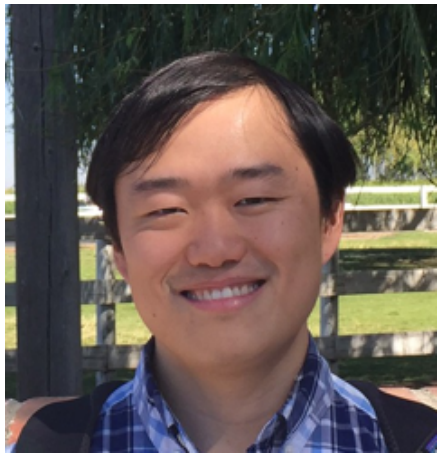
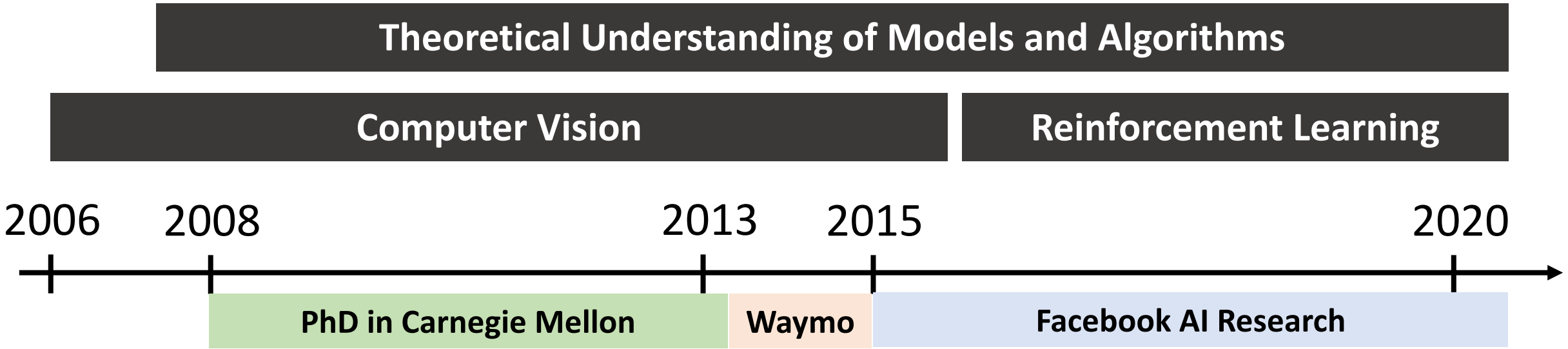
Rethinking Challenging Problems in Reinforcement Learning

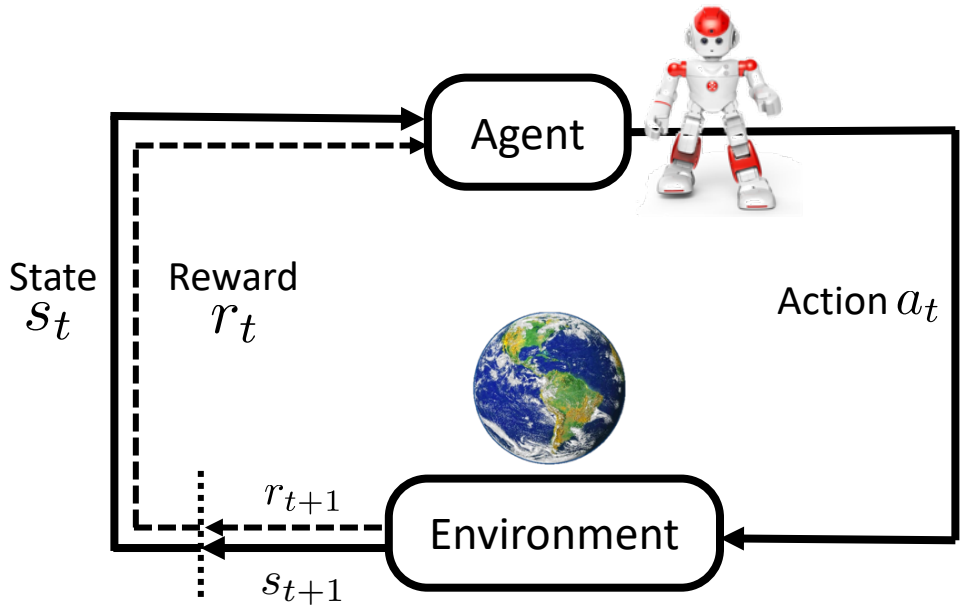
Yuandong Tian
Research Scientist
Facebook AI Research

Career Path

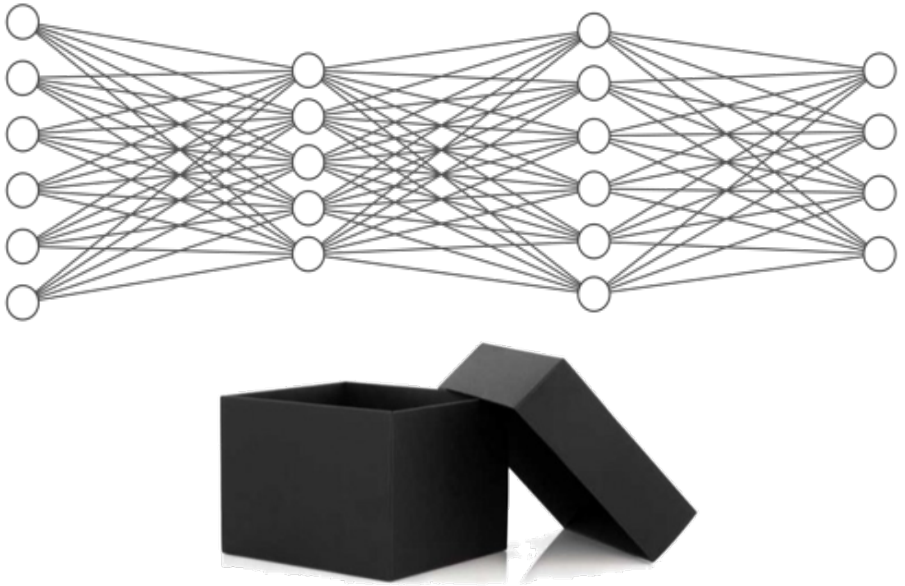



Theoretical Understanding of Models and Algorithms
Computer Vision
Reinforcement Learning
2006
2008
2013
2015
2020
PhD in Carnegie Mellon
Waymo
Facebook AI Research

Research Directions



Reinforcement Learning



Understanding of Deep Models

Challenging Problems in RL

Compounding Errors

Exploration

Demonstration

Instability

Safety

Sample Complexity

Agent Collaboration

Sim2Real

State Decomposition

Convergence

Off-policy Data

Skill Acquisition

Hierarchical RL

Challenging Problems in RL

Compounding Errors

Exploration

Demonstration

Instability

Safety

Sample Complexity

Agent Collaboration

Sim2Real

State Decomposition

Convergence

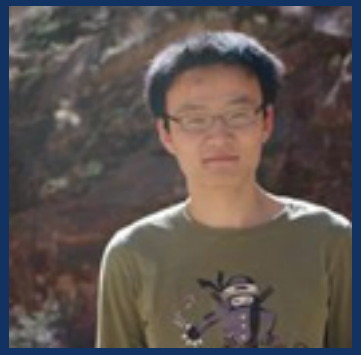
Off-policy Data

Skill Acquisition

Hierarchical RL

Joint Policy Search for Multi-agent Collaboration with Imperfect Information

Yuandong Tian



Qucheng Gong

Tina Jiang

Facebook AI Research

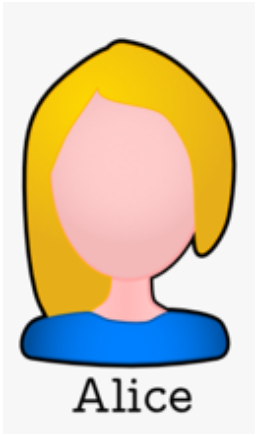
facebook Artificial Intelligence

NeurIPS 2020

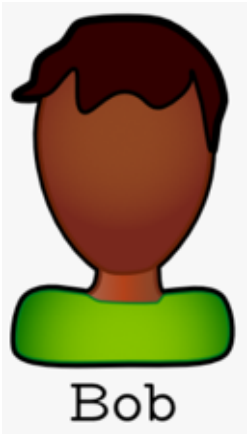
An Illustrative Example

Private Card

or



Public Signal
1 or 2 or 3



Guess
= ♥A

One possible solution (6 symmetric solutions):

Private card	Alice's Action	Bob's Action
♥ A	1	Guess ♥ A
♠ A	3	Guess ♠ A
--	2	--

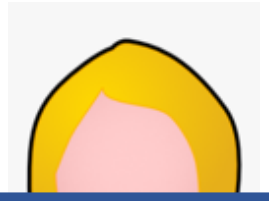
Not used

What if Allice and Bob never use signal 2,

but sending signal 2 come with additional rewards?

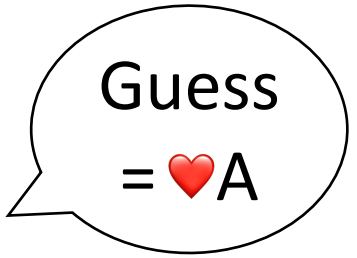
An Illustrative Example

Private Card



Public Signal

1 or 2 or 3



For pure multi-agent collaborative games, A **unilateral** optimization of policy doesn't improve overall value.

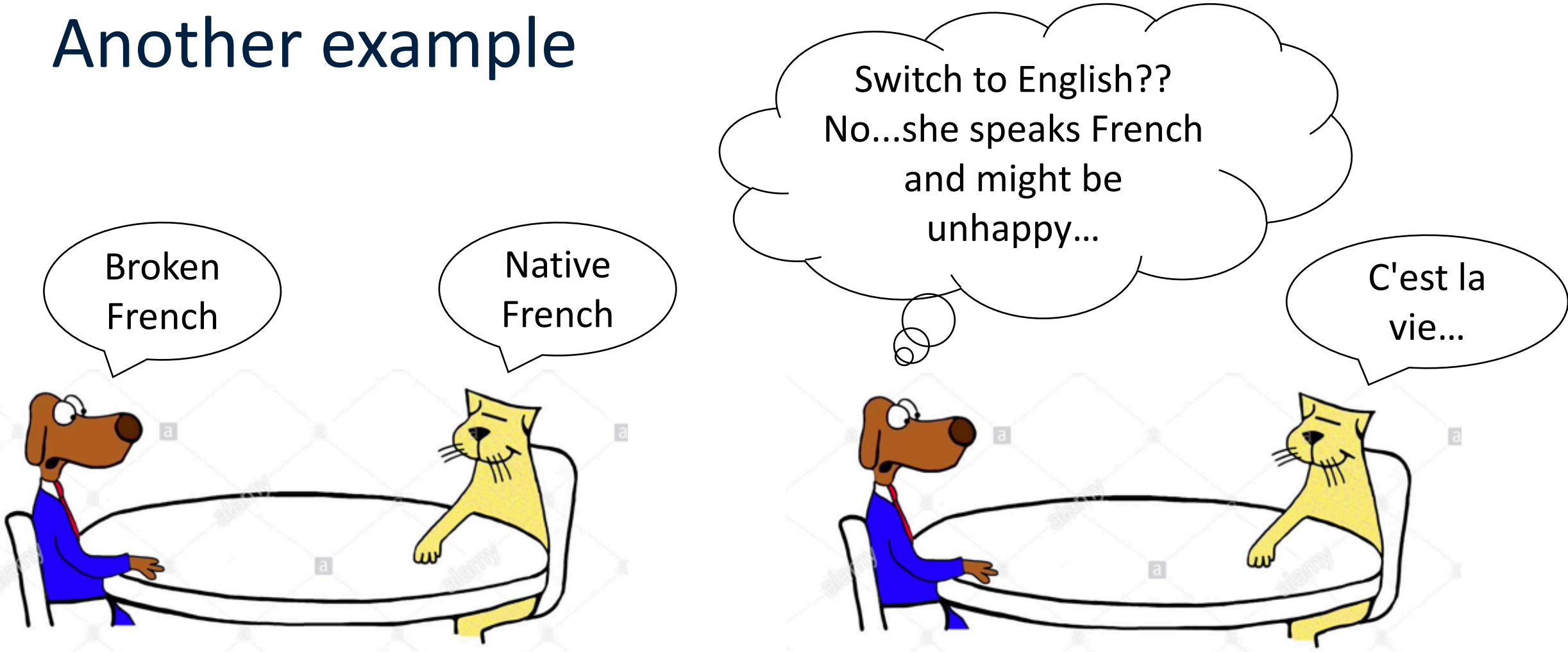
Or

♥ A	1	Guess ♥ A
♠ A	3	Guess ♠ A
--	2	--

Not used

but sending signal 2 come with additional rewards?

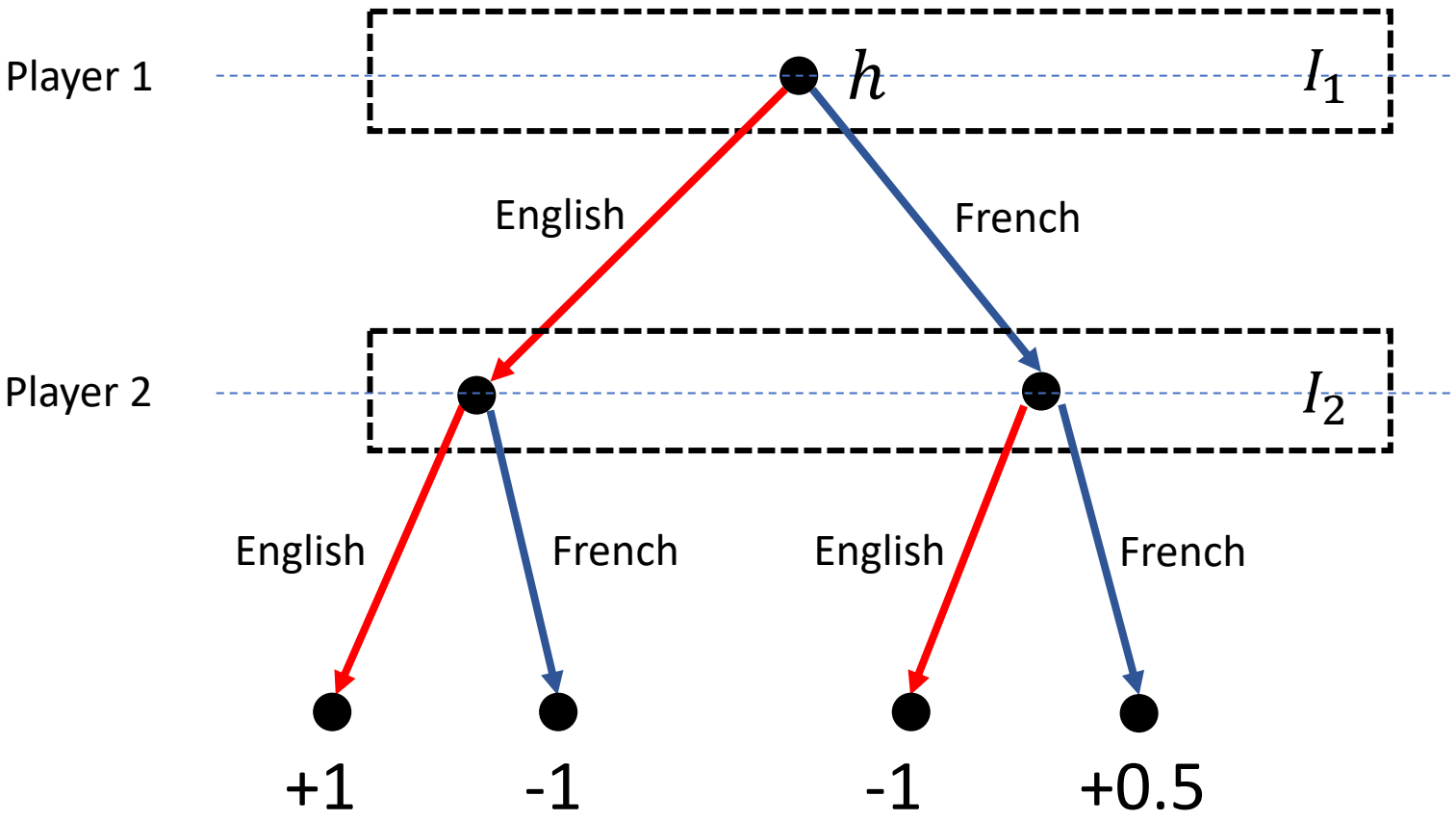
Another example



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

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

Communication Game



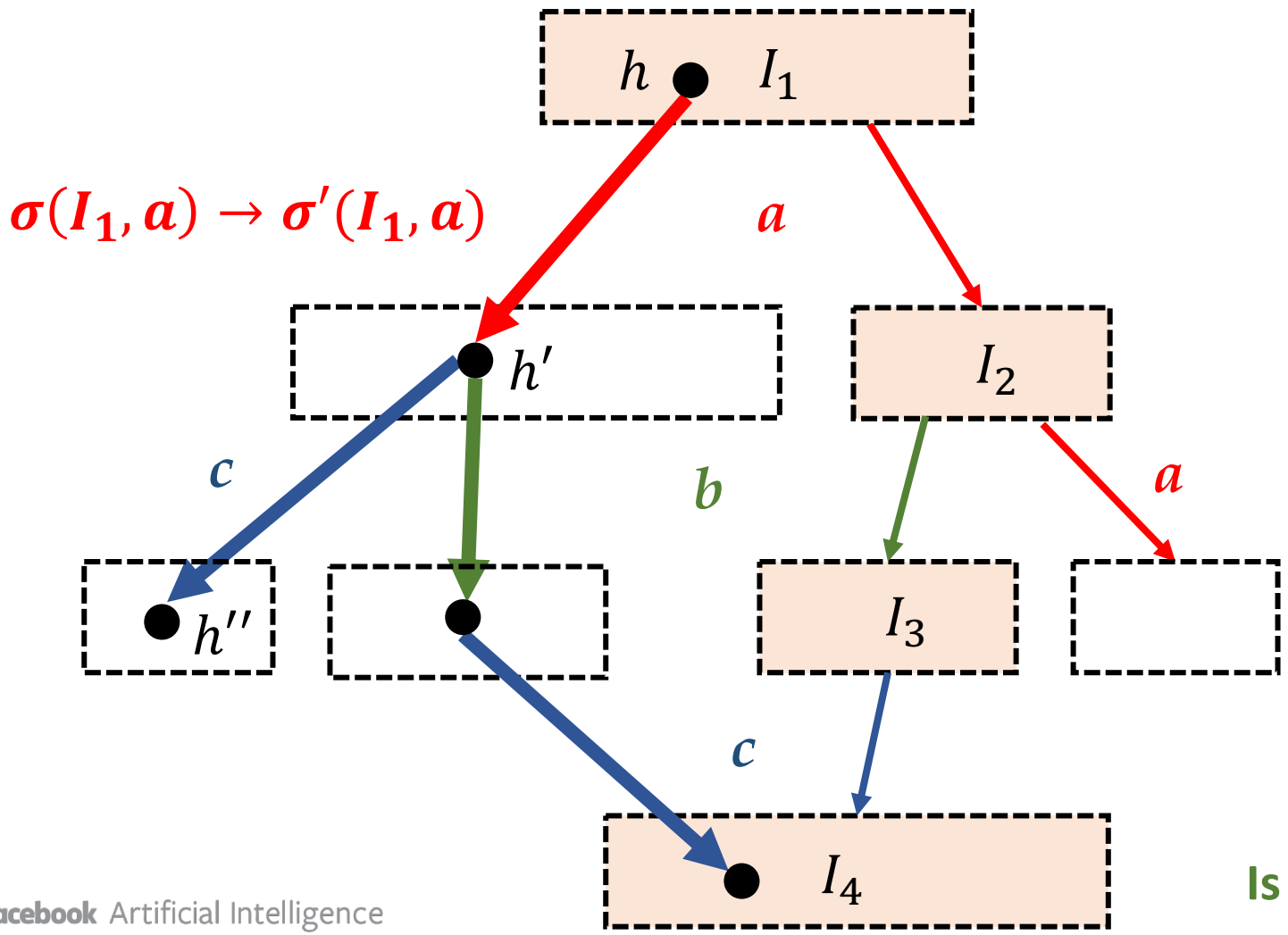
Player 2 makes the decision without knowing player 1's action.

