AlphaGo Zero, Monte Carlo Tree Search and Supervised Learning: Towards Theoretical Foundation

Devavrat Shah

Massachusetts Institute of Technology

Qiaomin Xie

Cornell University

Zhi Xue

Massachusetts Institute of Technology

AlphaGo Zero (AGZ)

nature.com > nature > articles > article

nature
International journal of science

Altmetric: 2151 Citations: 1 More detail »

Article

Mastering the game of Go without human knowledge

David Silver ✉, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel & Demis Hassabis

Nature **550**, 354–359 (19 October 2017)
doi:10.1038/nature24270
Download Citation

Received: 07 April 2017
Accepted: 13 September 2017
Published online: 18 October 2017

AlphaGo Zero (AGZ)

Abstract

A long-standing goal of artificial intelligence is an algorithm that learns, *tabula rasa*, superhuman proficiency in challenging domains. Recently, AlphaGo became the first program to defeat a world champion in the game of Go. The tree search in AlphaGo evaluated positions and selected moves using deep neural networks. These neural networks were trained by supervised learning from human expert moves, and by reinforcement learning from self-play. Here we introduce an algorithm based solely on reinforcement learning, without human data, guidance or domain knowledge beyond game rules. AlphaGo becomes its own teacher: a neural network is trained to predict AlphaGo's own move selections and also the winner of AlphaGo's games. This neural network improves the strength of the tree search, resulting in higher quality move selection and stronger self-play in the next iteration. Starting ***tabula rasa***, our new program AlphaGo Zero achieved superhuman performance, winning 100–0 against the previously published, champion-defeating AlphaGo.

AlphaGo Zero (AGZ)

Abstract

A long-standing goal of artificial intelligence is an algorithm that learns, *tabula rasa*, superhuman proficiency in challenging domains. Recently, AlphaGo became the first program to defeat a world champion in the game of Go. The tree search in AlphaGo evaluated positions and selected moves using deep neural networks. These neural networks were trained by supervised learning from human expert moves, and by reinforcement learning from self-play. Here we introduce an algorithm based solely on reinforcement learning, without human data, guidance or domain knowledge beyond game rules. AlphaGo becomes its own teacher: a neural network is trained to predict AlphaGo's own move selections and also the winner of AlphaGo's games. This neural network improves the strength of the tree search, resulting in higher quality move selection and stronger self-play in the next iteration. Starting ***tabula rasa***, our new program AlphaGo Zero achieved superhuman performance, winning 100–0 against the previously published, champion-defeating AlphaGo.

AlphaGo Zero (AGZ) at Wikipedia

From Wikipedia, the free encyclopedia

AlphaGo Zero is a version of DeepMind's Go software AlphaGo. AlphaGo's team published an article in the journal *Nature* on 19 October 2017, introducing AlphaGo Zero, a version created without using data from human games, and stronger than any previous version.[1] By playing games against itself, AlphaGo Zero surpassed the strength of AlphaGo Lee in three days by winning 100 games to 0, reached the level of AlphaGo Master in 21 days, and exceeded all the old versions in 40 days.[2]

Training artificial intelligence (AI) without datasets derived from human experts has significant implications for the development of AI with superhuman skills because expert data is "often expensive, unreliable or simply unavailable."[3] Demis Hassabis, the co-founder and CEO of DeepMind, said that AlphaGo Zero was so powerful because it was "no longer constrained by the limits of human knowledge".[4] David Silver, one of the first authors of DeepMind's papers published in *Nature* on AlphaGo, said that it is possible to have generalised AI algorithms by removing the need to learn from humans.[5]

In December 2017, a generalized version of AlphaGo Zero, named AlphaZero, beat the 3-day version of AlphaGo Zero by winning 60 games to 40, and with 8 hours of training it outperformed AlphaGo Lee on an Elo scale, as well as a top chess program (Stockfish) and a top Shōgi program (Elmo).[6][7]

AlphaGo Zero (AGZ) at Wikipedia

From Wikipedia, the free encyclopedia

AlphaGo Zero is a version of DeepMind's Go software AlphaGo. AlphaGo's team published an article in the journal *Nature* on 19 October 2017, introducing AlphaGo Zero, a version created without using data from human games, and stronger than any previous version.[1] By playing games against itself, AlphaGo Zero surpassed the strength of AlphaGo Lee in three days by winning 100 games to 0, reached the level of AlphaGo Master in 21 days, and exceeded all the old versions in 40 days.[2]

Training artificial intelligence (AI) without datasets derived from human experts has significant implications for the development of AI with superhuman skills because expert data is "often expensive, unreliable or simply unavailable."[3] Demis Hassabis, the co-founder and CEO of DeepMind, said that AlphaGo Zero was so powerful because it was "no longer constrained by the limits of human knowledge".[4] David Silver, one of the first authors of DeepMind's papers published in *Nature* on AlphaGo, said that it is possible to have generalised AI algorithms by removing the need to learn from humans.[5]

