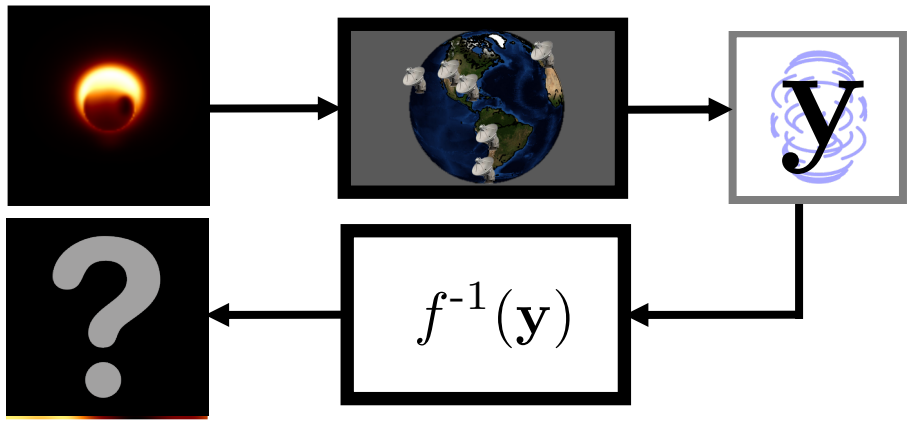
Caltech

Learning to Optimize as Policy Learning

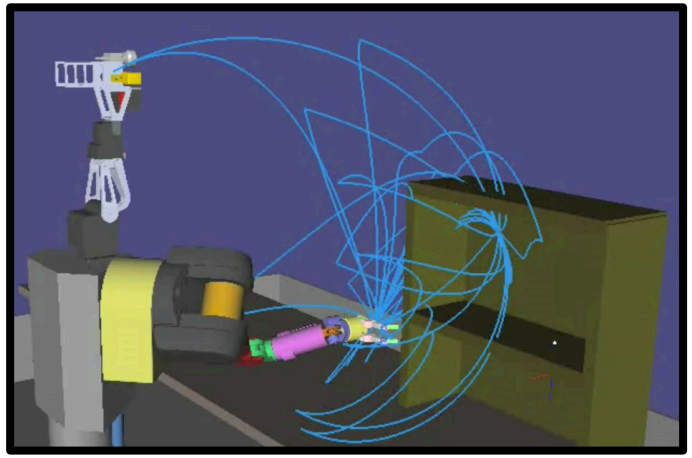
Yisong Yue

Optimization is a Fundamental Challenge

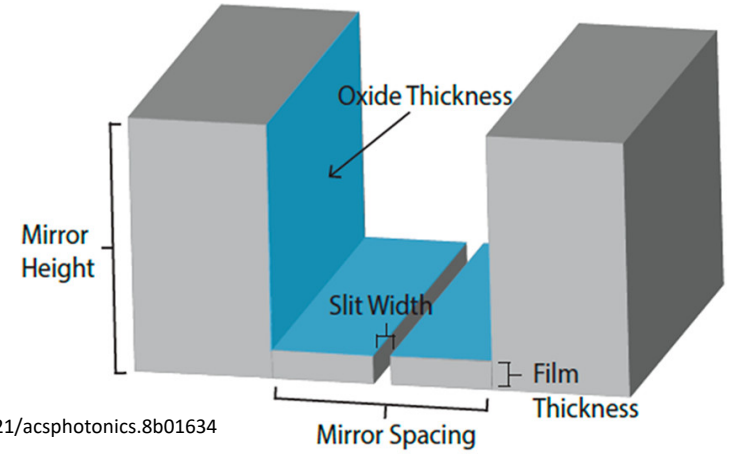
Inverse Problems



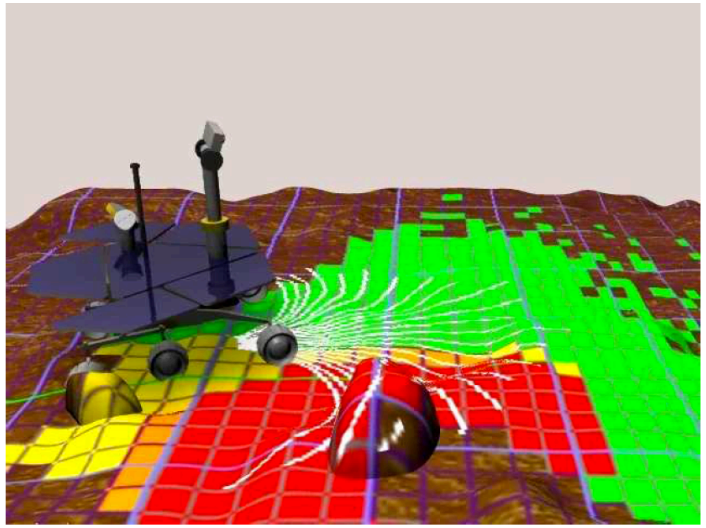
Planning



Design Tasks



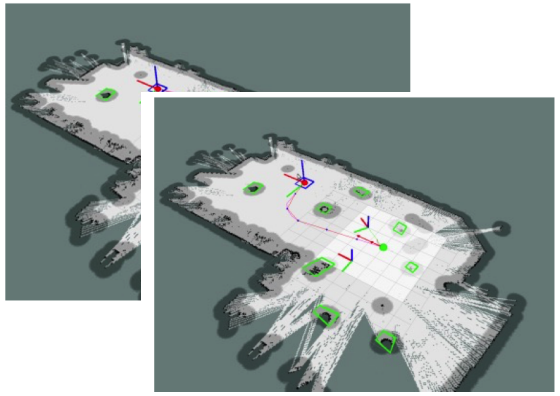
https://doi.org/10.1021/acsphotonics.8b01634



Optimization is Hard!

- High Dimensional / Combinatorial
- Real-Time Resource Constraints
- Poorly Conditioned / Poorly Initialized
- Tuning of Optimizers

Learning to Optimize



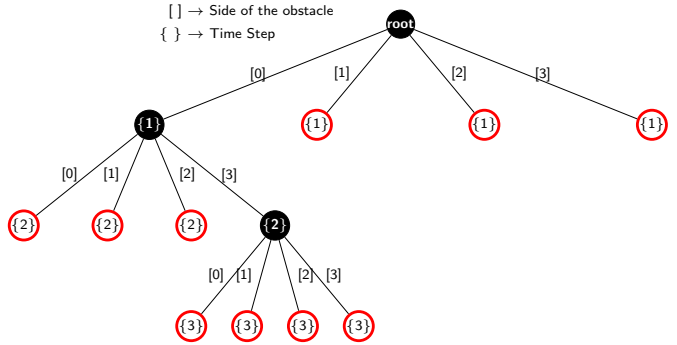
Distribution of Planning Problems

$$\min_{\mathbf{U}} \quad J(\mathbf{U}, \mathbf{X})$$

subject to,

(Dynamic Constraint) $x_{t+1} = Ax_t + Bu_t,$

(Safety Constraints) $h_t^{iT} x_t \leq g_t^i$



Compiled as Combinatorial Search Problems

- Many Solvers are Sequential
  - Tree-Search / Greedy
  - Gradient / Coordinate Descent
- **Can view solver as “agent” or “policy” making decisions**

Many Other People Work on this Topic!

CSC 2547 Fall 2019: Learning to Search

Overview

In planning, search, active learnir
face a series of similar tasks. We
even combine parts of previous s
foundational ideas, recent work,
self-improving tree-search meth
self-tuning gradient estimators, a
planning in POMDPs. Evaluation
research by the students. Studer
learning such as linear algebra, o

RESEARCH HIGHLIGHTS

Data-Driven Algorithm Design

By Rishi Gupta, Tim Roughgarden
Communications of the ACM, June 2020, Vol. 63 No. 6, Pages 87-94
10.1145/3394625
Comments

VIEW AS: SHARE:

Credit: Getty Images

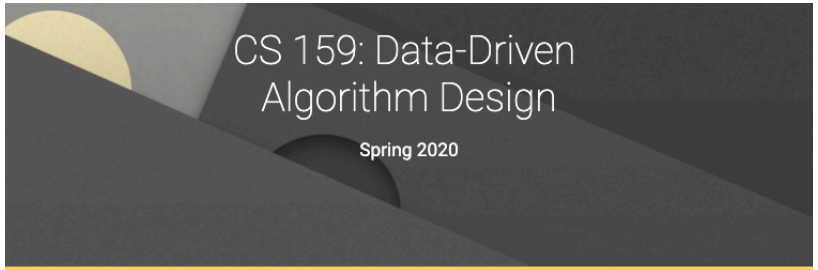
The best algorithm for a comput
depends on the "relevant inputs,
application domain and often de
Although there is a large literature on empirical approaches to selecting the best algorithm for a given application domain, there has been surprisingly little theoretical analysis of the problem.

We model the problem of identifying a good algorithm from data as a statistical learning problem. Our framework captures several state-of-the-art empirical and theoretical approaches to the problem, and our results identify conditions under which these approaches are guaranteed to perform well. We interpret our results in the contexts of learning greedy heuristics, instance feature-based algorithm selection, and parameter tuning in machine learning.

Workshop on Automated Algorithm Design

Aug 7 — Aug 9, 2019
Toyota Technological Institute

This workshop focuses on new machine learning techniques for automatically designing algorithms. Algorithms are central to modern computing, and they have lots of applications in our life. Yet, writing correct, efficient algorithms is a time-consuming and difficult task. It also often requires intuition and expertise to tailor algorithmic choices to specific instances that arise in particular appli
However, there have been a number
advancements that have allowed algo
selected or designed from specific al
families automatically, often leading to
of-the-art empirical performance or pr
performance guarantees on observed
distributions. In this workshop, we tak
view of the problem and seek to bring
researchers with different viewpoints a
approaches to the general challenge.



What is Data-Driven Algorithm Design?

Over the past several years, machine learning is increasingly used to (semi-)automatically
ems. Canonical examples include:
.
-bound.
r than gradient descent) for continuous
rs to get the best performance.
workshop.
order to learn well. There are two ways
s to learn from, and assume that future
ning instances.
stance.

Learning Meets Combinatorial Algorithms (LMCA)

Workshop at NeurIPS 2020

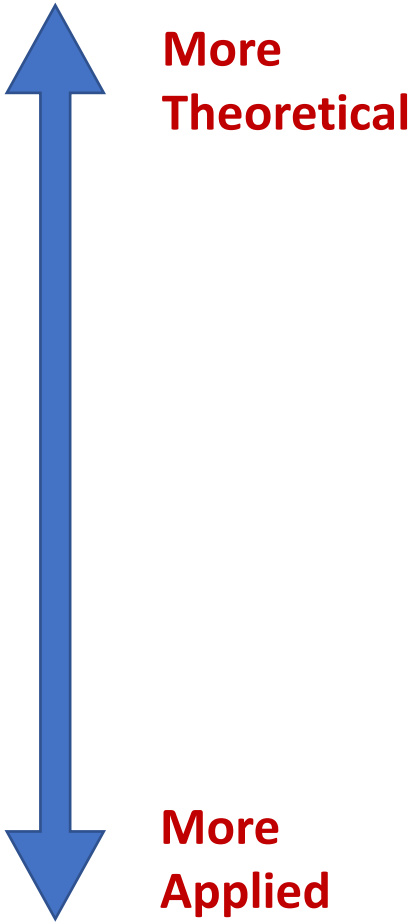
Why this Workshop?

Machine learning algorithms have been shown to generalize poorly on combinatorially demanding tasks. Recent research has demonstrated that merging combinatorial optimization with machine learning methods enables solving problems that require non-trivial combinatorial generalization beyond pattern matching. In this spirit, this workshop aims to bring the communities (machine learning and combinatorial optimization, operations research) together in order to motivate further research at the intersection. This involves:

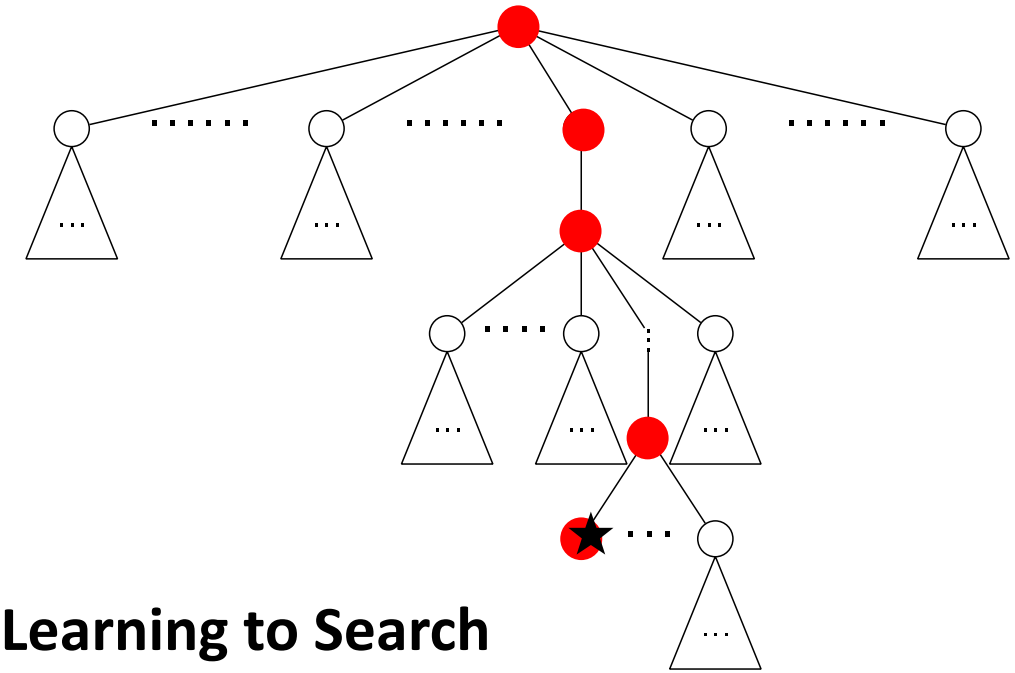
- Machine learning approaches aimed at improving combinatorial algorithms/solvers.
- Machine learning techniques to directly learn solvers for combinatorial problems.
- Hybrid architectures; pipelines containing both algorithmic/combinatorial and standard NN building blocks.
- Applications of the above.

Research Questions

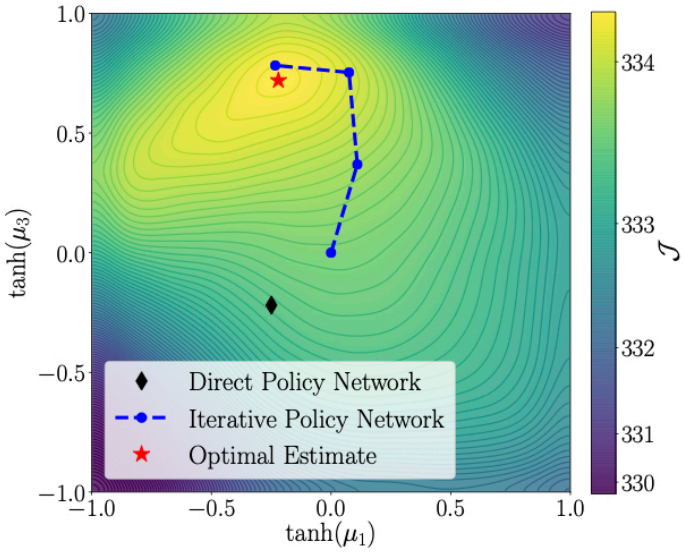
- How do we formalize the learning problem?
  - Is the formulation novel or standard?
- Do we need new algorithms?
- How do we measure progress?
  - Speedup in wall-clock time?
  - Savings in memory?
  - Final solution quality?
- Can we integrate into real systems?



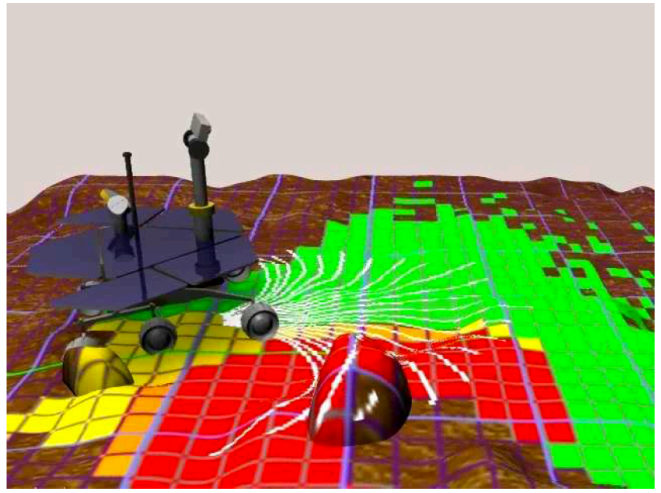
This Talk



Learning to Search



Continuous Optimization



Practical Applications

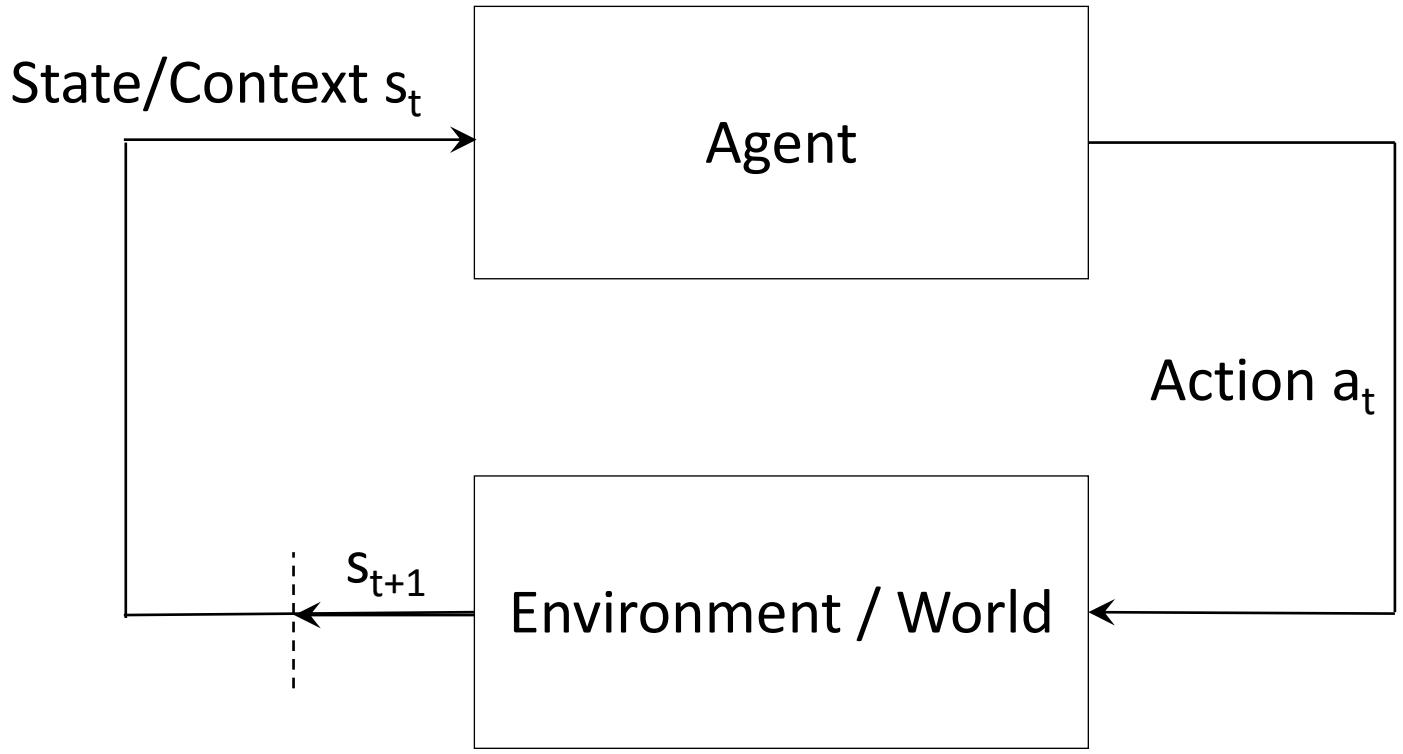
Policy Learning (Reinforcement & Imitation)

