Monte Carlo Tree Search

Exploration reduces uncertainty

Model-free setting

- Unknown environment dynamics $p(s', r|s, a)$

Model-based setting

- Known model of the world
  - Distribution model: $p(s', r|s, a)$
  - Sample model: $(s', r) \backsim G(s, a)$

Do we need exploration for model-based learning?

Exploration reduces uncertainty

Model-free setting

- Unknown environment dynamics $p(s', r|s, a)$

Model-based setting

- Known model of the world
  - Distribution model: $p(s', r|s, a)$
  - Sample model: $(s', r) \backsim G(s, a)$

Do we need exploration for model-based learning?

Exploration reduces uncertainty

Model-free setting

- Unknown environment dynamics $p(s', r|s, a)$

Model-based setting

- Known model of the world
  - Distribution model: $p(s', r|s, a)$
  - Sample model: $(s', r) \backsim G(s, a)$

Do we need exploration for model-based learning?

Still don't know the global values of any action!

We don’t know global value estimates

Given the model of the world, what is the best action?

1. Immediate rewards are **insufficient**
2. **Time constraints** on recovering sum of future rewards

We don't know global value estimates

Given the model of the world, what is the best action?

1. Immediate rewards are **insufficient**
2. **Time constraints** on recovering sum of future rewards

Planning

Transform

- known immediate rewards & dynamics

  into

- value / action-value estimates / explicit policy

Types of planning

Model-free – **learn** the policy by trial and error

Model-based – **plan** to obtain the policy

- Background planning (DP)
  - learning from simulated experience
  - improves estimates in arbitrary states

Types of planning

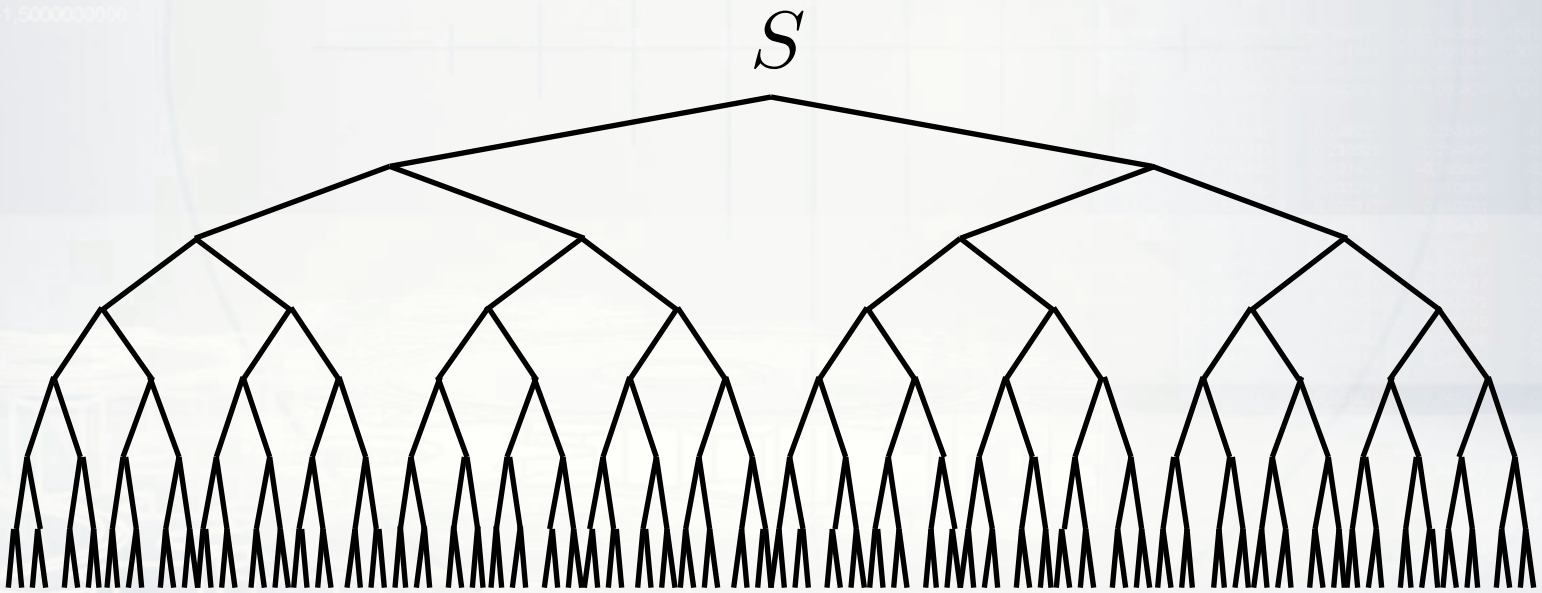
Model-free – **learn** the policy by trial and error