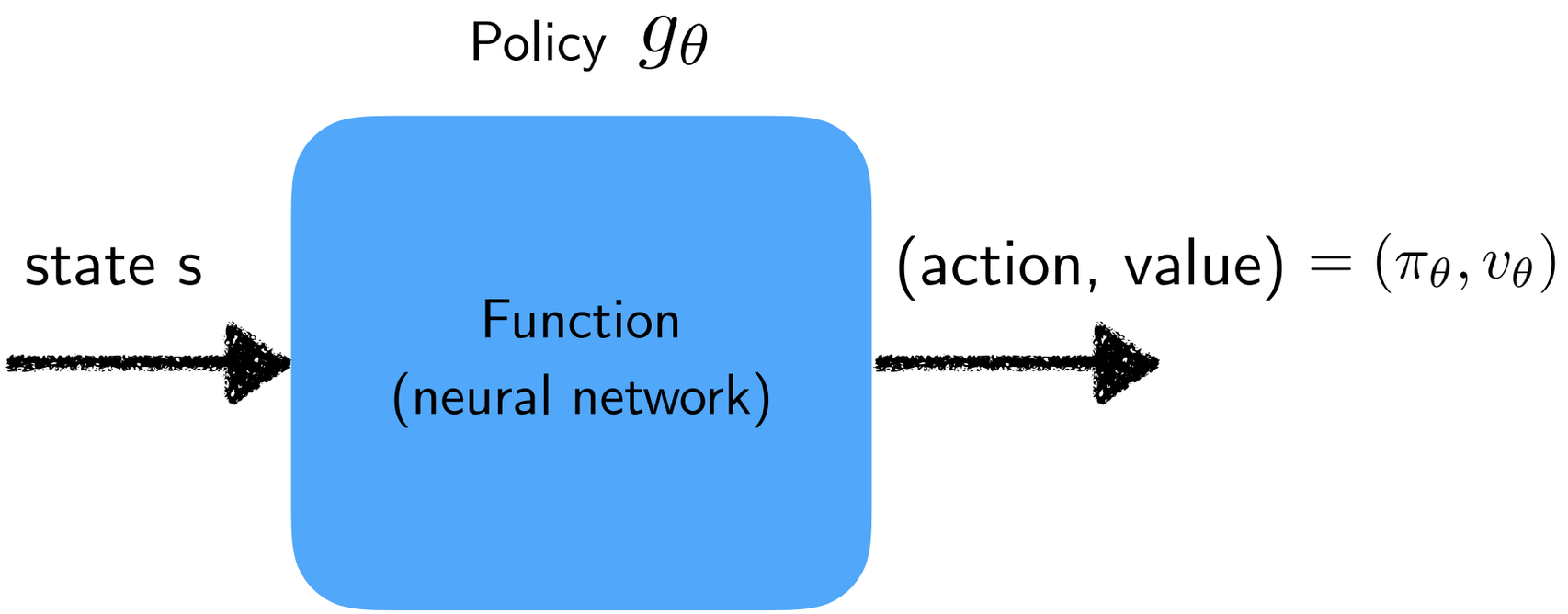
In December 2017, a generalized version of AlphaGo Zero, named AlphaZero, beat the 3-day version of AlphaGo Zero by winning 60 games to 40, and with 8 hours of training it outperformed AlphaGo Lee on an Elo scale, as well as a top chess program (Stockfish) and a top Shōgi program (Elmo).[6][7]

AlphaGo Zero (AGZ)

- Summary
  - World class policy can be learnt for two-player games (Go, Chess, ...)
  - By simply starting with *rules of the game*
- Questions
  - Why does it work and what's new?
    - Restricted to two player games?
    - Is it about the policy?
    - At core, what question is it solving? what is the "framework"?
  - When should it *not* work?

Policy of AGZ

- Learning policy through supervised learning



- Key questions:
    - How to generate training data

Policy of AGZ

- Training data
  - Monte-Carlo Tree Search (MCTS)
    - Finds approximately optimal action
    - Through simulating potential future outcomes for each possible action in "decision tree" form
      - To make most out of available resources (simulation steps) need to explore the "paths" in "decision tree" carefully
- Self-play
  - Good opponent will use the same policy as you!
  - Use policy learnt thus far to simulate future outcomes in MCTS

Policy of AGZ

- In summary, key ingredients
    - Learn policy using
        - Monte-Carlo Tree Search (MCTS)
        - And, (expressive enough) Supervised Learning (SL)
    - Simulation for training
        - Self-play

- Questions:
    - Do MCTS + SL learn policy well?
    - What policy are they learning under Self-play?

Part 1. Monte-Carlo Tree Search

The Formal Setup

- Infinite Horizon Discounted MDP (S, A, P, R, γ)
  - S : set of states
  - A : set of actions
  - $P(s' \mid s, a)$: transition kernel; the probability of transiting to s' by taking an action a at state s ; we'll assume *deterministic* (as in games)
  - $R(s, a)$: random reward.
  - $\gamma \in [0, 1)$: discount factor
- Policy $\pi(\cdot \mid s)$: probability of taking an action
- Value $V^{\pi}(s)$: expected discounted sum of rewards received following the policy π from initial state s

$$V^{\pi}(s) = \mathbb{E}_{\pi}\Big[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) | s_0 = s\Big]$$

