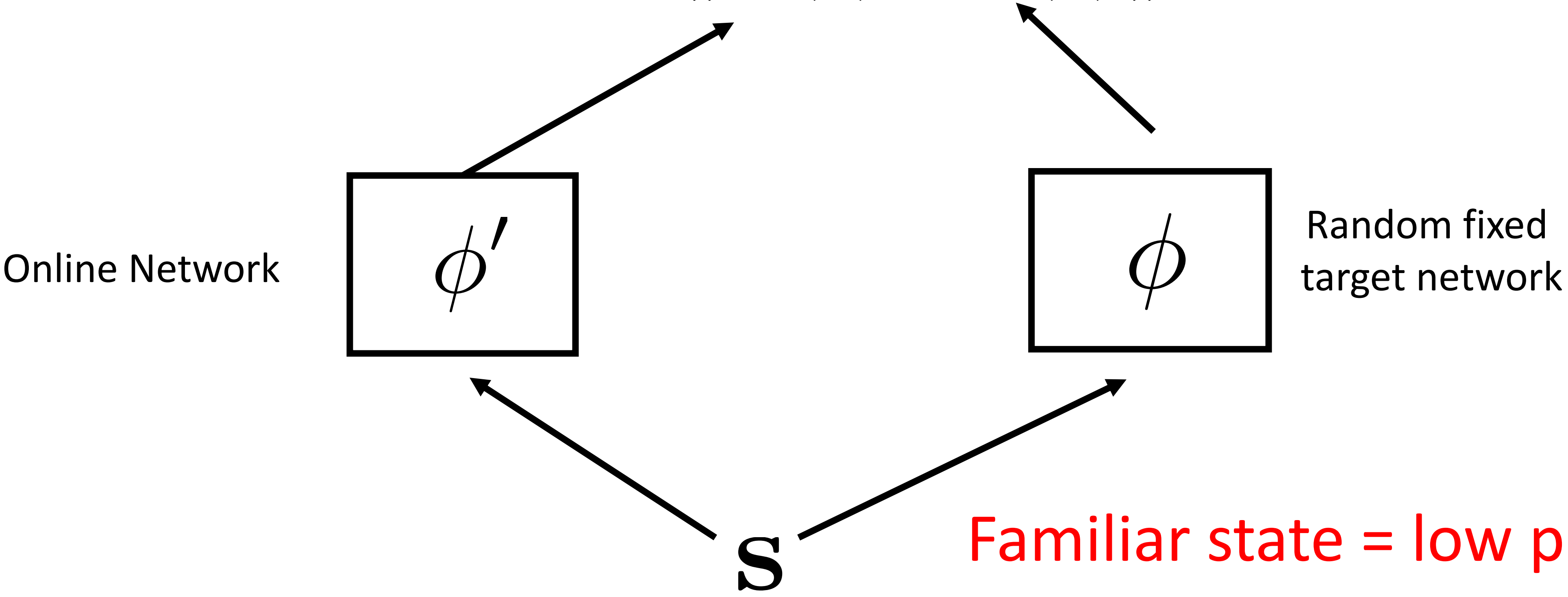
High visitation counts $N(s)$
Low Intrinsic reward

What if we have exponential #states?

[Bellemare, Marc, et al. "Unifying count-based exploration and intrinsic motivation." Advances in neural information processing systems. 2016]

Random Network Distillations (RND)

$$N(\mathbf{s}) \approx \frac{1}{\|\phi'(\mathbf{s}) - \phi(\mathbf{s})\|}$$

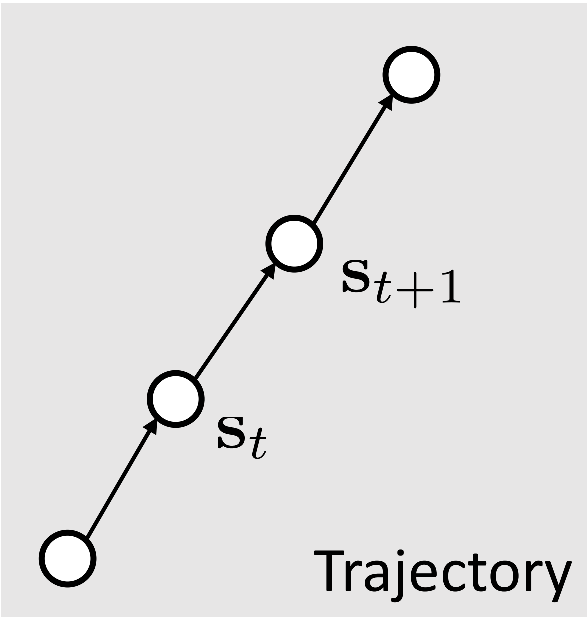


Familiar state = low prediction error

[Y. Burda et al, Exploration by Random Distillation Network, ICLR 2019]