Model-based – **plan** to obtain the policy

- Background planning (DP)
  - learning from simulated experience
  - improves estimates in arbitrary states
- Decision-time planning (heuristic search, MCTS)
  - focuses on current state
  - plans action selection for current state only
  - commit action only after planning is finished

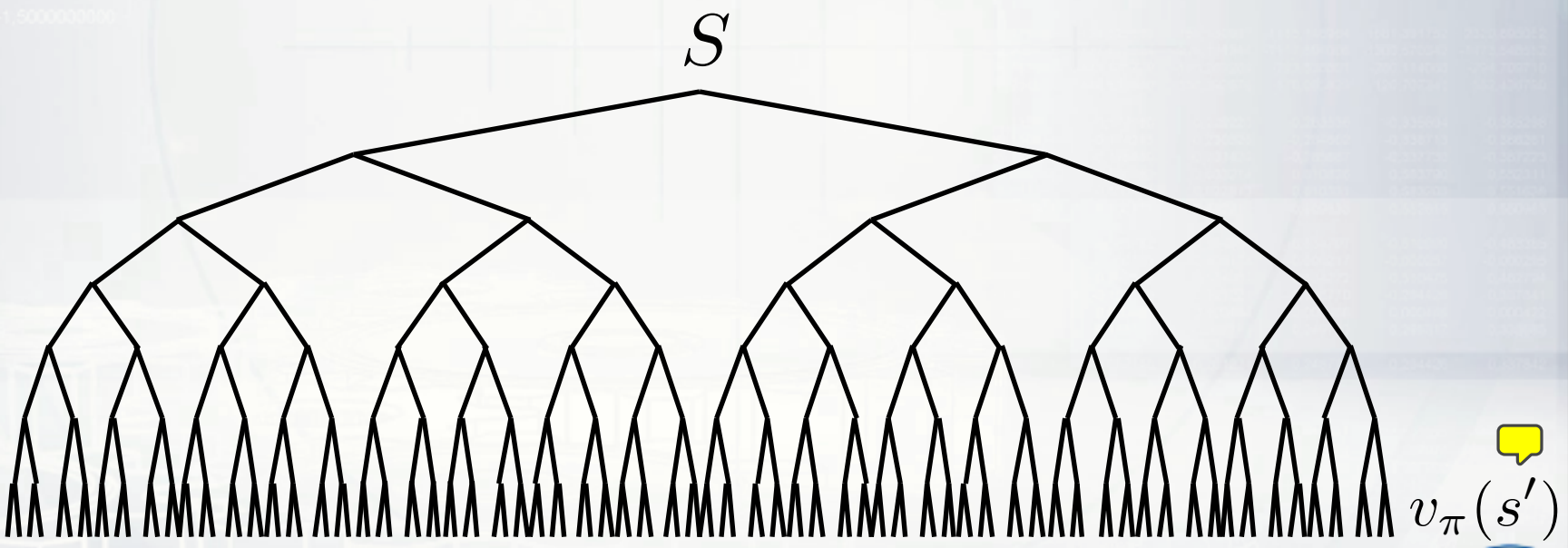
Guided planning – heuristic search

1. Exhaustive



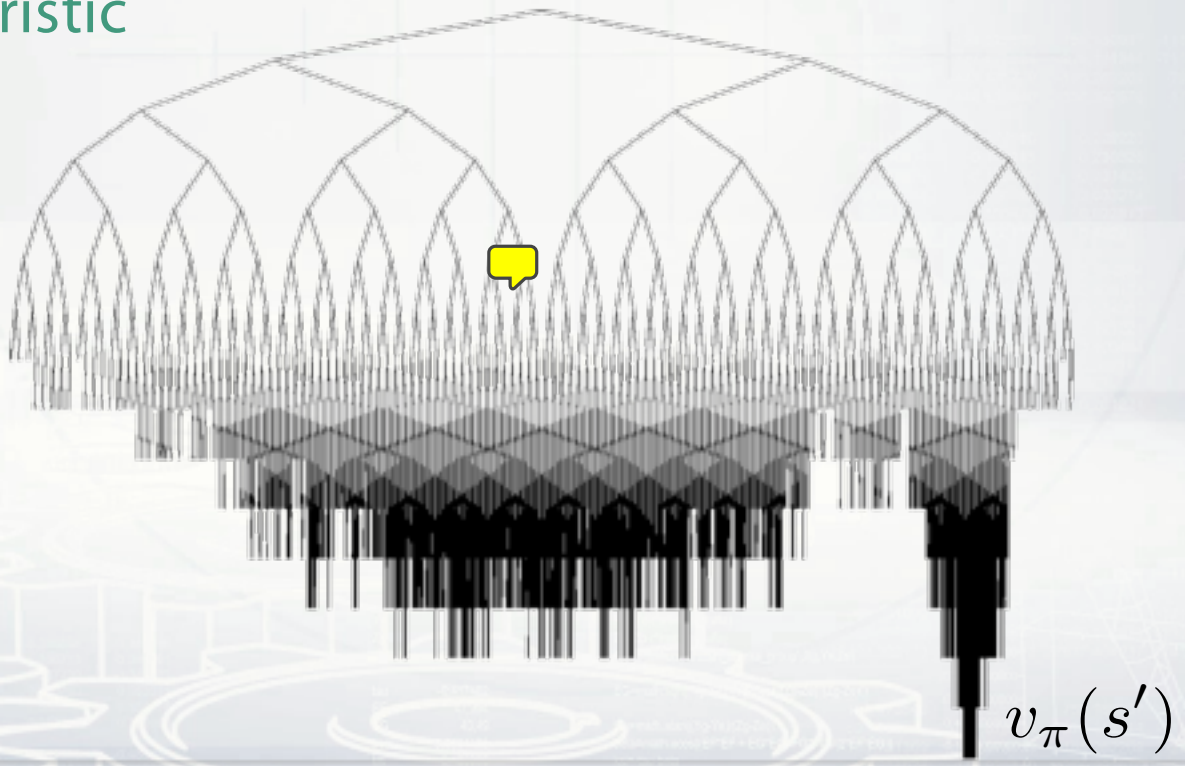
Guided planning – heuristic search

1. Exhaustive
2. Full-width with fixed depth
3. Full-width with fixed depth, approximate at leaves



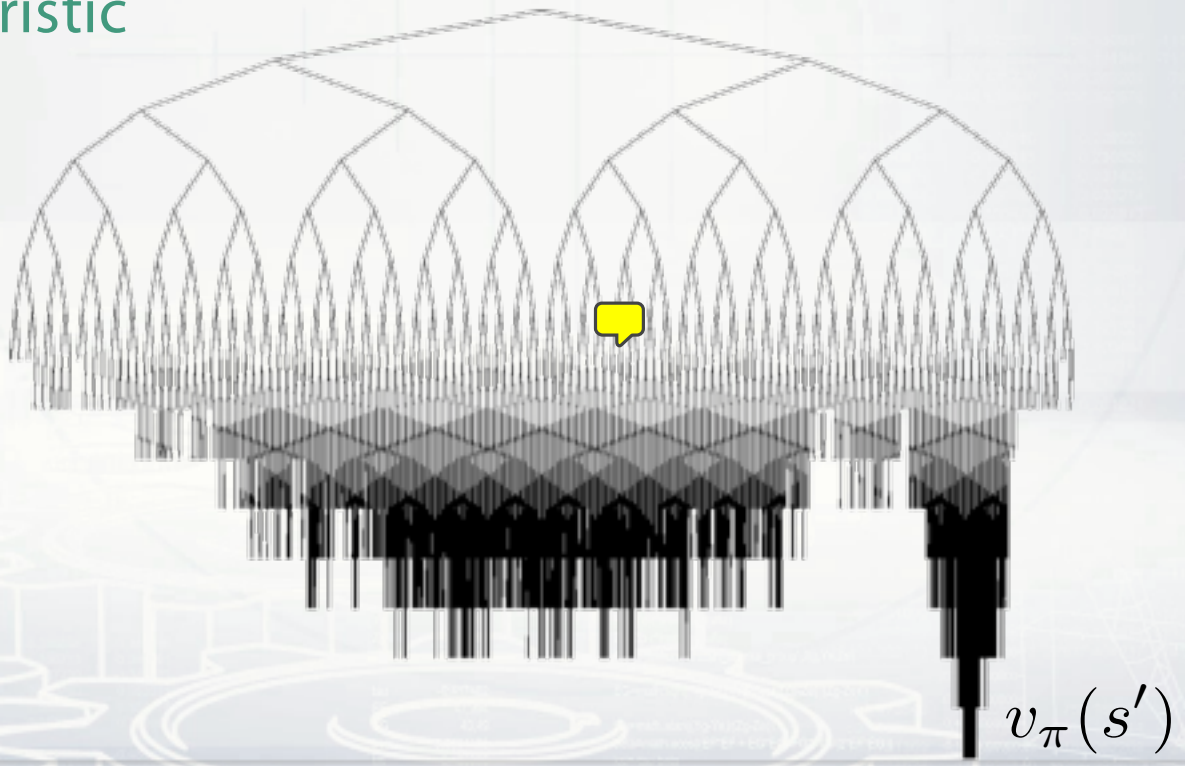
Guided planning – heuristic search

1. Exhaustive
2. Full-width with fixed depth
3. Full-width with fixed depth, approximate at leaves