optimal value

$$V^{*}(s) = \sup_{\pi} V^{\pi}(s)$$

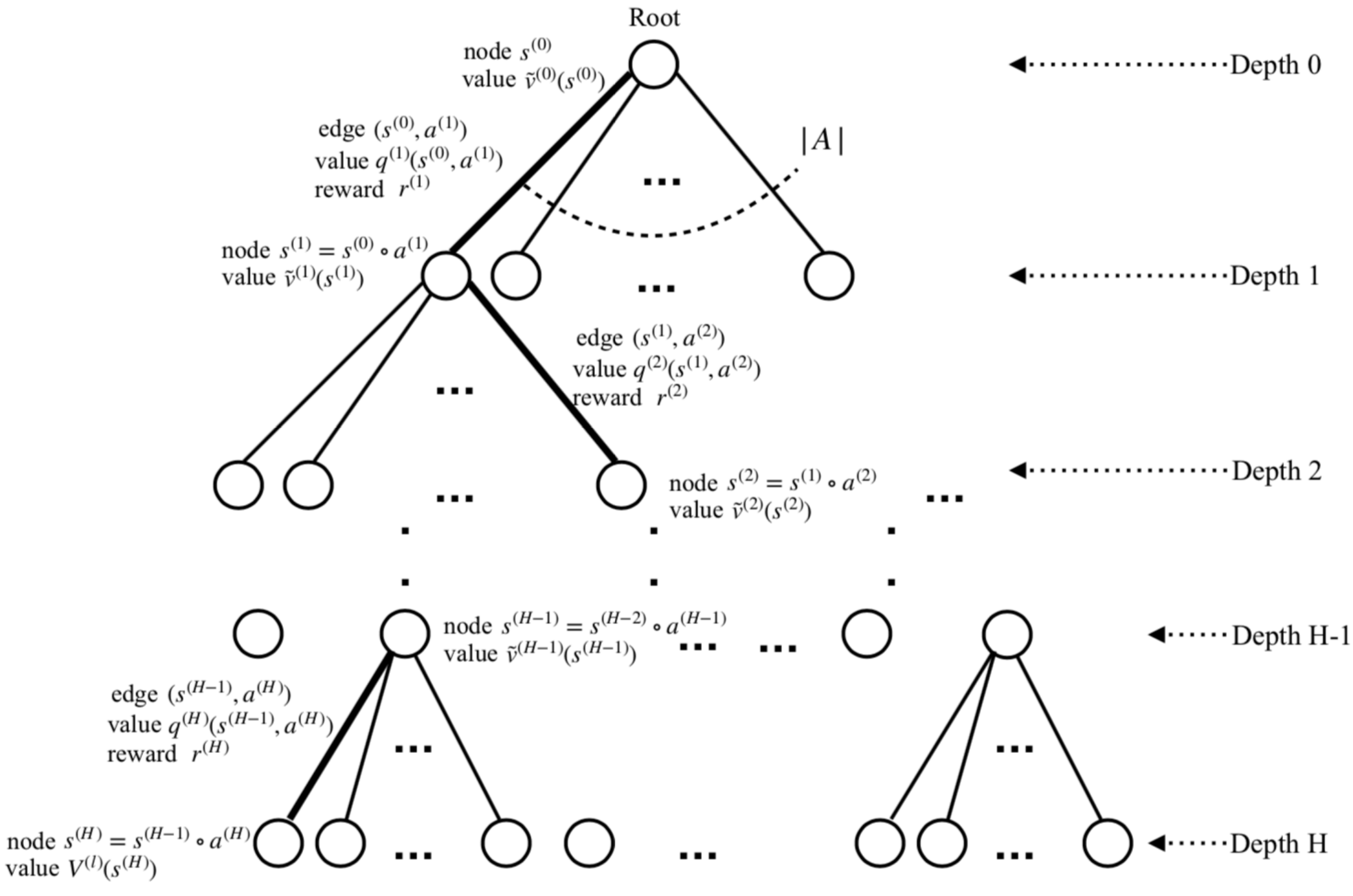
Monte Carlo Tree Search (MCTS)

- Monte Carlo simulation with infinite samples
  - Given state s, find the *optimal* action to take
  - Suppose reward of interest were only *immediate*, i.e. $\gamma = 0$
  - Then, for each feasible action
    - observe the (immediate, average) reward obtained
    - by sampling it potentially infinitely many times
- But we have reward that depends on
  - *immediate* and *future* action
  - more generally, all the future states and actions

Monte Carlo Tree Search (MCTS)

- Monte Carlo simulation with infinite samples (continued)
  - So why not repeat the sampling processing, that is
    - sample immediate action
    - observe next state, and sample action for it
    - and continue for *infinite* future
  - At each stage (i.e. future time step)
    - sample each action *infinitely* many times
  - Finally, add up rewards obtained along each potential future path
  - Take immediate action per path that provides best long-term reward!
- In effect, suggested by [Chang-Fu-Hu-Marcus, Operations Research 2005]

Monte Carlo Tree Search (MCTS)



Monte Carlo Tree Search (MCTS)

- Two issues with the "ideal" scenario we considered
    - One. Sampling for infinite future
    - Two. Infinite samples required for each stage
- Addressing first challenge
    - Truncate, i.e. stop at some large enough depth
    - And use the a value function estimation at leaves (including 0)
- Addressing second challenge
    - Use finitely many samples
    - And use them cleverly amongst different paths

Monte Carlo Tree Search (MCTS)

- MCTS with finite samples
  - Suppose reward of interest were only *immediate*
  - That is, at a given state s
    - reward obtained by playing action a is R(s, a)
  - Therefore, the best action a given state s is simply

$$\arg\max \mathbb{E}[R(s, a)]$$

  - This is the classical stochastic Multi-Arm Bandit (MAB)
  - Therefore, in this simple case
    - We can use policy for stochastic MAB

Stochastic Multi-Arm Bandit

Stochastic Multi-Arm Bandit

- Upper confidence bound

Stochastic Multi-Arm Bandit

- Upper confidence bound
    - Given A different arms: 1, ..., A

Stochastic Multi-Arm Bandit

- Upper confidence bound
    - Given A different arms: 1, ..., A
    - Given t-1 observations thus far