BeBold

Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph Gonzalez, Yuandong Tian



BeBold = **Be**yond the **Bo**undary of Exp**l**ore**d** Regions

RND for t+1 RND for t

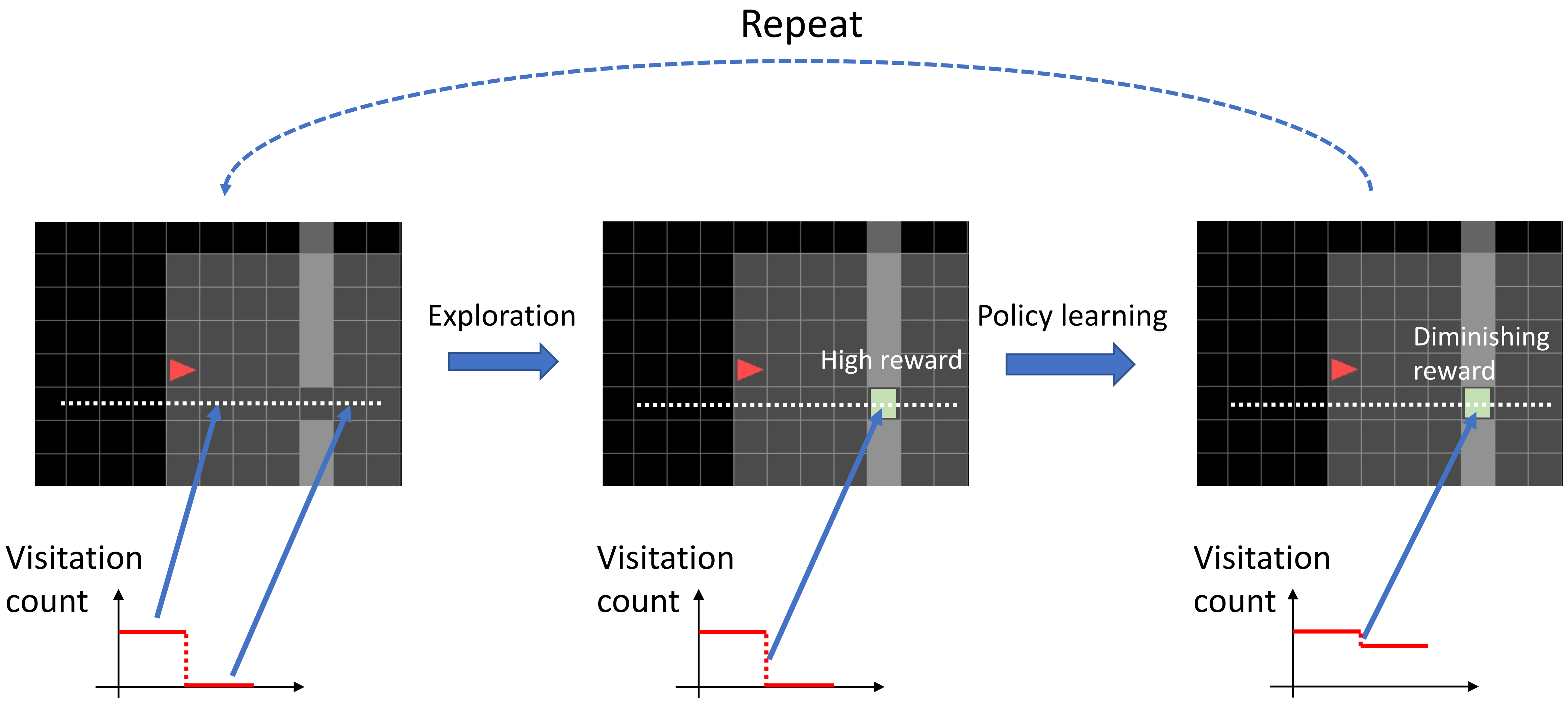
$$r^i(\mathbf{s}_t, \mathbf{a}_t) = \max(||\phi(\boldsymbol{o}_{t+1}) - \phi'(\boldsymbol{o}_{t+1})||_2 - ||\phi(\boldsymbol{o}_t) - \phi'(\boldsymbol{o}_t)||_2, 0) * \mathbb{1}\{N_e(\boldsymbol{o}_{t+1}) = 1\})$$

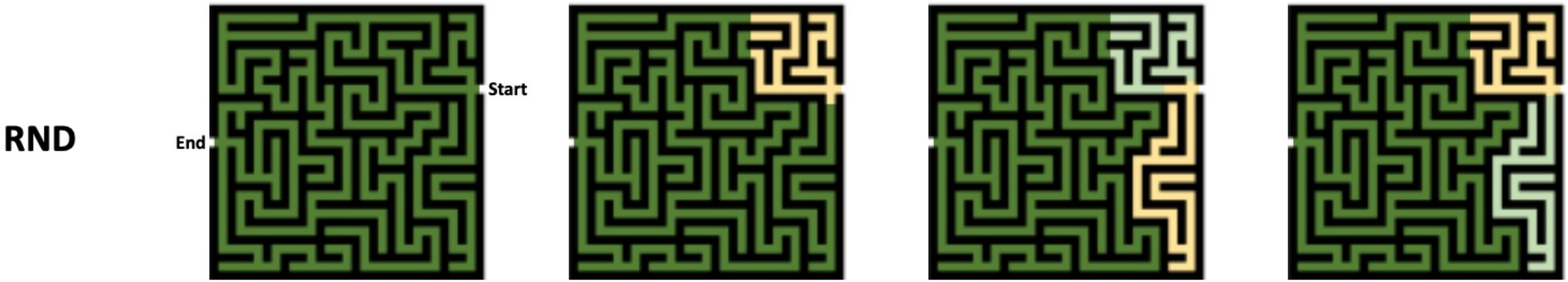
Observation (rather than full state)