Goal: Find "Optimal" Policy

Imitation Learning:
Optimize imitation loss

Reinforcement Learning:
Optimize environmental reward

Learning-based Approach for Sequential Decision Making



Imitation Learning Tutorial (ICML 2018)

https://sites.google.com/view/icml2018-imitation-learning/

Yisong Yue

yyue@caltech.edu

@YisongYue

yisongyue.com

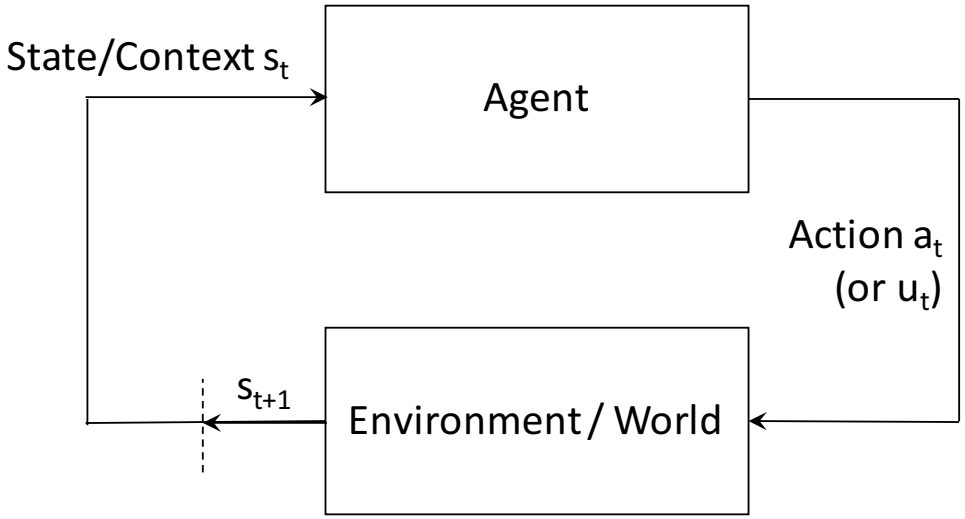
Hoang M. Le

hmle@caltech.edu

@HoangMinhLe

hoangle.info

Basic Formulation



(Typically a Neural Net)

- Policy: $\pi(s) \rightarrow P(a)$

State Action

- Roll-out: $\tau = \langle s_0, a_0, s_1, a_1, s_2, \ldots \rangle$ (aka trace or trajectory)

Transition Function: P(s'|s,a)

- Objective: $\sum_i r(s_i, a_i)$

Example: Learning to Search

Integer Program

$$\max \; -\sum_{i=1}^{5} x_i,$$

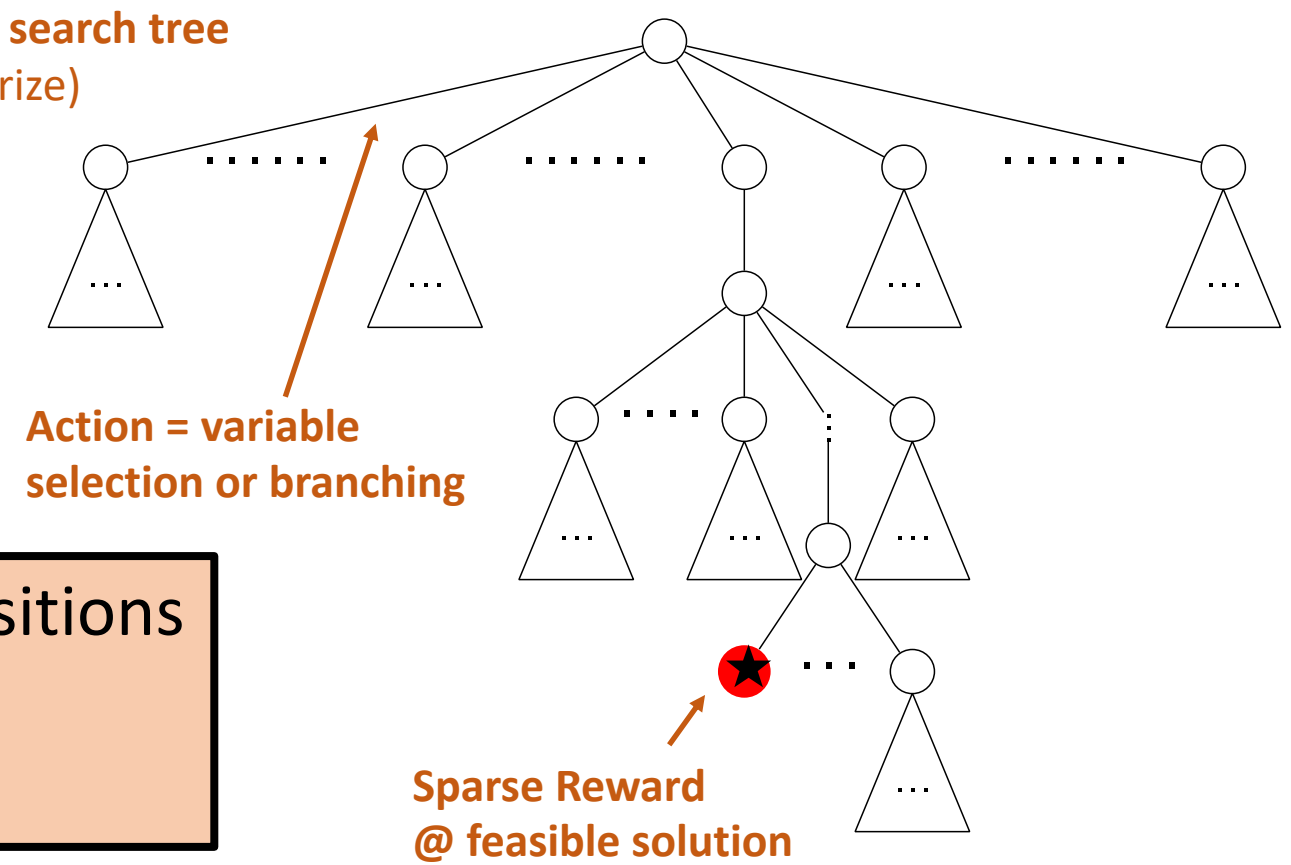
subject to:

$$x_1 + x_2 \geq 1,$$

$$x_2 + x_3 \geq 1,$$

$$x_3 + x_4 \geq 1,$$

Tree-Search (Branch and Bound)

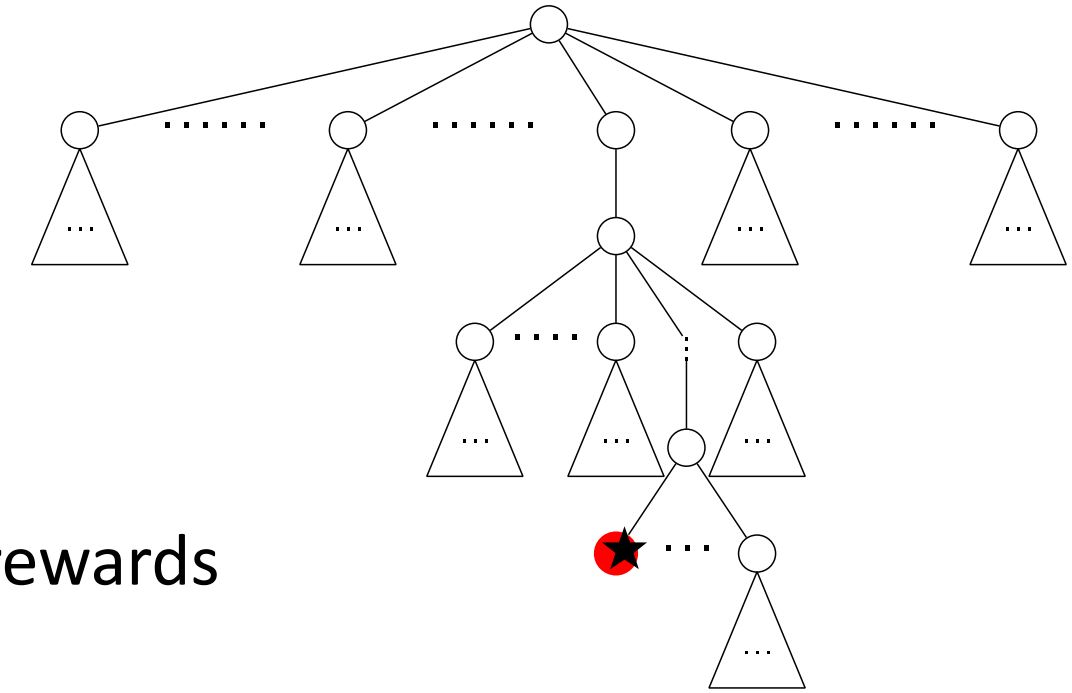


- Deterministic State Transitions
- Massive State Space
- Sparse Rewards

[He et al., 2014][Khalil et al., 2016] [Song et al., arXiv]

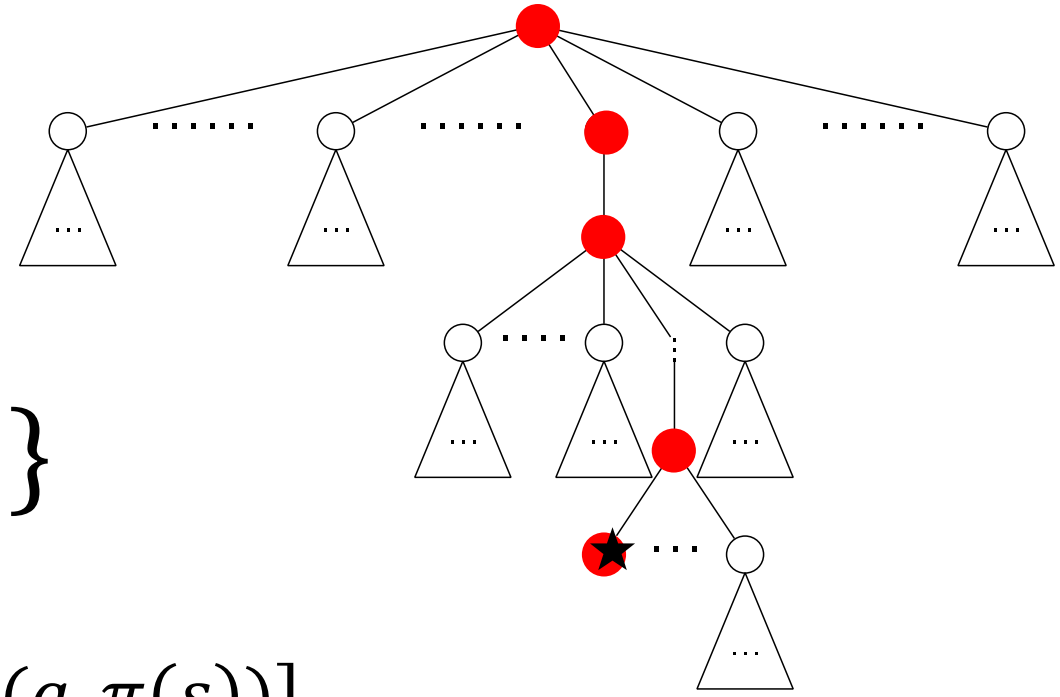
Learning to Optimize for Tree Search

- Idea #1: Treat as Standard RL
- Randomly explore for high rewards
  - **Very hard exploration problem!**
- Issues: massive state space & sparse rewards

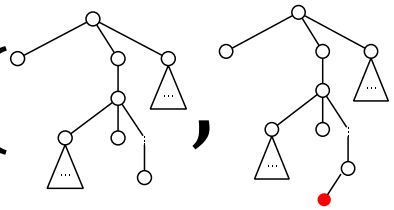


Learning to Optimize for Tree Search

- Idea #2: Treat as Standard IL
- Convert to Supervised Learning
  - Assume access to solved instances



"Demonstration Data"

- Training Data: $D_0 = \{($  $)\}$

- Basic IL: $\underset{\pi \in \Pi}{\operatorname{argmin}} L_{D_0}(\pi) \equiv E_{(s,a)\sim D_0}[\ell(a, \pi(s))]$

Behavioral Cloning

Challenges w/ Imitation Learning

- Issues with Behavioral Cloning
  - Minimize L_{D_0} … implications?
  - If π makes a mistake early, subsequent state distribution ≈ D_0 ??
  - Some extensions to Interactive IL [He et al., NeurIPS 2014]

Our Approach is also Interactive IL

- Demonstrations not Available on Large Problems
  - How to (formally) bootstrap from smaller problems?
  - **Bridging the gap between IL & RL**

Our Approach gives one solution

Retrospective Imitation
(Bridging IL & RL)

Jialin Song

Ravi Lanka

- Given:
  - Family of Distributions of Search problems
    - Family is parameterized by size/difficulty
  - Solved Instances on the Smallest/Easiest Instances
    - “Demonstrations”

Difficulty levels: k=1,…,K

- Goal:
  - Interactive IL approach
  - Can Scale up from Smallest/Easiest Instances
  - Formal Guarantees

Connections to Curriculum Learning & Transfer Learning

Learning to Search via Retrospective Imitation, Jialin Song, Ravi Lanka, et al., arXiv

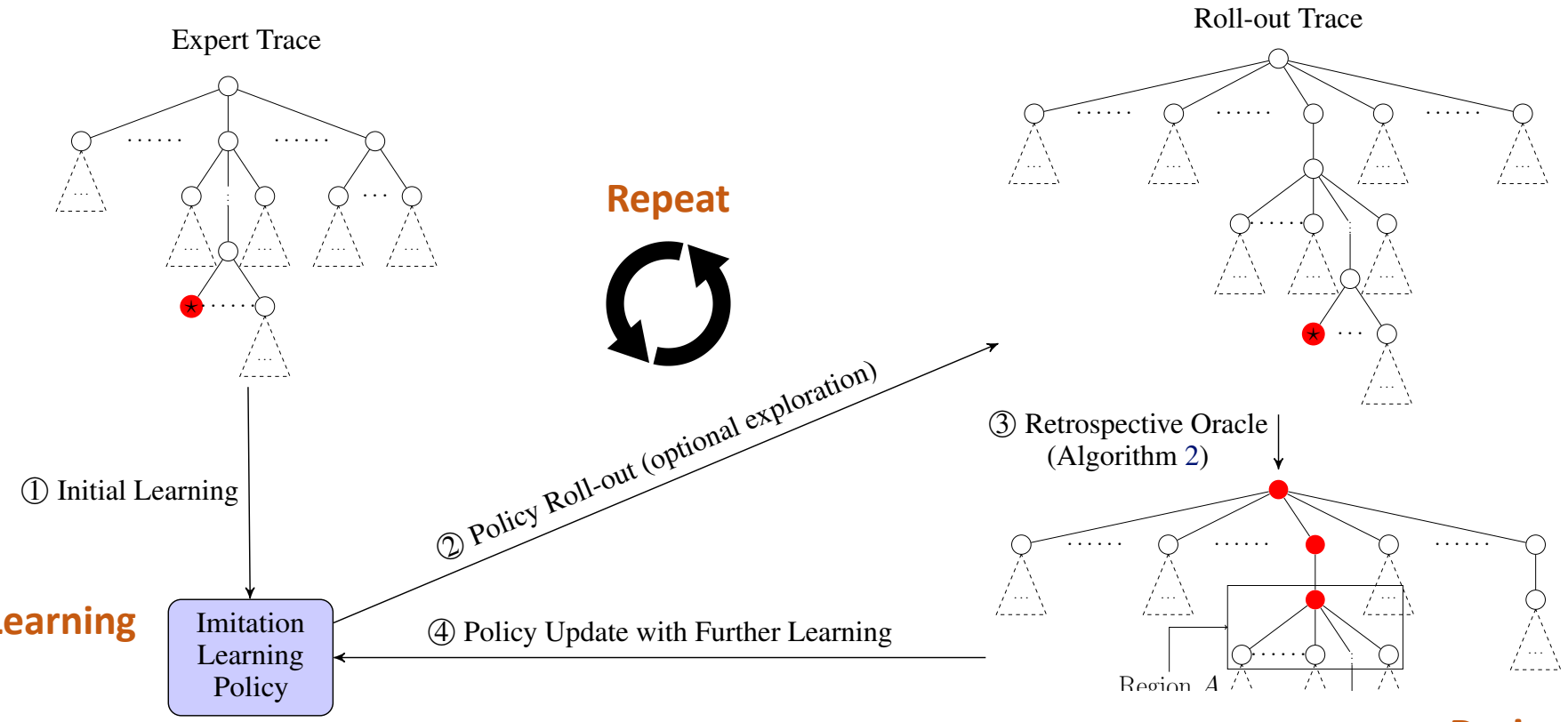
Retrospective Imitation

- Two-Stage Algorithm
- Core Algorithm
  - Fixed problem difficulty
  - Reductions to Supervised Learning
- Full Algorithm w/ Scaling Up
  - Uses Core Algorithm as Subroutine

Interactive IL w/ Sparse Environmental Rewards

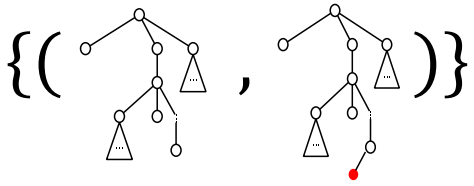
Learning to Search via Retrospective Imitation, Jialin Song, Ravi Lanka, et al., arXiv

Retrospective Imitation (Core Algorithm)



Supervised Learning Reduction

State s Action a



Derived from Sparse Environmental Rewards

Learning to Search via Retrospective Imitation, Jialin Song, Ravi Lanka, et al., arXiv

Retrospective Imitation (Full Algorithm)