Stochastic Multi-Arm Bandit

- Upper confidence bound
  - Given A different arms: 1, ..., A
  - Given t-1 observations thus far
  - Define *index* of arm i as

$$\hat{\mu}_i^{T_i(t-1)} + B_{t,T_i(t-1)}$$

Stochastic Multi-Arm Bandit

- Upper confidence bound
  - Given A different arms: 1, ..., A
  - Given t-1 observations thus far
  - Define *index* of arm i as

# of times arm i is selected thus far

$$\hat{\mu}_i^{T_i(t-1)} + B_{t,T_i(t-1)}$$

empirical mean

bonus term reflecting probabilistic concentration

$$\mathbb{P}\left(\mu_i \geq \hat{\mu}_i^{T_i(t-1)} + B_{t,T_i(t-1)}\right) \leq \frac{1}{t}$$

Stochastic Multi-Arm Bandit

- Upper confidence bound
  - Given A different arms: 1, ..., A
  - Given t-1 observations thus far
  - Define *index* of arm i as

# of times arm i is selected thus far

$$\hat{\mu}_i^{T_i(t-1)} + B_{t,T_i(t-1)}$$

empirical mean

bonus term reflecting probabilistic concentration

$$\mathbb{P}\left(\mu_i \geq \hat{\mu}_i^{T_i(t-1)} + B_{t,T_i(t-1)}\right) \leq \frac{1}{t}$$

  - At time t, select arm that maximizes the index

Monte Carlo Tree Search (MCTS) [Kocsis-Szepesvari 2006]

Monte Carlo Tree Search (MCTS) [Kocsis-Szepesvari 2006]

- When rewards are independent (and identical per arm) random variables

$$B_{t,s} \sim \sqrt{\frac{\log t}{s}}$$

Monte Carlo Tree Search (MCTS) [Kocsis-Szepesvari 2006]

- When rewards are independent (and identical per arm) random variables

$$B_{t,s} \sim \sqrt{\frac{\log t}{s}}$$

- Inspired by this, Monte Carlo Tree Search policy
  - known as Upper Confidence bound for Trees (UCT)
  - suggests use of index of the form at each stage in the tree

empirical mean + bonus term

of reward along the best future path computed recursively by viewing immediate future step as another MAB instance

of the above form

Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS)

- Bonus term is *tricky*
  - if we assume independence of reward across stages of MCTS tree

$$B_{t,s} \sim \sqrt{\frac{\log t}{s}}$$

  - now, independence is *not true*
    - the reward function beyond *leaf* level of tree in effect depends on
      - the regret of MAB corresponding to node a layer below
    - seems that required *exponential* concentration for regret is unlikely to hold in general beyond special condition [cf. Audibert et al 2009]
    - indeed, there is *no correct* proof of UCT by [Kocsis-Szepesvari 2006]
    - btw, what does AGZ do?

Monte Carlo Tree Search (MCTS)

- Bonus term is *tricky*
  - UCT by [Kocsis-Szepesvari 2006]

$$B_{t,s} \sim \sqrt{\frac{\log t}{s}}$$

  - AGZ

$$B_{t,s} \sim \frac{\sqrt{t}}{s}$$

- We formally find that a better (?) choice is

$$B_{t,s} \sim \sqrt{\frac{\sqrt{t}}{s}}$$

Non-Stationary Multi-Arm Bandit

Non-Stationary Multi-Arm Bandit

- Let reward for each arm satisfies: for some $\beta > 1,\ \zeta > 0,\ \frac{1}{2} \leq \eta < 1$

$$\mathbb{P}\Big(|n\hat{\mu}_i^n - n\mu_i| \geq n^\eta z\Big) \leq \frac{\beta}{z^\zeta}, \quad \forall\ n \geq 1,\ z \geq 1$$

Non-Stationary Multi-Arm Bandit

- Let reward for each arm satisfies: for some $\beta > 1,\ \zeta > 0,\ \frac{1}{2} \leq \eta < 1$

$$\mathbb{P}\Big(|n\hat{\mu}_i^n - n\mu_i| \geq n^\eta z\Big) \leq \frac{\beta}{z^\zeta},\quad \forall\ n \geq 1,\ z \geq 1$$

- Use UCB with bonus term

$$B_{t,s} = \frac{\beta^{1/\zeta} t^{\eta(1-\eta)}}{s^{1-\eta}}$$

Non-Stationary Multi-Arm Bandit

- Let reward for each arm satisfies: for some $\beta > 1,\ \zeta > 0,\ \frac{1}{2} \le \eta < 1$

$$\mathbb{P}\Big(|n\hat{\mu}_i^n - n\mu_i| \ge n^\eta z\Big) \le \frac{\beta}{z^\zeta}, \quad \forall\ n \ge 1,\ z \ge 1$$

- Use UCB with bonus term

$$B_{t,s} = \frac{\beta^{1/\zeta} t^{\eta(1-\eta)}}{s^{1-\eta}}$$

- Informal Statement of Theorem [Shah-Xie-Xu 2019].
  - Let $\hat{\mu}^n$ be empirical average of reward collected in *n* steps of algorithm
  - Let $\mu^\star = \max_i \mu_i$
  - Then $\mathbb{P}\Big(|n\hat{\mu}^n - n\mu^\star| \ge n^\eta z\Big) \le \frac{\beta'}{z^{\zeta'}}, \quad \forall\ n \ge 1,\ z \ge 1$
    - for $\zeta' = \zeta\eta(1-\eta) - 1, \quad \beta' > 1$