4. Heuristic



Guided planning – heuristic search

1. Exhaustive
2. Full-width with fixed depth
3. Full-width with fixed depth, approximate at leaves
4. Heuristic



Guided planning – heuristic search

Heuristic search – an umbrella term for many algorithms

- lookahead planning guided by heuristic function
  - select only “valuable” nodes to expand

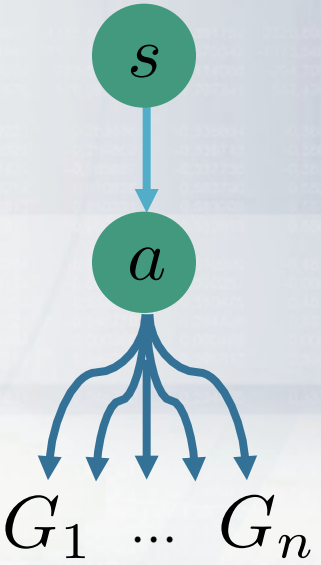
Heuristic search for RL

+ resources are focused only on valuable paths
+ nearest states contribute the most to the return
– bad planning results when the model is unreliable
– dependence on the quality of heuristic

Obtain value estimates with rollouts

Obtain value estimates with rollouts

1. While within computational budget:
   a. From state S make action A
   b. Follow rollout policy until episode terminates
   c. Obtain return G_i
2. Return $Q(s, a) = \frac{1}{n} \sum_{i=1}^{n} G_i$



+ No function approximation
+ Trials are independent → parallelization
– Dependence on # of returns averaged

Using rollouts for action selection

Greedy policy – policy improvement as in Value Iteration

$$\pi(s) \leftarrow \operatorname{argmax}_a \hat{q}_{rollout}(s, a)$$

- No estimation of q_*
- No further policy improvement

Good rollout policy is needed!

Using rollouts for action selection

Greedy policy – policy improvement as in Value Iteration

$$\pi(s) \leftarrow \text{argmax}_a \hat{q}_{rollout}(s,a)$$

- No estimation of q_*
- No further policy improvement

} Good rollout policy is needed!