Episodic visitation count (Hash Table)

[T. Zhang et al, BeBold: Exploration Beyond the Boundary of Explored Regions, arXiv 2020]

BeBold


Repeat
Exploration
Policy learning
High reward
Diminishing
reward
Visitation
count
Visitation
count
Visitation
count

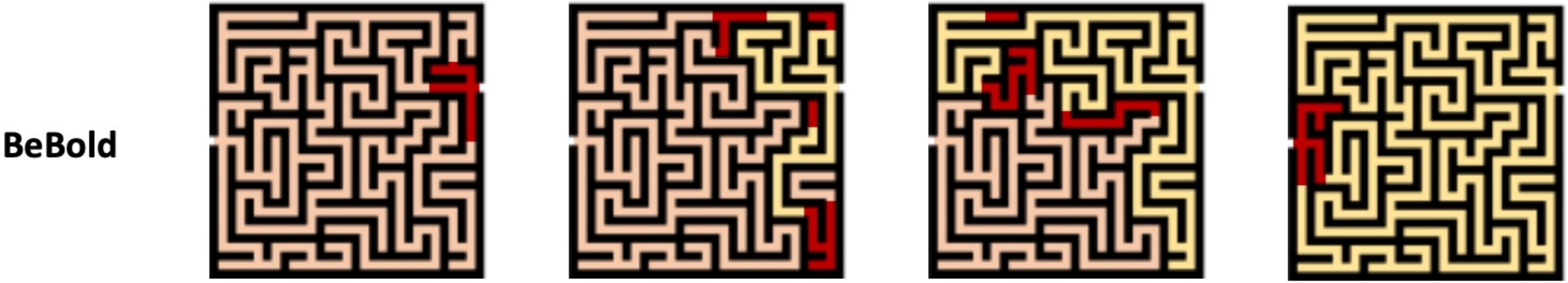


1. RND assigns high IR (dark green) throughout the environment

2. RND temporarily focuses on the upper right corner (yellow)

3. RND by chance starts exploring the bottom right corner heavily, resulting in the IR at top right higher than bottom right