Non-Stationary Multi-Arm Bandit

- Let reward for each arm satisfies: for some $\beta > 1,\ \zeta > 0,\ \frac{1}{2} \le \eta < 1$

$$\mathbb{P}\Big(|n\hat{\mu}_i^n - n\mu_i| \ge n^\eta z\Big) \le \frac{\beta}{z^\zeta}, \quad \forall\ n \ge 1,\ z \ge 1$$

- Use UCB with bonus term

$$B_{t,s} = \frac{\beta^{1/\zeta} t^{\eta(1-\eta)}}{s^{1-\eta}}$$

- Informal Statement of Theorem [Shah-Xie-Xu 2019].
  - Let $\hat{\mu}^n$ be empirical average of reward collected in *n* steps of algorithm
  - Let $\mu^\star = \max_i \mu_i$
  - Then $\mathbb{P}\Big(|n\hat{\mu}^n - n\mu^\star| \ge n^\eta z\Big) \le \frac{\beta'}{z^{\zeta'}}, \quad \forall\ n \ge 1,\ z \ge 1$
    - for $\zeta' = \zeta\eta(1-\eta) - 1, \quad \beta' > 1$
- An *excellent* choice is $\eta = \frac{1}{2}$

Part 2. Supervised Learning

Supervised Learning

Supervised Learning

- AGZ proposes to use "expressive" neural network

- We shall utilize non-parametric nearest neighbor method
    - manages to learn underlying model (function) as long as its *reasonable*
        - with sample complexity that scales with intrinsic dimension
        - well understood, allows us to focus on the problem at hand

- Recall
    - MCTS uses the learnt value function for leaf node valuation
    - And produces samples for supervised learning to learn

Supervised Learning

Supervised Learning

- Key question
  - does this converge to optimal policy?
  - in particular, does MCTS act as "policy improvement" operator?

Supervised Learning

- Key question
  - does this converge to optimal policy?
  - in particular, does MCTS act as "policy improvement" operator?
- Informal Statement of Theorem [Shah-Xie-Xu 2019].
  - Suppose we have an estimate of value function $\hat{V}$ such that for all *leaf* nodes, s, in MCTS for some $\varepsilon_0 > 0$

$$|\hat{V}(s) - V^\star(s)| \leq \varepsilon_0$$

  - Then, after *n* simulation of the MCTS described earlier, it produces estimate of value function for the given node within error (on average)

$$\delta\varepsilon_0 + O(n^{-1/2})$$

Supervised Learning

- Key question
  - does this converge to optimal policy?
  - in particular, does MCTS act as "policy improvement" operator?
- Informal Statement of Theorem [Shah-Xie-Xu 2019].
  - Suppose we have an estimate of value function $\hat{V}$ such that for all *leaf* nodes, s, in MCTS for some $\varepsilon_0 > 0$

$$|\hat{V}(s) - V^\star(s)| \leq \varepsilon_0$$

  - Then, after *n* simulation of the MCTS described earlier, it produces estimate of value function for the given node within error (on average)

$$\delta\varepsilon_0 + O(n^{-1/2})$$

choice of $\eta = \frac{1}{2}$ leads to this minimal error exponent

Supervised Learning

- Consider states visited by policy based on MCTS + SL
  - In each step, MCTS suggested action is taken with high chance
  - But, a random action is taken with small chance
- Let the underlying MDP be "regular enough"
  - State space be d dimensional unit cube (for simplicity)
- Informal Statement of Theorem [Shah-Xie-Xu 2019].
  - Given $\varepsilon > 0$, algorithm finds $\hat{V}$, an estimate of value function such that

$$\mathbb{E}[\|\hat{V} - V^\star\|_\infty] \leq \varepsilon$$

  - using total samples $\widetilde{O}\left(\varepsilon^{-d+4}\right)$
- Nearly matches minimax lower bound $\widetilde{\Omega}\left(\varepsilon^{-d+2}\right)$ for any policy

Part 3. Self-Play

Two-player game, Self-play and Training Data

- Game tuple: $(\mathcal{S}, \mathcal{A}, r, \gamma, P)$
  - States (including turn of player) $\mathcal{S}$
  - Actions (for both players) $\mathcal{A}$
  - Reward $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
  - Discount $\gamma \in [0, 1]$
  - Probability transition Kernel $P(s'|s, a) \in [0, 1],\ s, s' \in \mathcal{S}, a \in \mathcal{A}$
- Policy: $\pi = (\pi_1, \pi_2)$ where $\pi_i(\cdot|s)$ is distribution over $\mathcal{A}$ for all $s \in \mathcal{S}$
- Value and Reward:

$$V_{\pi_1,\pi_2}(s) = \mathbb{E}_{a_t, s_{t+1}, a_{t+1}, \ldots}\left[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k})|s_t = s\right],$$

$$Q_{\pi_1,\pi_2}(s, a) = \mathbb{E}_{s_{t+1}, a_{t+1}, \ldots}\left[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k})|s_t = s, a_t = a\right].$$

Two-player game, Self-play and Training Data

- Zero-sum:
    - Reward earned by player 2 is negative of reward earned by player 1

$$r^1(s,a) = -r^2(s,a)$$

    - We will consider reward of player 1 as *the reward (reference)*
        - Player 1: maximizer, Player 2: minimizer

- Self-play:
    - Player 2 is the attempting to play the *worst* possible policy
        - Given choice of policy for player 1

Optimal policy, Nash equilibrium

- Optimal Counter Policy:
  - policy of player 1, π_1 is optimal counter policy for player 2's policy π_2
  - if $V_{\pi_1,\pi_2} \geq V_{\pi_1',\pi_2}, \quad \text{for all } \pi_1'$
  - and similarly for π_2 with respect to π_1 (but inequality *reversed*)