- Good rollout policy may require much of valuable time
  - May result in less reliable estimates of $\hat{q}_{rollout}(s,a)$
  - Dependence on quality of rollout policy
  - \+ Quite good results even for **random** rollout policy
  - No estimates preserved between successive states
  - Does not respect uncertainty in action-value estimates

Recap: discussed so far

- Planning: model in, policy out
- Crucial constraint in planning: time
- Accurate planning: guided by heuristic
- Variant of heuristic: MC estimate
- Simplest policy: greedy w.r.t. to heuristic

Not yet covered

- Preserving estimates between successive time steps
- Dealing explicitly with uncertainty in estimates
- Guiding search with the heuristic

Monte Carlo Tree Search at a glance

- Amazingly successful decision-time algorithms
- Distant relatives of heuristic search
- Usually requires much computation time
- Estimate action-values with MC rollouts
- Preserve value estimates between transitions
- Maintain two policies
  - tree policy – respects uncertainty, guides the search
  - rollout policy – fast & as close to optimal as possible

Scheme of Monte Carlo Tree Search

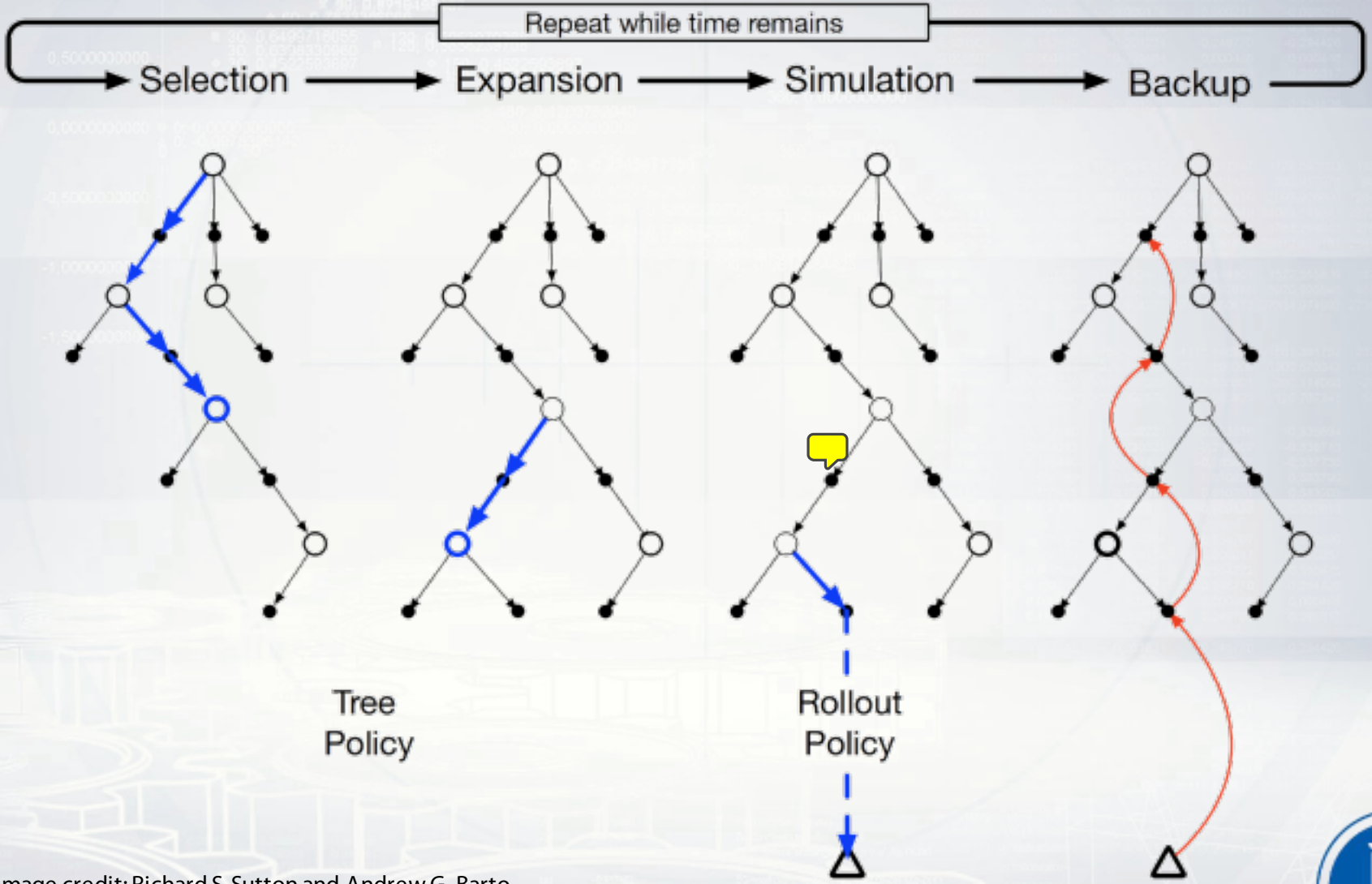


Image credit: Richard S. Sutton and Andrew G. Barto
"Reinforcement Learning: An Introduction", Draft of second edition (2017)