(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*

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.

Is there a good way to track value changes?

Policy-change Density

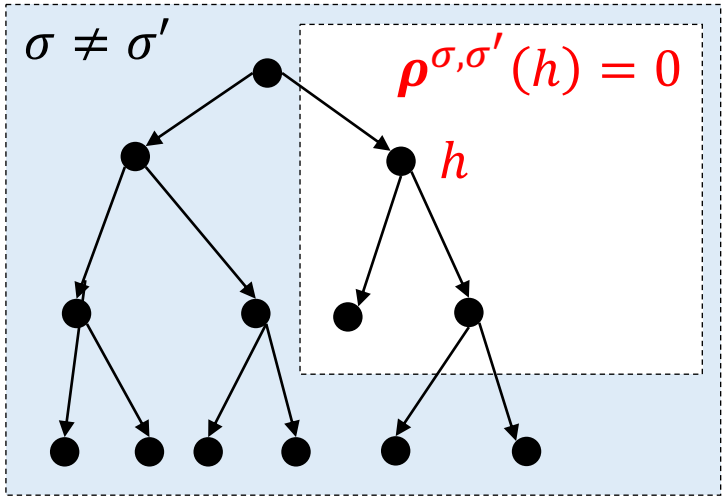
Density $$\rho^{\sigma,\sigma'}(h) = \pi^{\sigma'}(h)\left[\sum_{a\in A(I)} \sigma'(I,a)v^{\sigma}(ha) - v^{\sigma}(h)\right]$$

Two key properties:

(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

Main Theorem

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

Overall value changes due to policy change

All active Infosets ($\sigma' \neq \sigma$)

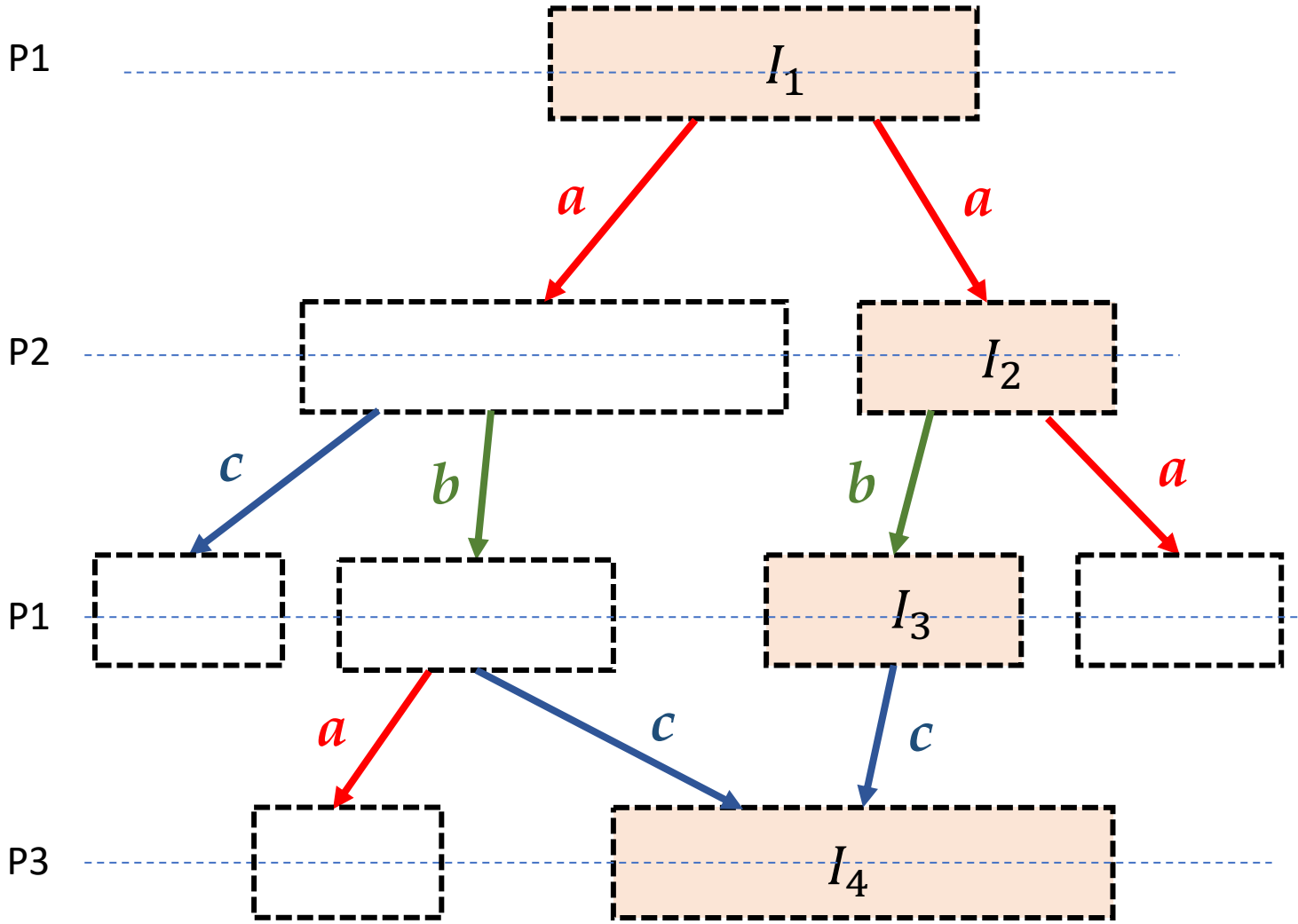
Inactive Infosets doesn't matter!!

JPS (**J**oint **P**olicy **S**earch)

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

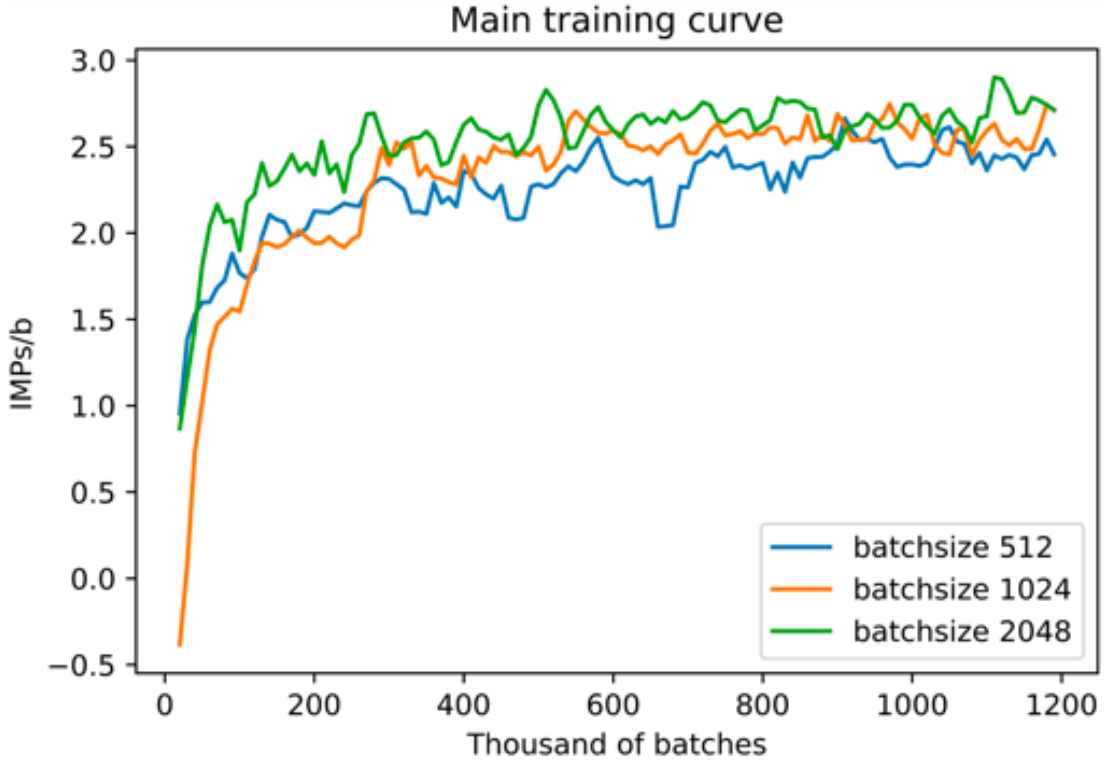
Contract Bridge Bidding

	N ♠A9743 ♥K8763 ♦A6 ♣7	
W ♠None ♥QJ952 ♦109 ♣KQ10982		E ♠Q82 ♥104 ♦QJ85432 ♣J
	S ♠KJ1065 ♥A ♦K7 ♣A6543	

West	North	East	South
			1♠
2♠ [1]	2NT [2]	Pass	3♣
Pass	4♣ [3]	Pass	4NT [4]
Pass	5♠ [5]	Pass	7♠
Pass	Pass	Pass	

(1) Hearts and a minor. (2) Spade support, forcing to game. (3) Short clubs. (4) Keycard Blackwood. (5) Two key cards and the queen of spades, treating his fifth card as the equivalent of the queen.

- **100** years of history
- Imperfect Information
- Collaborative + Competitive
- Large State Space ($5.4*10^{28}$)



A2C Self-play

Double-Dummy Evaluation against SoTA software

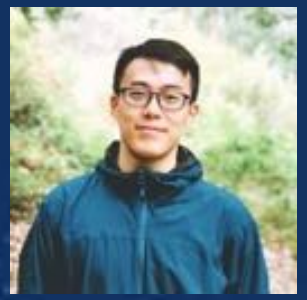
Methods	Vs. WBridge5 (1000 games) (IMPs/board)
Previous SoTA (Rong et al, 2019)	+ 0.25 (on 64 games)
Our A2C baseline	+ 0.29 ± 0.22
1% JPS (2 days)	+ 0.44 ± 0.20
5% JPS (2 days)	+ 0.37 ± 0.19
1% JPS (14 days)	**+ 0.63 ± 0.20**

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

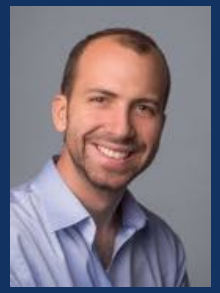
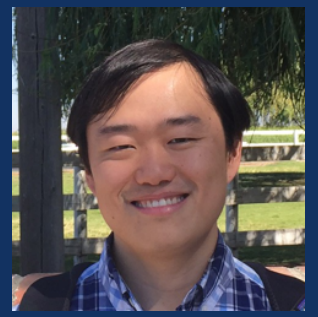
BeBold: Exploration **Be**yond the **Bo**undary of Exp**l**ore**d** Regions

Tianjun Zhang[1,4] Huazhe Xu[1,4] Xiaolong Wang[1,2] Yi Wu[3] Kurt Keutzer[1]

Joseph E. Gonzalez[1] Yuandong Tian[4]





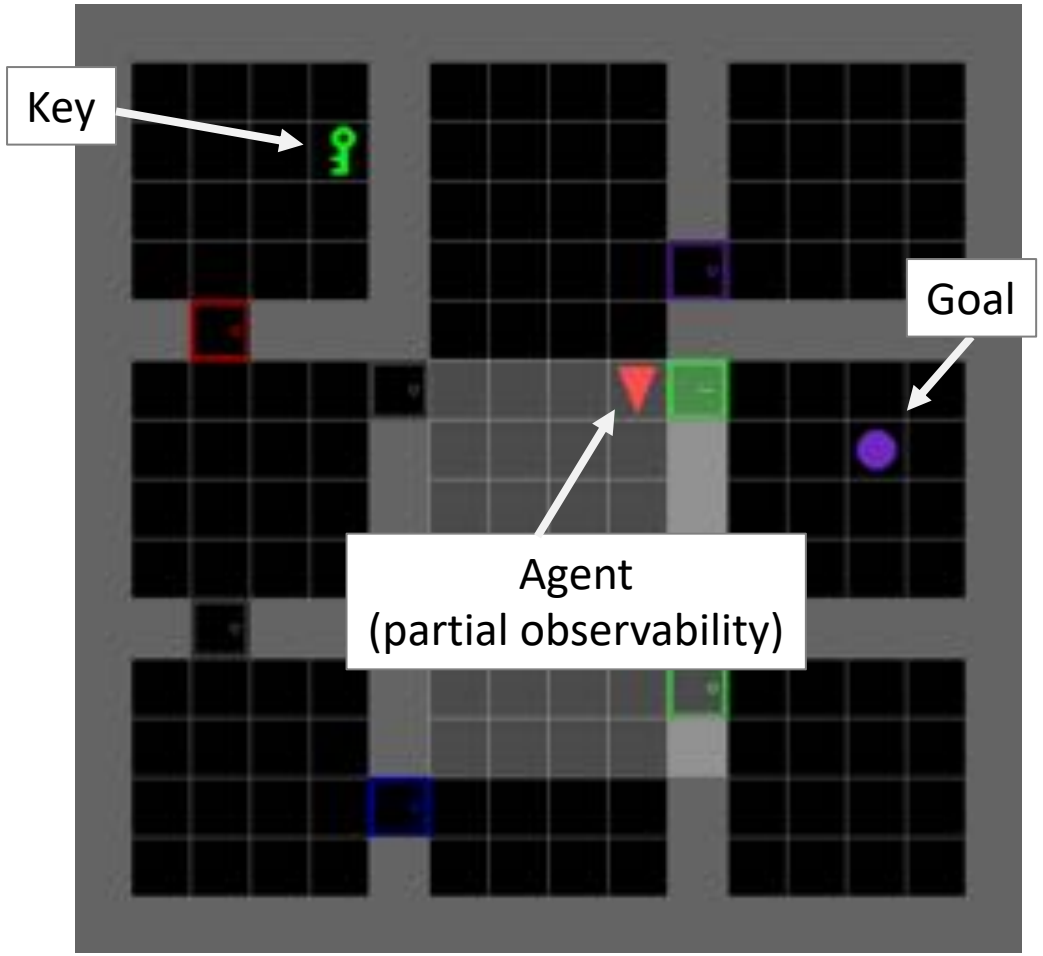
[1]UC Berkeley [2]UCSD [3]Tsinghua University [4]FaceBook AI Research

