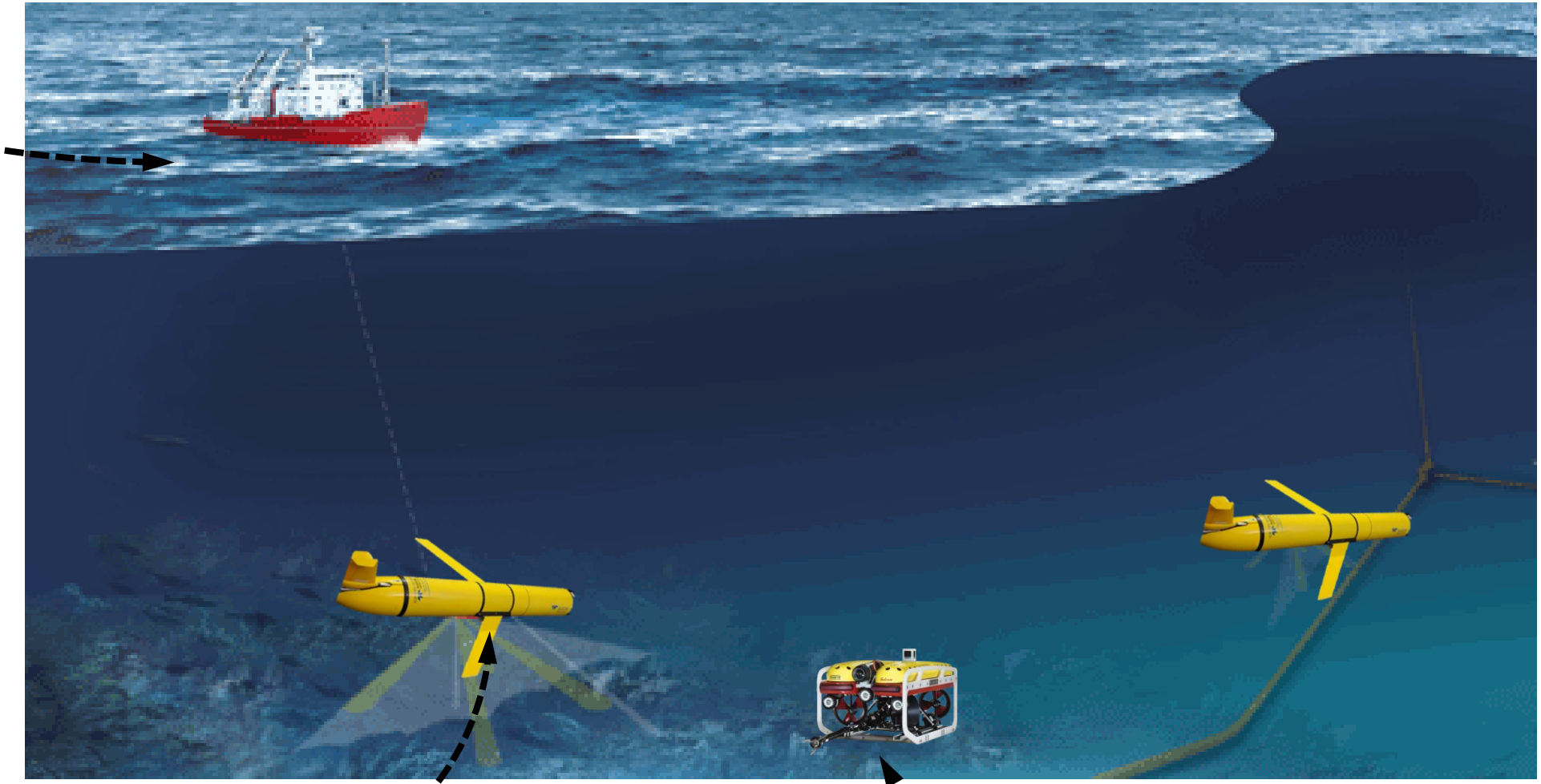
Monte Carlo Tree Search for Collaboration