Scheme of Monte Carlo Tree Search

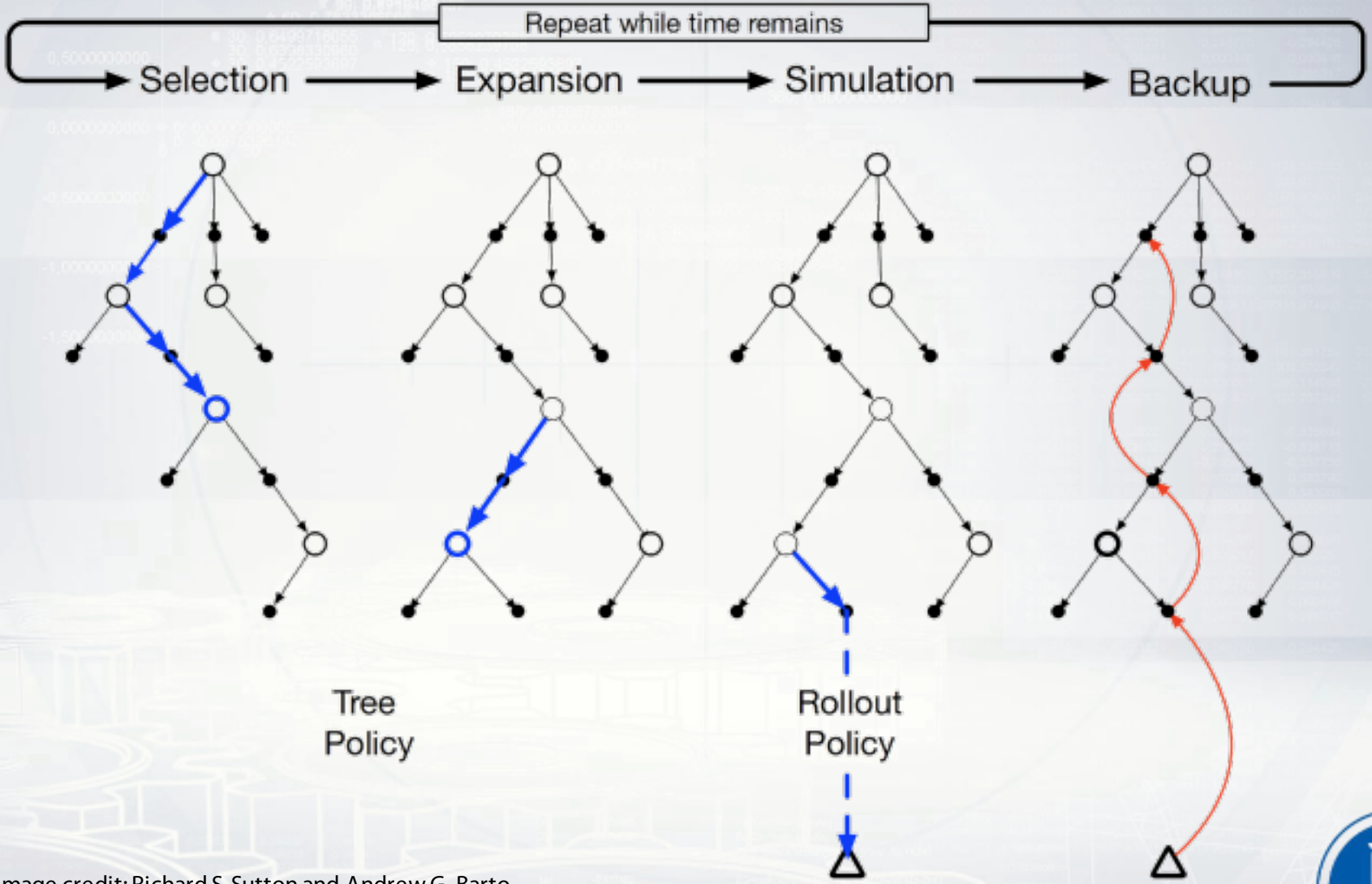


Image credit: Richard S. Sutton and Andrew G. Barto
"Reinforcement Learning: An Introduction", Draft of second edition (2017)