facebook Artificial Intelligence

BAIR
BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH

Imaged by Heritage Auctions, HA.com

Environment with Sparse Reward

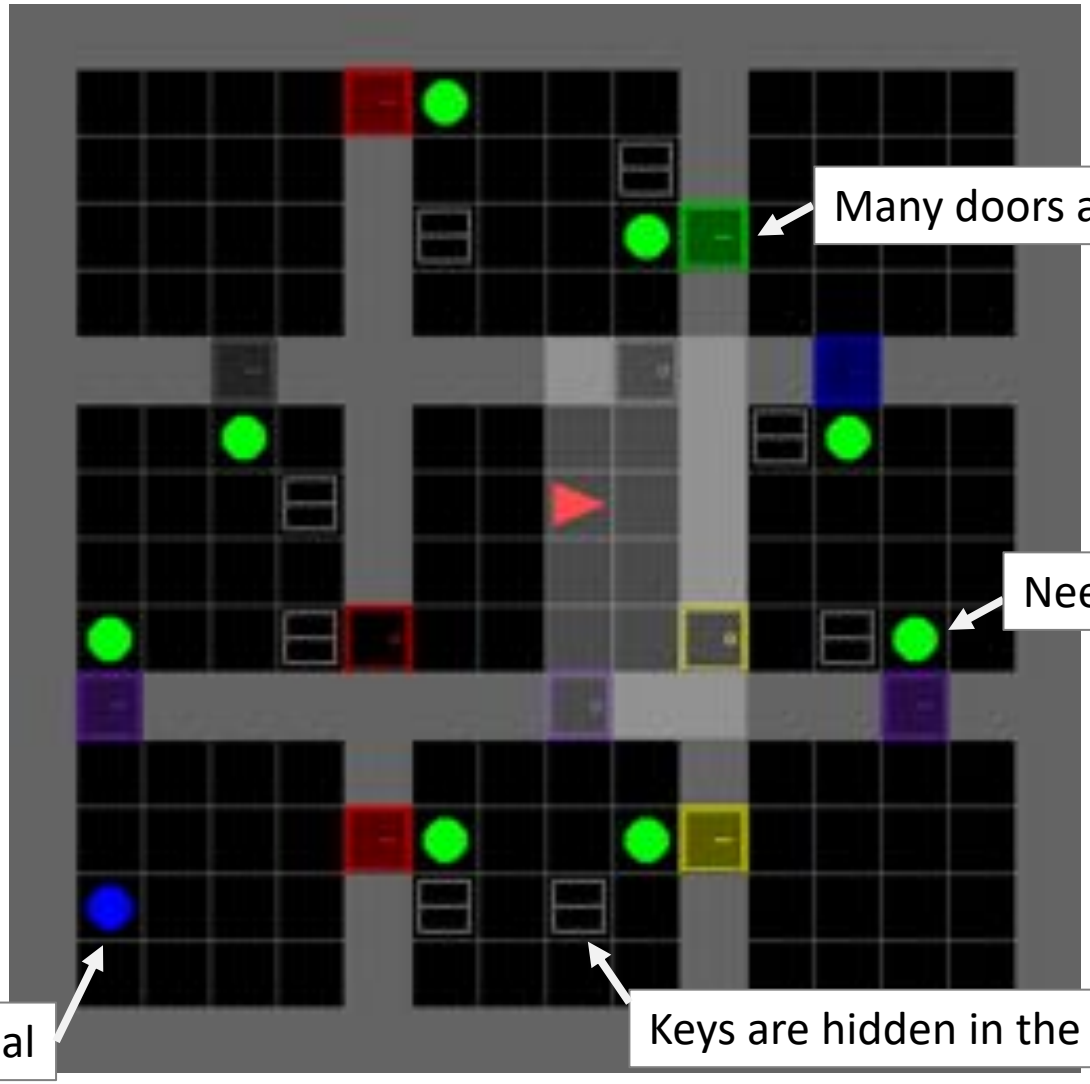


No external reward

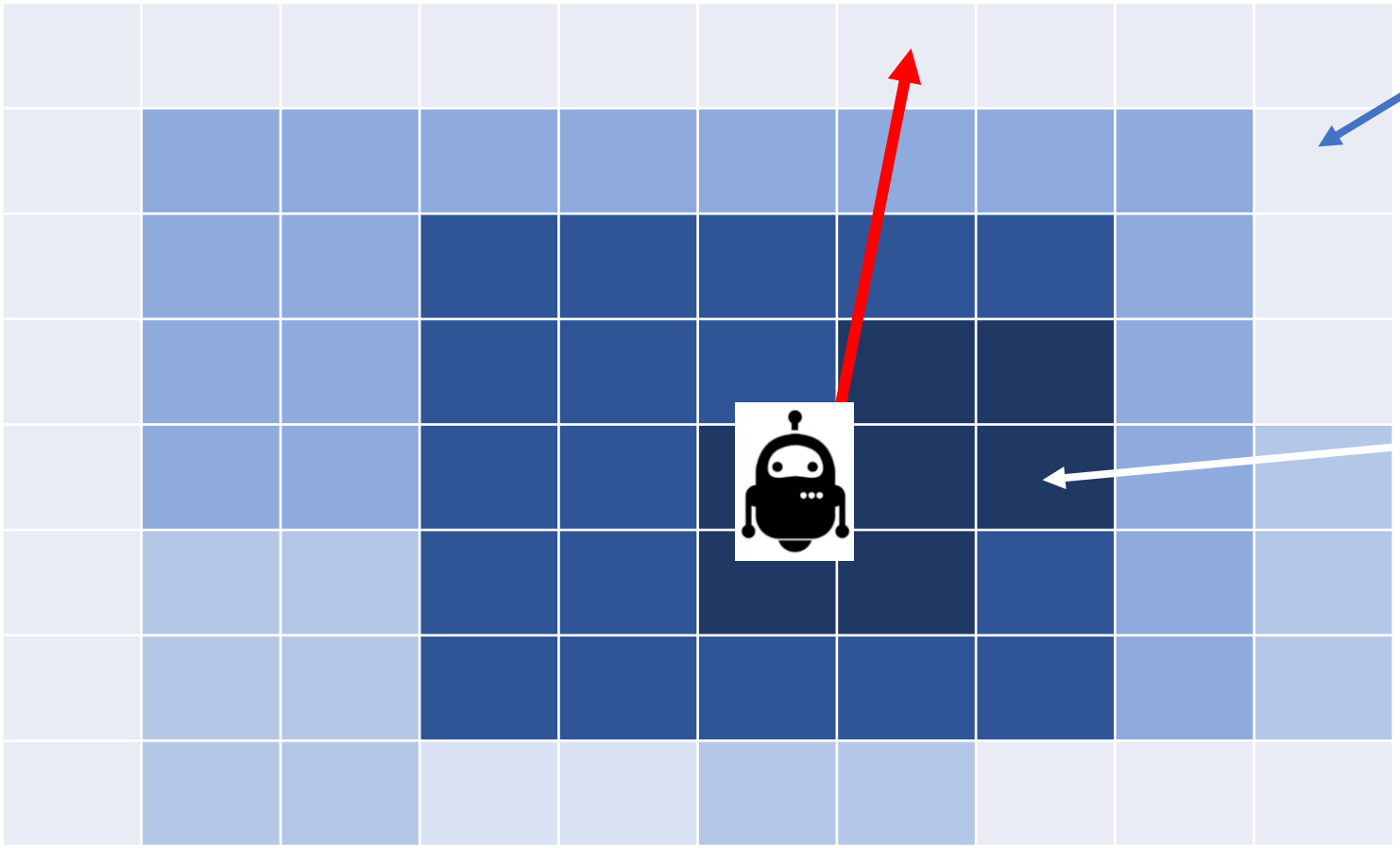
when agent wonders around.
when agent picks the key
when agent opens all doors
when agent opens the locked door
…

until the agent reaches the goal

And more complicated situations...



Count-based Exploration



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

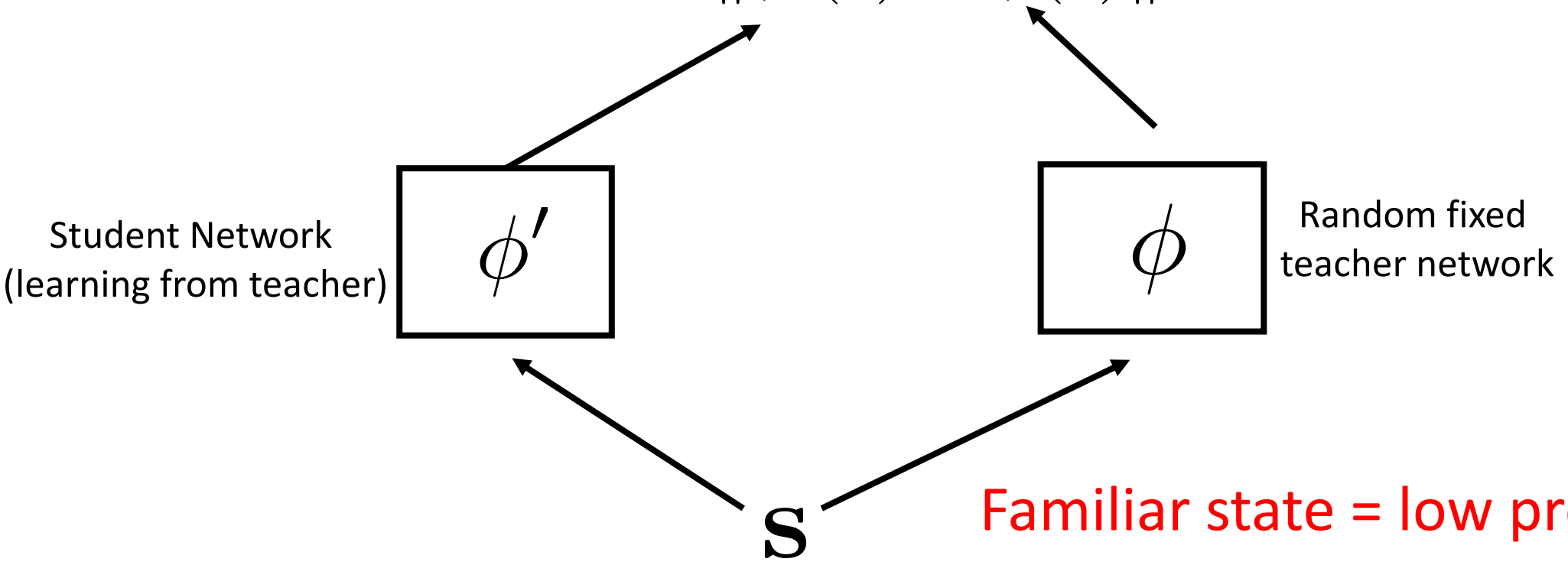
High visitation counts $N(s)$
Low Intrinsic reward

What if we have exponential #states?

[Bellemare, Marc, et al. "Unifying count-based exploration and intrinsic motivation." NeurIPS 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]

Issues in RND

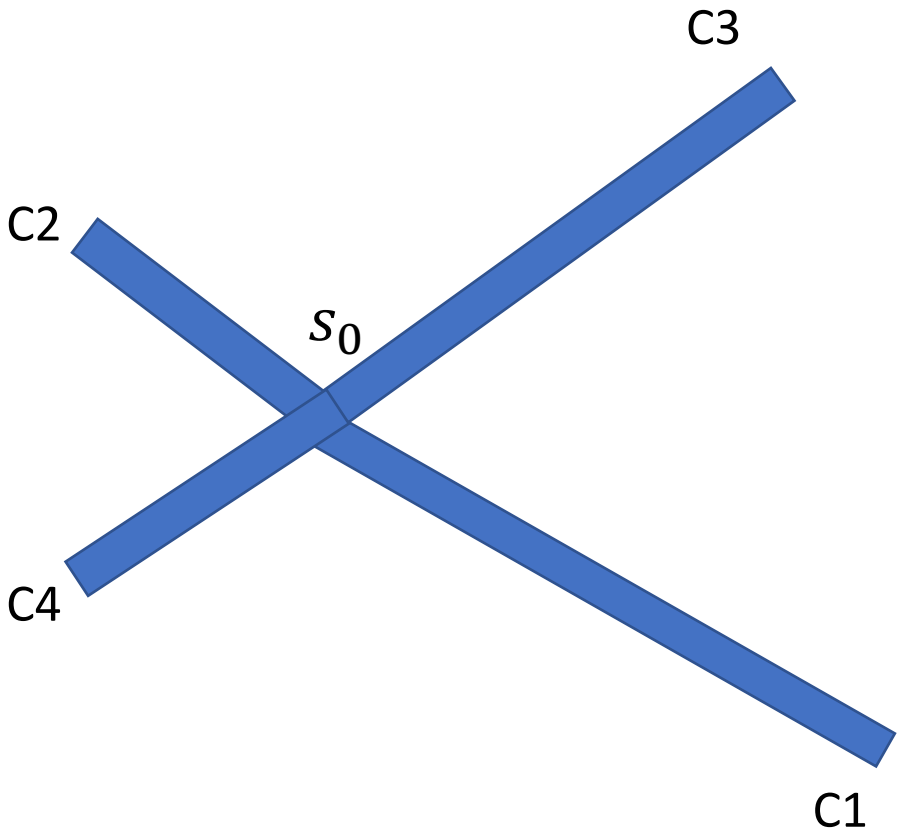
1. RND assigns high IR (dark green) throughout the environment

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

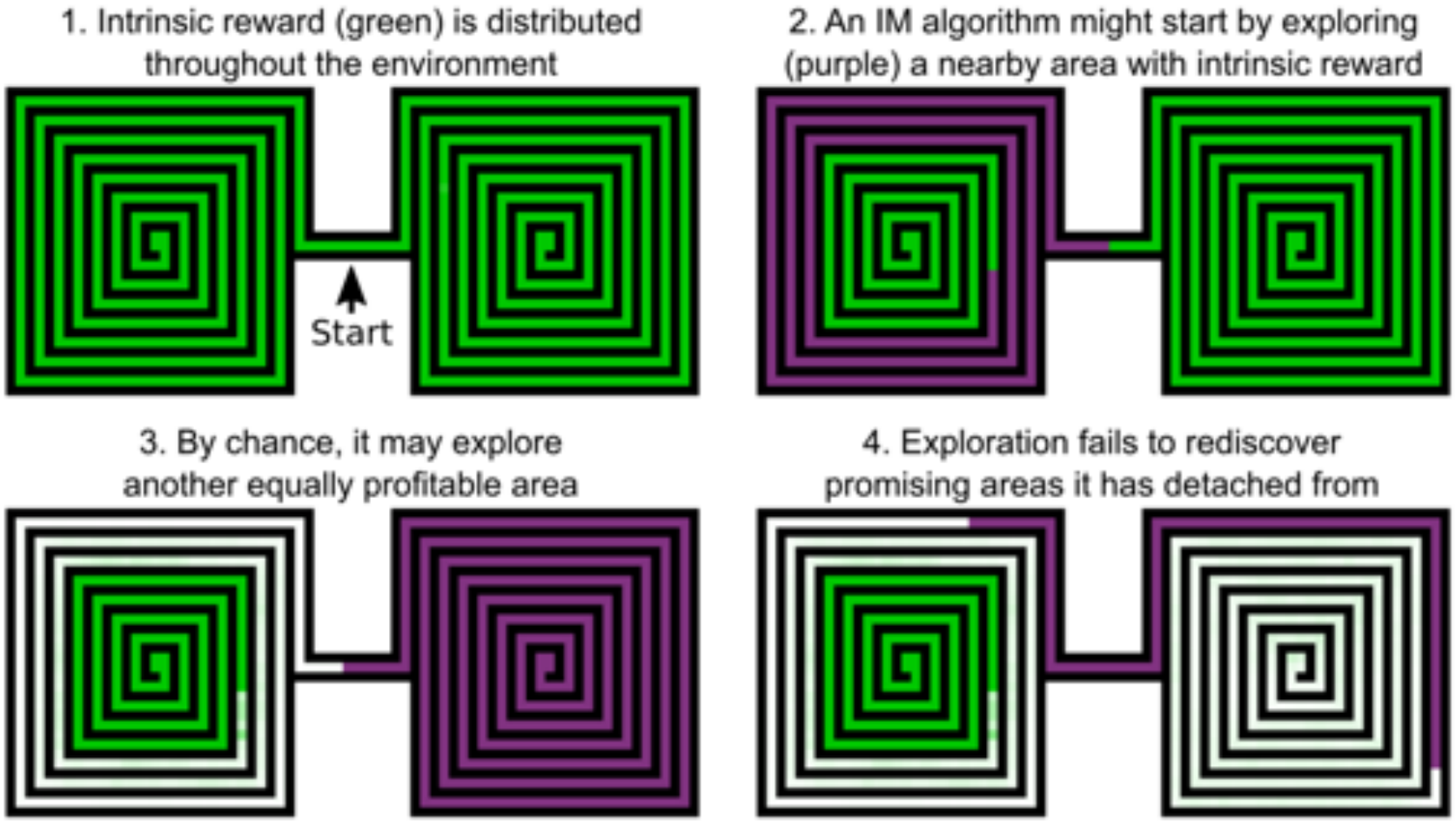
4. RND re-explores the upper right and forgets the bottom right, gets trapped

Multi-Corridor Problem



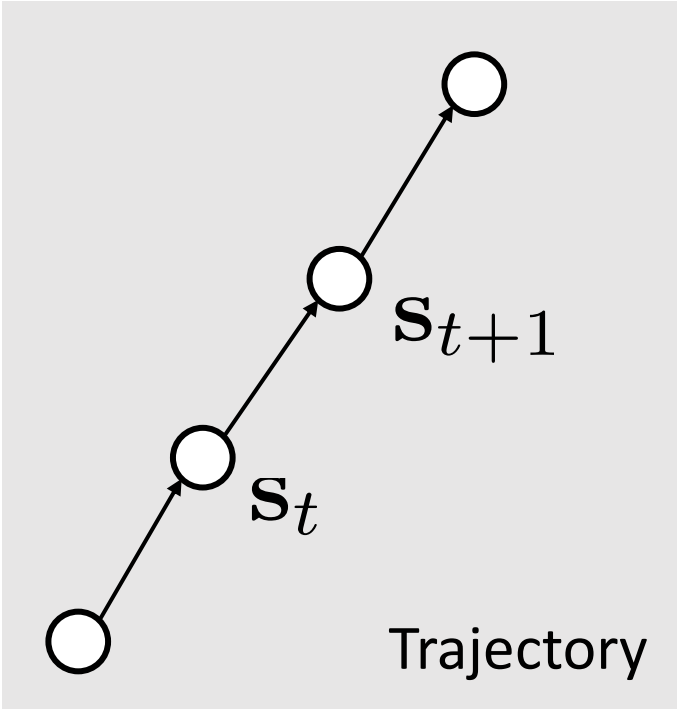
	C1	C2	C3	C4	Entropy
Length	40	10	30	10	–
Count-Based	68K	2K	3K	36K	1.23
BeBold Tabular	27K	51K	13K	17K	**1.80**
RND	26K	38K	14K	32K	1.92
BeBold	27K	24K	33K	27K	**1.99**

Detachment in RND



[A. Ecoffet et al, Go-Explore: a New Approach for Hard-Exploration Problems, arxiv 2019]

BeBold



$$r^i(\mathbf{s}_t, \mathbf{a}_t) = \max\left(\frac{1}{N(\mathbf{s}_{t+1})} - \frac{1}{N(\mathbf{s}_t)}, 0\right) * \mathbb{1}\{N_e(\mathbf{s}_{t+1}) = 1\}$$

Intrinsic Reward

Inverse of visitation counts

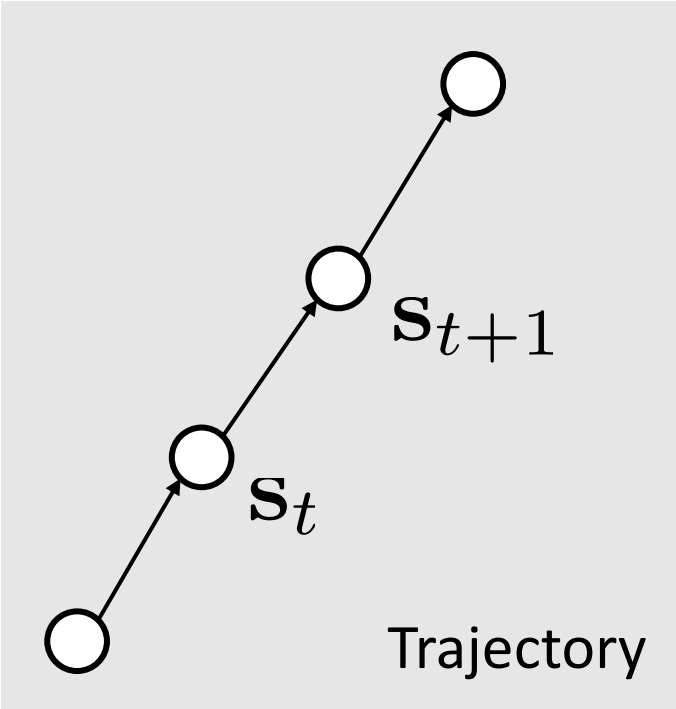
Episodic visitation count

BeBold (**Be**yond the **Bo**undary of Explore**d** Regions)