4. RND re-explores the upper right and forgets the bottom right, gets trapped



1. BeBold assigns high IR (dark red) near the start and low IR for the rest (light red)

2. BeBold pushes every direction to the frontier of exploration uniformly (yellow)

3. BeBold continuously pushes the exploration frontier

4. BeBold reaches the end of exploration

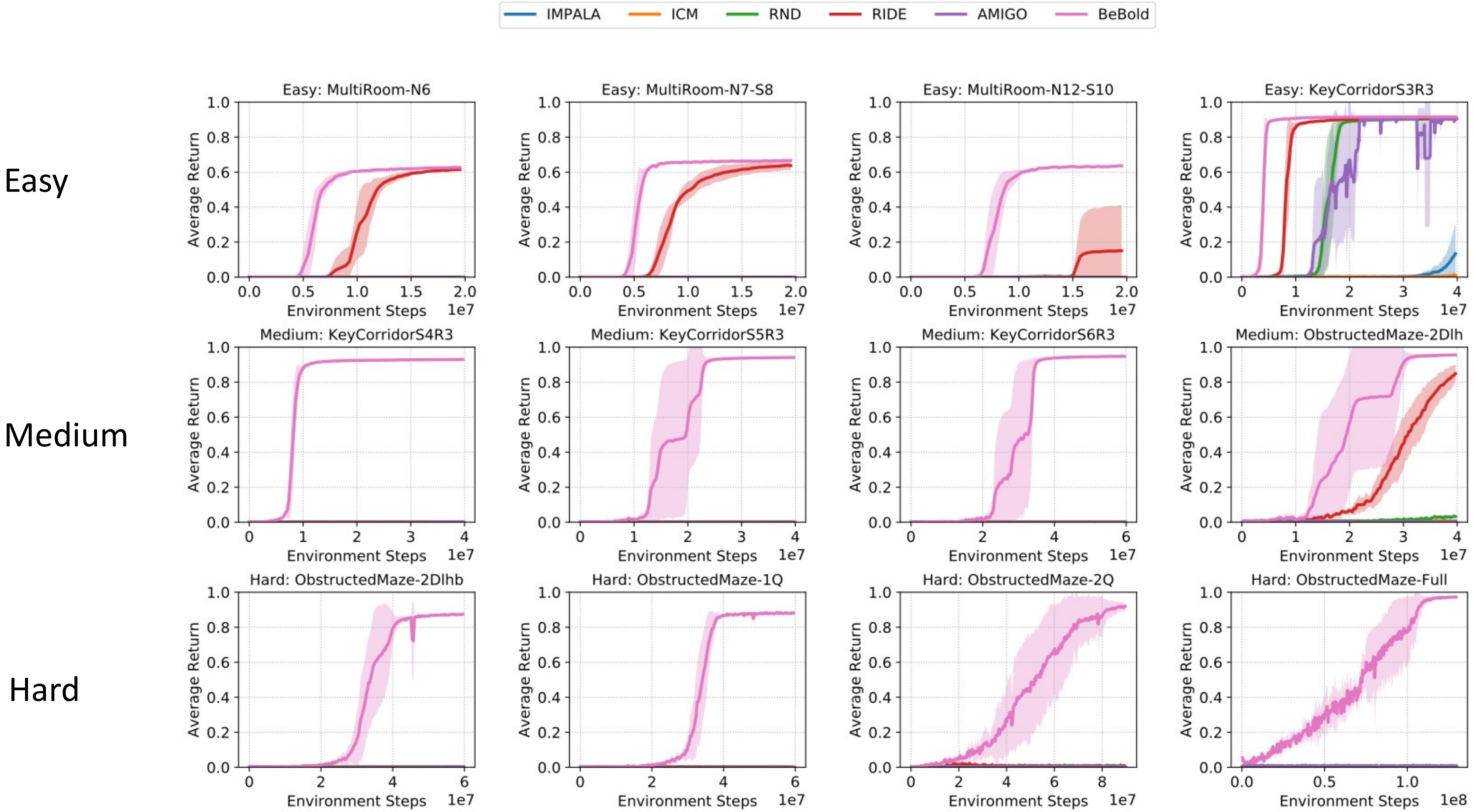
MiniGrid

	MRN6	MRN7S-8	MRN12-S10	KCS3R3	KCS4R3	KCS5R3	KCS6R3	OM2Dl-h	OM2Dl-hb	OM1Q	OM2Q	OMFULL
ICM				✓								
RND				✓				✓				
RIDE	✓	✓	✓	✓	✓			✓				
AMIGO				✓								
BeBold	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

✓ : Solved within 120M steps

*MR is short for MultiRoom, KC is for KeyCorridor, OM is for ObstructedMaze

[Chevalier-Boisvert, Maxime, Lucas Willems, and Suman Pal. "Minimalistic gridworld environment for openai gym." GitHub repository (2018)]

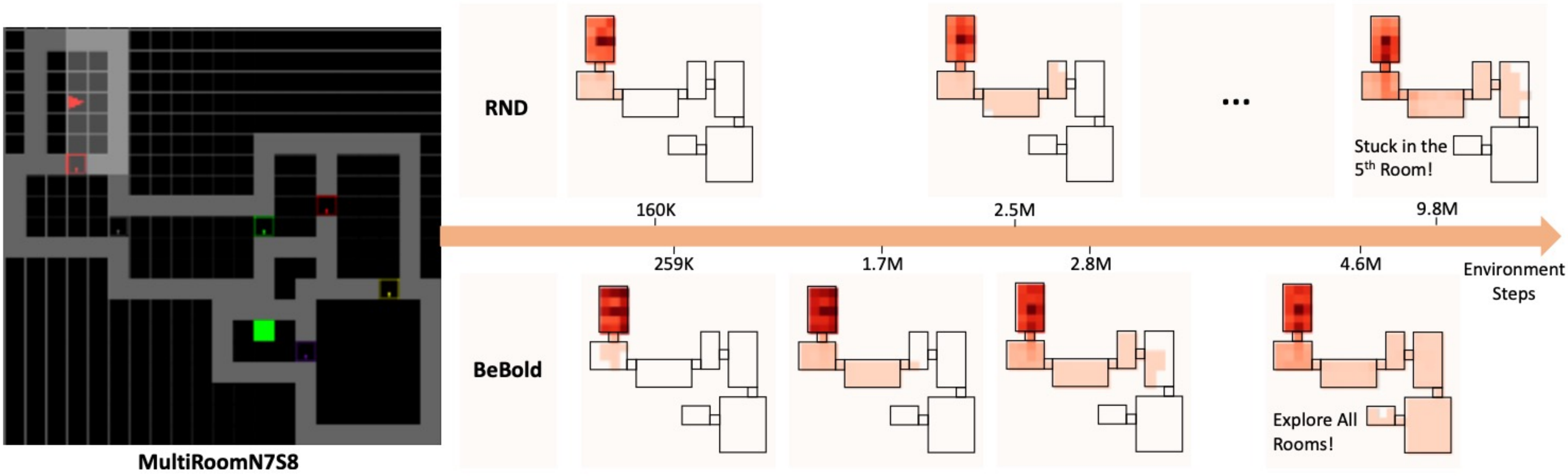


AMIGO: [C. Andres, et al. "Learning with AMIGo: Adversarially Motivated Intrinsic Goals." ICLR 2021]