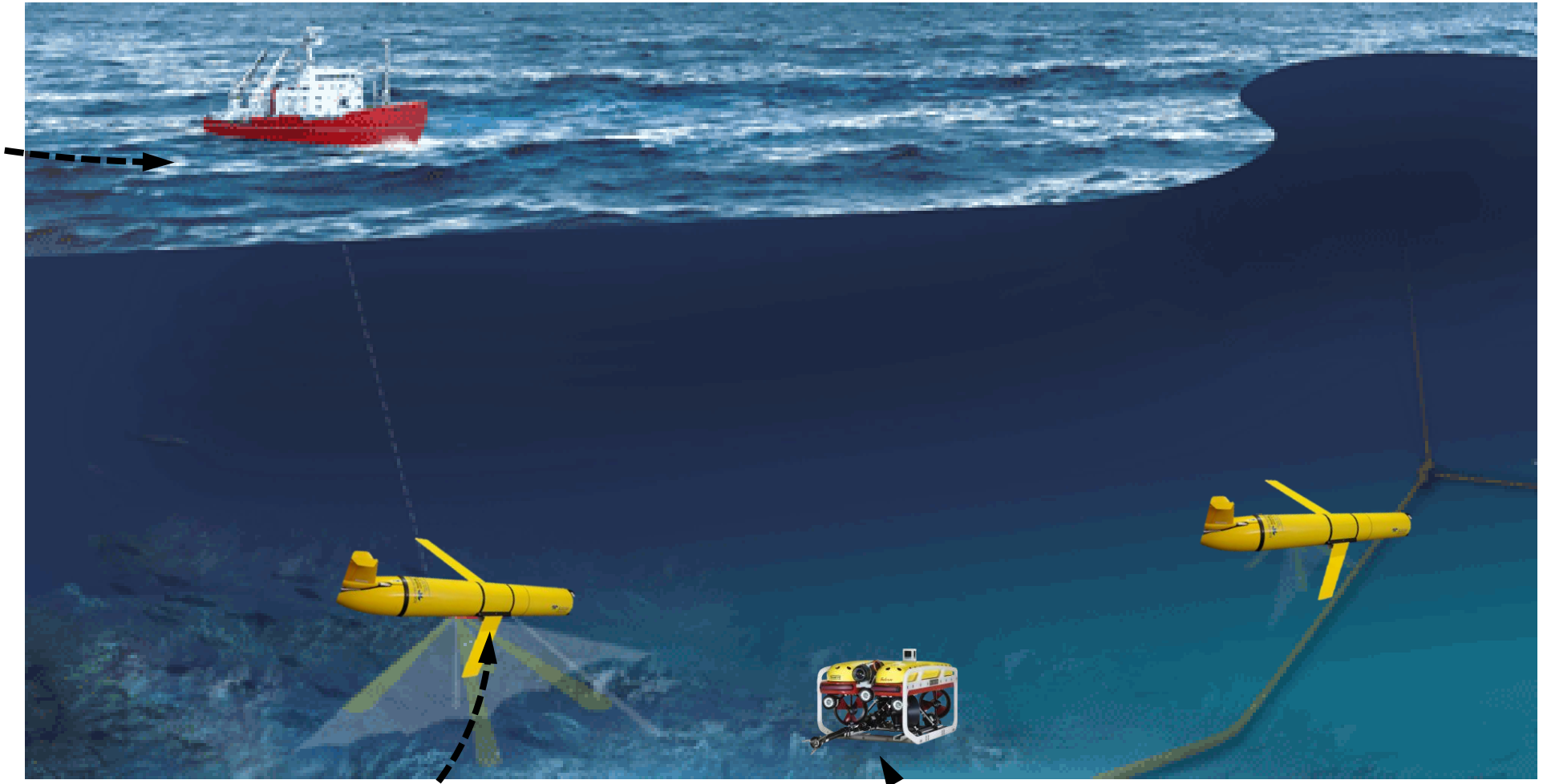
Can Pu, Filippos Sotiropoulos, Gabriel Margolis, Jacob Guggenheim, Zachary Duguid