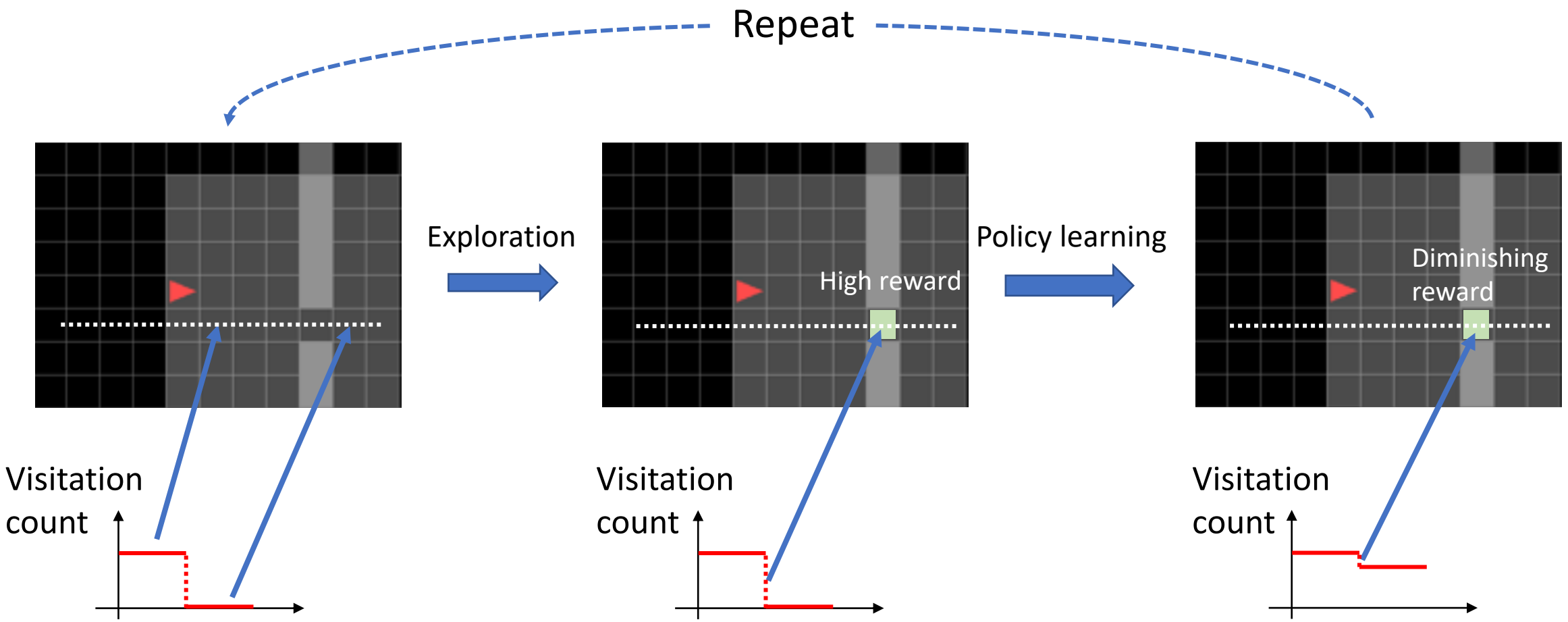
$$r^i(\mathbf{s}_t, \mathbf{a}_t) = \max\left(\frac{1}{N(\mathbf{s}_{t+1})} - \frac{1}{N(\mathbf{s}_t)}, 0\right)$$

Intrinsic Reward

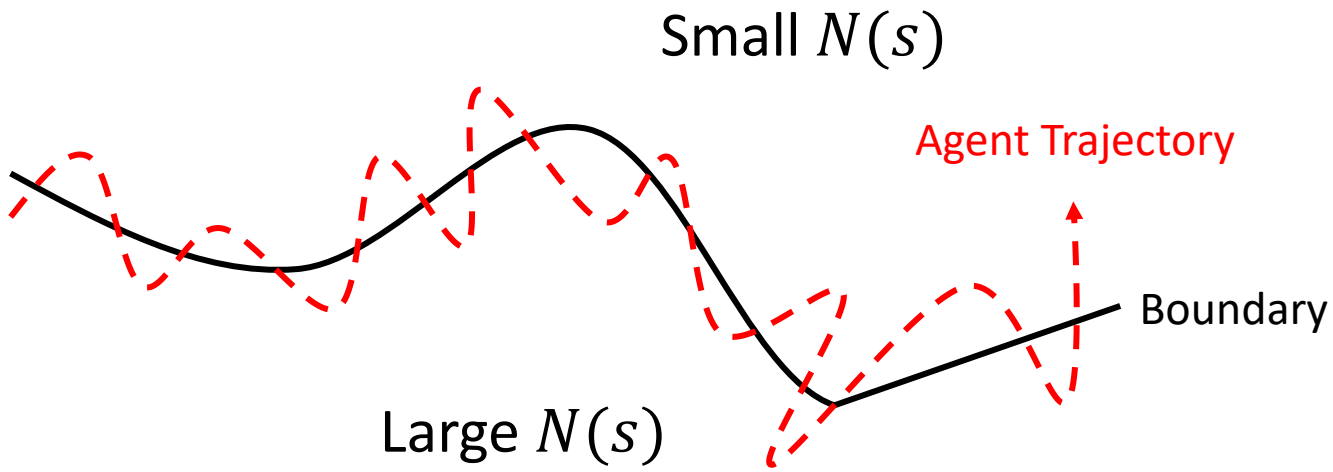
Inverse of visitation counts



BeBold (**Be**yond the **Bo**undary of Explore**d** Regions)



BeBold



$$r^i(\mathbf{s}_t, \mathbf{a}_t) = \max\left(\frac{1}{N(\mathbf{s}_{t+1})} - \frac{1}{N(\mathbf{s}_t)}, 0\right) * \mathbb{1}\{N_e(\mathbf{s}_{t+1}) = 1\}$$

Intrinsic Reward

Episodic visitation count

BeBold (final form)

RND for t+1

RND for t

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

Hash Table

Observation (rather than full state)

RND

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

BeBold

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	OM2DI-h	OM2DI-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

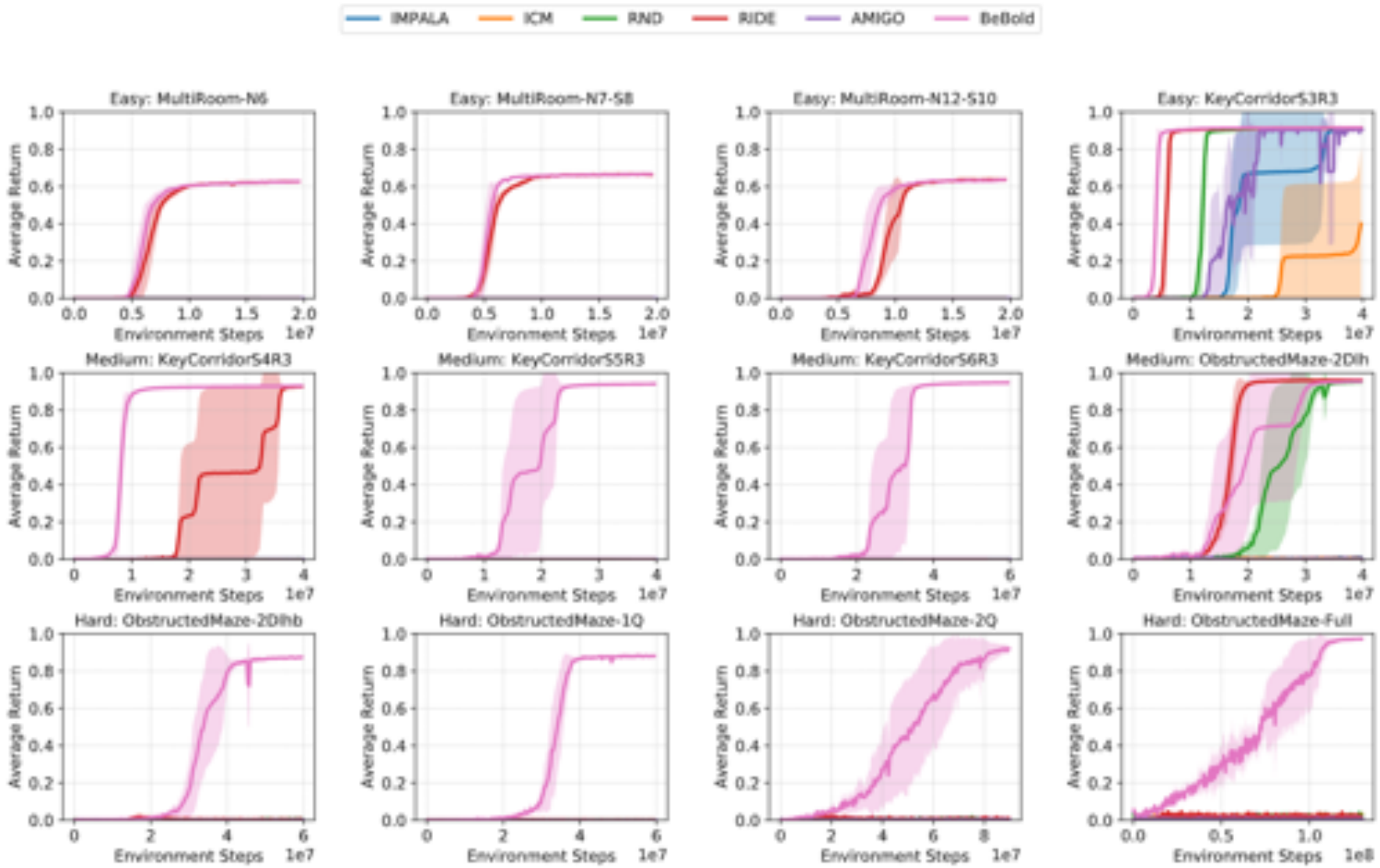
[Chevalier-Boisvert, Maxime, Lucas Willems, and Suman Pal. "Minimalistic gridworld environment for openai gym." GitHub repository (2018)]

MiniGrid

Easy

Medium

Hard



AMIGO: [Campero, Andres, et al. "Learning with AMIGo: Adversarially Motivated Intrinsic Goals." arXiv preprint arXiv:2006.12122 (2020)]

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

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

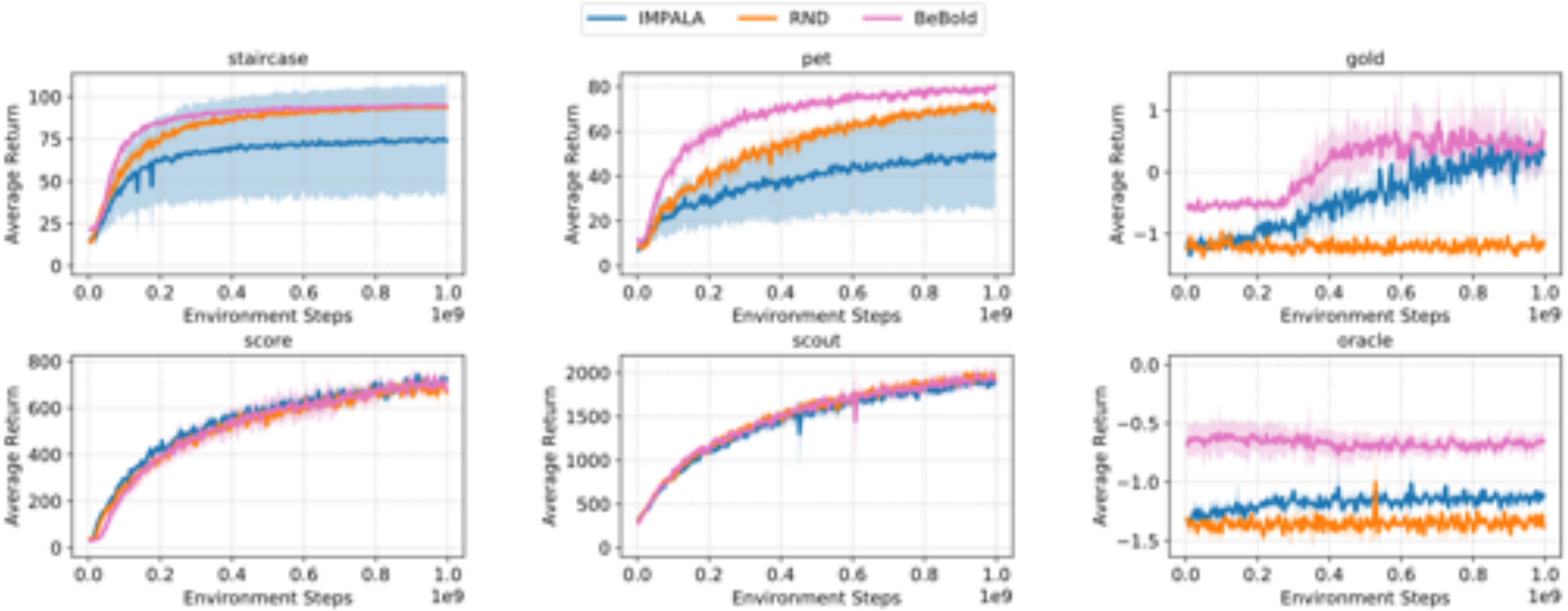
Pure Exploration

NetHack

Agent States

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

6 Tasks in NetHack



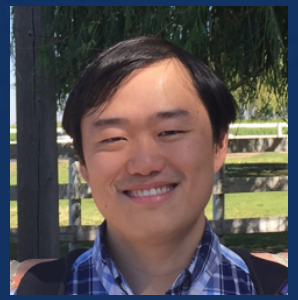
Learning Action Space in Monte Carlo Tree Search


Linnan Wang[1]


Saining Xie[2]


Teng Li[2]


Rodrigo Fonseca[1]


Yuandong Tian[2]

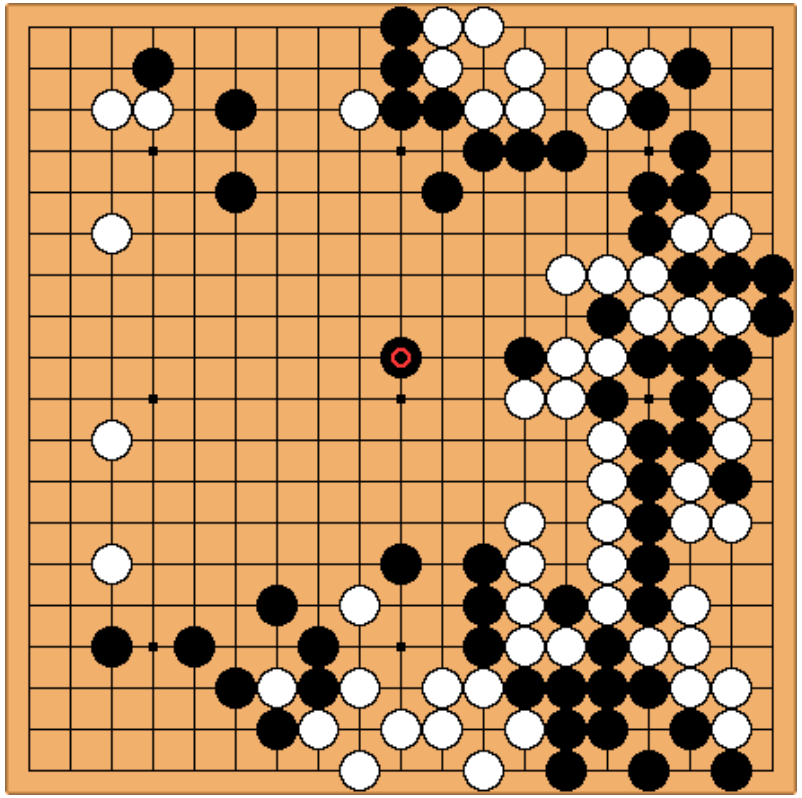
[1]Brown University, [2]Facebook AI Research

Code is available: https://github.com/facebookresearch/LaMCTS

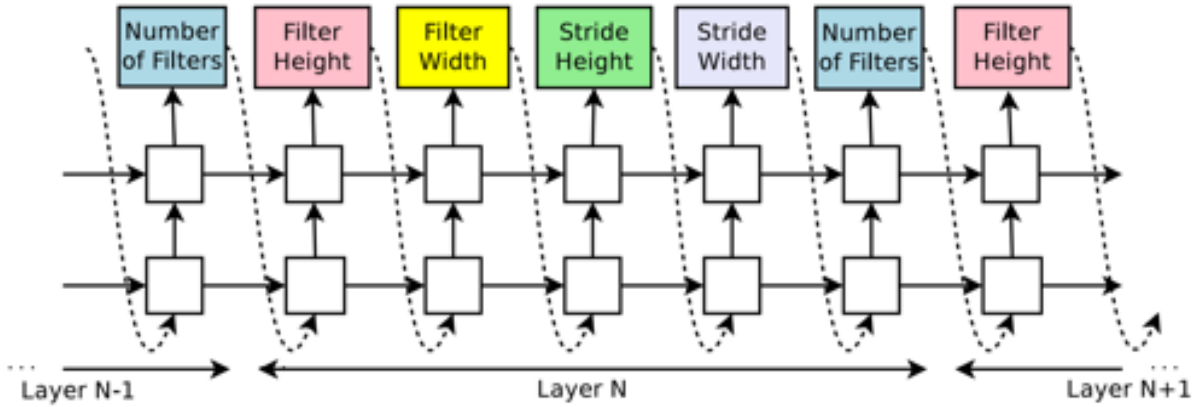
facebook Artificial Intelligence

LA-MCTS

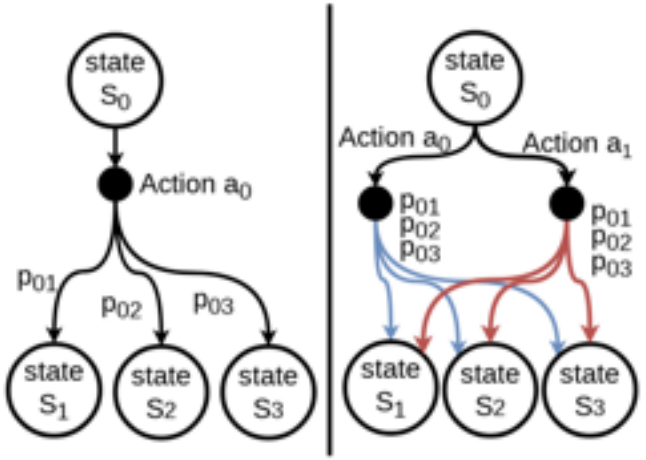
Predefined Action Space



Fixed action space = R^{361}

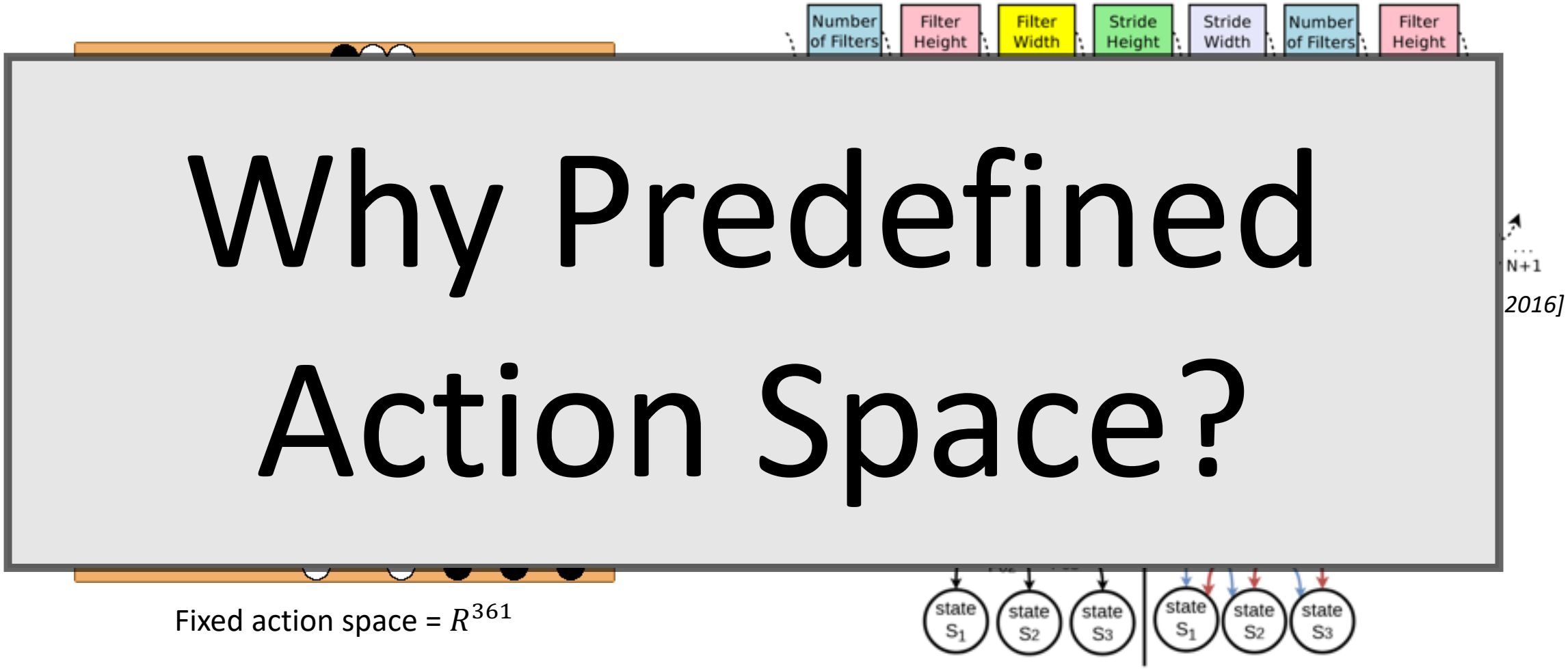


[B. Zoph and Q. Le, Neural Architecture Search with Reinforcement Learning, 2016]