RIDE: [R. Roberta, and Tim Rocktäschel. "RIDE: Rewarding Impact-Driven Exploration for Procedurally-Generated Environments.", ICLR 2020]

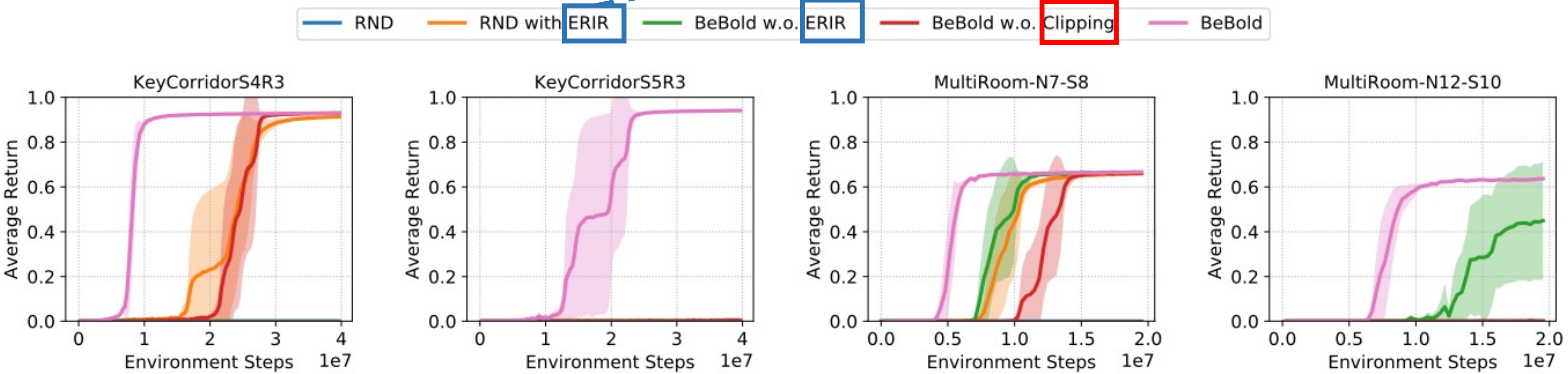
ICM: [P. Deepak, et al. "Curiosity-driven exploration by self-supervised prediction." CVPR Workshops. 2017.]