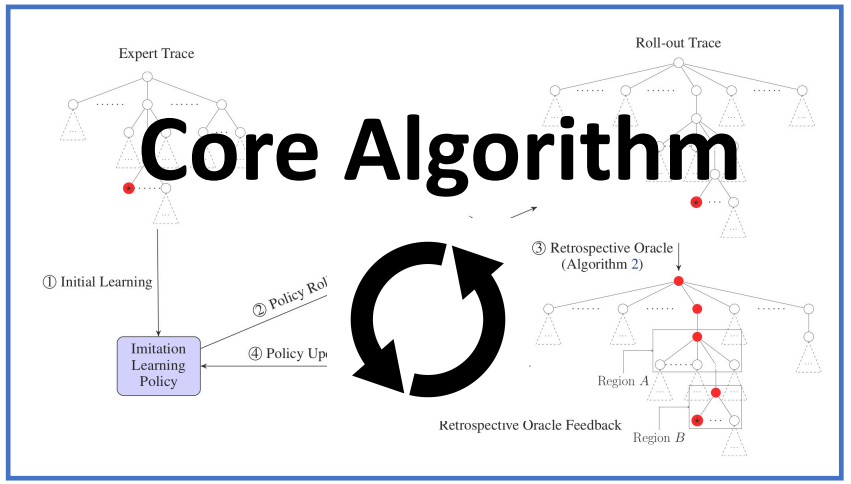
Initialize k=1

Initialize Gurobi/SCIP/CPlex

Problem Difficulty k

Base Solver

Instances & Demonstrations



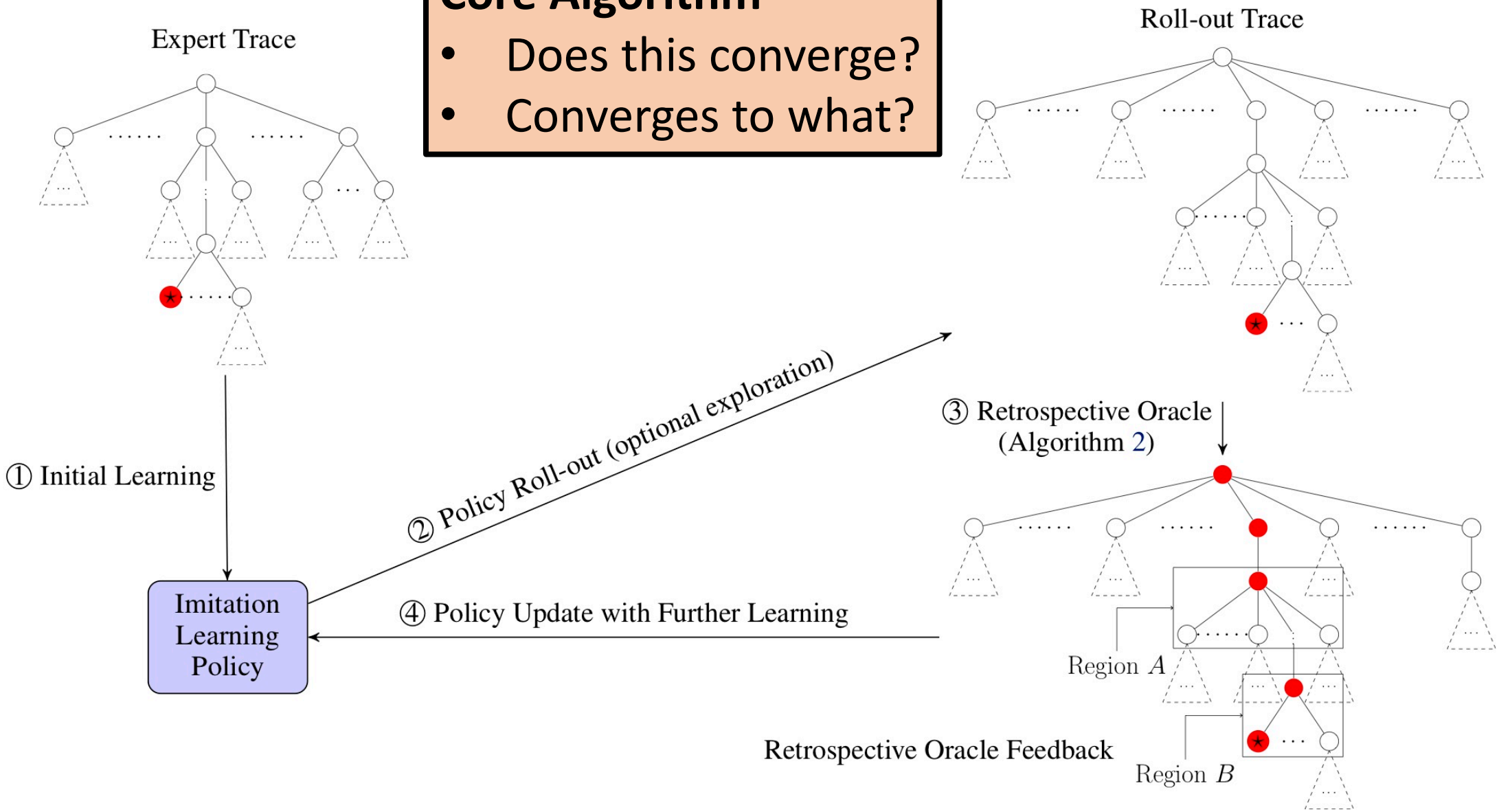
k=k+1

Use trained π

Learning to Search via Retrospective Imitation, Jialin Song, Ravi Lanka, et al., arXiv

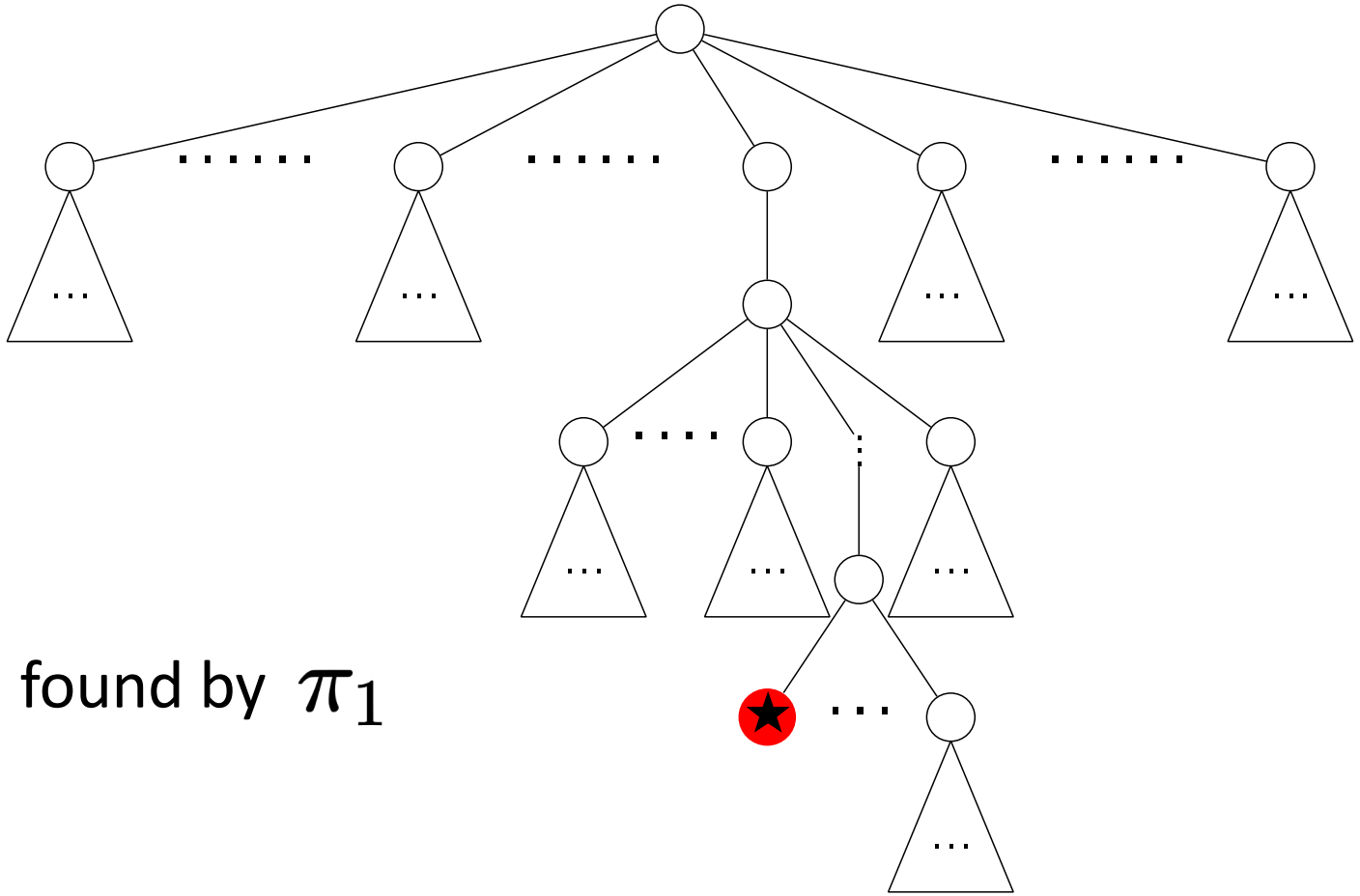
Core Algorithm

- Does this converge?
- Converges to what?



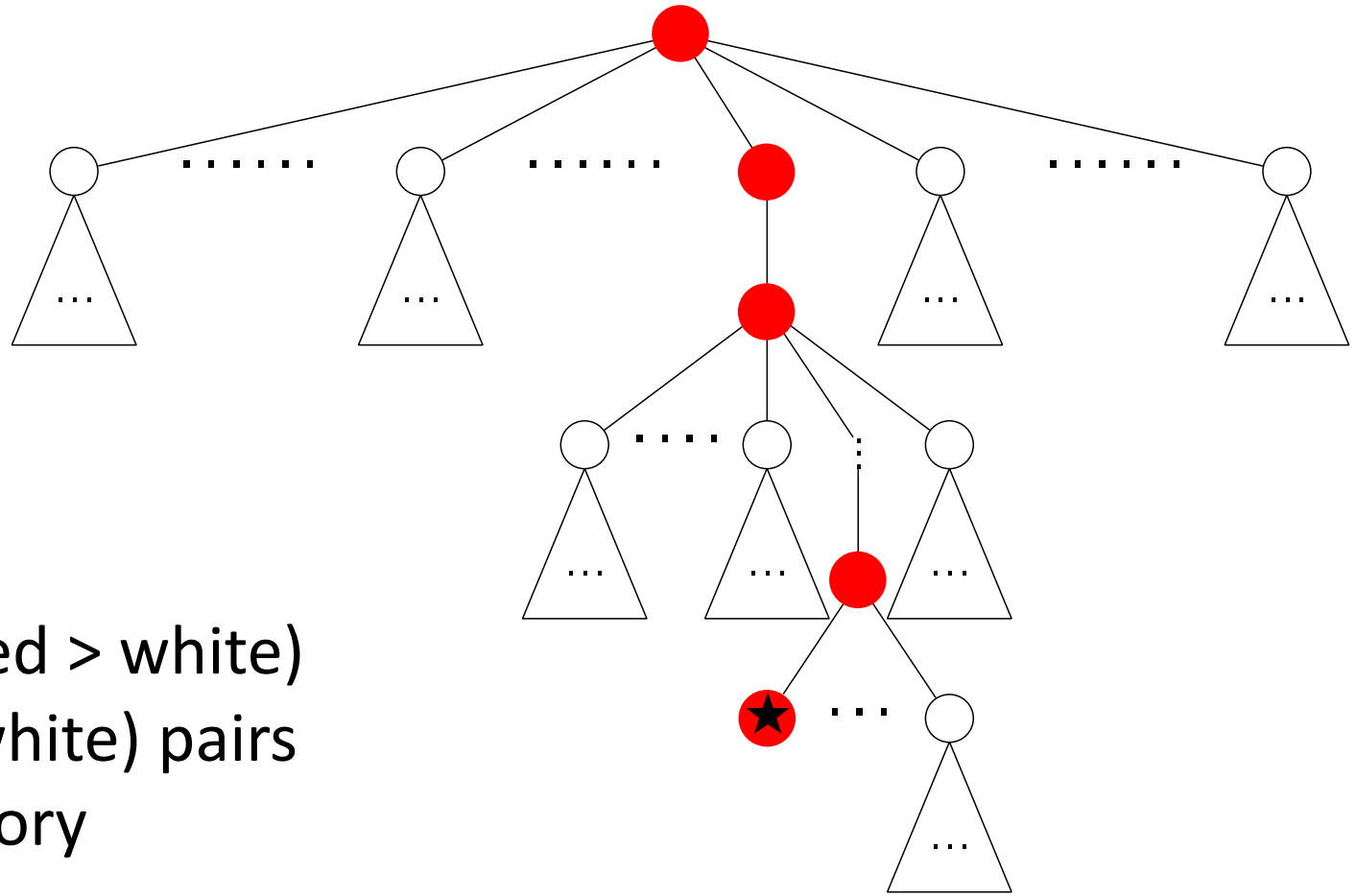
Learning to Search via Retrospective Imitation, Jialin Song, Ravi Lanka, et al., arXiv

π_1 Policy Rollout



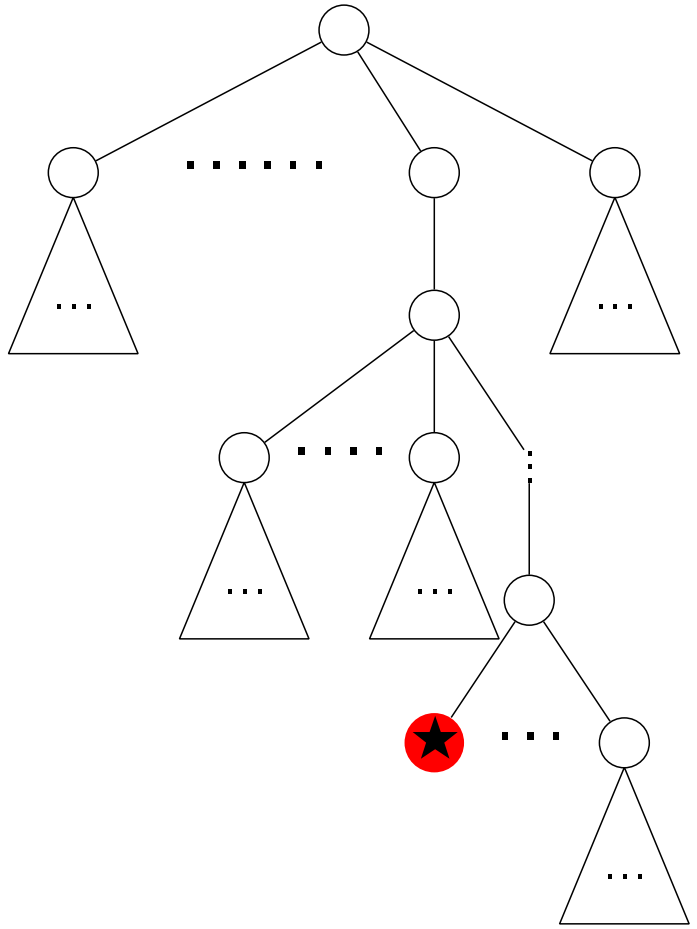
★: best solution found by π_1

Retrospective Oracle Feedback

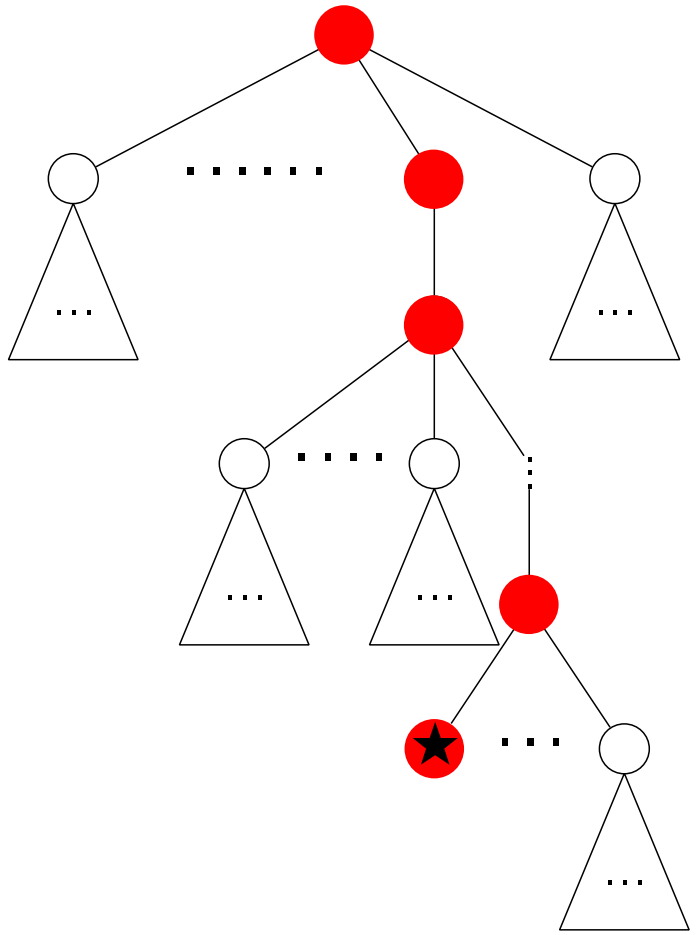


Feedback: (red > white)
for all (red, white) pairs
in the trajectory

π_2 Policy Rollout

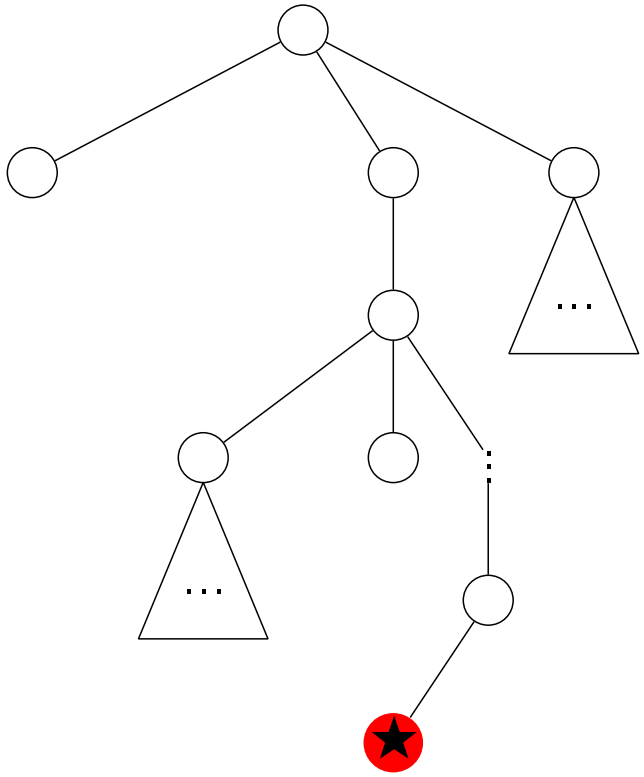


Retrospective Oracle Feedback



Feedback: (red > white)
for all (red, white) pairs
in the trajectory

π_3 Policy Rollout



Core Algorithm Summary & Guarantees

- Sequence of Learning Reductions
- Leverages Retrospective Oracle to Define "Correct"
  - Relies on sparse environmental rewards
- Converges to near-optimal policy in class
  - Offloads computational challenges to Supervised Learning Oracle
- For supervised learning error ε:

(caveats apply, see paper)

$$\text{Expected Search Length} = \frac{H^*}{1-2\varepsilon}$$

Optimal Search Length (typically # integer variables)

Learning to Search via Retrospective Imitation, Jialin Song, Ravi Lanka, et al., arXiv