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

Predefined Action Space



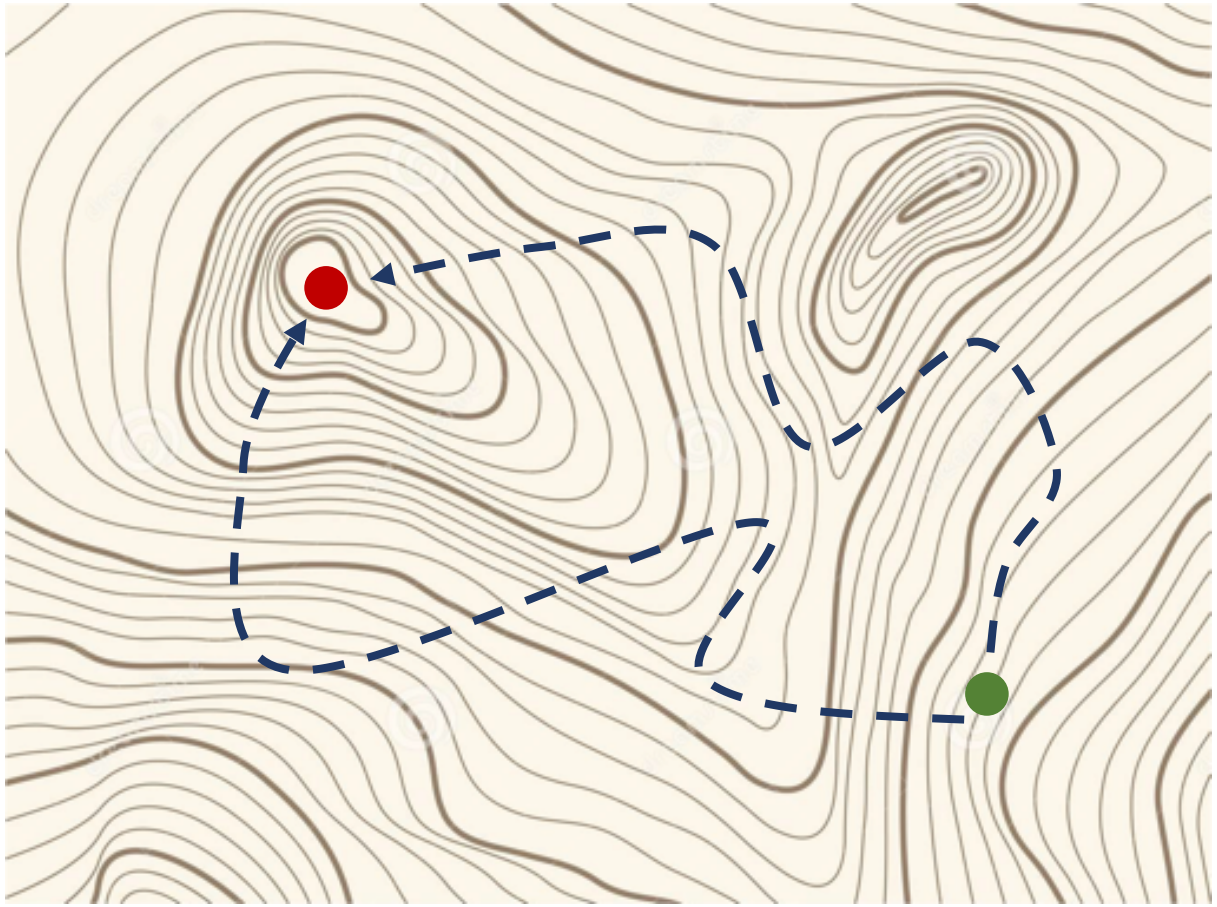
Why Predefined Action Space?

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

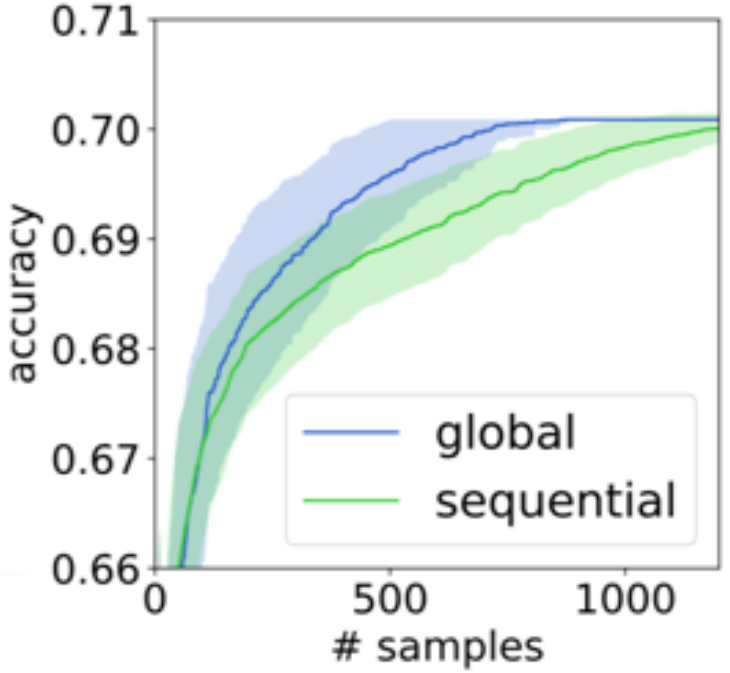
We don’t care how we get it.

Motivating Examples

Global is better!

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

1364 networks.

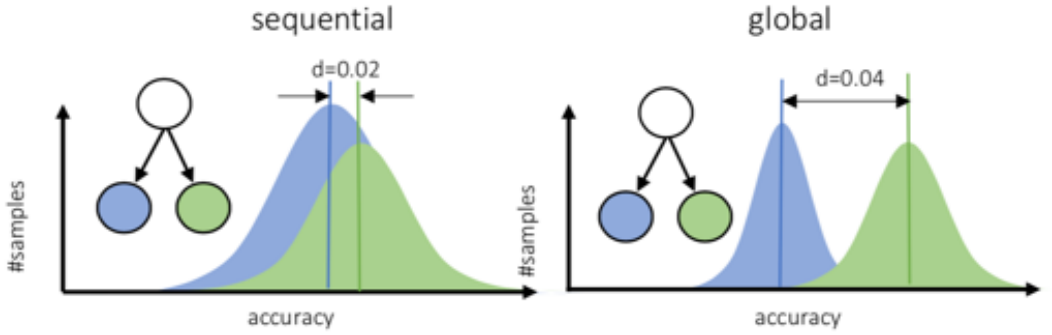


Goal: Find the network
with the best accuracy using fewest trials.

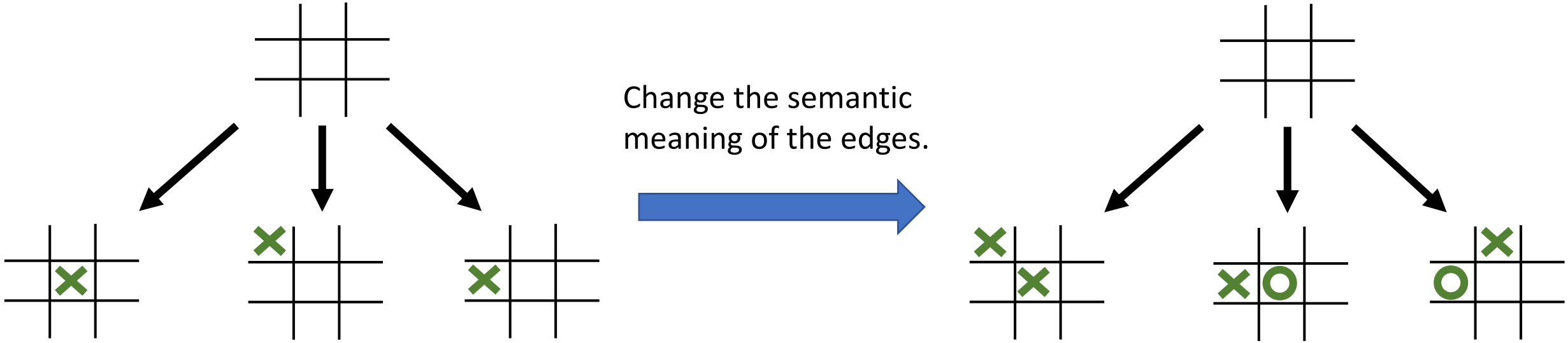
Action space

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

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

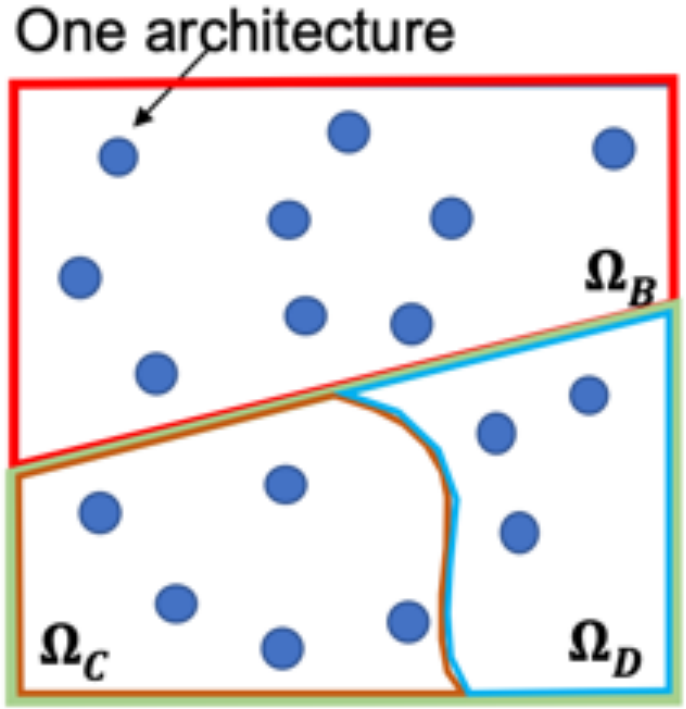


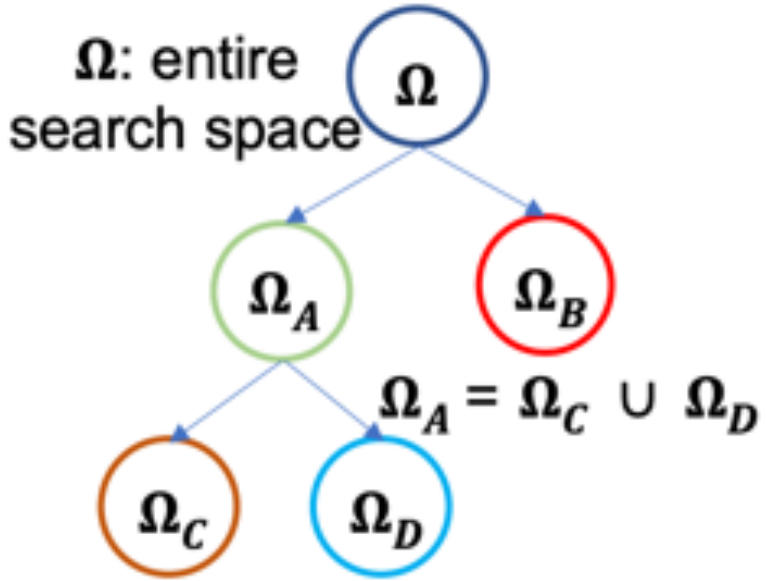
The Meaning of Learning Action Space



Not allowed in games, but doable in optimization.

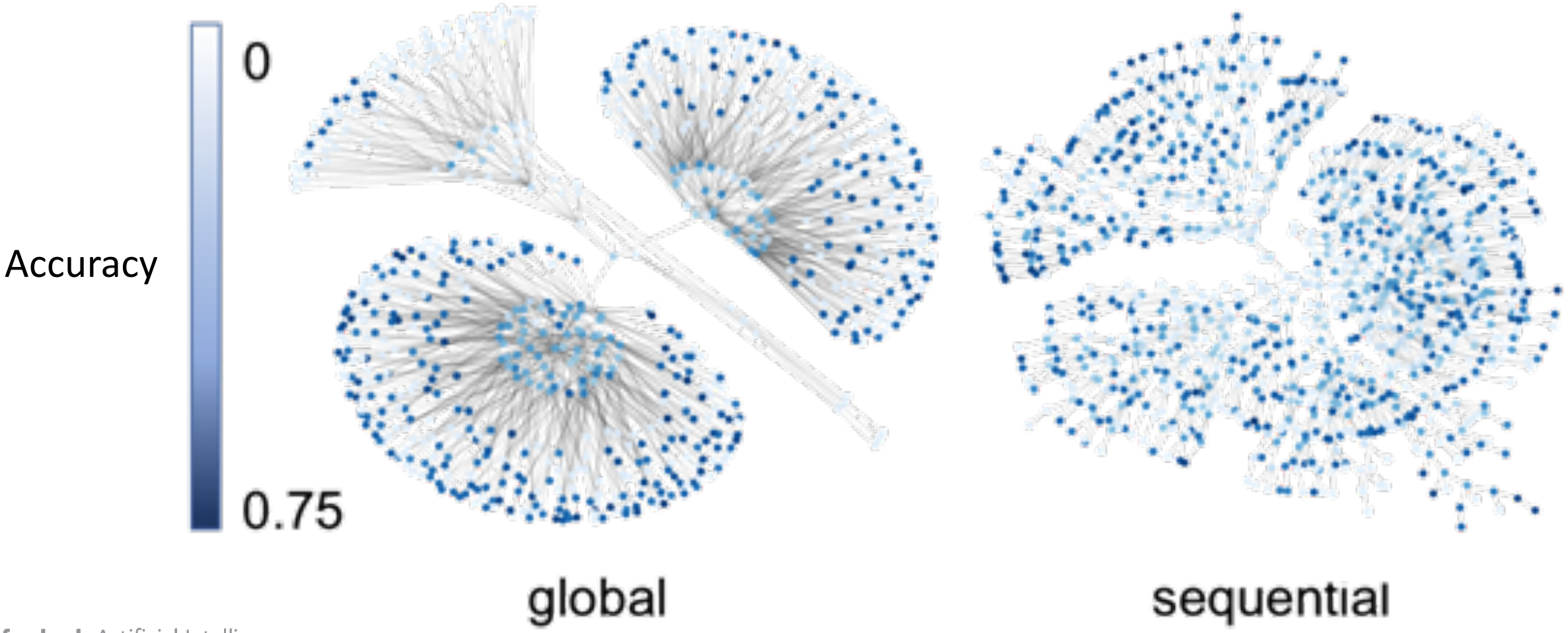
How to learn the action space?