Scenario


Surface Ship
Autonomous Underwater Vehicles
ROV exploration vehicle

Scenario

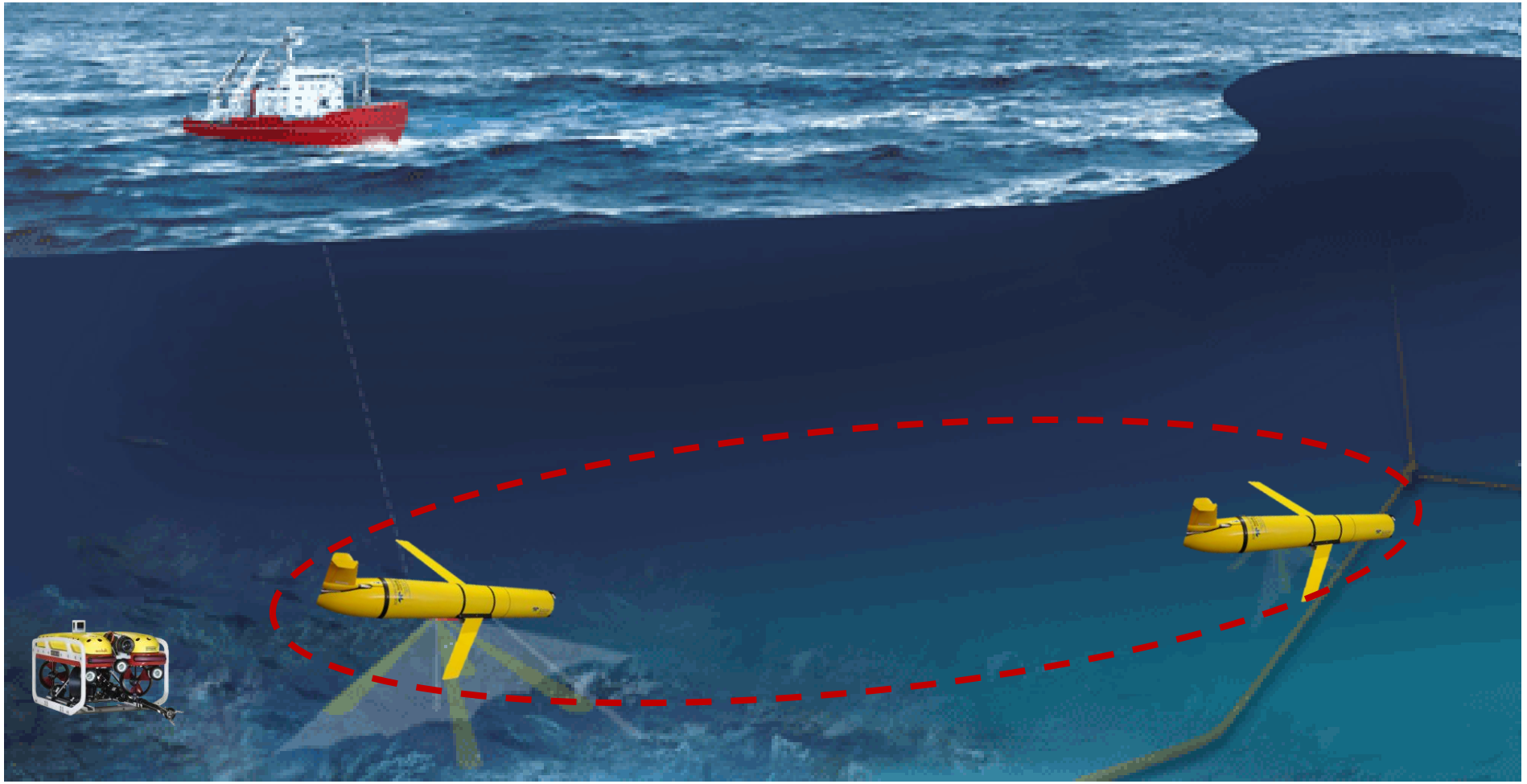
Rough Mapping



Explore areas of interest

Gather samples and other scientific tasks

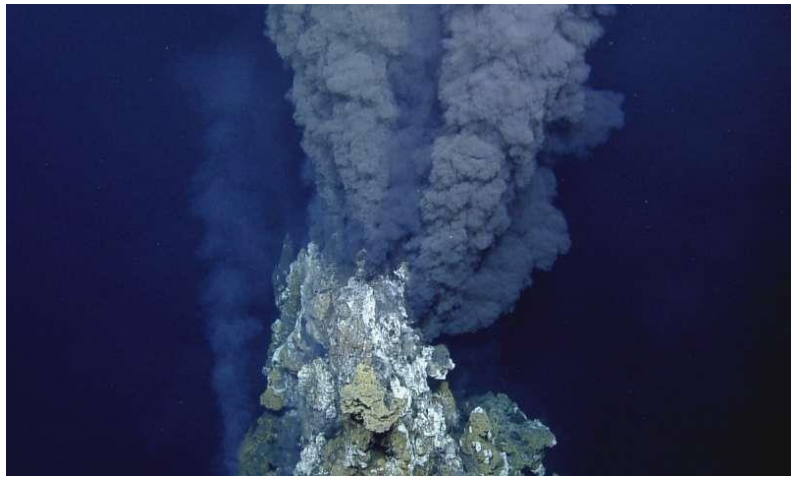
Scenario