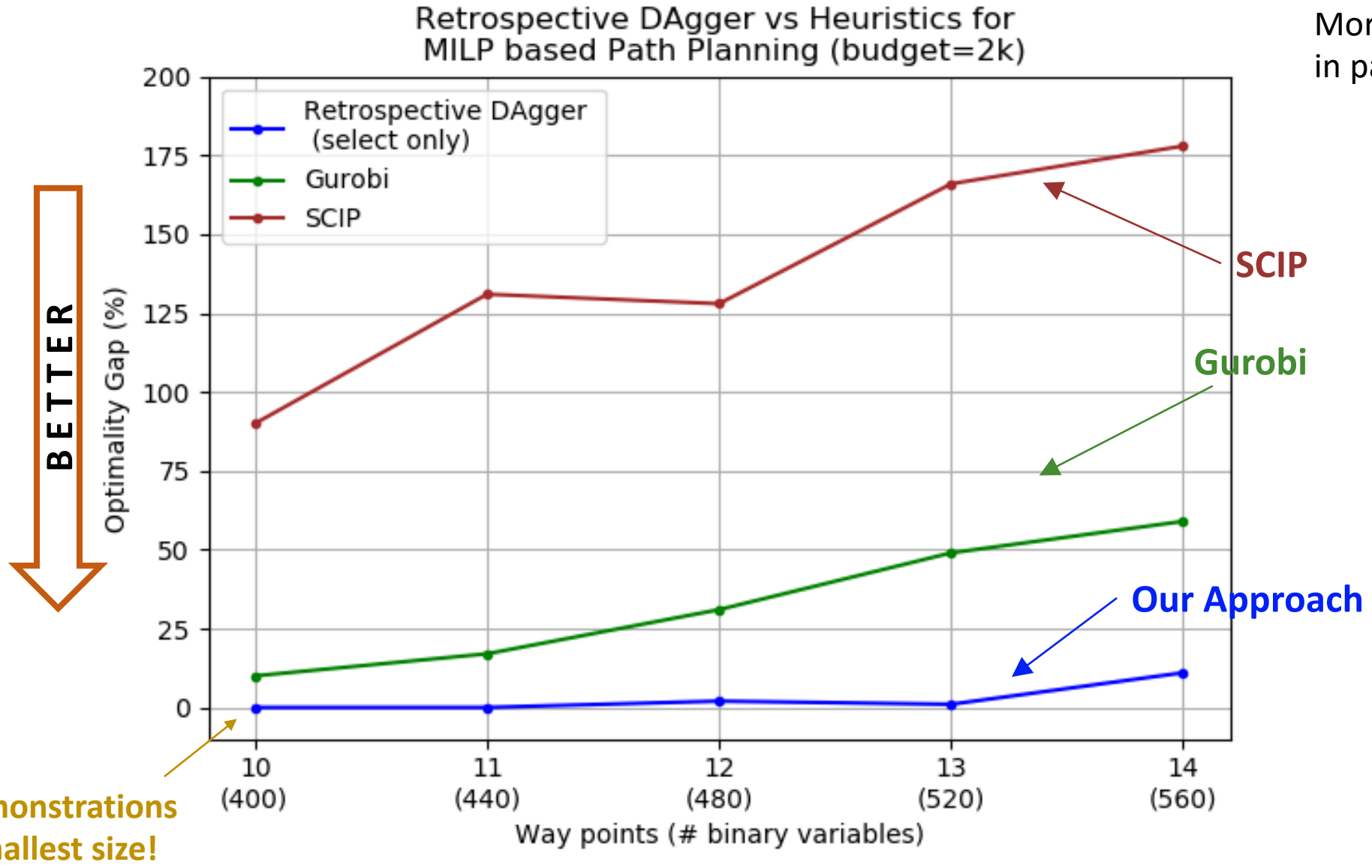
Guarantees for Full Algorithm

- Run π^k on problems of difficulty k+1
  - Initial demonstrations for the harder problem instances
- **Suppose:** we could have run external solver on harder instances

  Gurobi/SCIP/CPlex/Etc…
- **Suppose:** search trace includes feasible solution of external solver
- Then π^k is as good as using original external solver!
  - (might take longer to converge)

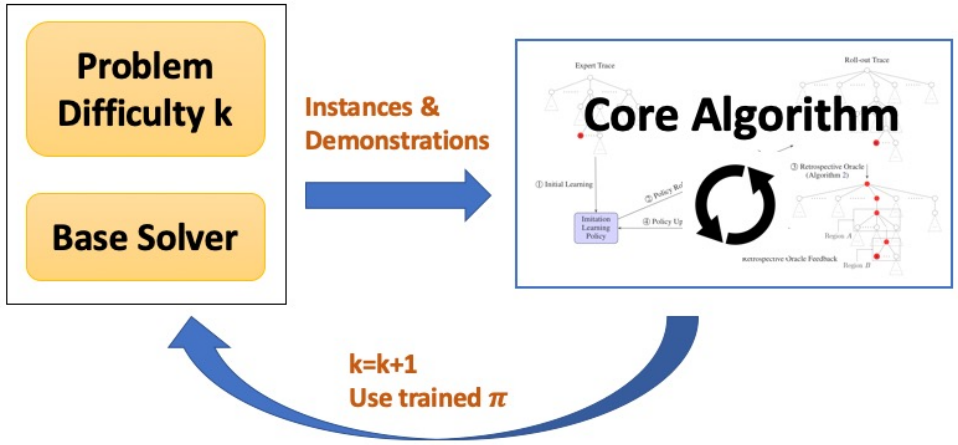
Learning to Search via Retrospective Imitation, Jialin Song, Ravi Lanka, et al., arXiv

More experiments in paper



Learning to Search via Retrospective Imitation, Jialin Song, Ravi Lanka, et al., arXiv

Retrospective Imitation

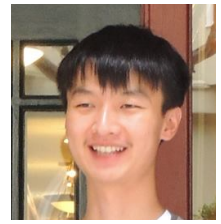


- Two-Stage Algorithm
  - Leverages Supervised Learning Oracle
- Initial demonstrations on small problems
- Exploits sparse environmental reward
  - "Retrospective Oracle"
- Iteratively scale up to harder problems

Blends Imitation & Reinforcement Learning

A Formal Notion of Curriculum Learning

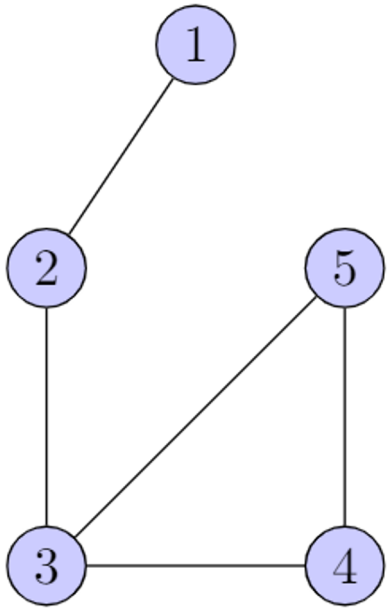
Co-Training for Policy Learning
(Multiple Views)



Jialin Song

Ravi Lanka

Example: Minimum Vertex Cover



Graph View

[Khalil et al., 2017]

$$\max \; -\sum_{i=1}^{5} x_i,$$

subject to:

$$x_1 + x_2 \geq 1,$$
$$x_2 + x_3 \geq 1,$$
$$x_3 + x_4 \geq 1,$$
$$x_3 + x_5 \geq 1,$$
$$x_4 + x_5 \geq 1,$$
$$x_i \in \{0, 1\}, \forall i \in \{1, \cdots, 5\}$$

Integer Program View
(Branch & Bound View)

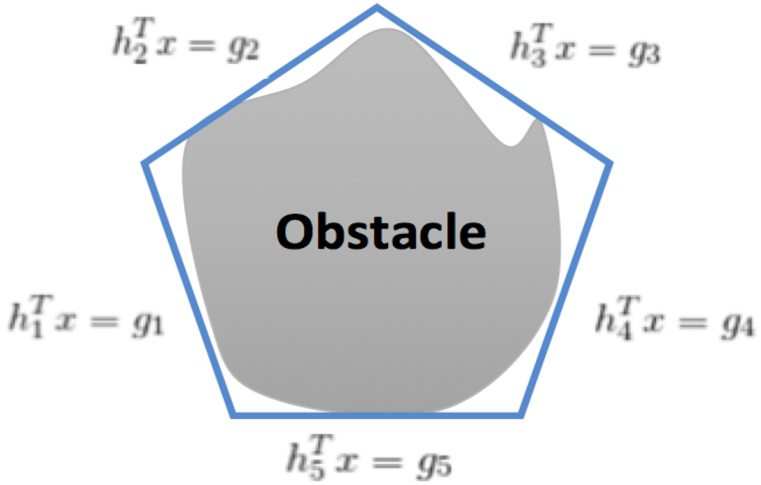
[He et al., 2014]

Co-Training for Policy Learning
(Multiple Views)

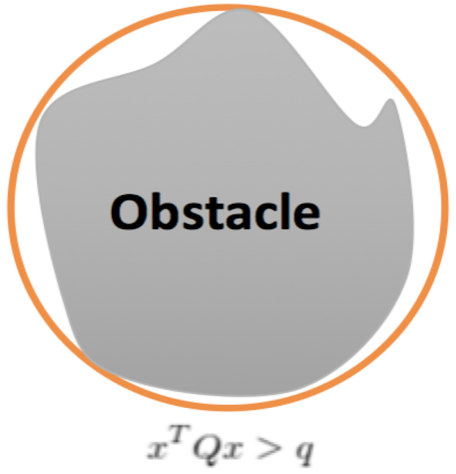
Jialin Song

Ravi Lanka

Example: Different Types of Integer Programs



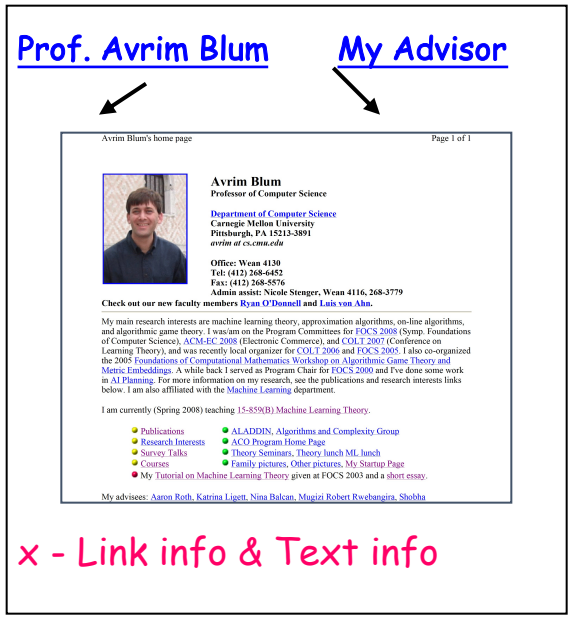
ILP

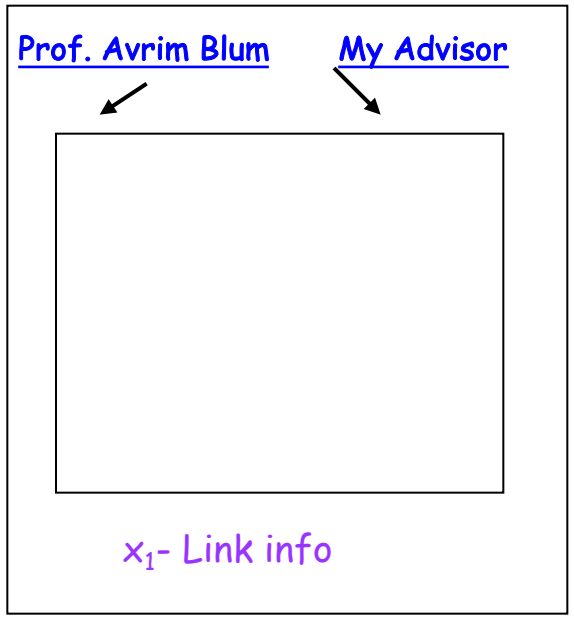


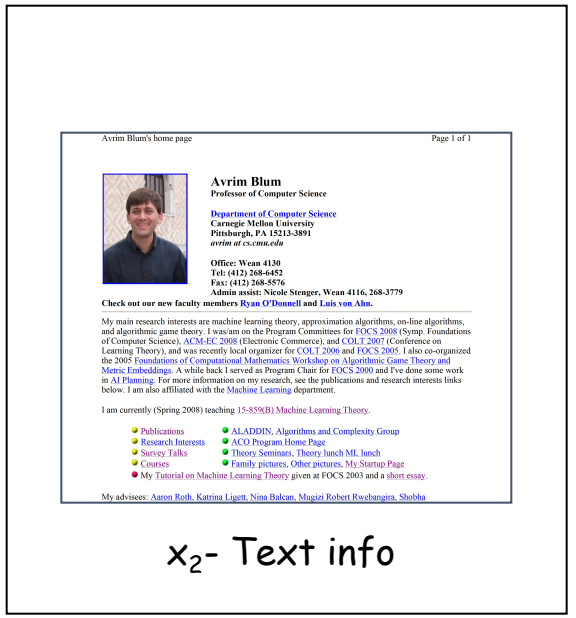
QCQP

Co-Training [Blum & Mitchell, 1998]

- Many learning problems have different sources of information
- Webpage Classification: Words vs Hyperlinks





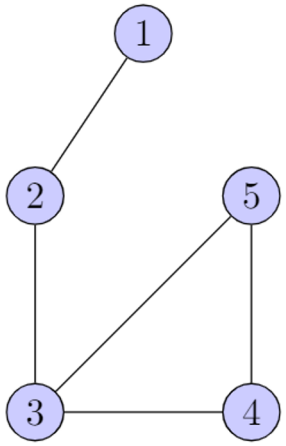


(Taken from Nina Balcan's slides)

What's Different about Policy Co-Training?

- Sequential Decisions vs 1-Shot Decisions
- (Sparse) Environmental Feedback
  - Can collect more "labels"
- Different Action Spaces
  - Graph vs Branch-and-Bound

(Not always applicable)



$$\max -\sum_{i=1}^{5} x_i,$$

subject to:

$$x_1 + x_2 \geq 1,$$
$$x_2 + x_3 \geq 1,$$
$$x_3 + x_4 \geq 1,$$
$$x_3 + x_5 \geq 1,$$
$$x_4 + x_5 \geq 1,$$
$$x_i \in \{0, 1\}, \forall i \in \{1, \cdots, 5\}$$

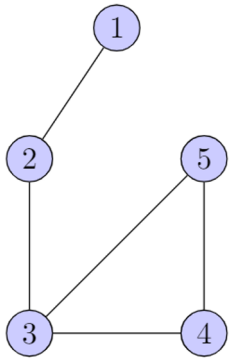
Co-training for Policy Learning, Jialin Song, Ravi Lanka, et al., UAI 2019

Intuition

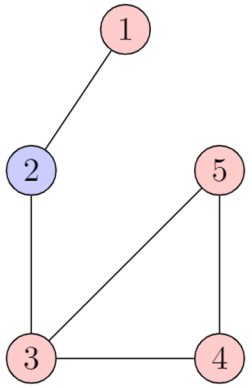
[1] “Learning combinatorial optimization algorithms over graphs” [Khalil et al., 2017]
[2] “Learning to Search in Branch and Bound Algorithms” [He et al., 2014]
[3] “Learning to Search via Retrospective Imitation” [Song et al., 2019]

MVC Instance

E.g., [1]



π^1



Demonstration

E.g., [2,3]

$$\max -\sum_{i=1}^{5} x_i,$$
subject to:
$$x_1 + x_2 \geq 1,$$
$$x_2 + x_3 \geq 1,$$
$$x_3 + x_4 \geq 1,$$
$$x_3 + x_5 \geq 1,$$
$$x_4 + x_5 \geq 1,$$
$$x_i \in \{0, 1\}, \forall i \in \{1, \cdots, 5\}$$

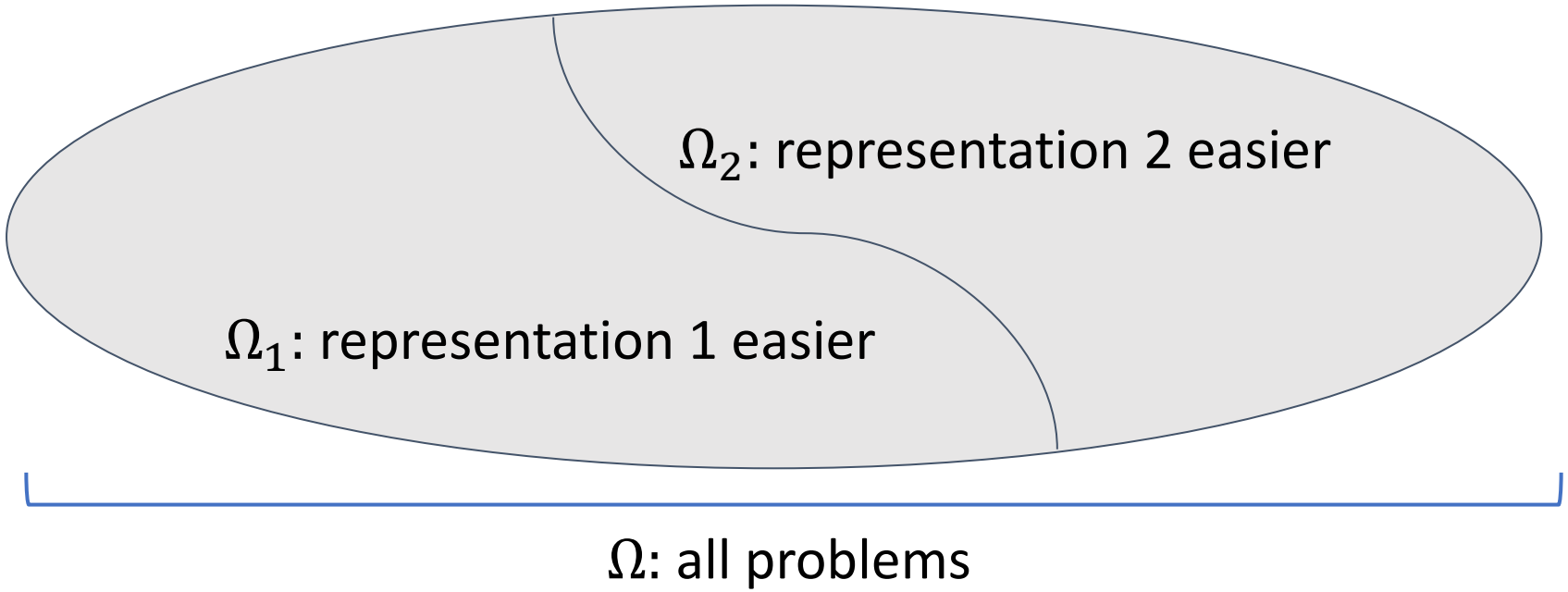
π^2

$x_1 = 0$
$x_2 = 1$
$x_3 = 1$
$x_4 = 1$
$x_5 = 0$

Better!