Scheme of Monte Carlo Tree Search

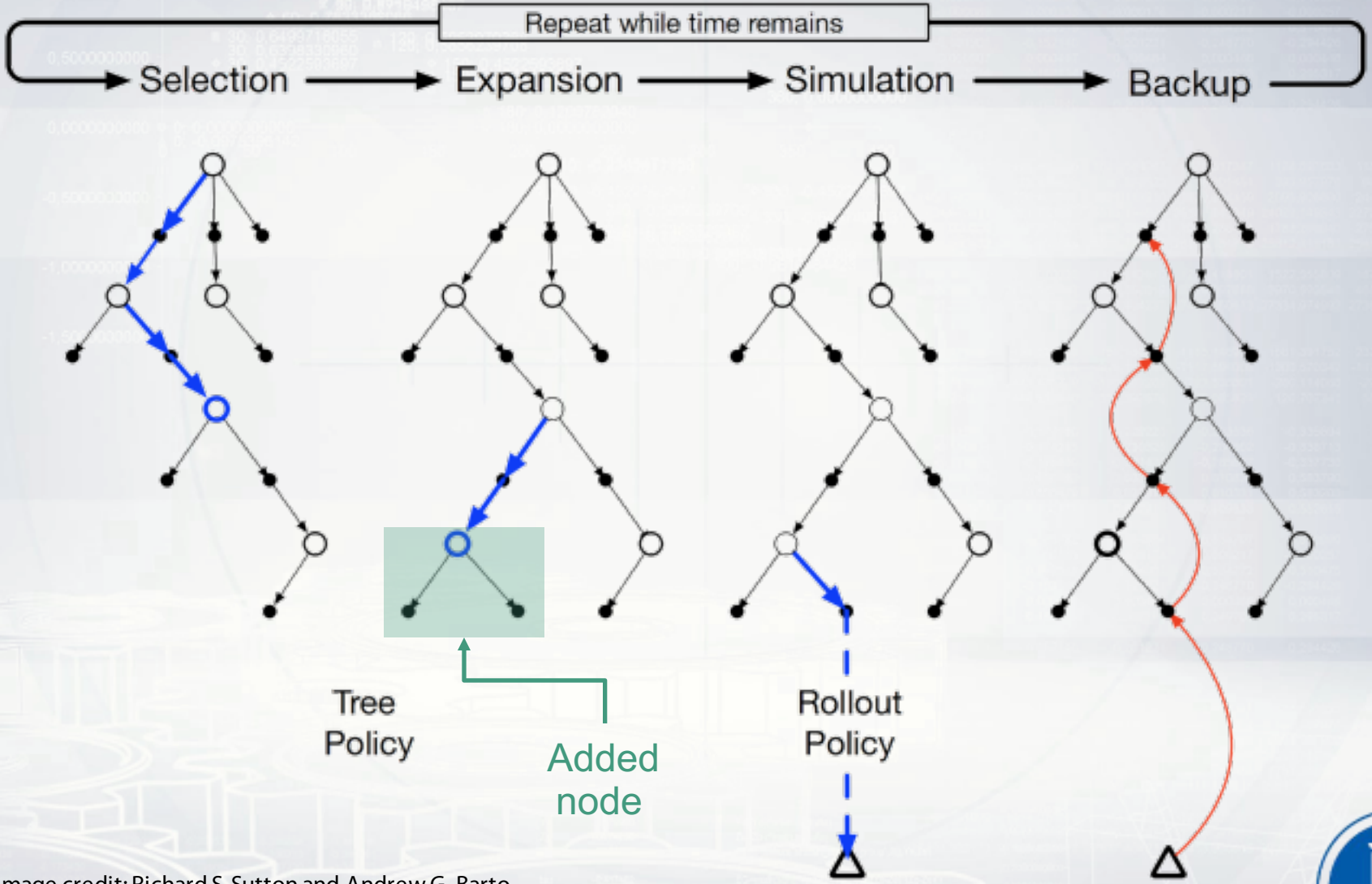


Image credit: Richard S. Sutton and Andrew G. Barto
"Reinforcement Learning: An Introduction", Draft of second edition (2017)