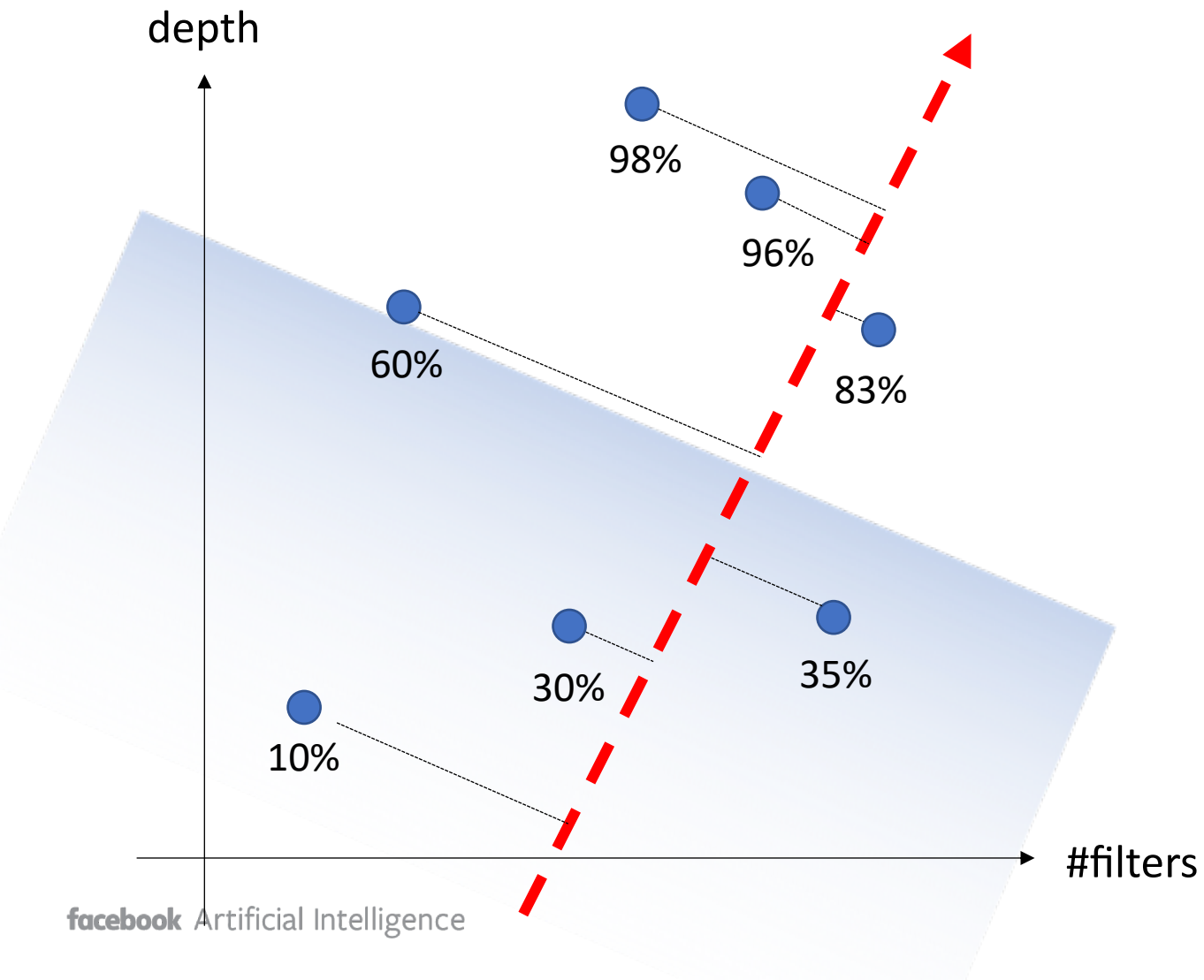


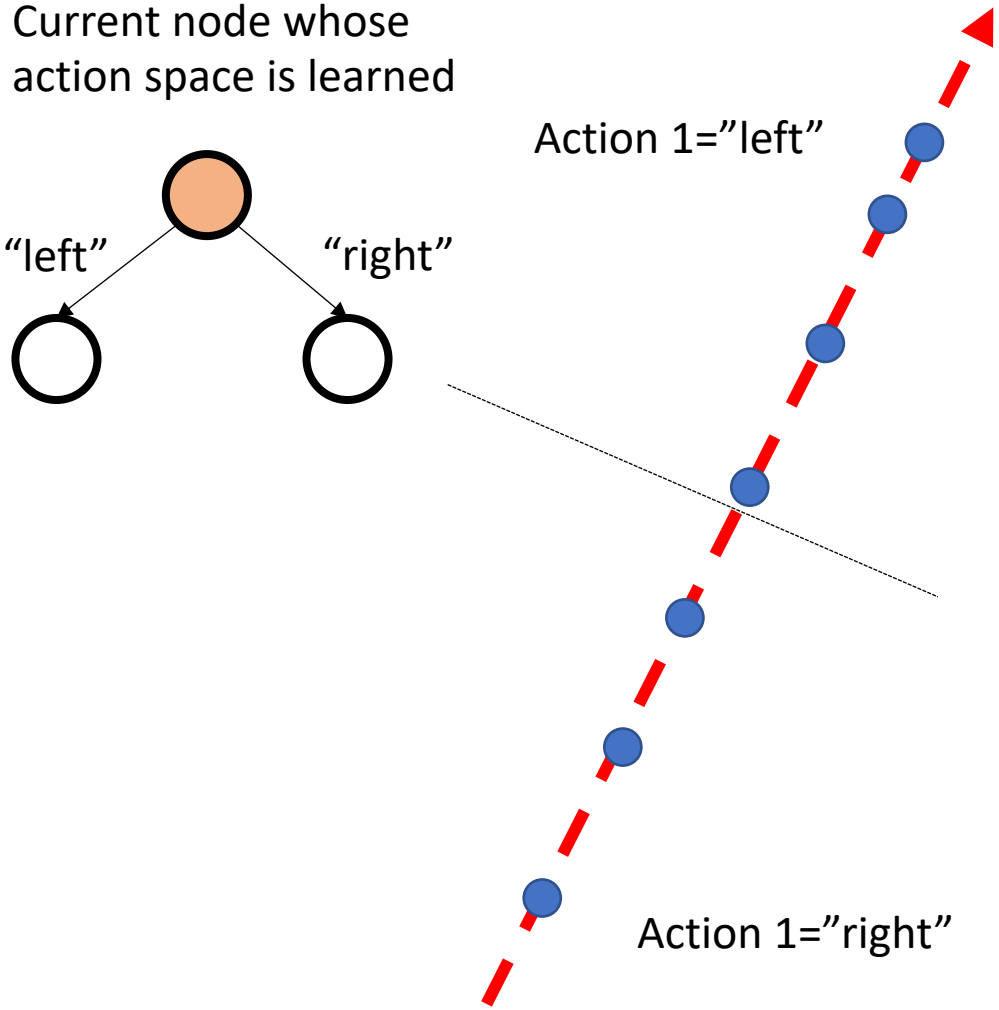
Partition = Action

Different Partition → Different Value Distribution

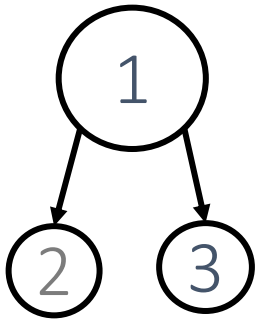


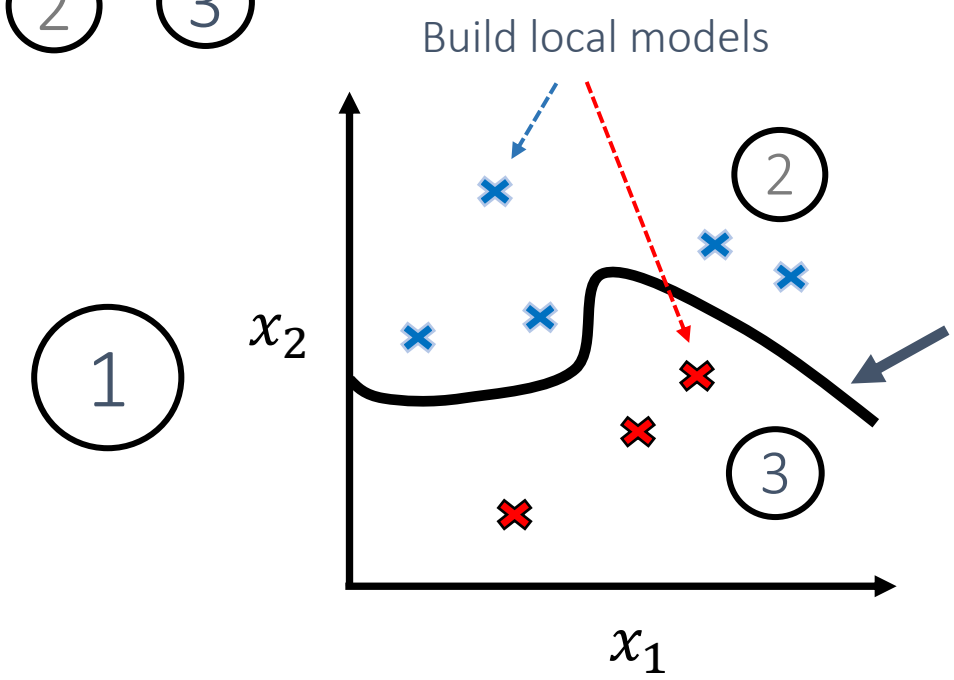
Learn action space



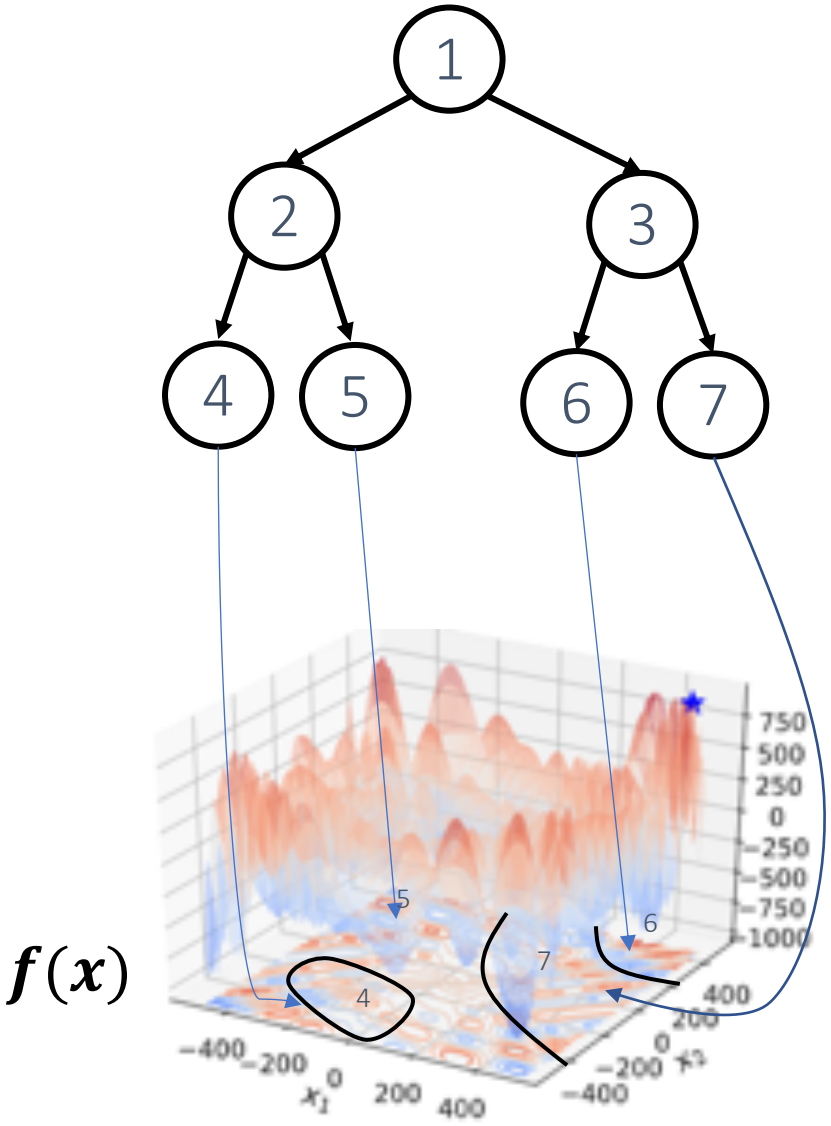


Nonlinear Partition



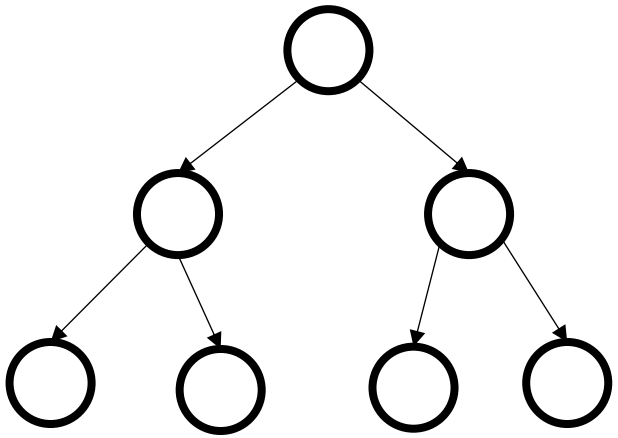


Nonlinear boundary learnt by SVM



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

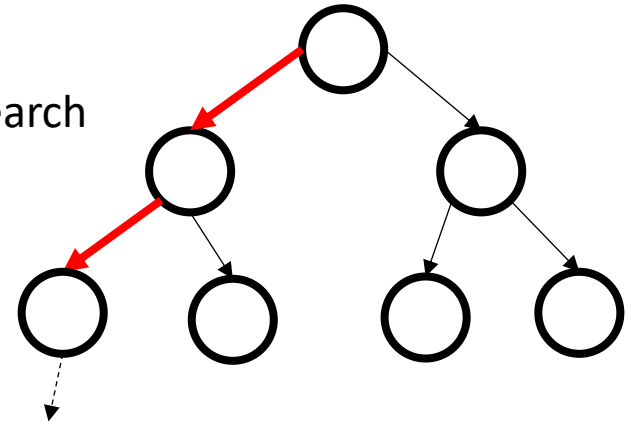
Approach



Fixed action branches
(but not action space)

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

Monte Carlo Tree Search
(MCTS)

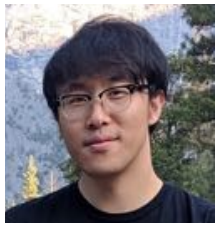


Getting the true quality $f(x)$ for the solution x

(b) Train the action space.

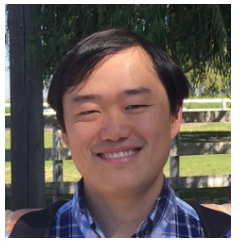
	Accuracy
(filter=2, depth=5)	85%
(filter=3, depth=7)	92%


Linnan Wang[1]


Saining Xie[2]


Teng Li[2]


Rodrigo Fonseca[1]

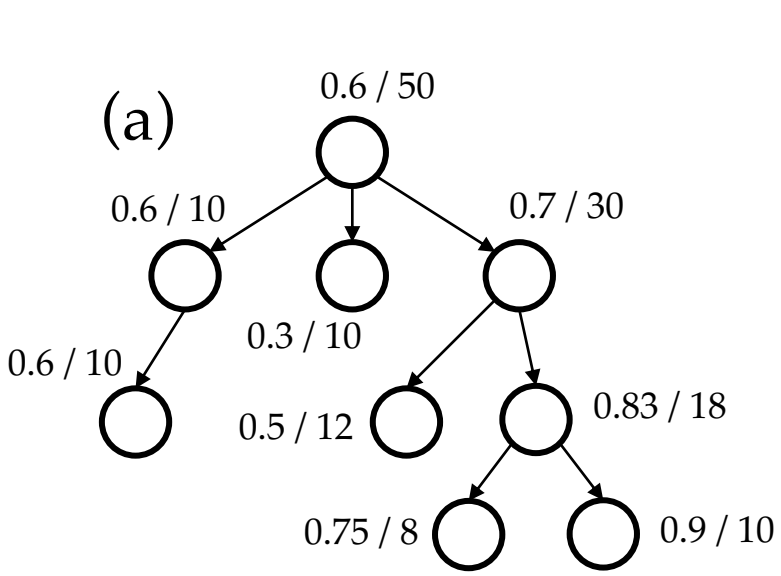

Yuandong Tian[2]

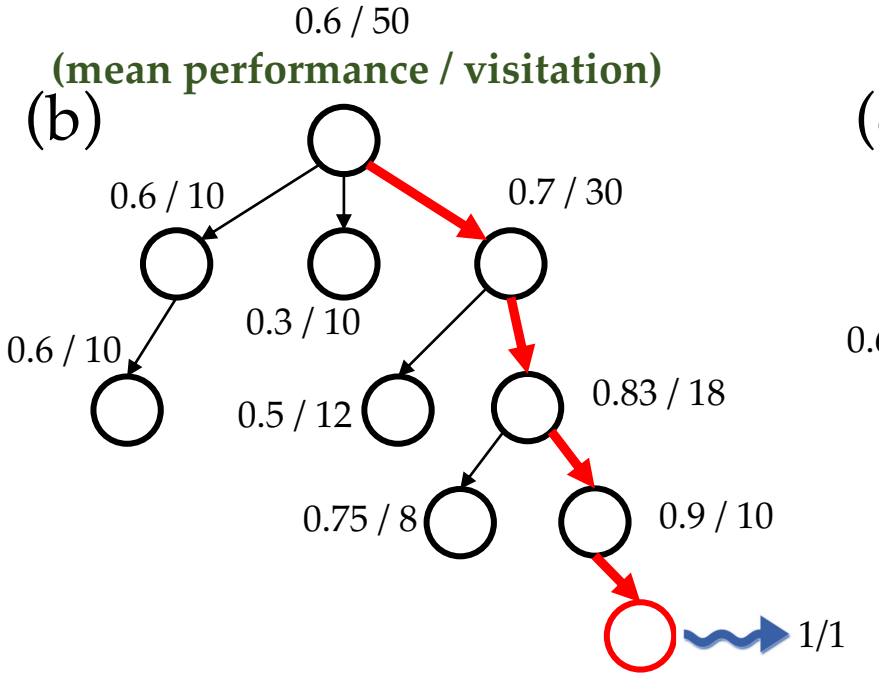
[L. Wang et al, Sample-Efficient Neural Architecture Search by Learning Action Space, arXiv]

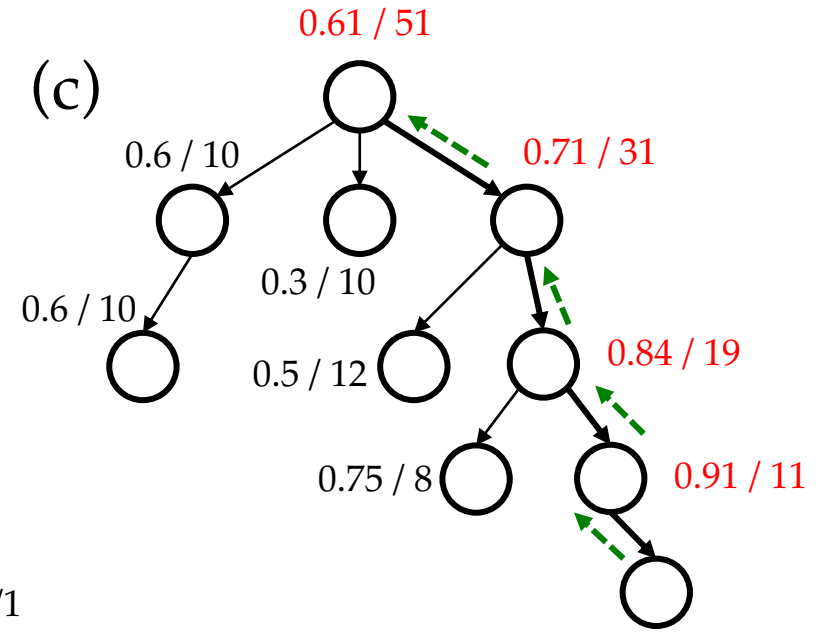
[L. Wang et al, Learning Search Space Partition for Black-box Optimization using Monte Carlo Tree Search, NeurIPS 2020]