Pure Exploration in MiniGrid

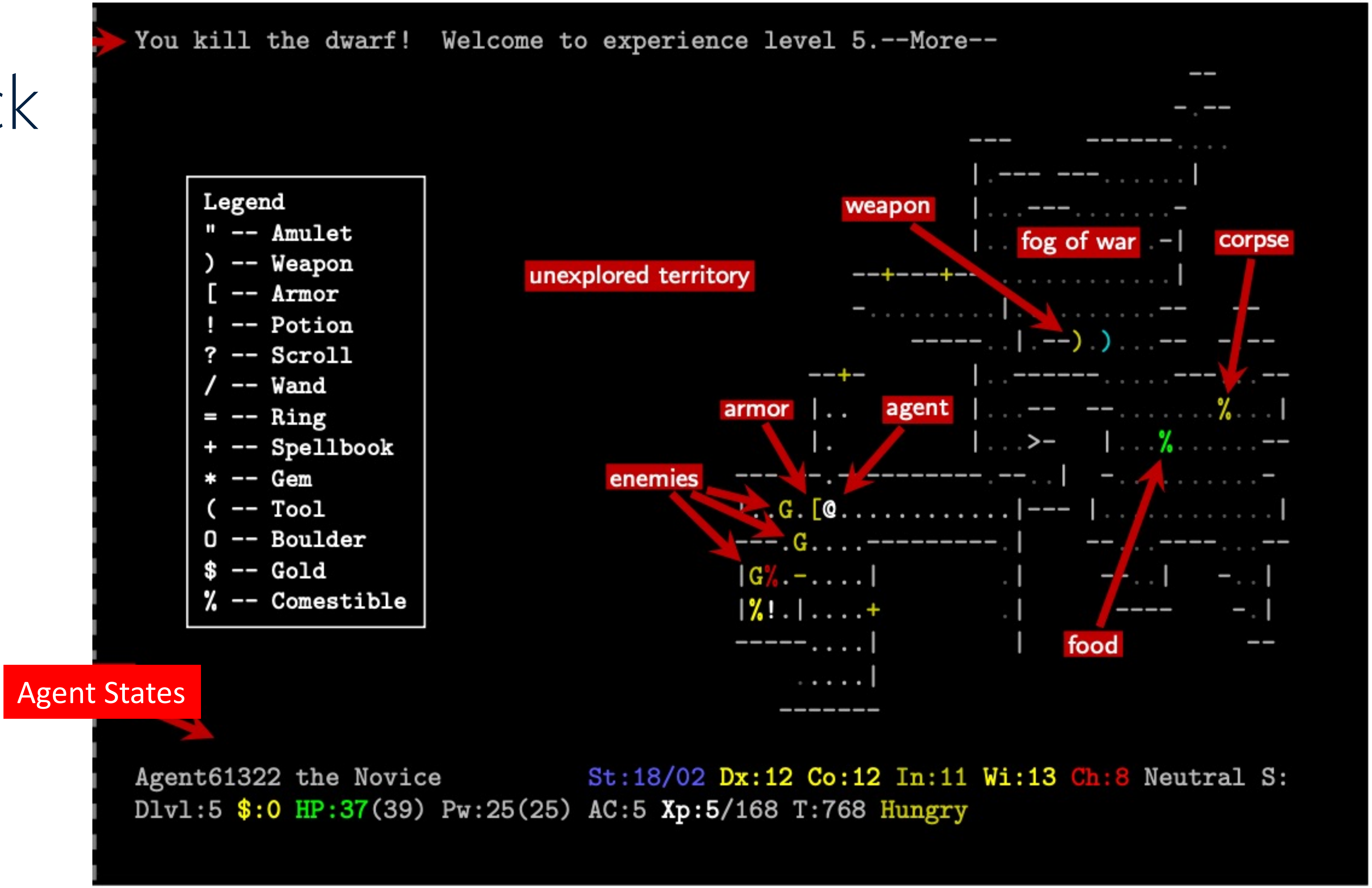


Ablation Study

$$r^i(\mathbf{s}_t, \mathbf{a}_t) = \boxed{\max}(||\phi(\boldsymbol{o}_{t+1}) - \phi'(\boldsymbol{o}_{t+1})||_2 - ||\phi(\boldsymbol{o}_t) - \phi'(\boldsymbol{o}_t)||_2, \boxed{0}) * \mathbb{1}\{\boxed{N_e(\boldsymbol{o}_{t+1}) = 1}\}$$