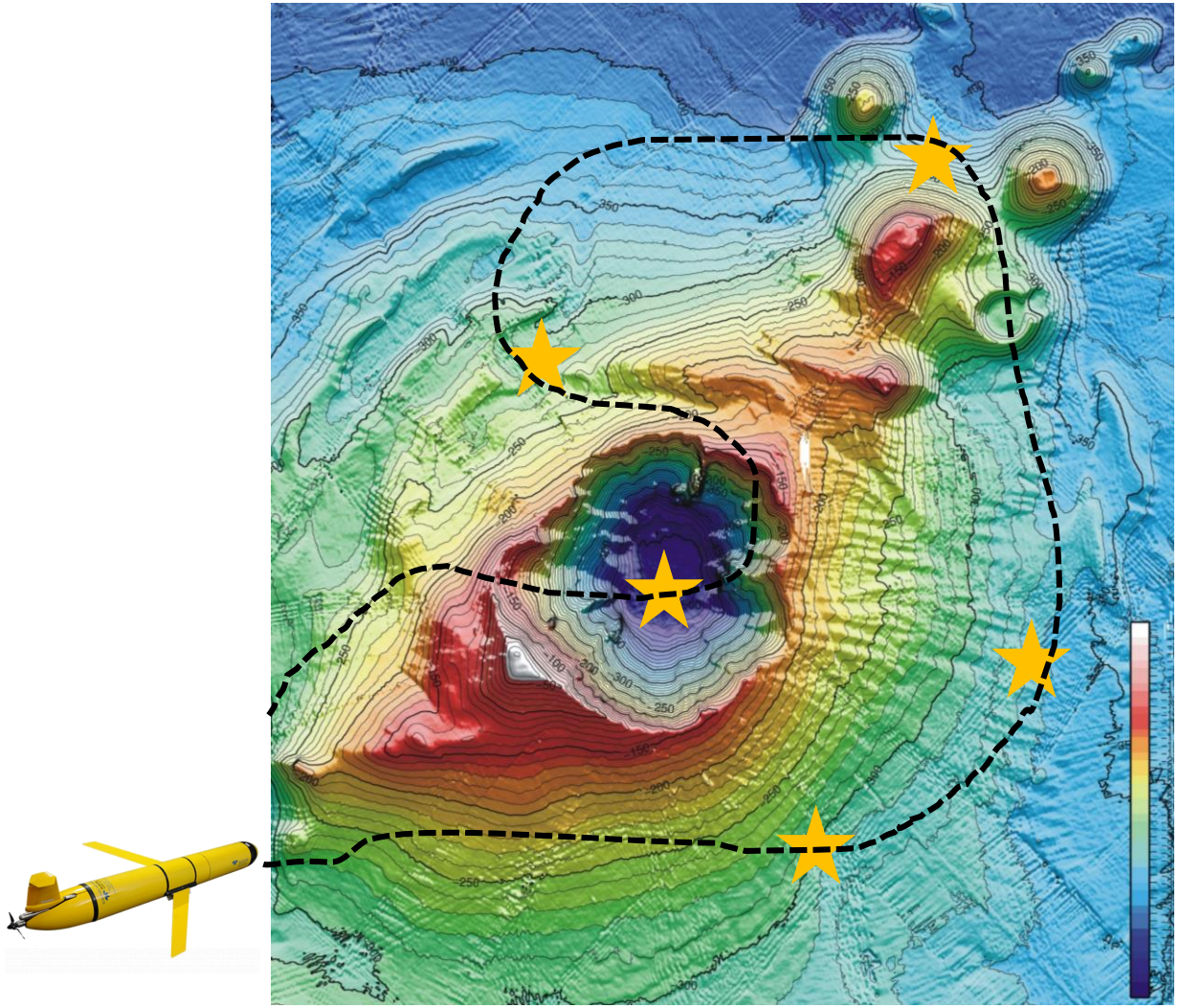


Focus on the AUV navigation

AUV Task

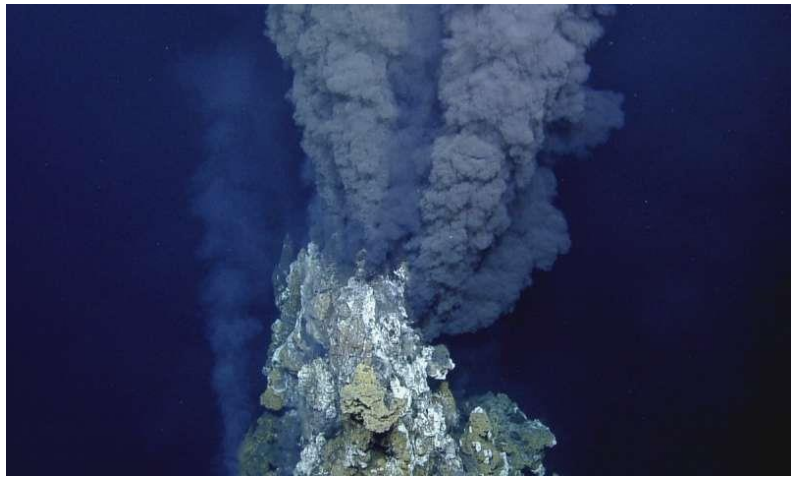
- AUV navigating through the Kolumbo volcanic crater.
- Trying to collect scientific information
- There may be many areas of value to visit