Monte Carlo Tree Search

Search towards the good nodes while keeping exploration in mind





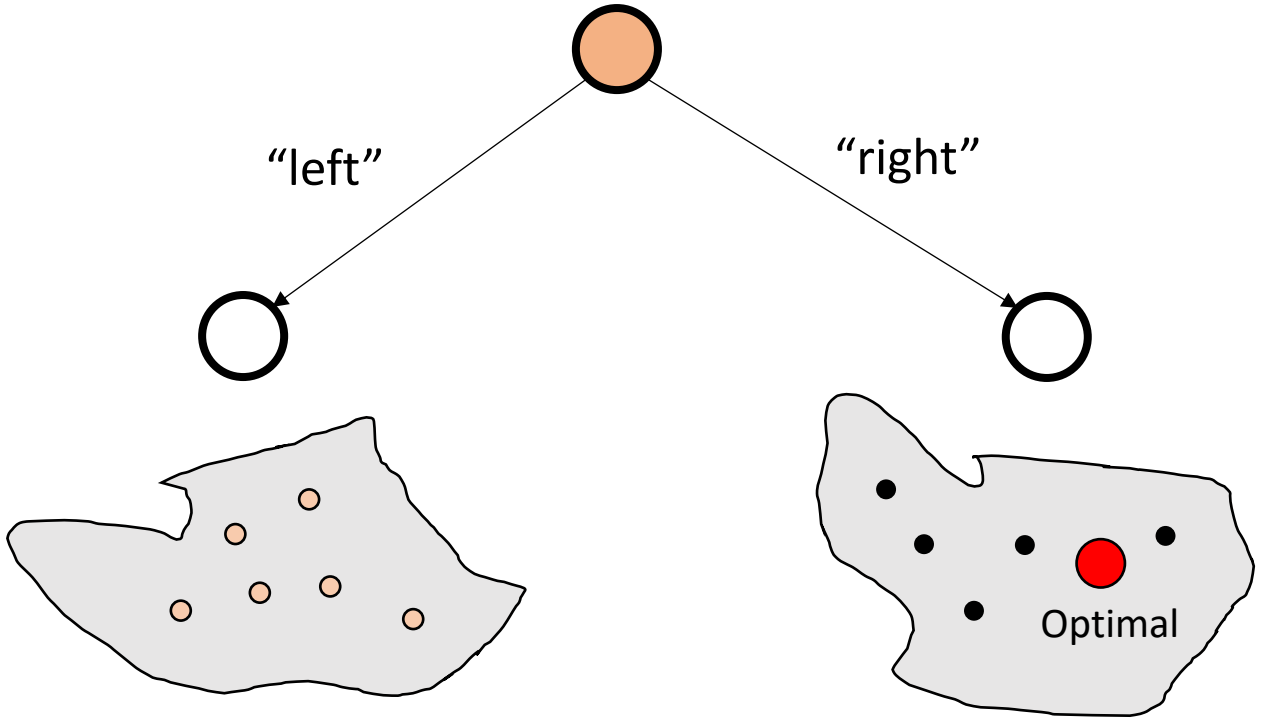


$$a_t = \arg\max_a Q(s_t, a) + u(s_t, a)$$

Exploration

$$u(s,a) = c_{\text{puct}} P(s,a) \frac{\sqrt{\sum_b N(s,b)}}{1 + N(s,a)}$$

Why Exploration is Important

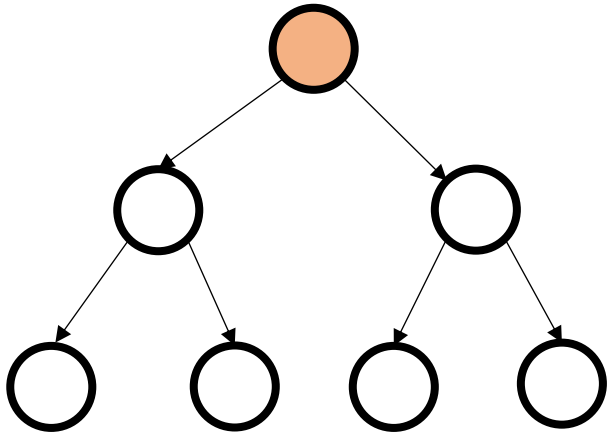


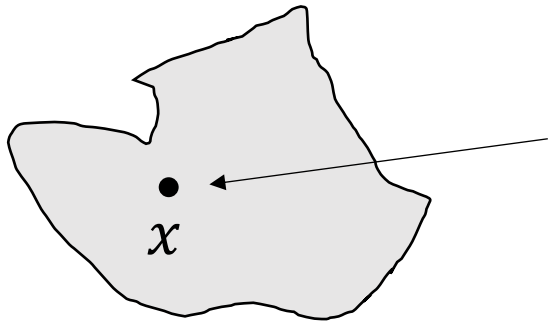
OK solutions
but not optimal

Most solutions are bad but there
exists an optimal one

- Bad solution
- OK solution
- Optimal solution

Sample in a Leaf





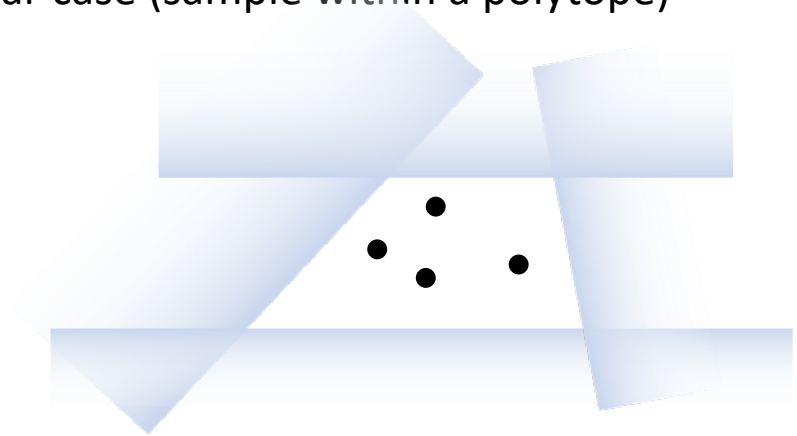
Random sample a point
Get the true function value

$$\boldsymbol{x} \rightarrow \boldsymbol{f}(\boldsymbol{x})$$

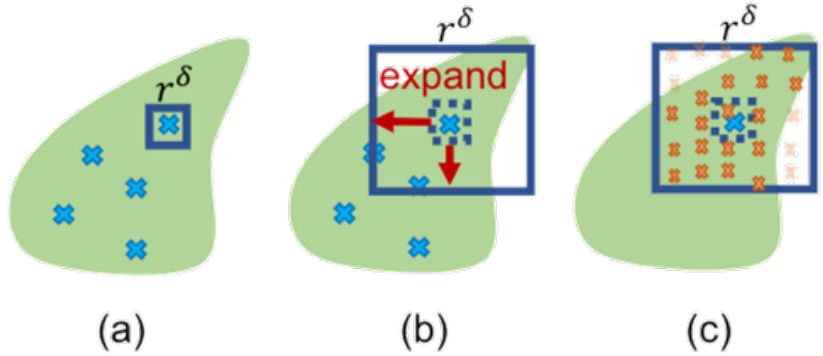
(can be expensive)

How?

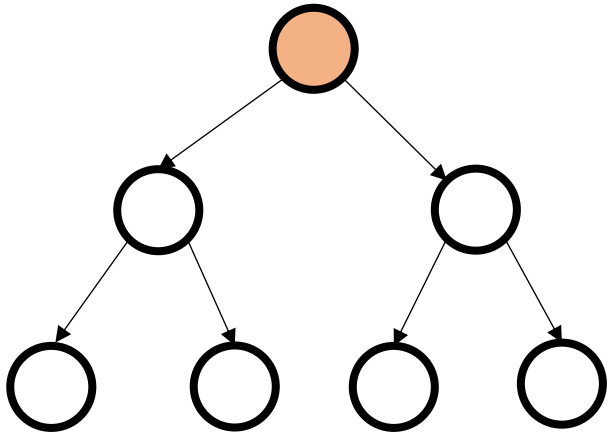
Linear case (sample within a polytope)

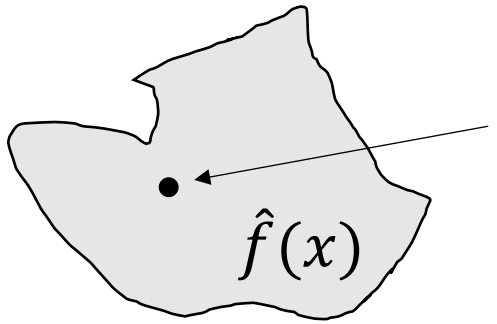


Nonlinear case (reject sampling)



Sample in a Leaf





Fit a function $\hat{f}(x)$ in this local region (e.g., Gaussian Process)

Find its maximal $\hat{x}^* = \arg\max \hat{f}(x)$

Our method is a Meta-Method

Only need to fit a **local** GP
(much lower sample complexity)

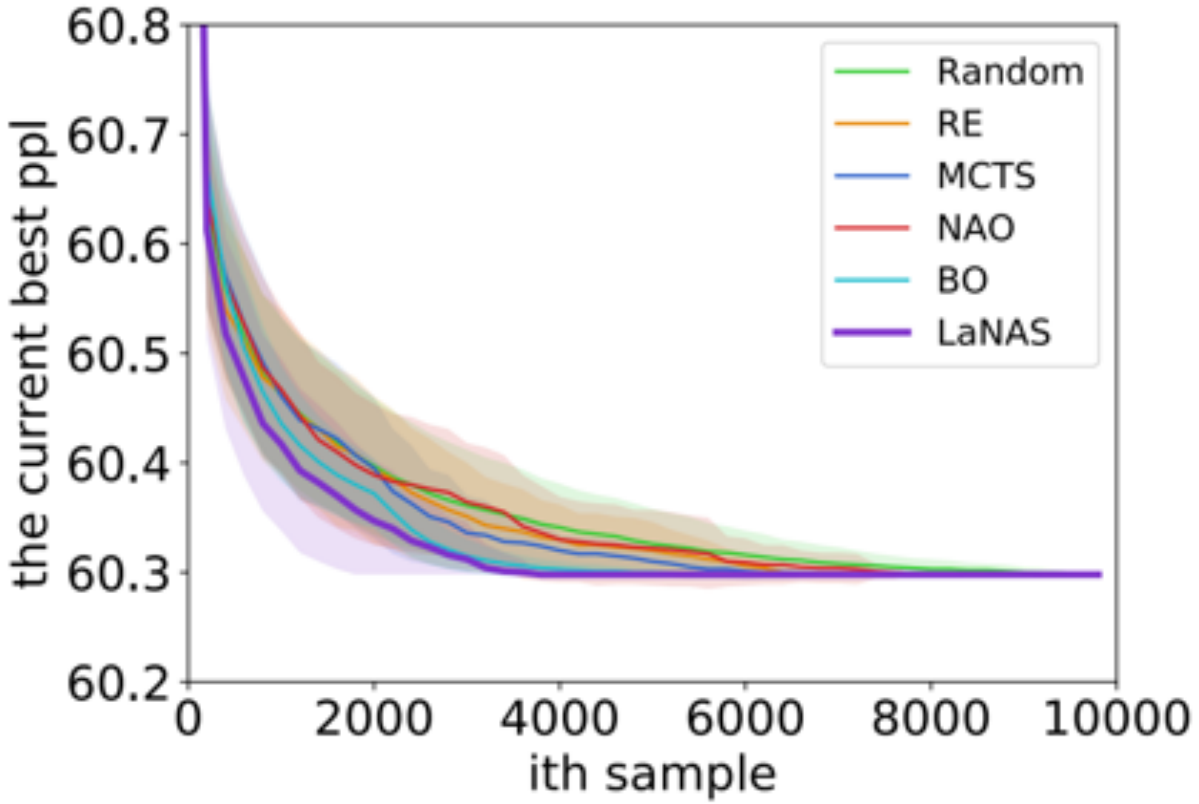
Function value is more **uniform** within a leaf
(easy for fitting)

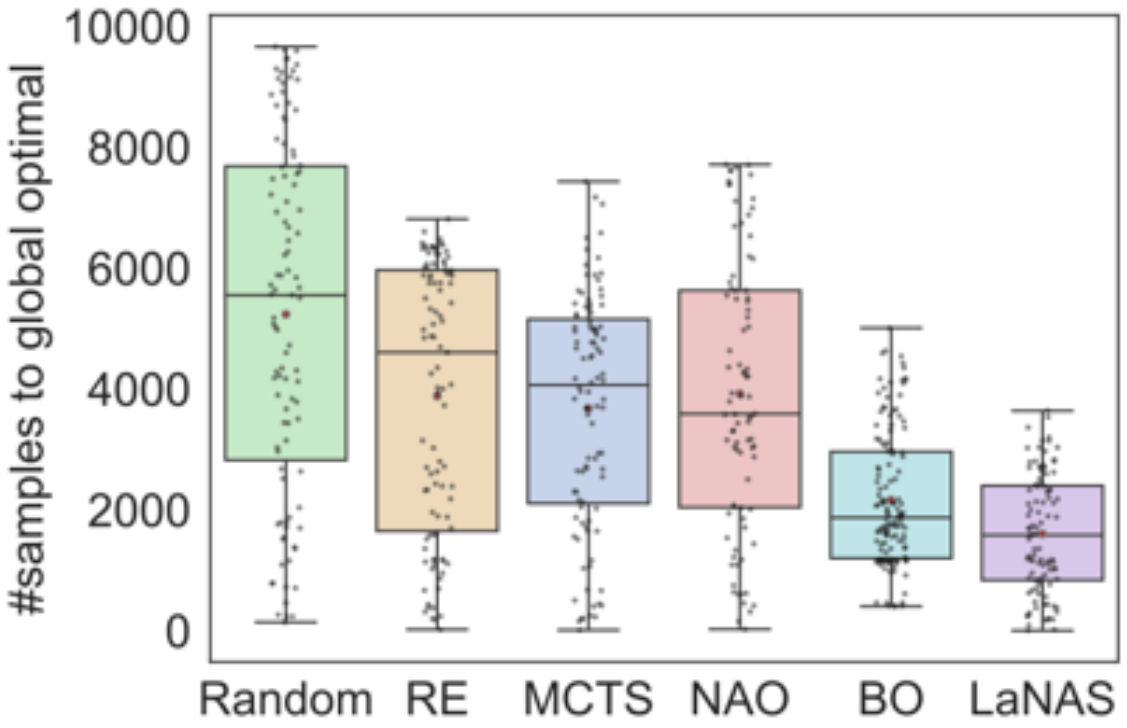
The fitting is needed only when #samples is small
Low computational cost

Accurate estimation of the leaf value
(uniform sample gives mean rather than max)

Performance

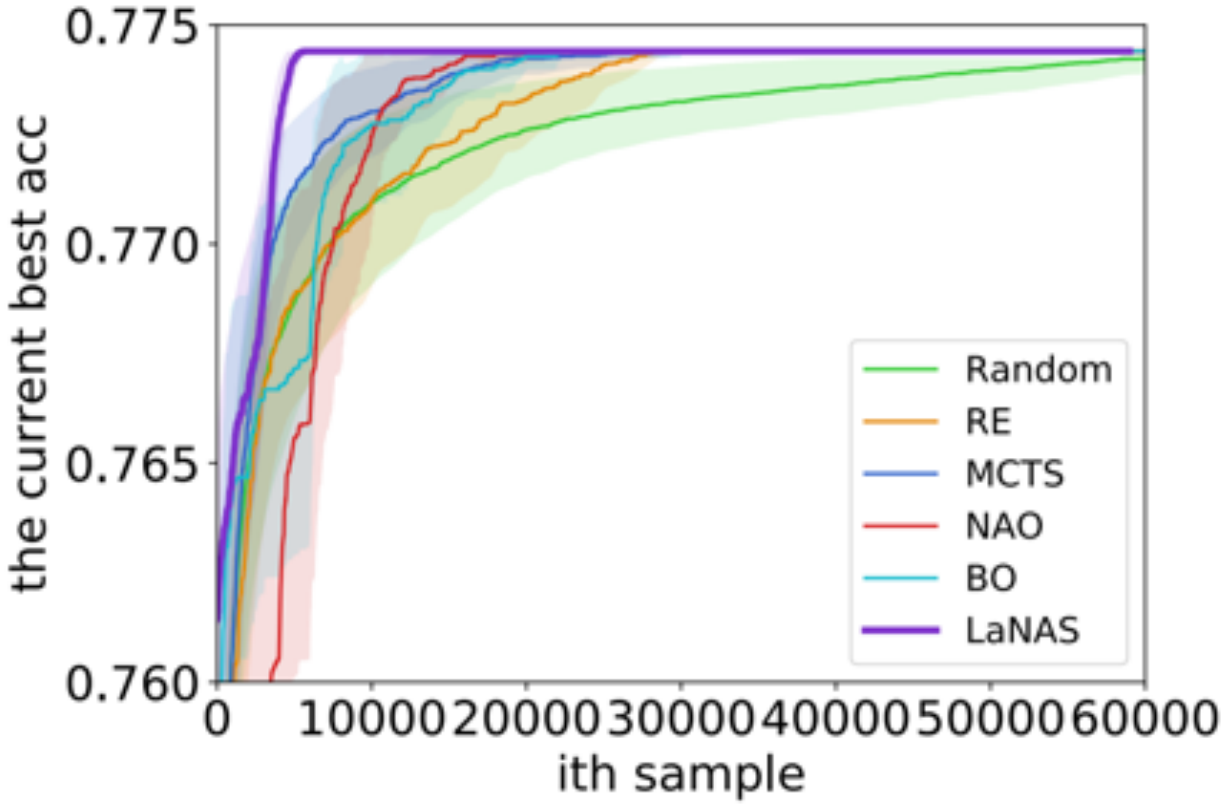
Customized dataset: LSTM-10K (PTB)

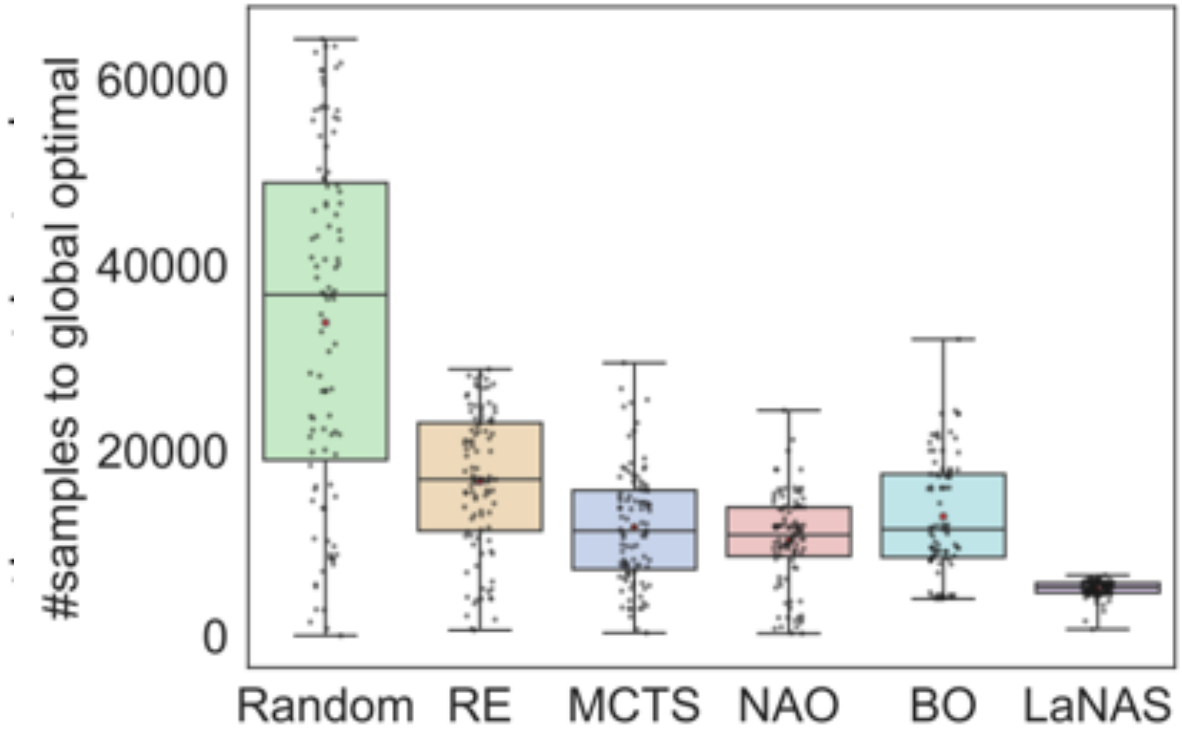




Performance

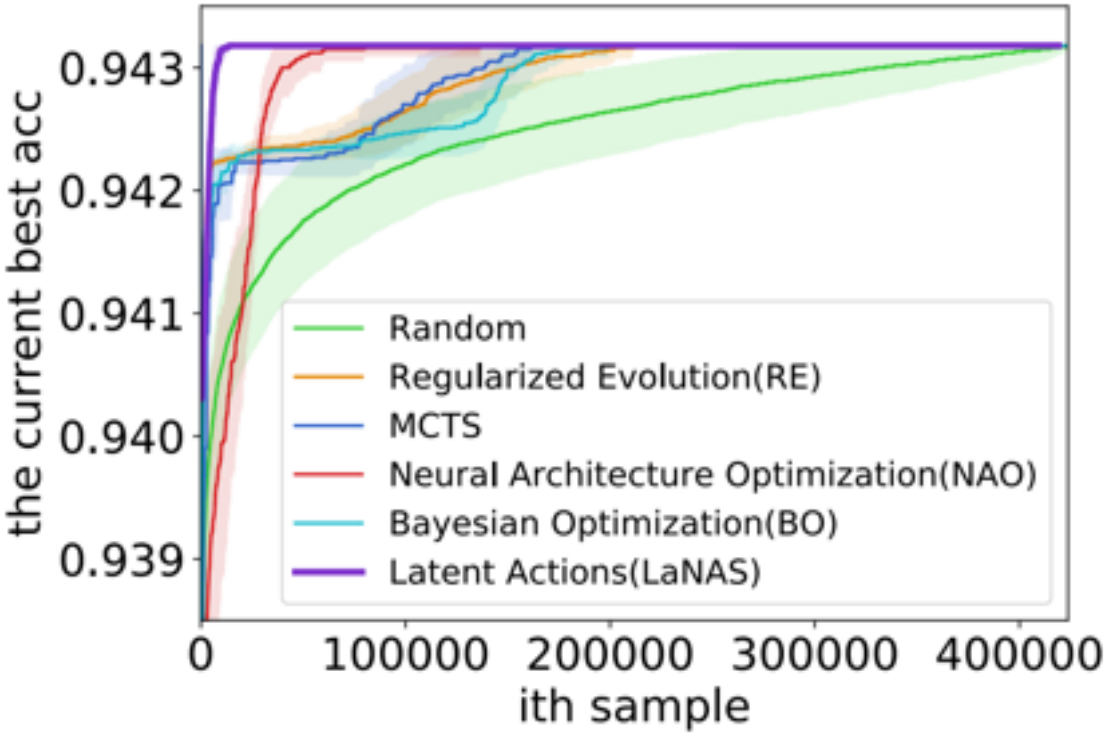
Customized dataset: ConvNet-60K (CIFAR-10, VGG style models)