- For $\gamma < 1$:
  - exists a unique, simultaneously optimal counter policies [Hansen 13]

$$\pi^\star = (\pi_1^\star, \pi_2^\star)$$

  - it is also Nash equilibrium for the game

- Interest is in finding this policy using supervised learning

Learning Optimal Counter Policy

- Initialize
    - Value function (and Q-function, equivalently policy) estimate
- Iteratively improve upon it
    - Using MCTS produce new estimate of Value function at given state
        - with help of existing value function estimate
        - and, best counter policy at each step for simulating "model"
    - Using thus generated sample(s)
        - learn updated Value function (and Q-function)

Learning Optimal Counter Policy

- Policy to transition between states while learning
  - Given state s, action a and estimate $\hat{Q}$

$$\hat{\pi}(a|s) \propto \exp\left(\hat{Q}(s,a)/\tau\right)$$

  - In above "Boltzmann policy", temperature parameter $\tau > 0$
- This choice of policy encourages exploration
  - while retaining good performance
- Proposition.
  - If $\mathbb{E}[|\hat{Q}(a,s) - Q^\star(a,s)|] \leq \varepsilon,\ \forall\ a$
  - Then $\mathbb{E}[\mathrm{D}_{\mathrm{KL}}(\hat{\pi}(\cdot|s), \pi^\star(\cdot|s))] \leq \dfrac{2|A|}{\tau}\varepsilon$

Learning Optimal Counter Policy

- Use of expressive enough supervised learning
    - generalizes well with sufficiently many samples

- Then, the iterative algorithm where
    - each iteration consists of visiting multiple states
        - at each state, MCTS with Boltzmann policy is utilized
        - the samples generated are used to update
            - Value (and Q-)function estimates

- Obtains ε approx. of Value function after $O\big(\log 1/\varepsilon\big)$ iterations

Part 4. Beyond Games

Beyond games, Robust MDP

Beyond games, Robust MDP

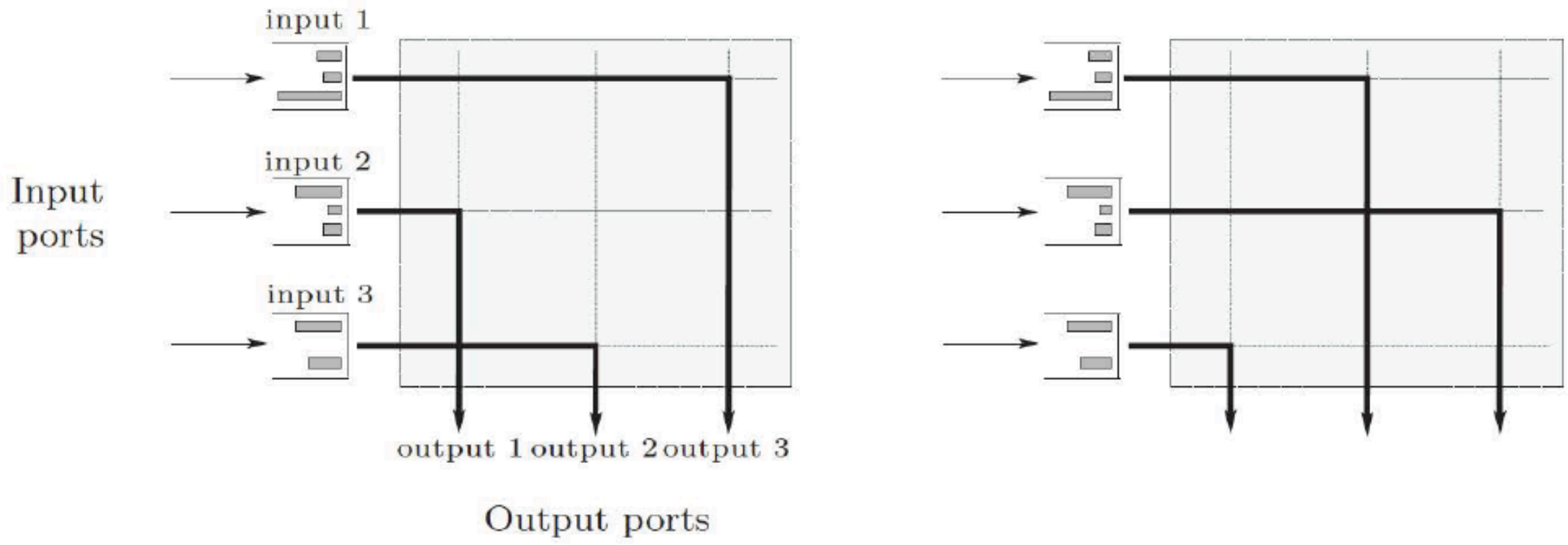
- Thus far, we discussed AGZ for two-player zero-sum

- A natural transformation to extend for generic Decision Process
    - By viewing generic decision making as a two-player game

- Specifically
    - Player 1 = agent, Player 2 = nature
    - Constraints (on nature) play crucial role in modeling
    - Idealized policy is Nash equilibrium
    - And, AGZ policy extends naturally

Data-free Reinforcement Learning

- Example of Switch Scheduling



$$Q_{ij}(t+1) = \max\left\{Q_{ij}(t) - \sigma_{ij}(t), 0\right\} + \text{Ber}(\lambda_{ij}(t))$$