Theoretical Insight

- Different representations differ in hardness
- Goal: quantify improvement

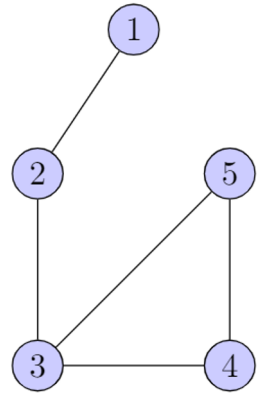


Co-training for Policy Learning, Jialin Song, Ravi Lanka, et al., UAI 2019

(Towards) a Theory of Policy Co-Training

- Two MDP "views": M^1 & M^2
  - $f^{1\to 2}(\tau^1) \Longrightarrow \tau^2$ (and vice versa)

    "Trajectory" / "Rollout"
  - Realizing τ^1 on $M^1 \Longleftrightarrow$ realizing τ^2 on M^2



$$\max -\sum_{i=1}^{5} x_i,$$

subject to:

$$x_1 + x_2 \geq 1,$$
$$x_2 + x_3 \geq 1,$$
$$x_3 + x_4 \geq 1,$$
$$x_3 + x_5 \geq 1,$$
$$x_4 + x_5 \geq 1,$$
$$x_i \in \{0, 1\}, \forall i \in \{1, \cdots, 5\}$$

- **Question:** when does having two views/policies help?
  - Policy Improvement (next slide)
    - Builds upon [Kang et al., ICML 2018]
  - Optimality Gap for Shared Action Spaces (in paper)
    - Builds upon [DasGupta et al., NeurIPS 2002]

Policy Improvement Bound (Summary)

Jialin Song

Ravi Lanka

$$J(\pi'^1) \geq J_{\pi^1}(\pi'^1) - \frac{2\gamma\left(\alpha_\Omega^1 \varepsilon_\Omega^1 + 4\beta_{\Omega_2}^2 \varepsilon_{\Omega_2}^2\right)}{(1-\gamma)^2} + \delta_{\Omega_2}^2$$

- Minimizing $\beta_{\Omega_2}^2$ → low disagreement between π^2 vs π^1
- Maximizing $\delta_{\Omega_2}^2$ → high performance gap π^2 over π^1 on some MDPs

Builds upon theoretical results from [Kang et al., ICML 2018]

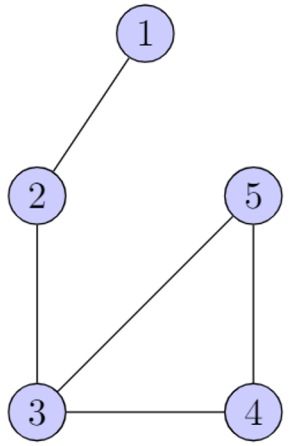
CoPiEr Algorithm (Co-training for Policy Learning)

Sample $M \sim \Omega$

Rollout

Run $\pi^1 \rightarrow \tau^1$

Run $\pi^2 \rightarrow \tau^2$



M^1

$$\max -\sum_{i=1}^{5} x_i,$$

subject to:

$$x_1 + x_2 \geq 1,$$
$$x_2 + x_3 \geq 1,$$
$$x_3 + x_4 \geq 1,$$
$$x_3 + x_5 \geq 1,$$
$$x_4 + x_5 \geq 1,$$
$$x_i \in \{0, 1\}, \forall i \in \{1, \cdots, 5\}$$

M^2

Update (only showing 1 view)

Augmented Obj: $\tilde{J}(\pi') = J_\pi(\pi') - \lambda L(\pi', \tau')$

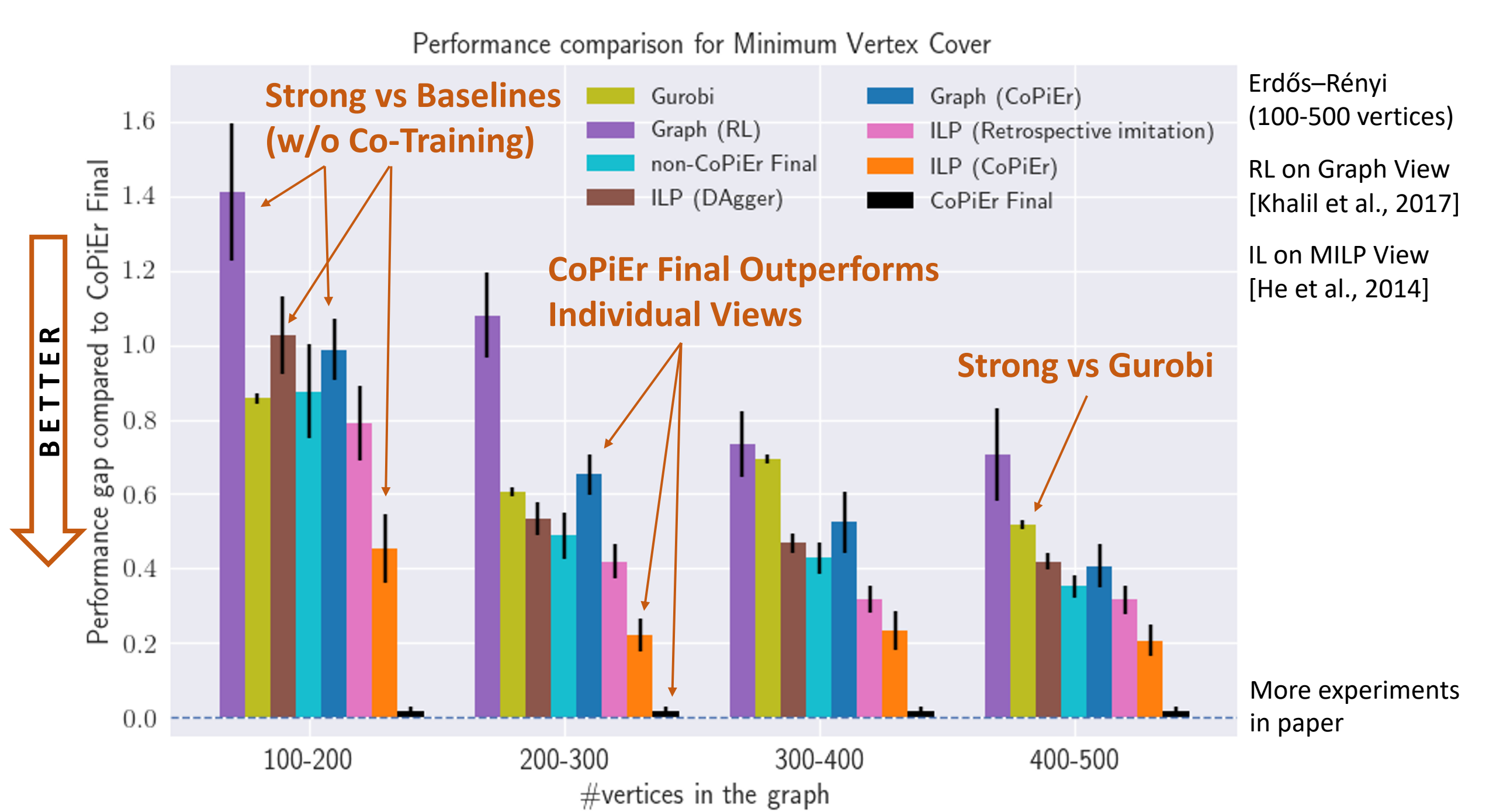
Take gradient step

Exchange (only showing 1 version)

If π^1 better: $\tau'^2 = f^{1\rightarrow 2}(\tau^1), \tau'^1 = \emptyset$

If π^2 better: $\tau'^1 = f^{2\rightarrow 1}(\tau^2), \tau'^2 = \emptyset$

Co-training for Policy Learning, Jialin Song, Ravi Lanka, et al., UAI 2019



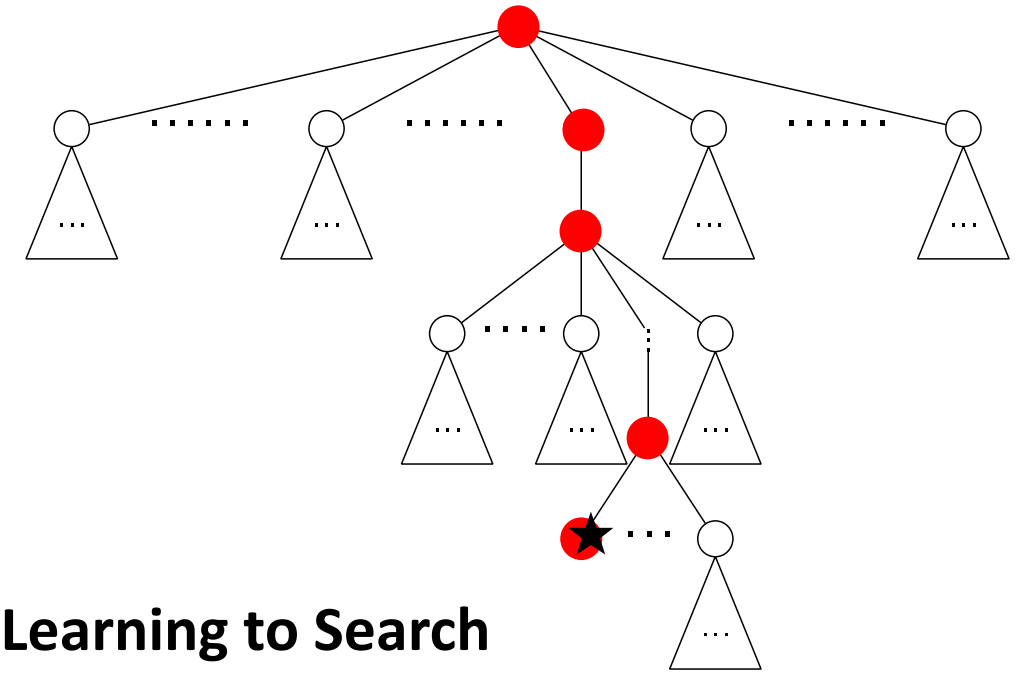
Erdős–Rényi (100-500 vertices)

RL on Graph View [Khalil et al., 2017]

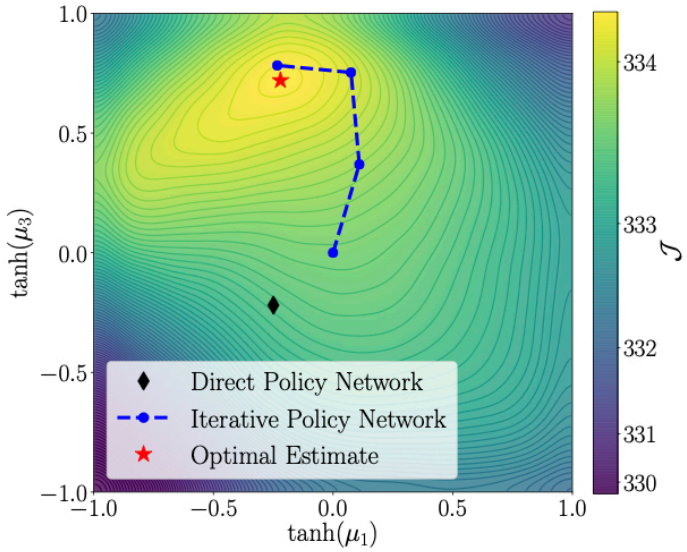
IL on MILP View [He et al., 2014]

More experiments in paper

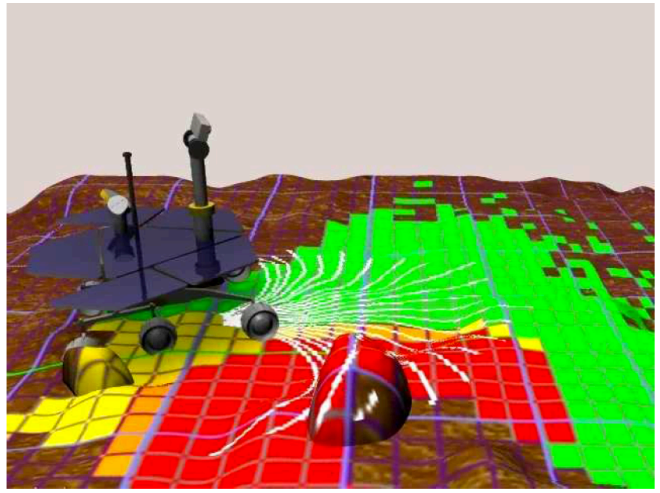
This Talk



Learning to Search

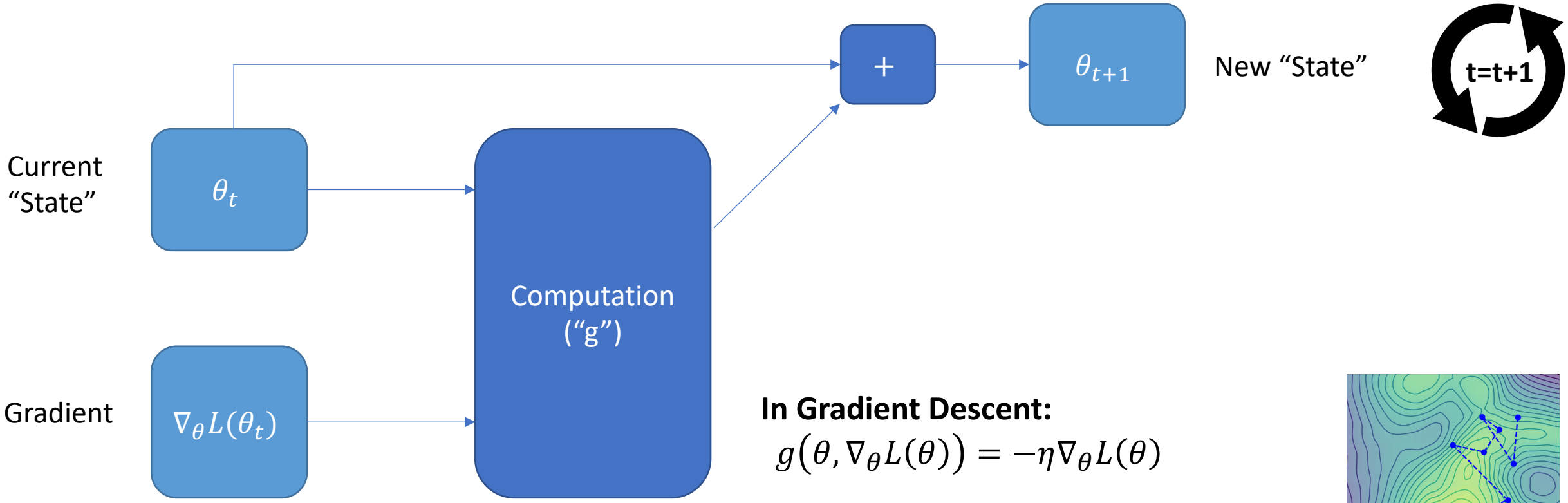


Continuous Optimization



Practical Applications

Optimization as a Computation Graph

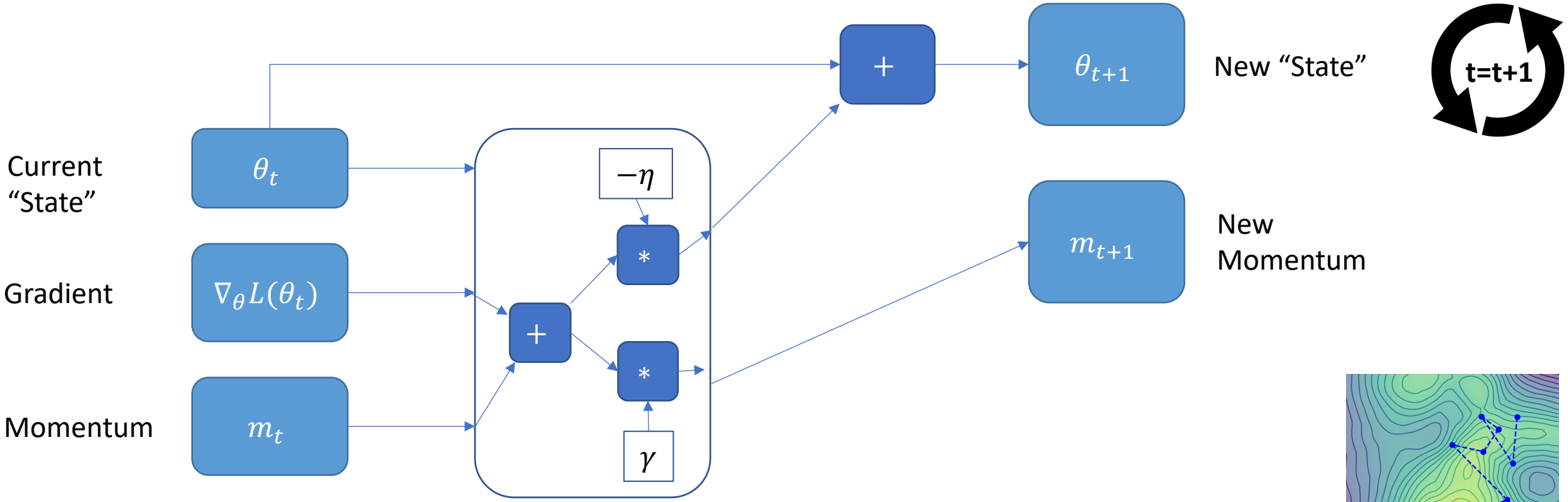


In Gradient Descent:

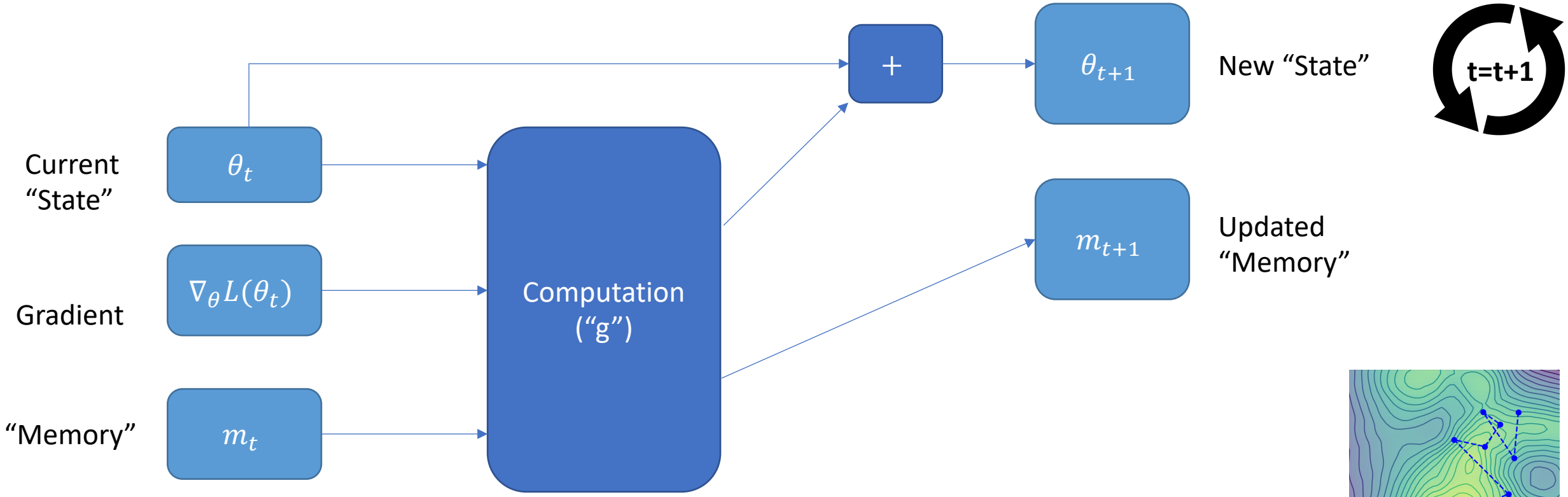
$$g\big(\theta, \nabla_\theta L(\theta)\big) = -\eta \nabla_\theta L(\theta)$$