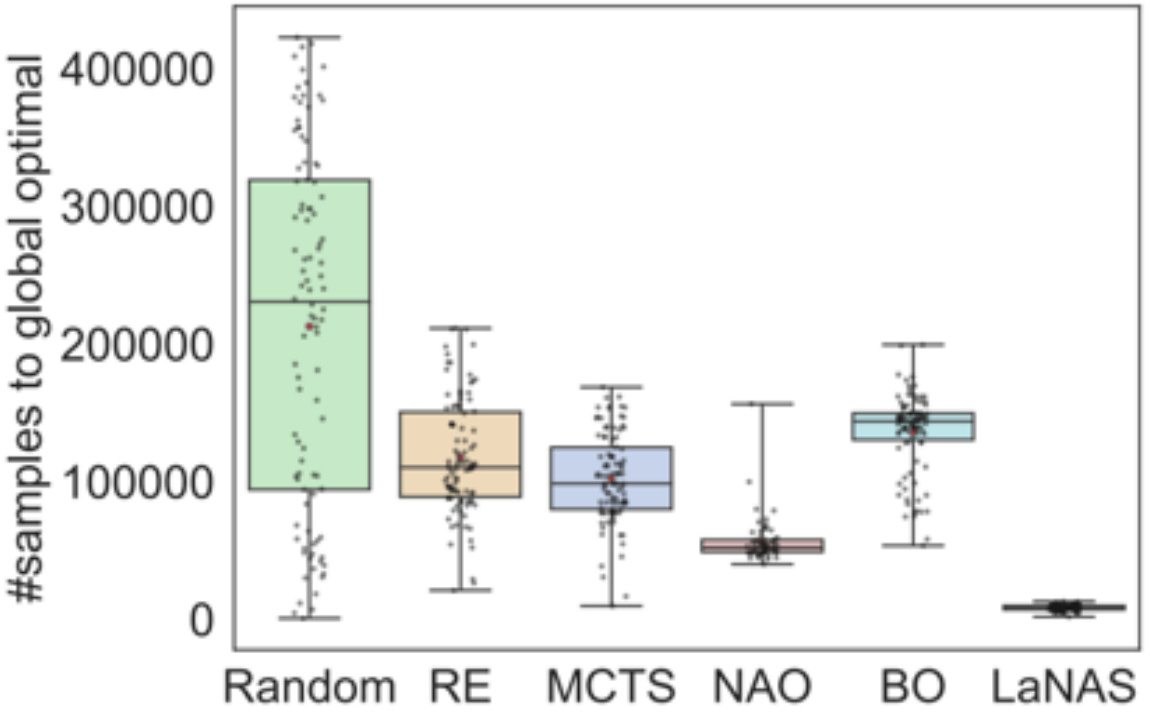




Performance

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

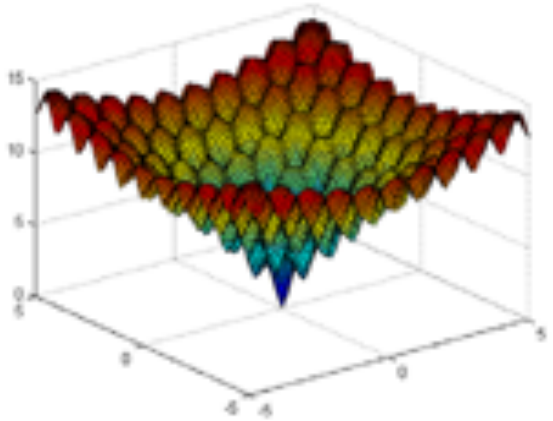
M: number of samples selected.

Open Domain

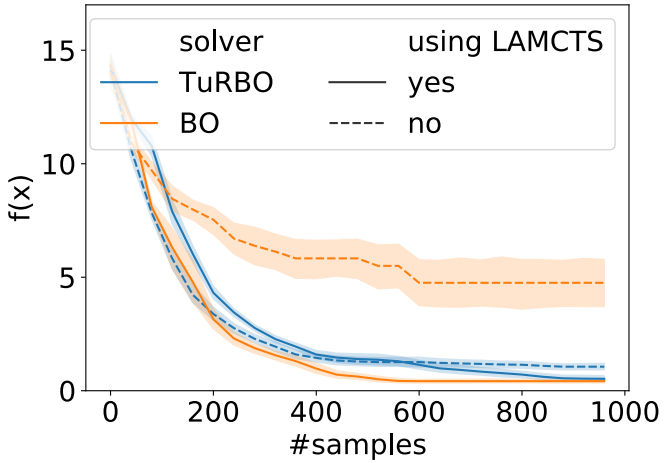
ImageNet
(mobile setting
Flop < 600M)

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**

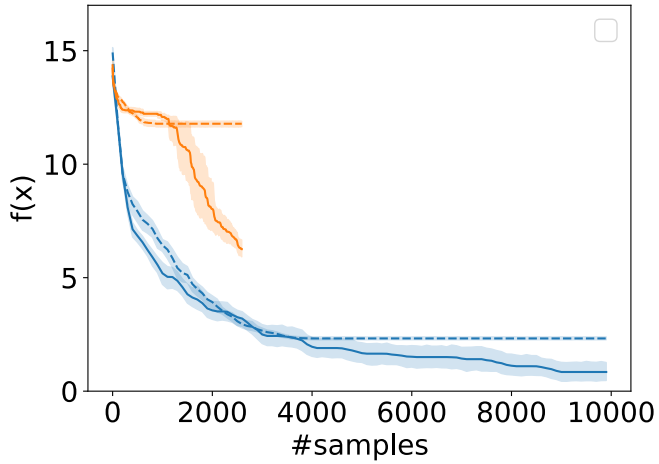
La-MCTS as a meta method $x^* = \arg\min_{x \in \Omega} f(x)$



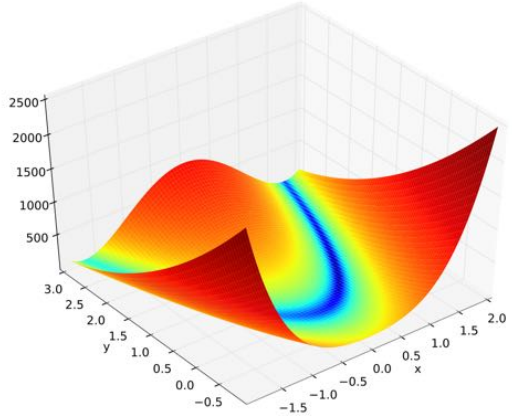
Ackley-2d



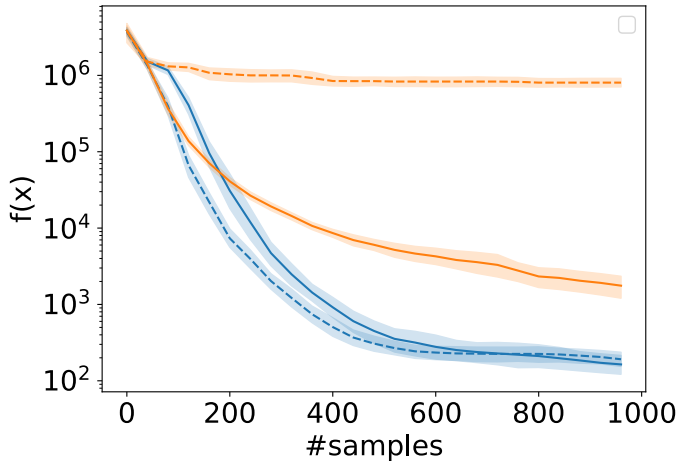
Ackley-20d



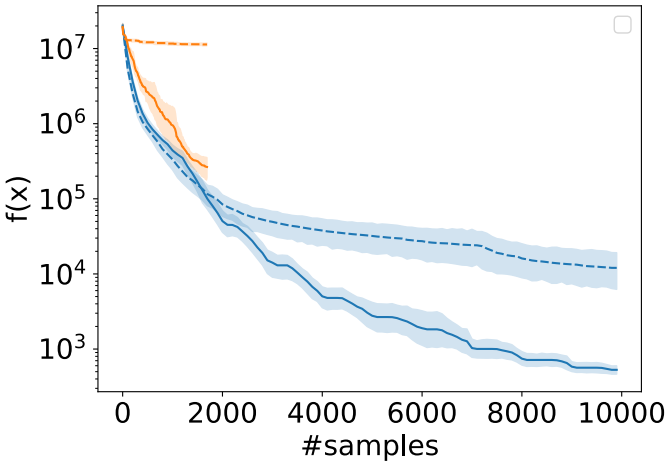
Ackley-100d



Rosenbrock-2d

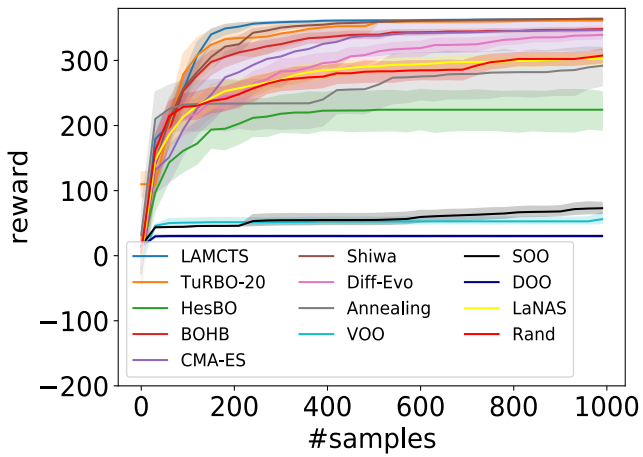


Rosenbrock-20d

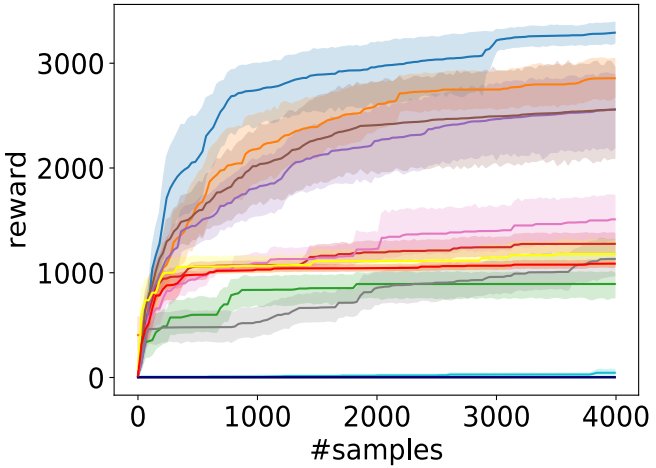


Rosenbrock-100d

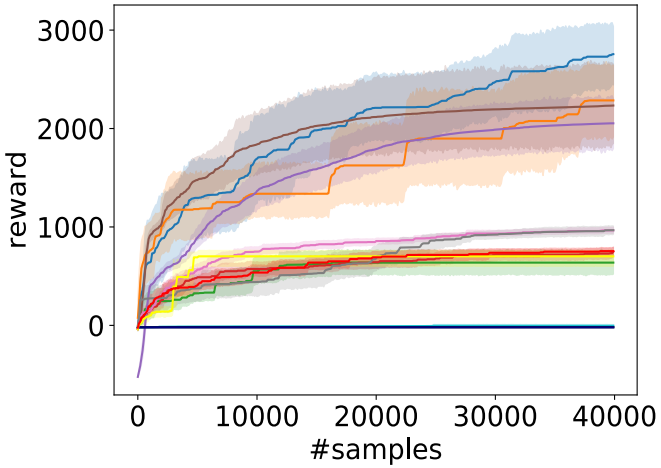
Optimizing linear policy for Mujoco tasks



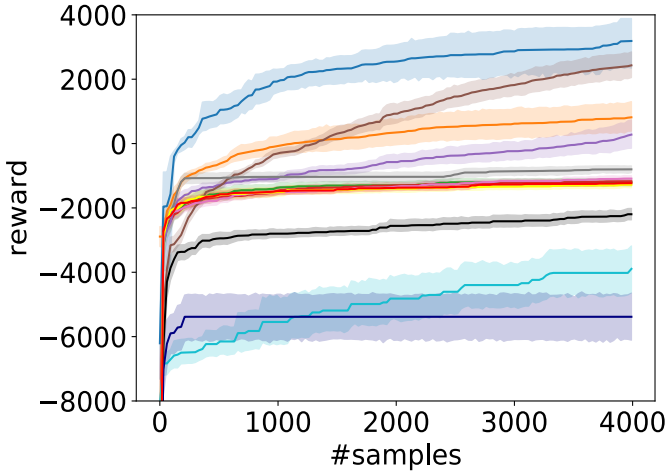
(a) Swimmer, #params = 16



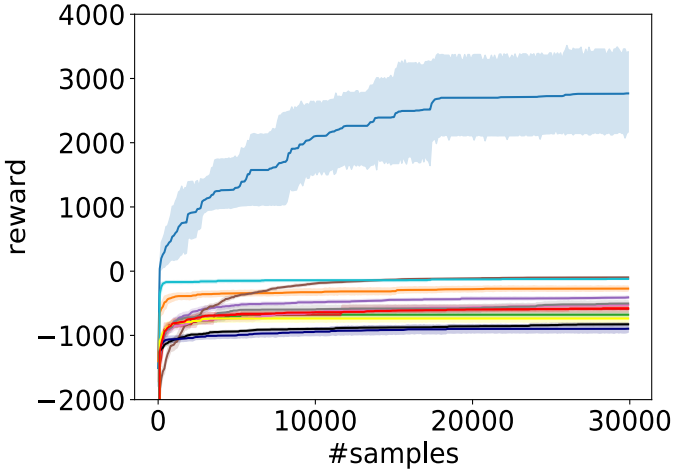
(b) Hopper, #params = 33



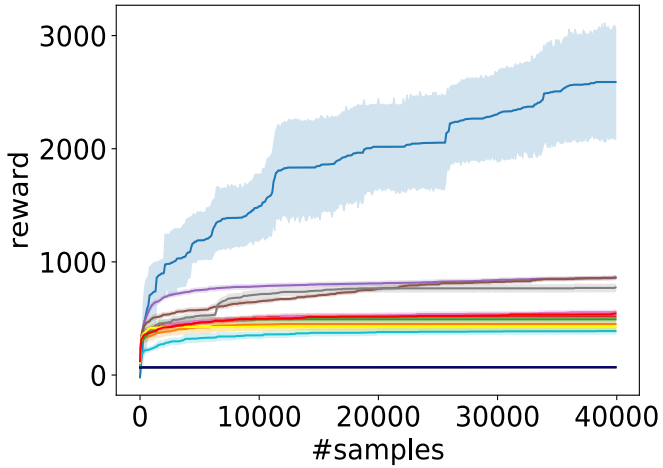
(c) Walker-2d, #params = 102



(d) Half-Cheetah, #params =102



(e) Ant, #params = 888



(f) Humanoid, #params = 6392

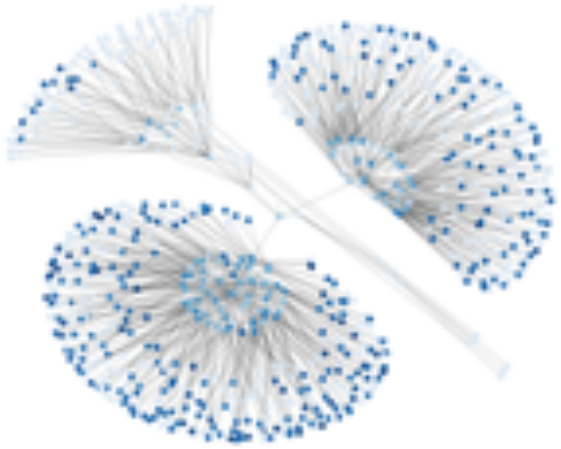
Limitations

Task	Reward Threshold	The average episodes (#samples) to reach the threshold				
		LA-MCTS	ARS V2-t [54]	NG-lin [55]	NG-rbf [55]	TRPO-nn [54]
Swimmer-v2	325	**132**	427	1450	1550	N/A
Hopper-v2	3120	2897	1973	13920	8640	10000
HalfCheetah-v2	3430	3877	1707	11250	6000	4250
Walker2d-v2	4390	N/A($r_{best} = 3314$)	24000	36840	25680	14250
Ant-v2	3580	N/A($r_{best} = 2791$)	20800	39240	30000	73500
Humanoid-v2	6000	N/A($r_{best} = 3384$)	142600	130000	130000	unknown

N/A stands for not reaching reward threshold.
r_{best} stands for the best reward achieved by LA-MCTS under the budget in Fig. 3.

Too many explorations might hurt in Mujoco tasks.

Code is public now!



LA-MCTS

https://github.com/facebookresearch/LaMCTS

Thanks!