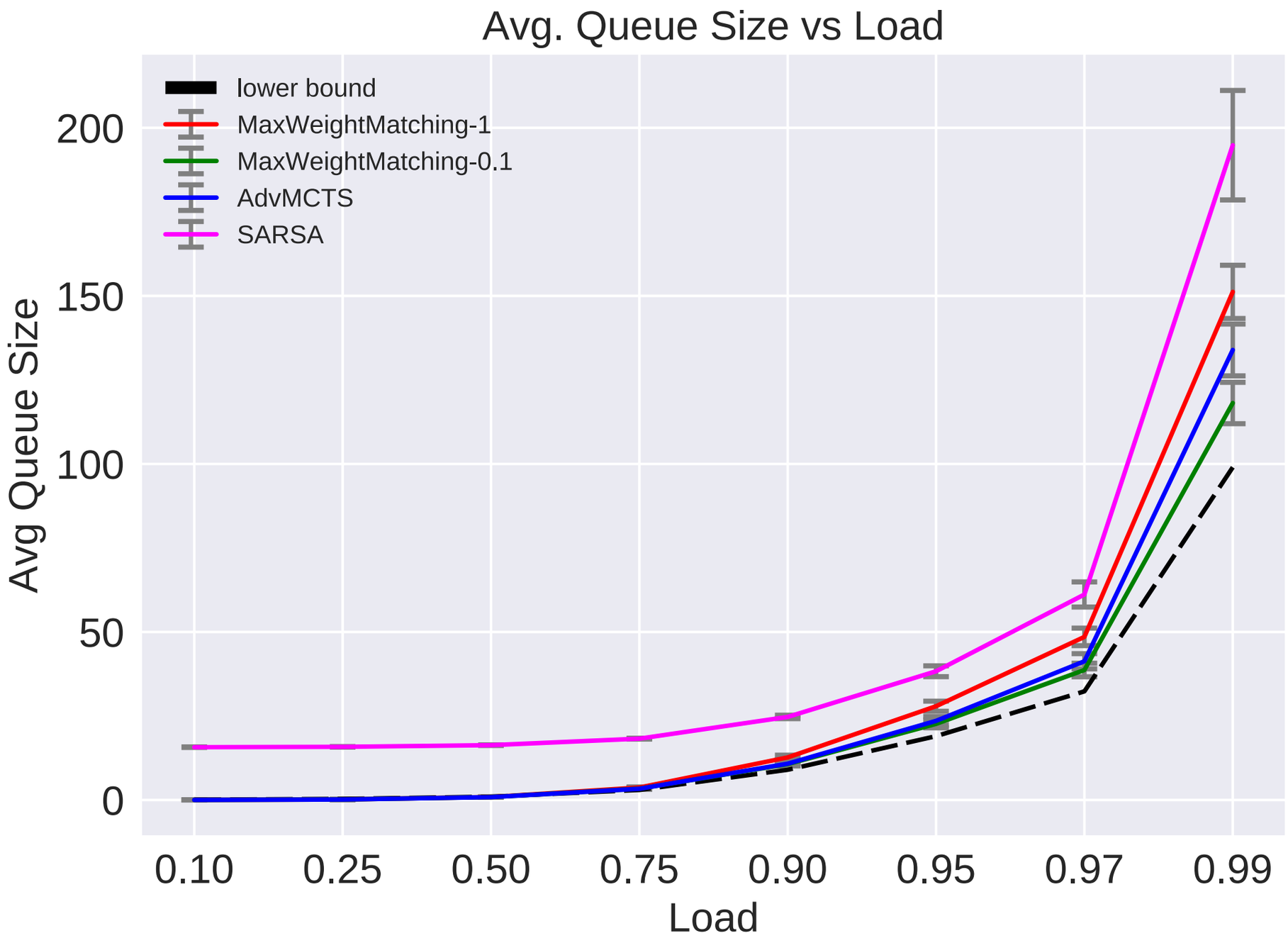
$$r(S, A) \triangleq -\sum_{i,j}(q_{ij} \text{ of state } S').$$