Example: Gradient Descent w/ Momentum

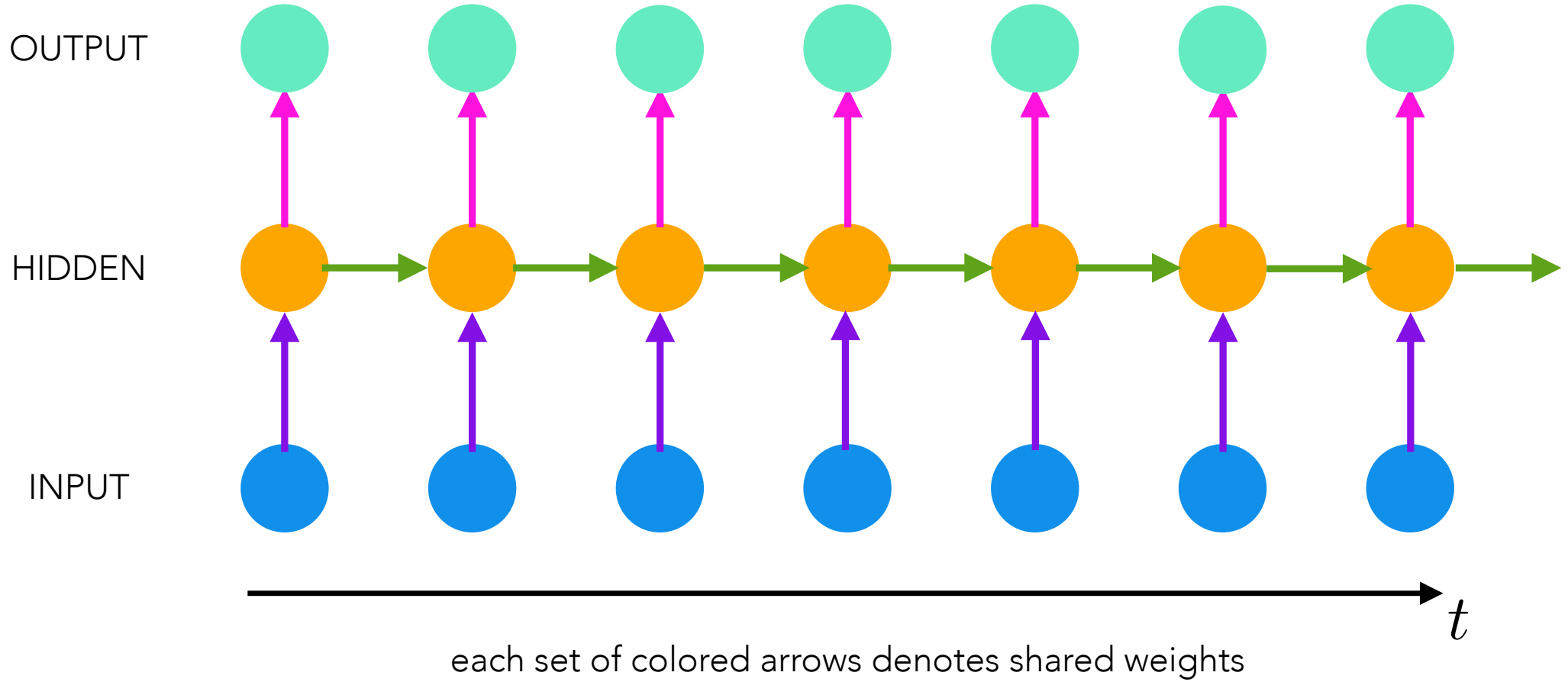
$$g\big(\theta, \nabla_\theta L(\theta)\big) = -\eta(\nabla_\theta L(\theta) + m)$$
$$m = \gamma\big(m + \nabla_\theta L(\theta)\big)$$



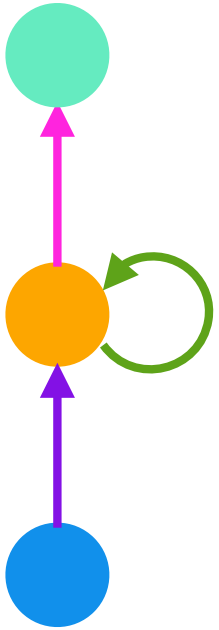
(Differentiable) Learning to Optimize



Recall: Recurrent Neural Networks



Recall: Recurrent Neural Networks



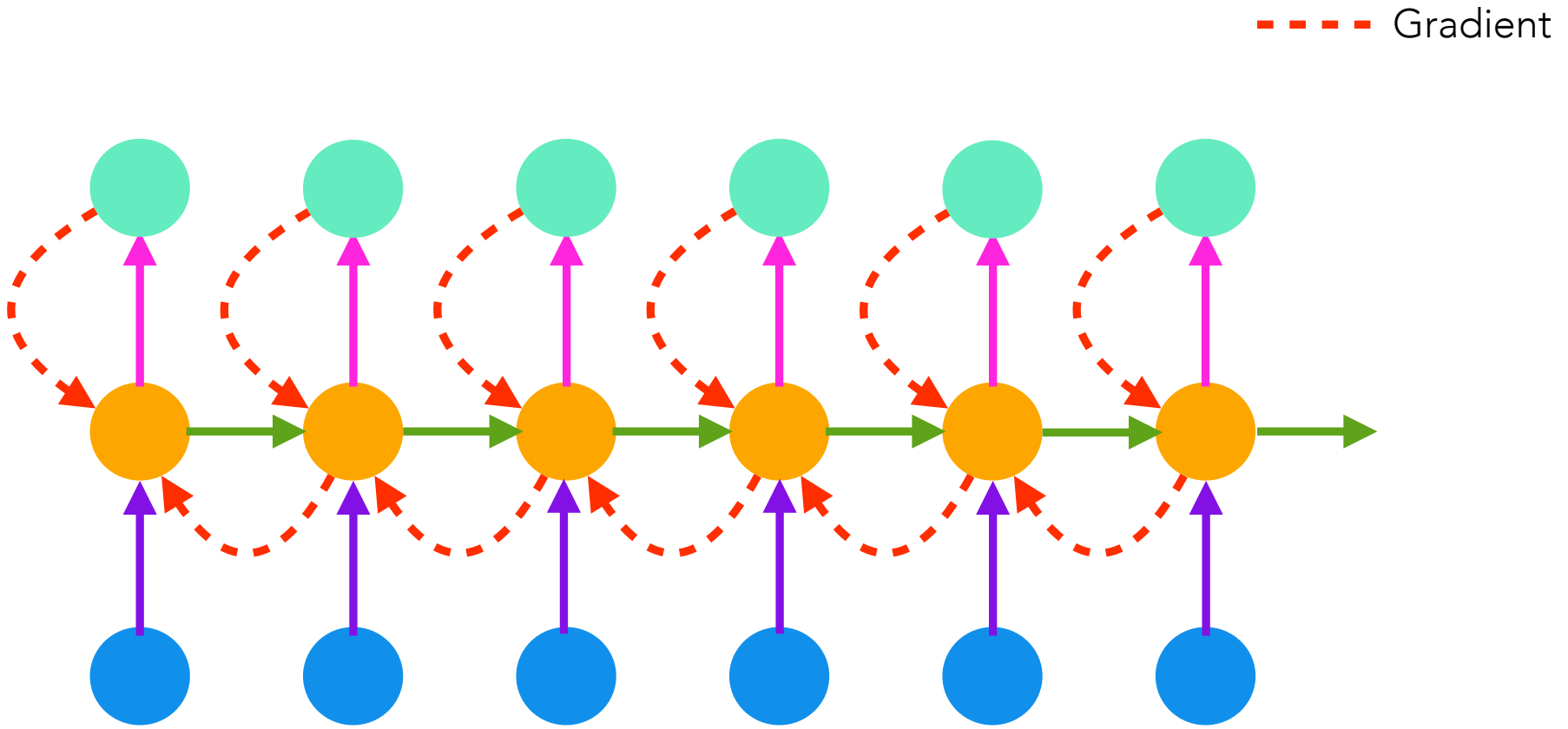
Hidden State

$$\mathbf{h}_t = \sigma(\mathbf{W}_\mathbf{h}^\intercal[\mathbf{h}_{t-1}, \mathbf{x}_t])$$

Output

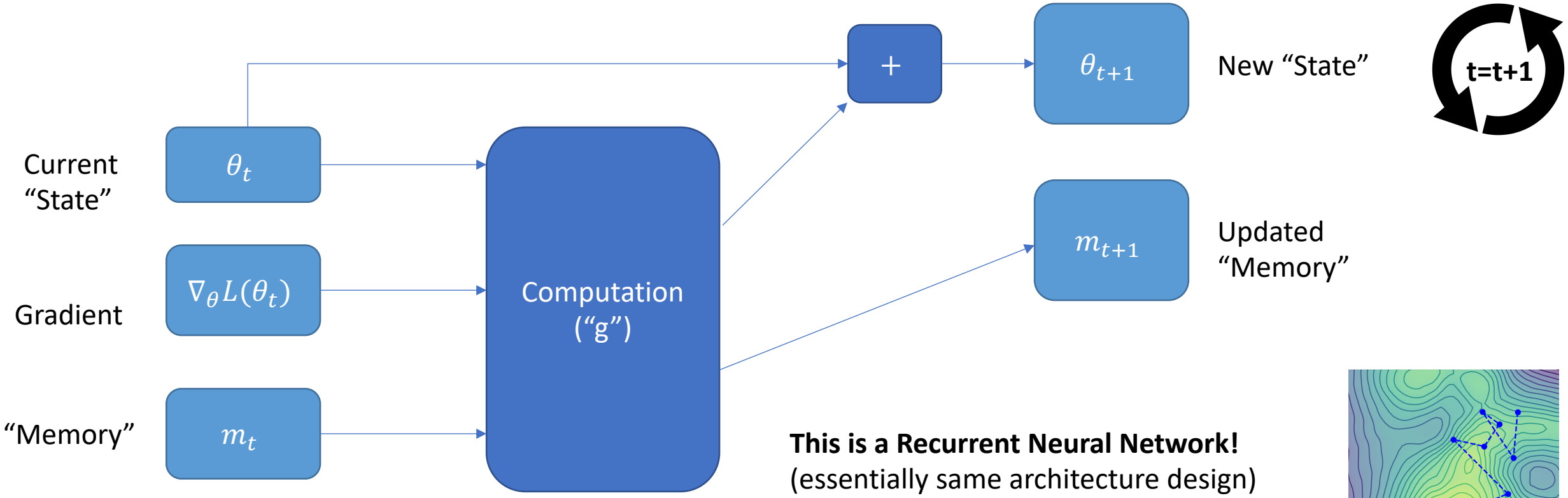
$$\mathbf{y}_t = \sigma(\mathbf{W}_\mathbf{y}^\intercal \mathbf{h}_t)$$

Backpropagation Through Time



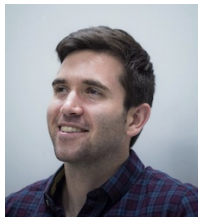
Learning to Optimize as (Recurrent) Deep Learning

(backprop learning signal)



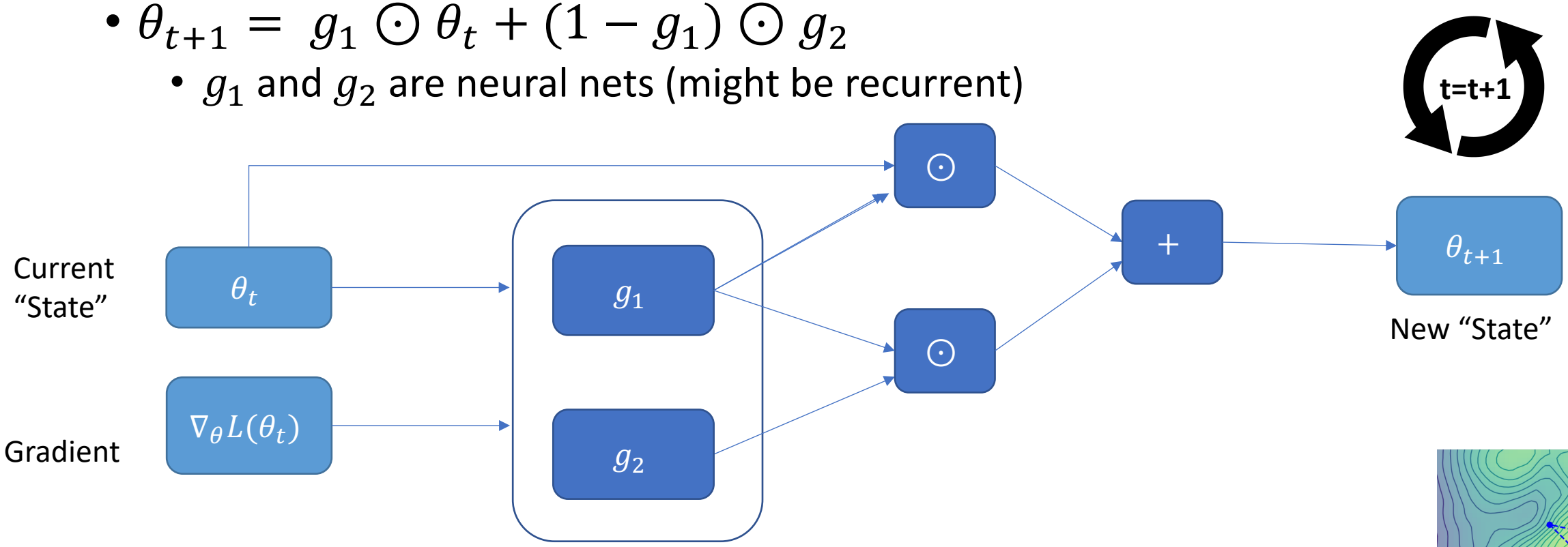
This is a Recurrent Neural Network!
(essentially same architecture design)

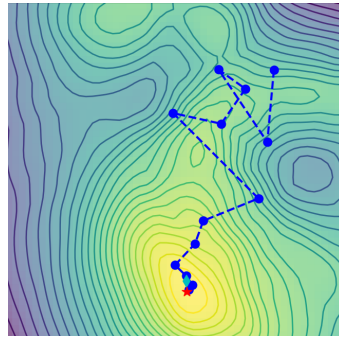
Gating Update Rule



Joe Marino

- $\theta_{t+1} = g_1 \odot \theta_t + (1 - g_1) \odot g_2$
  - g_1 and g_2 are neural nets (might be recurrent)



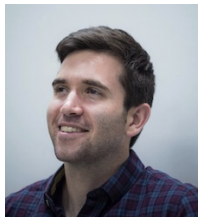


Iterative Amortized Inference, Joe Marino et al., ICML 2018

A General Framework for Amortizing Variational Filtering, Joe Marino et al, NeurIPS 2018

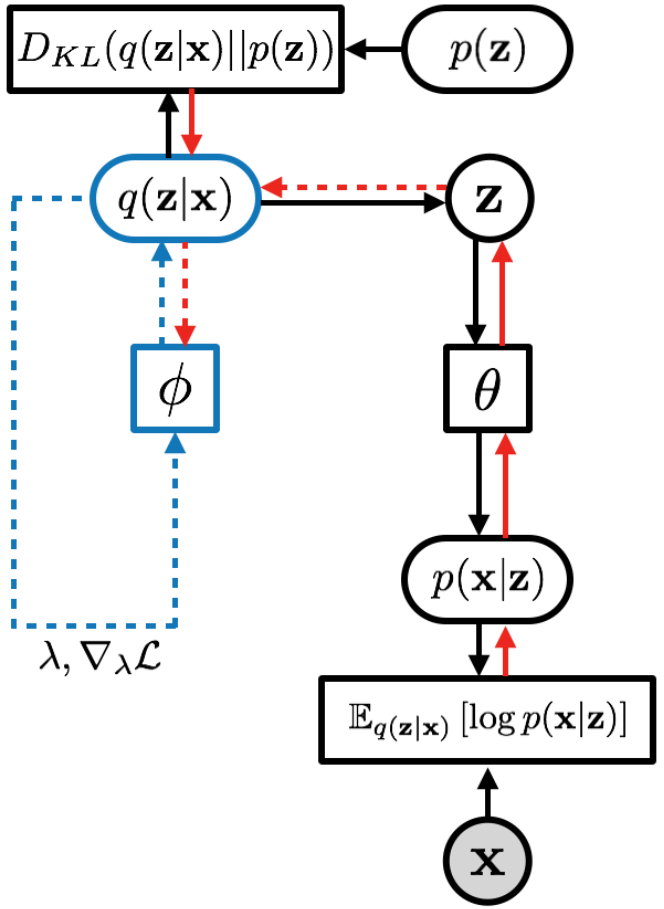
Iterative Amortized Policy Optimization, Joe Marino et al., arXiv

Setup More Complicated
(Variational Inference)



Joe Marino

- Blue part is the learned optimizer
- Due to the complexities of variational inference

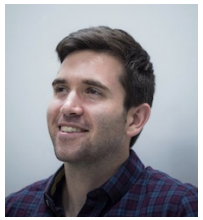


Iterative Amortized Inference, Joe Marino et al., ICML 2018

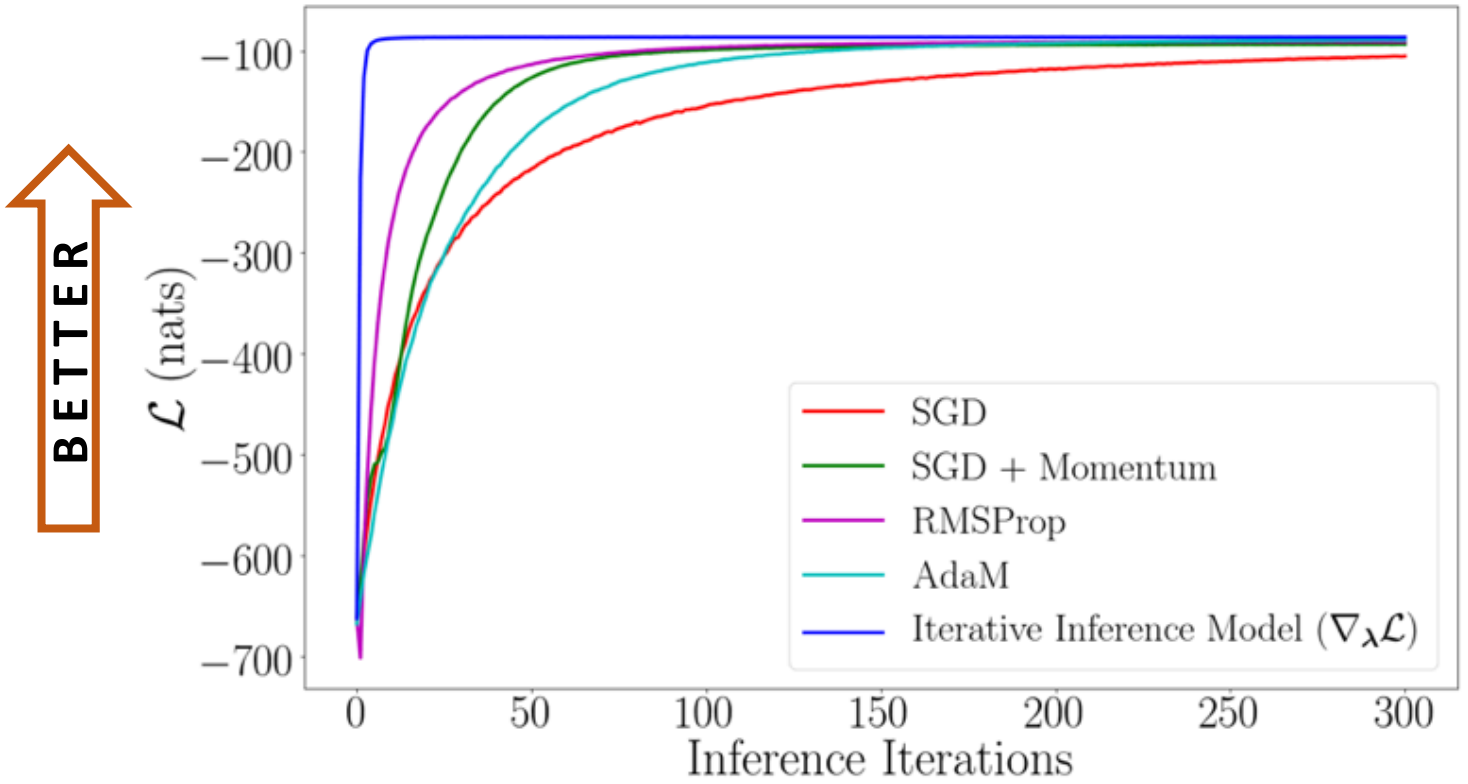
A General Framework for Amortizing Variational Filtering, Joe Marino et al, NeurIPS 2018