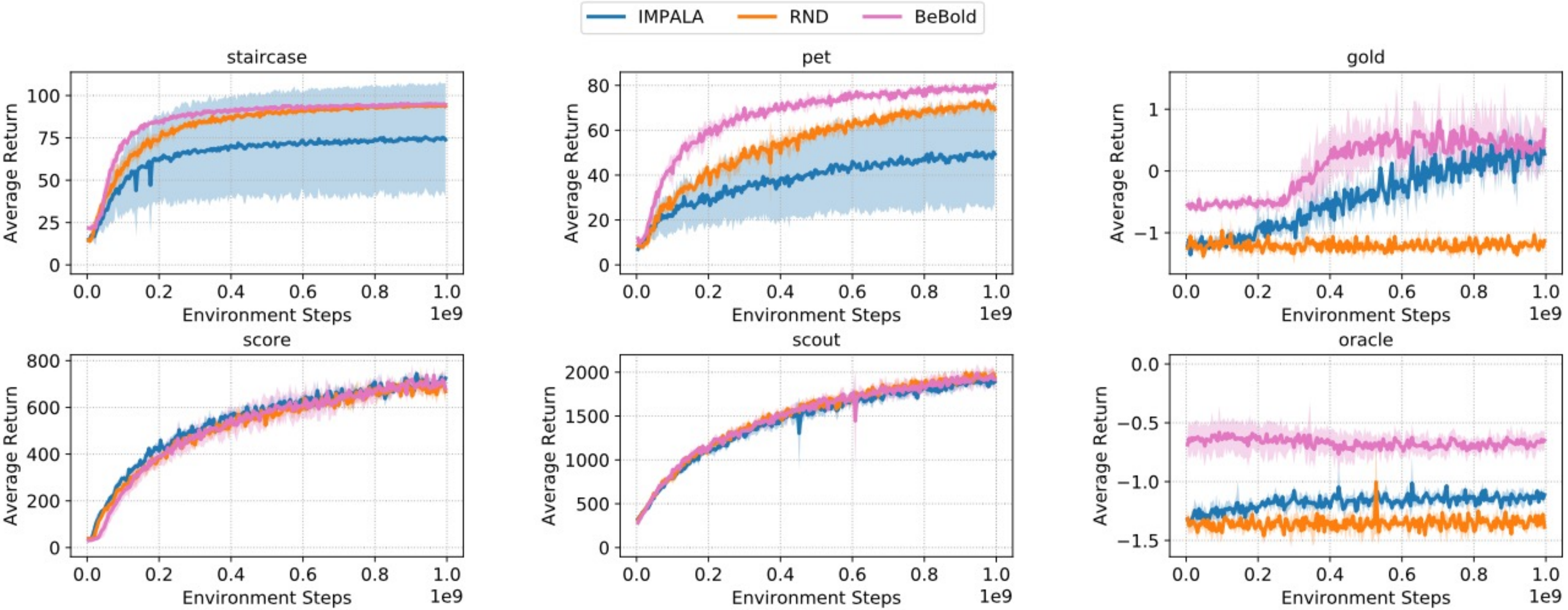


NetHack

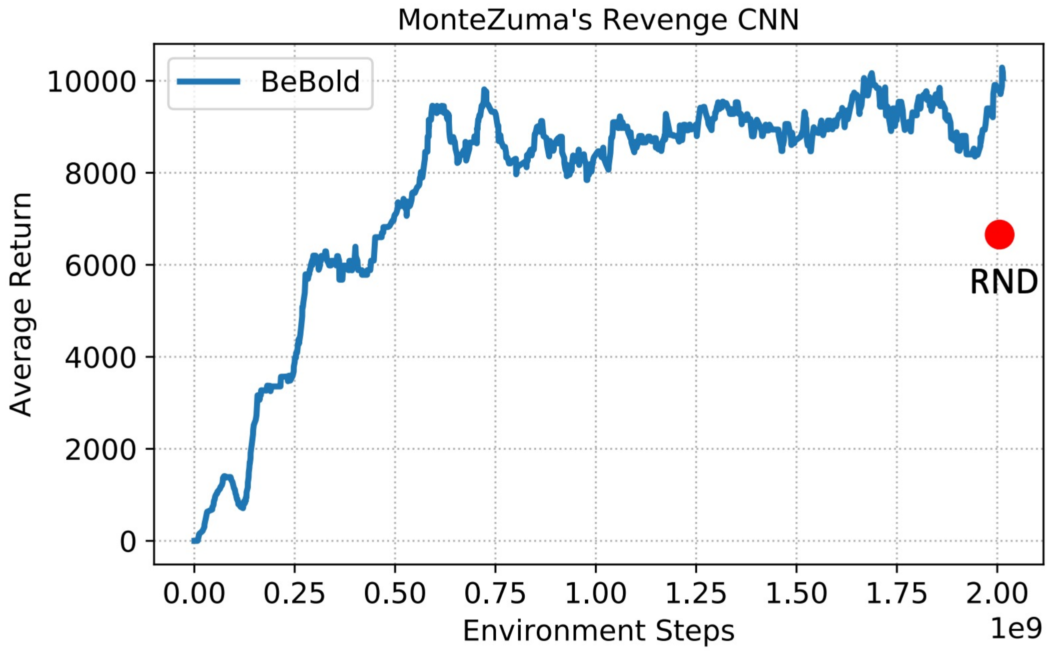


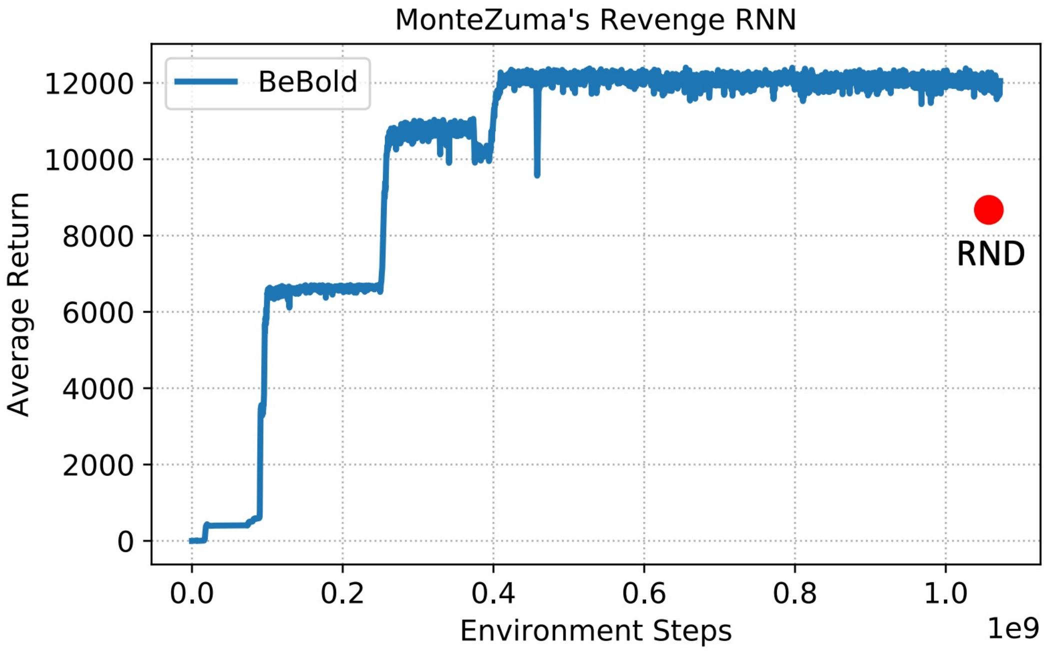
[Küttler, Heinrich, et al. "The NetHack Learning Environment." arXiv preprint arXiv:2006.13760 (2020)]