Data-free Reinforcement Learning

- Example of Switch Scheduling

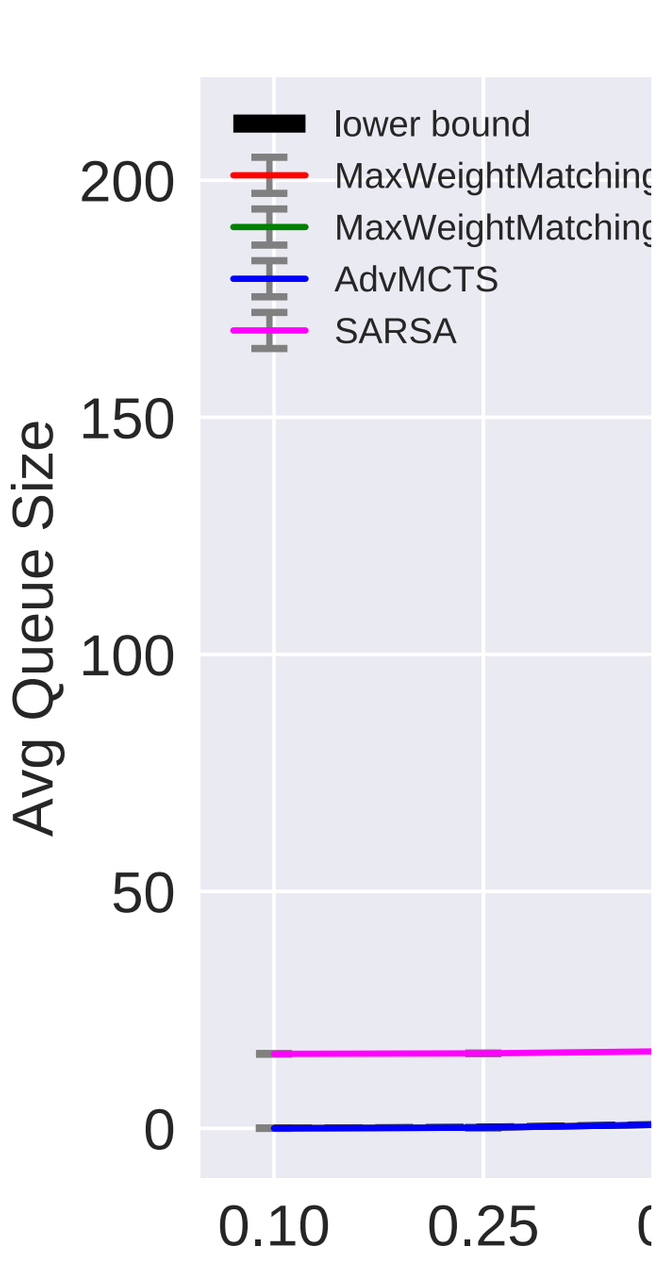
Data-free Reinforcement Learning

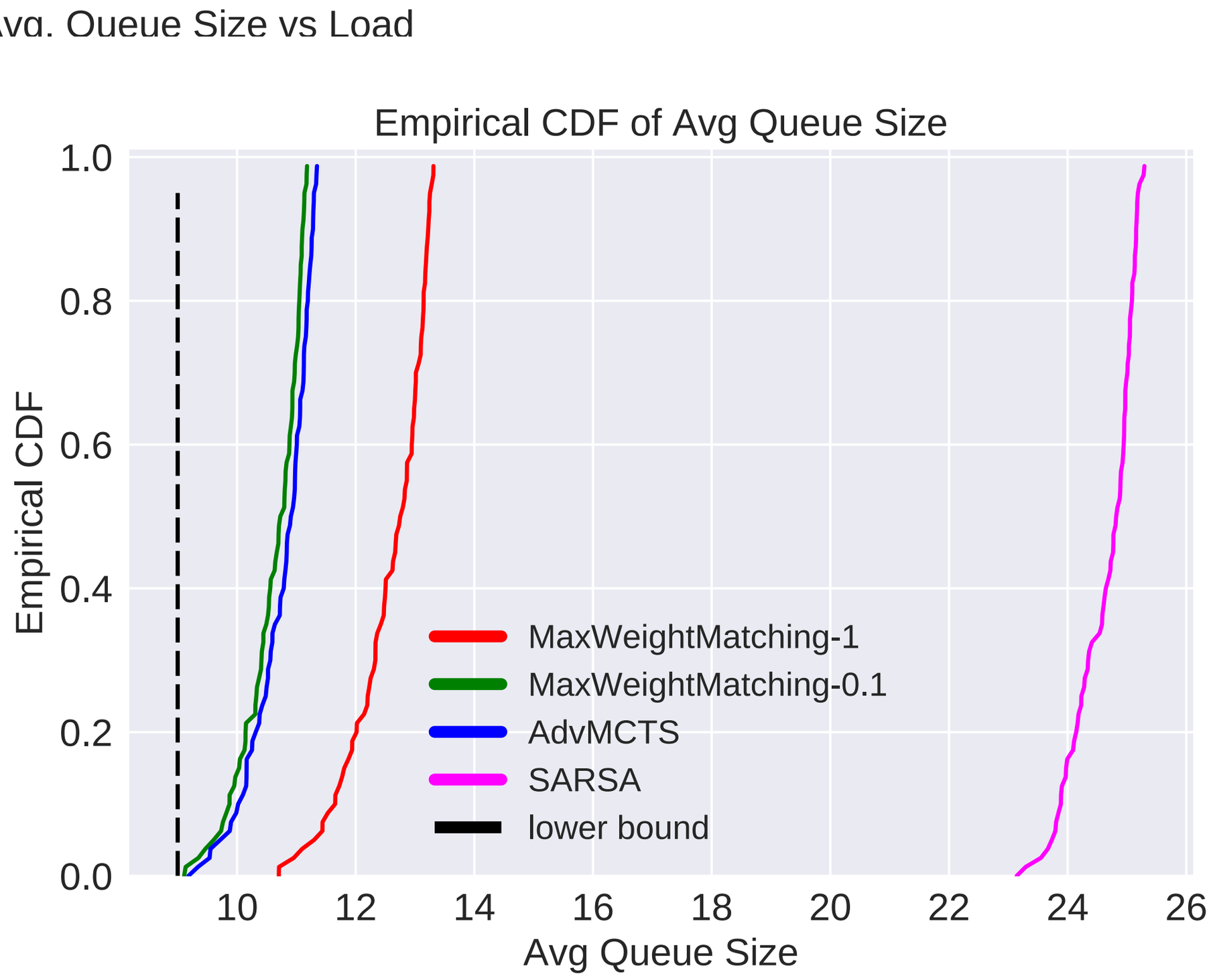
- Example of Switch Scheduling



Data-free Reinforcement Learning

- Example of Switch Scheduling





Part 5. We'r Done

Parting Remarks

Parting Remarks

- Successes like AGZ need explanation

- Key components behind AGZ's success
    - Use of MCTS with Supervised Learning
        - Acts as a policy improvement operator
        - Nearly achieves minimax optimality
    - Self-play is looking for Nash equilibrium
        - Might be applicable beyond Games

- There is something special about MCTS + SL
    - Beyond standard RL approaches (e.g. Q learning)