Iterative Amortized Policy Optimization, Joe Marino et al., arXiv

Iterative Amortized Inference
(for Deep Probabilistic Models)



Joe Marino

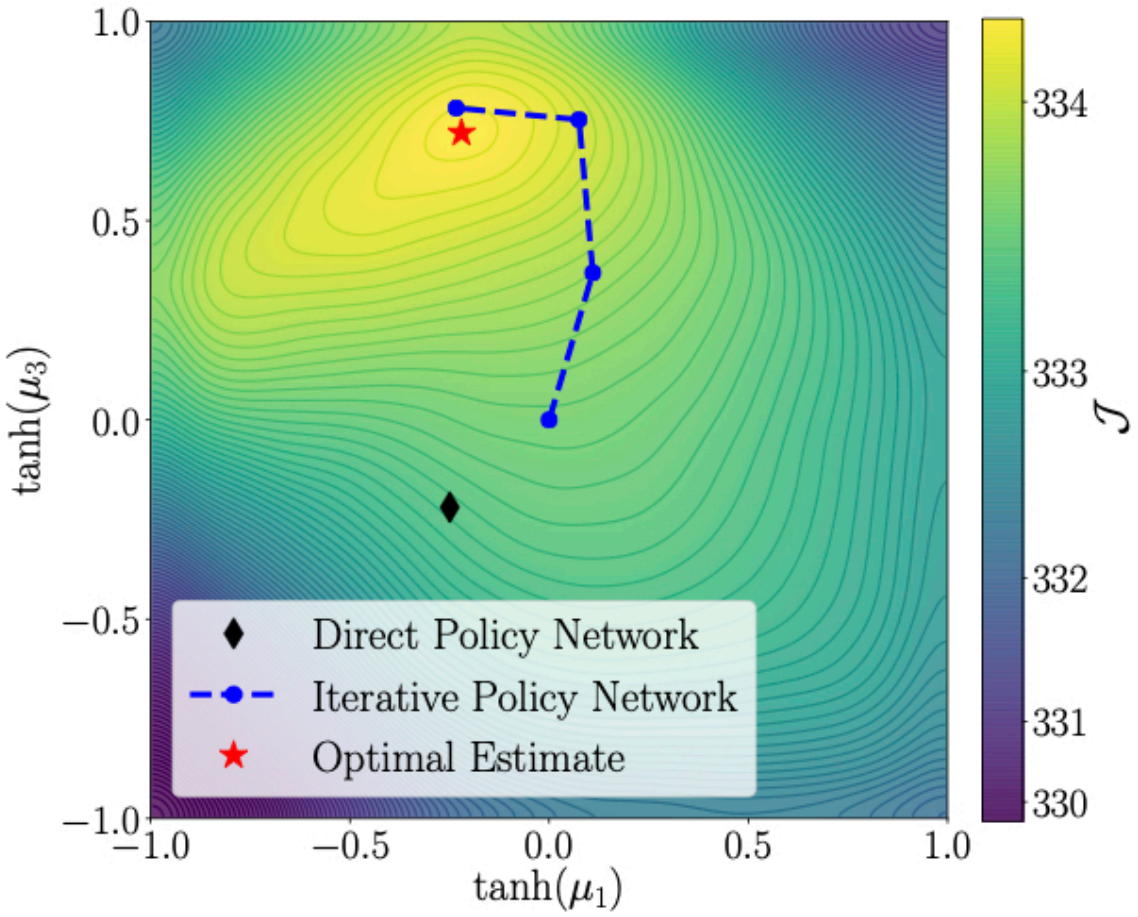


Iterative Amortized Inference, Joe Marino et al., ICML 2018

Related Work

- The Differentiable Cross-Entropy Method
  - [Amos & Yarats] https://arxiv.org/abs/1909.12830
- Learning to Learn
  - [Andrychowicz et al.] https://arxiv.org/abs/1606.04474
- Differentiable MPC
  - [Amos et al.] https://arxiv.org/abs/1810.13400
- Deep MRI Reconstruction
  - [Liang et al.] https://arxiv.org/abs/1907.11711
- RNA Secondary Structured Prediction
  - [Chen et al.] https://arxiv.org/abs/2002.05810
- And Many More!

Aside: Amortization Gap



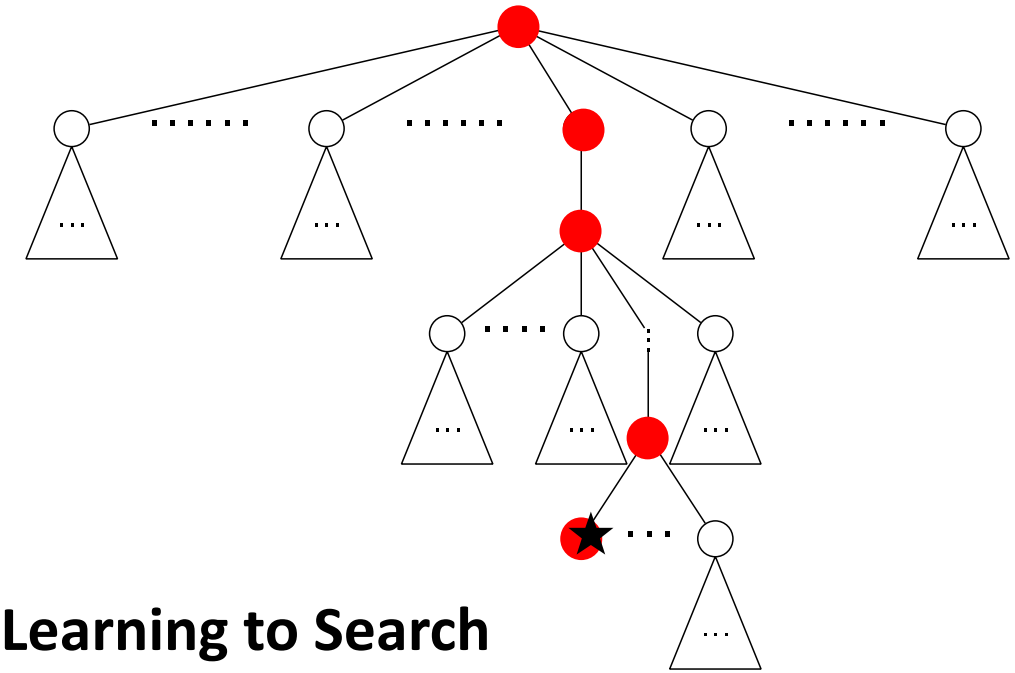
"Amortization"

- Learn a NN to predict solution
- Spend compute on (pre-)training
- Run-time optimization is fast.

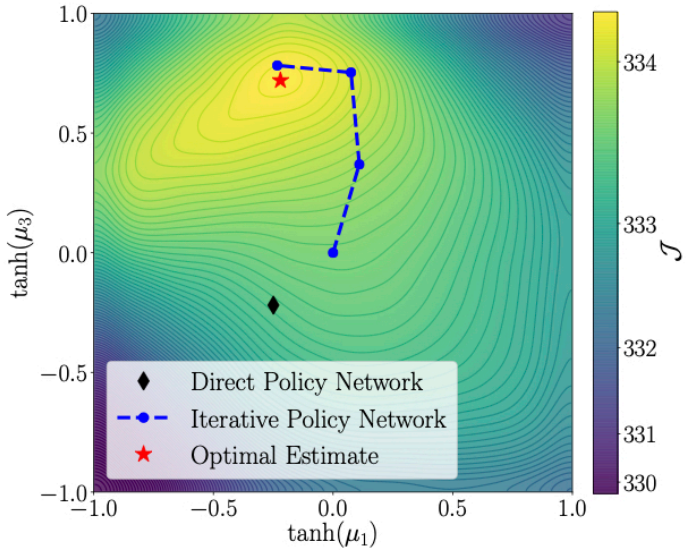
"Amortization Gap"

- 1-shot amortization (aka "Direct")
- Cannot accurately predict solution

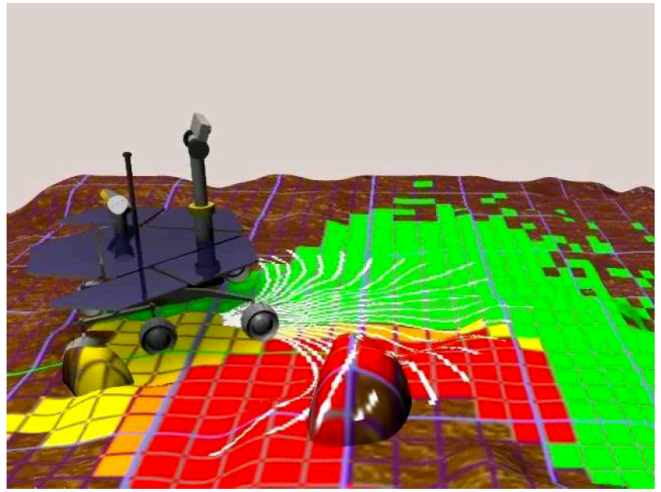
This Talk



Learning to Search



Continuous Optimization



Practical Applications

What Matters in Practice?

- **Story so far:** solution quality vs #iterations
  - Baseline solver might be very fast per iteration
  - Baseline solver might have smart pre-conditioner
  - Baseline solver might generalize better
  - Etc…

- **Next Step:** solution quality vs wall-clock time

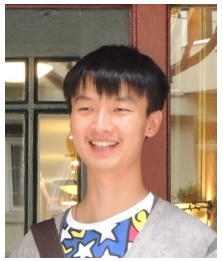
Many Solvers Can be Very Fast….

- … if you know some key structural properties of the problem.

- **Paradigm 1:** Predict the key variables that are hard
  - E.g., backdoor variables [https://www.cs.cornell.edu/gomes/pdf/2009_dilkina_cpaior_backdoors.pdf]
  - Set backdoor variables first, then run solver.

- **Paradigm 2:** Predict a decomposition
  - E.g., Large Neighborhood Search [https://arxiv.org/abs/2004.00422]
  - Run solver on smaller problems (should be fast)

Large Neighborhood Search



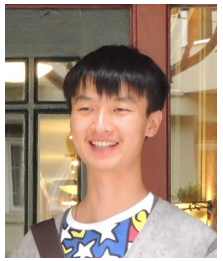
Jialin Song

1. Partition variables into X_1, X_2, X_3, …, X_m
2. Freeze all variables except one partition at a time
   - Run solver on partition
3. Repeat from Step 1

- **How to partition?**
  - Use learning to predict!

A General Large Neighborhood Search Framework for Solving Integer Programs, Jialin Song, et al., NeurIPS 2020

Large Neighborhood Search



Jialin Song

1. Partition variables into X_1, X_2, X_3, …, X_m
2. Freeze all variables except one partition at a time
   - Run solver on partition
3. Repeat from Step 1

- **How to partitio**
  - Use learning t

Benefits:

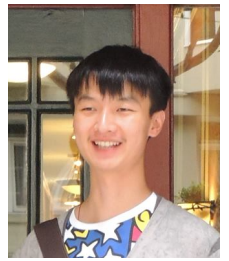
1. Can leverage state-of-the-art solvers & their implementations
2. Can be competitive in wall-clock time

Drawbacks:

1. Reliant on existing solver being good for sub-problems
2. Reliant on being able to find those sub-problems

A General Large Neighborhood

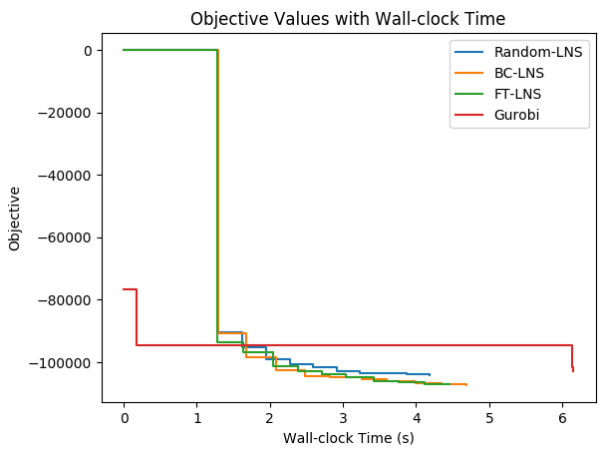
Some Empirical Results



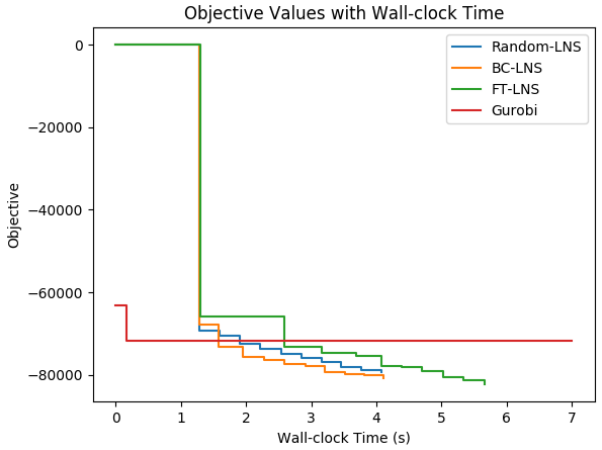
Jialin Song

1000x Faster in Wall-Clock Time!

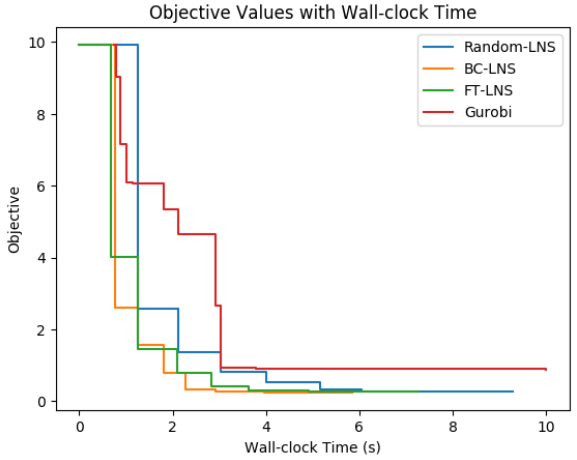
BETTER



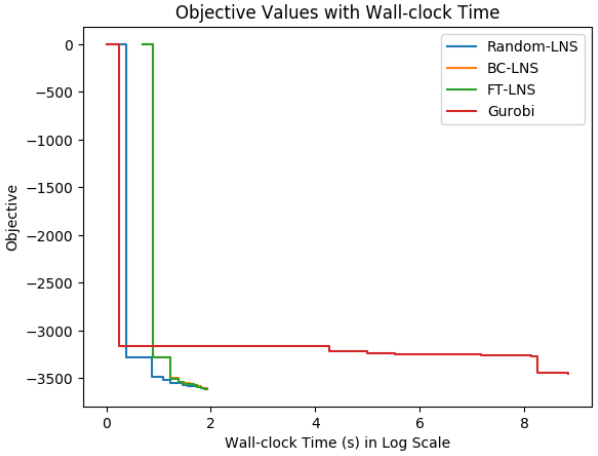
(a) Combinatorial auction of 2000 items and 4000 bids from regions distribution.



(b) Combinatorial auction of 2000 items and 4000 bids from arbitrary distribution.



(c) Risk-aware path planning for navigating through 30 obstacles.

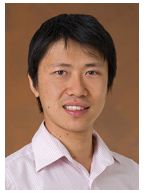


(d) Maximum cut over a Barabási-Albert random graph with 500 vertices.