6 Tasks in NetHack

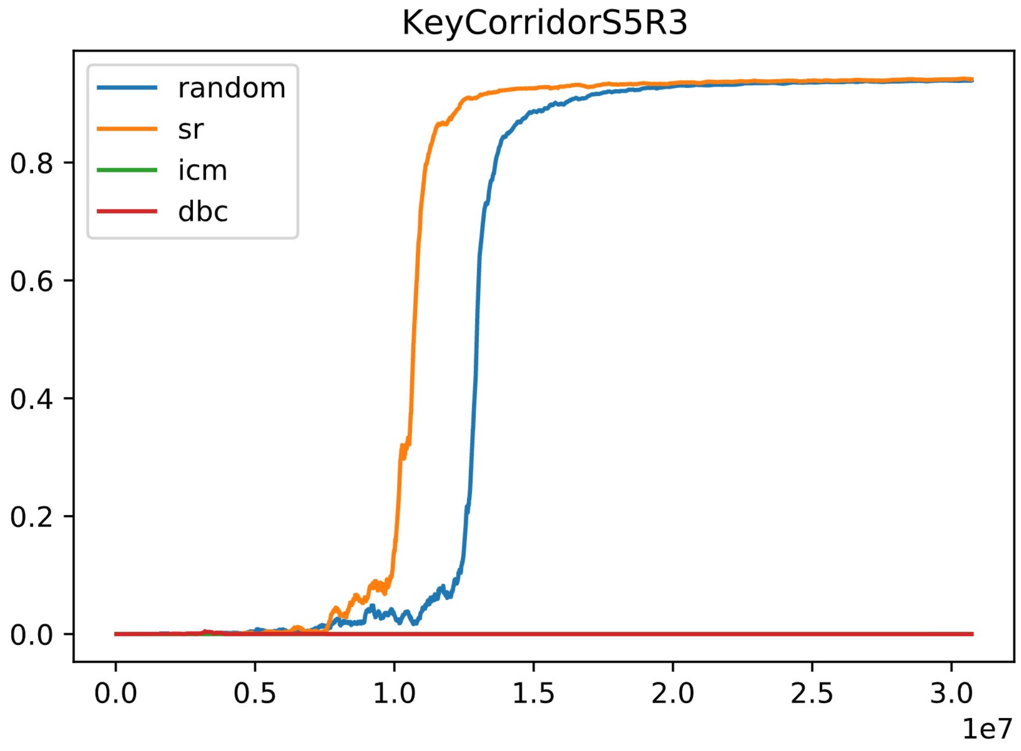


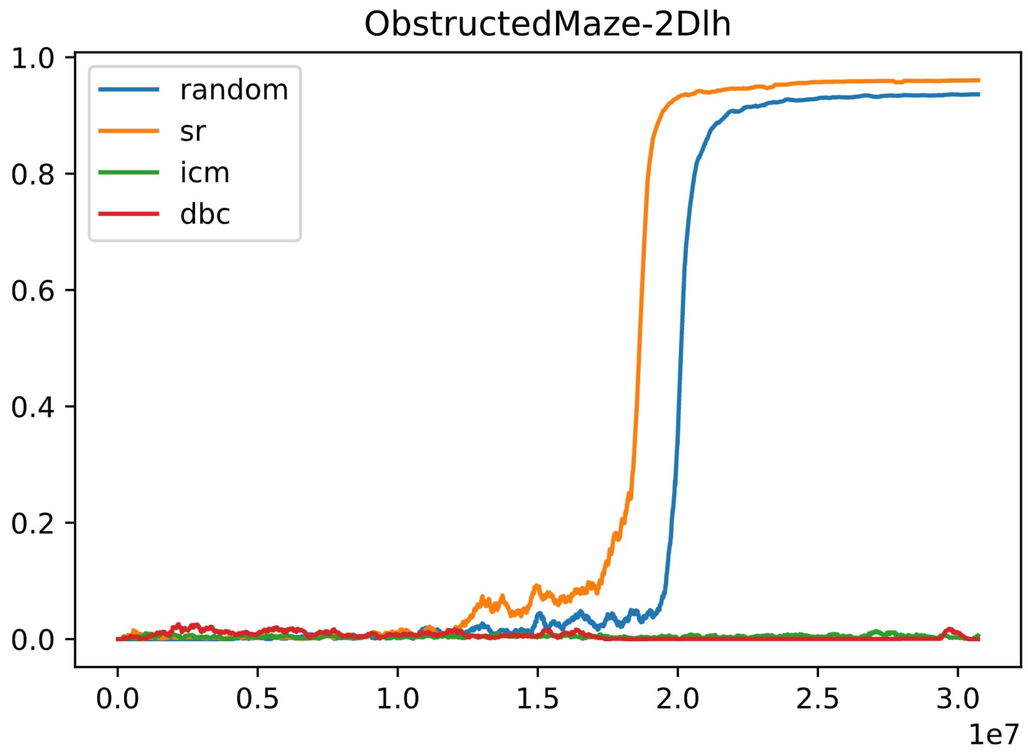
MonteZuma’s Revenge





Huge Performance Difference with different Representations





Random = vanilla BeBold
DBC = deep bisimulation control
SR = Successor Representation

Thanks!