Tree policy: Upper Confidence Bounds for Trees

- Exploitation – actions with existing good estimates
- Exploration – outcomes of rarely tested actions

Tree policy: Upper Confidence Bounds for Trees

- Exploitation – actions with existing good estimates
- Exploration – outcomes of rarely tested actions

Treating action selection as multiarmed bandit (UCB1)

$$\pi_{tree}(s) = \text{argmax}_a \hat{Q}(s,a) + 2C\sqrt{\frac{2\ln N(s)}{N(s,a)}}$$

Tree policy: Upper Confidence Bounds for Trees

- Exploitation – actions with existing good estimates
- Exploration – outcomes of rarely tested actions

Treating action selection as multiarmed bandit (UCB1)

Exploitation

$$\pi_{tree}(s) = \text{argmax}_a \hat{Q}(s,a) + 2C\sqrt{\frac{2\ln N(s)}{N(s,a)}}$$

Exploration

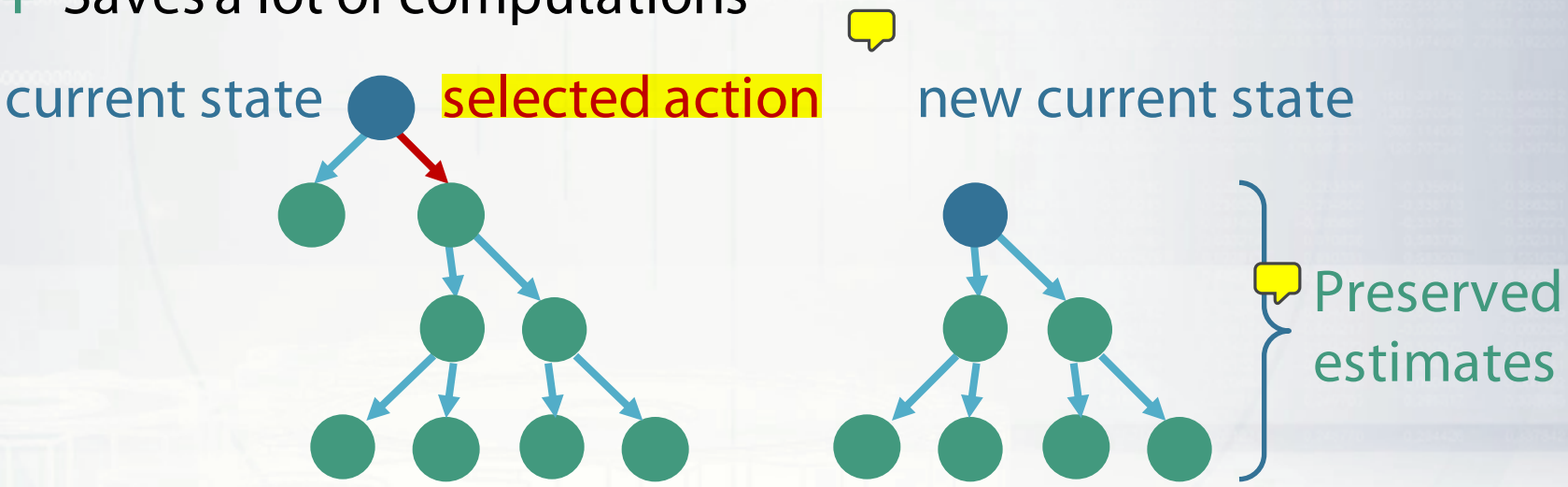
- Good theoretical properties
- Constant C tunes the amount of exploration
- If $N(s,a) = 0$, then action is always selected

Action selection

1. Max – root child with the highest Q
   - Simple and effective
2. Robust – most visited root child
   - Effective against outliers, may be suboptimal
3. Max-robust – both highest visit count & highest Q
   - May require more planning time
4. Secure child – maximizes a lower confidence bound
   - Paranoid mode, real life applicable

MCTS: benefits and drawbacks

+ Context independent – no hand designed heuristic
+ Asymmetric search – more promising directions first
+ Anytime – has the answer if stopped at any time
+ Saves a lot of computations



+ Simple in implementation
– Dependence on quality of rollout policy
– Computationally expensive