A General Large Neighborhood Search Framework for Solving Integer Programs, Jialin Song, et al., NeurIPS 2020

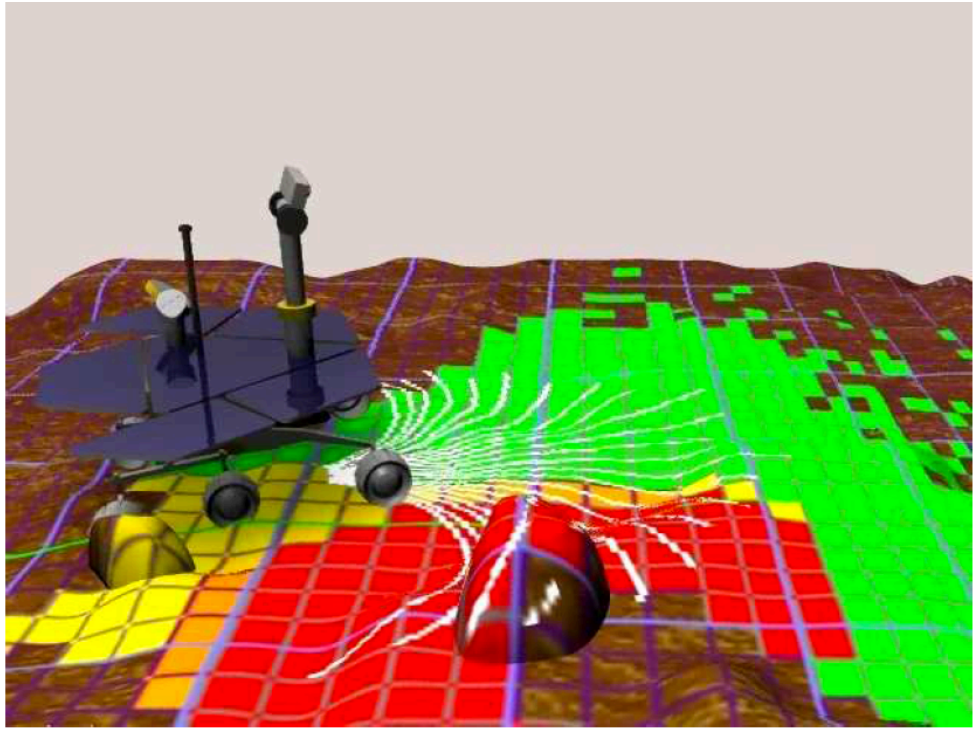
Ongoing: Integration with ENav

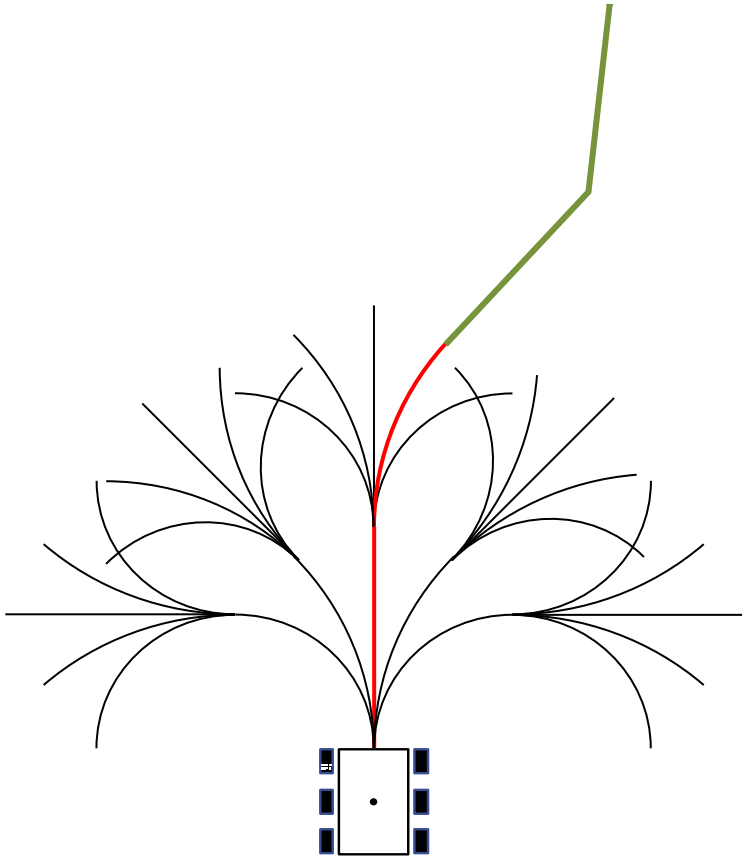

Shreyansh Daftry


Hiro Ono


Olivier Toupet


Neil Abcouwer

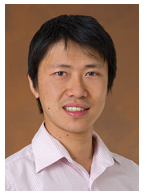




NASA JPL

The Problem with ENav

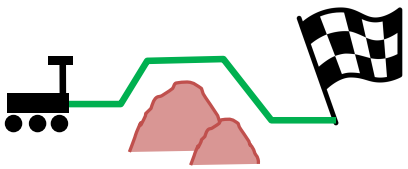

Shreyansh Daftry


Hiro Ono


Olivier Toupet

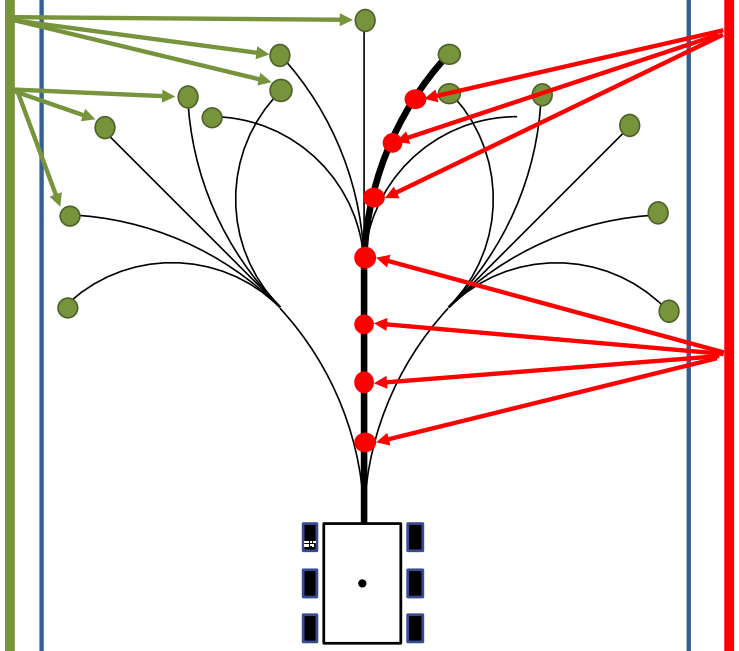

Neil Abcouwer

Global Planner



- Gives cost from the end of tree to goal
- Routes computed on 200m x 200m map
- 1 m resolution
- Considers slope, roughness, KOZ, KIZ

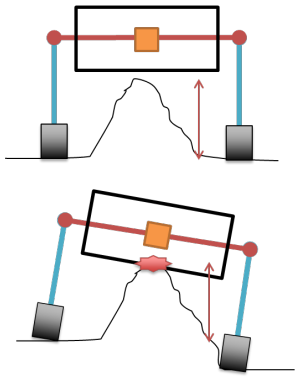
Local Planner



- Selects best path for the next 6m from finite # of options

ACE

(Approx. Clearance Est.)



- Evaluates clearance using approx. kinematics
- Run every 25cm
- Also checks tilt, wheel drop, etc

ACE Evaluation is Expensive!

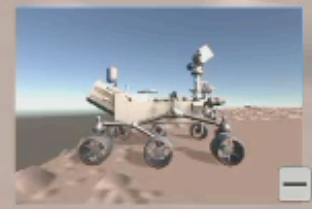
Behind Rover Camera

Overhead Camera

SCLK: 1,478,818,216.76650

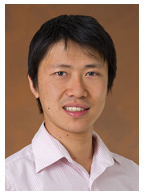
Left Camera

Right Camera



Preliminary Results


Shreyansh Daftry


Hiro Ono


Olivier Toupet


Neil Abcouwer

Baseline ENav (Cycle Time(s))

	7 %	10 %	12 %	15 %
20 deg.	1.45	3.38	n/a	n/a
15 deg.	1.00	2.49	2.89	3.58
10 deg.	0.99	2.39	3.17	2.06
5 deg.	0.77	2.38	3.79	2.09
0 deg.	0.98	2.57	3.44	5.18

MLNav (Cycle Time(s))

	7 %	10 %	12 %	15 %
20 deg.	0.80	0.99	n/a	n/a
15 deg.	0.48	0.99	1.20	1.45
10 deg.	0.47	1.20	1.38	1.41
5 deg.	0.54	0.93	1.25	1.28
0 deg.	0.47	1.11	1.33	1.78

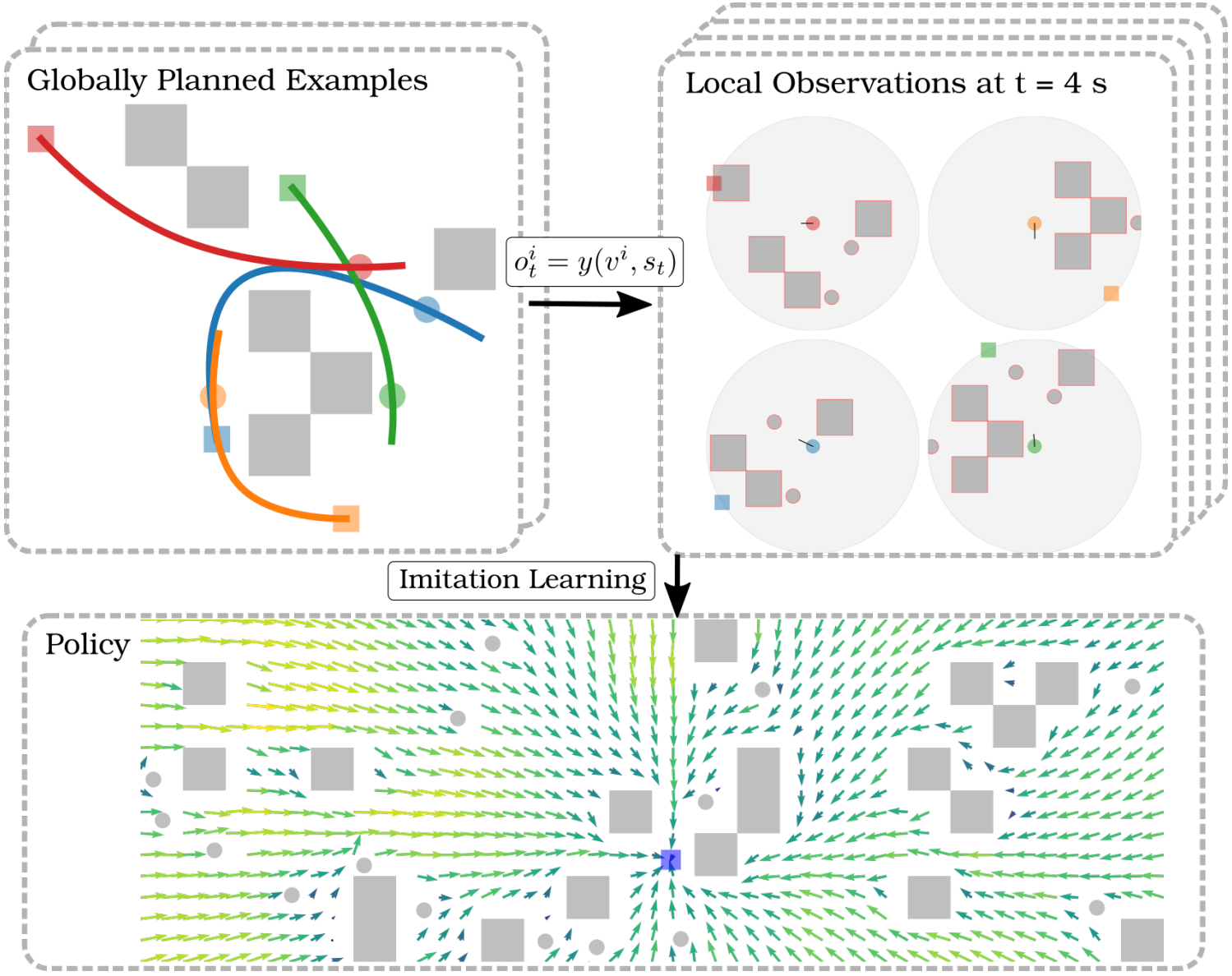
Machine Learning Based Path Planning for Improved Rover Navigation, Neil Abcouwer et al., (under review)

Learned Decentralized Planner

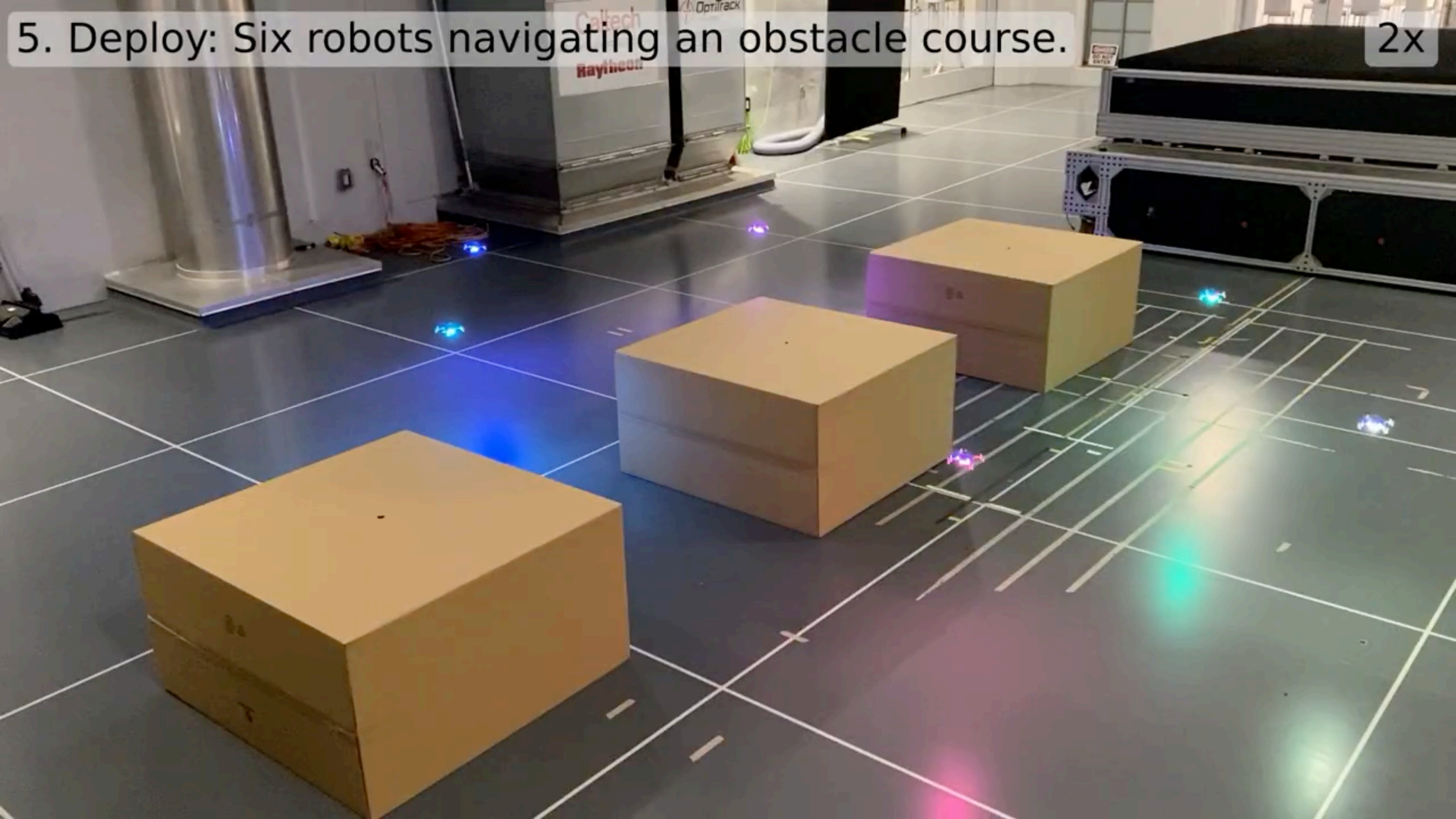
(enforcing safety)

Ben Riviere

Wolfgang Hoenig

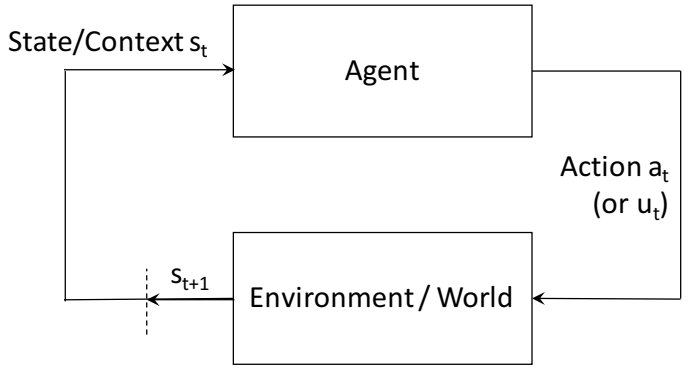


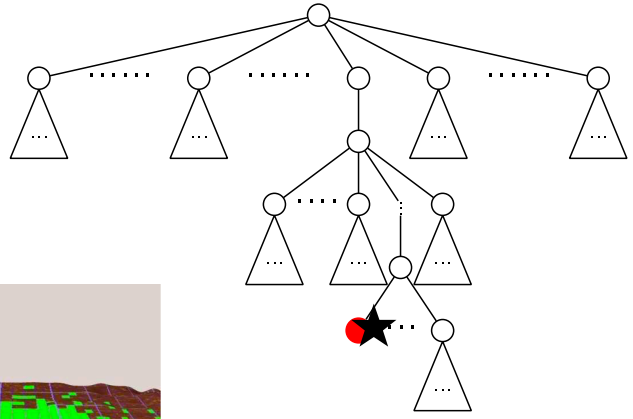
GLAS: Global-to-Local Safe Autonomy Synthesis for Multi-Robot Motion Planning with End-to-End Learning,
Benjamin Rivière, et al., R-AL 2020

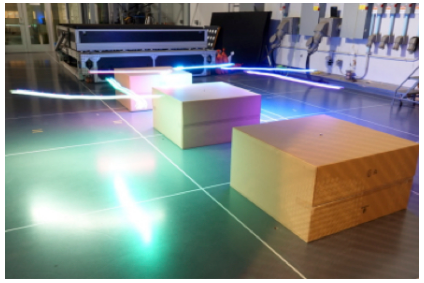

5. Deploy: Six robots navigating an obstacle course.
2x

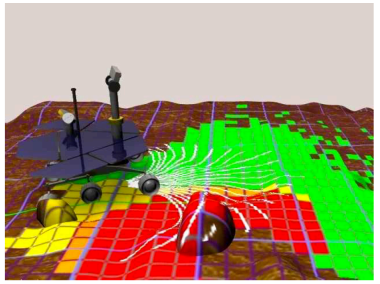
Summary: Learning to Optimize

- Optimization as Sequential Decision Making
- Formulate New Learning Problems
  - Builds upon RL/IL
- Interesting Algorithms
  - Theoretical Analysis/Guidance
  - Good Empirical Performance
- Exciting Applications!



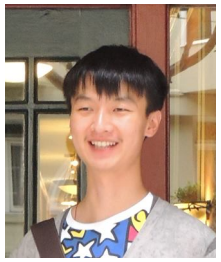






Ben Riviere

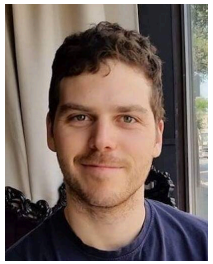
Wolfgang Hoenig



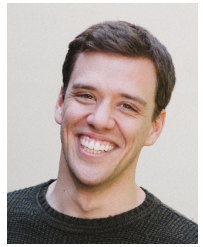
Jialin Song

Ravi Lanka

Joe Marino



Alex Piche



Milan Cvitkovic

Neil Abcouwer

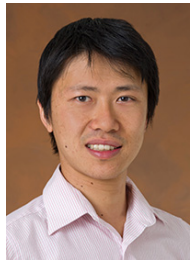
Shreyansh Daftry

Siddarth Venkatraman

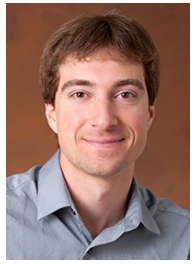
Alessandro Ialongo

Soon-Jo Chung

Stephan Mandt



Hiro Ono



Tyler del Sesto

Bistra Dilkina

Olivier Toupet

Aadyot Bhatnagar

Albert Zhao

Learning to Search via Retrospective Imitation, Jialin Song, Ravi Lanka, et al., arXiv

Co-Training for Policy Learning, Jialin Song, Ravi Lanka, et al., UAI 2019

A General Large Neighborhood Search Framework for Solving Integer Programs, Jialin Song, et al., NeurIPS 2020

Iterative Amortized Inference, Joe Marino et al., ICML 2018

A General Framework for Amortizing Variational Filtering, Joe Marino et al, NeurIPS 2018

Iterative Amortized Policy Optimization, Joe Marino et al., arXiv

Machine Learning Based Path Planning for Improved Rover Navigation, Neil Abcouwer et al., (under review)

GLAS: Global-to-Local Safe Autonomy Synthesis for Multi-Robot Motion Planning with End-to-End Learning, Ben Rivière et al., R-AL 2020