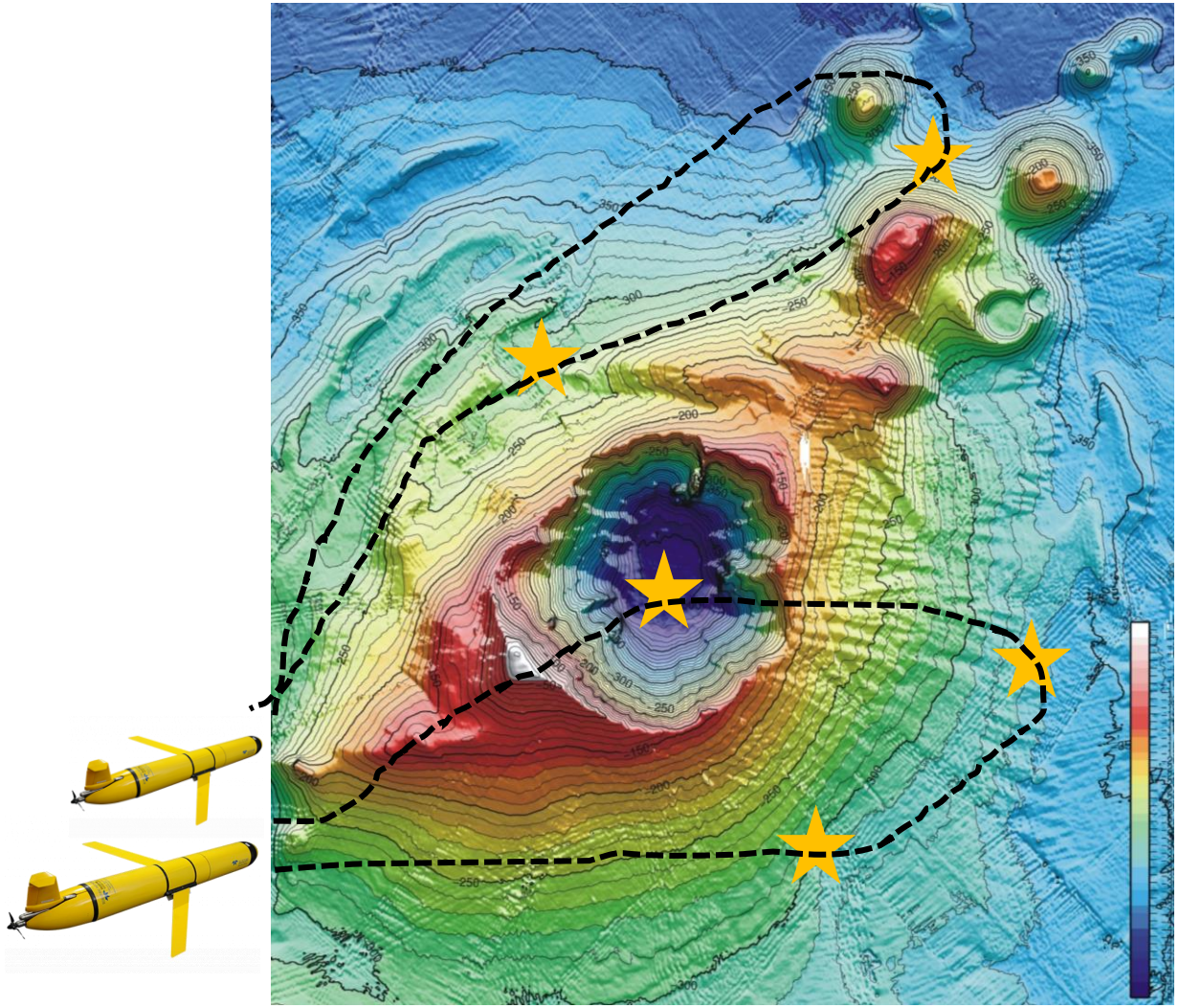




AUV Task

- AUV navigating through the Kolumbo volcanic crater.
- Trying to collect scientific information
- There may be many areas of value to visit
  - Using multiple AUV's can save time





Challenges

Key features of the problem:

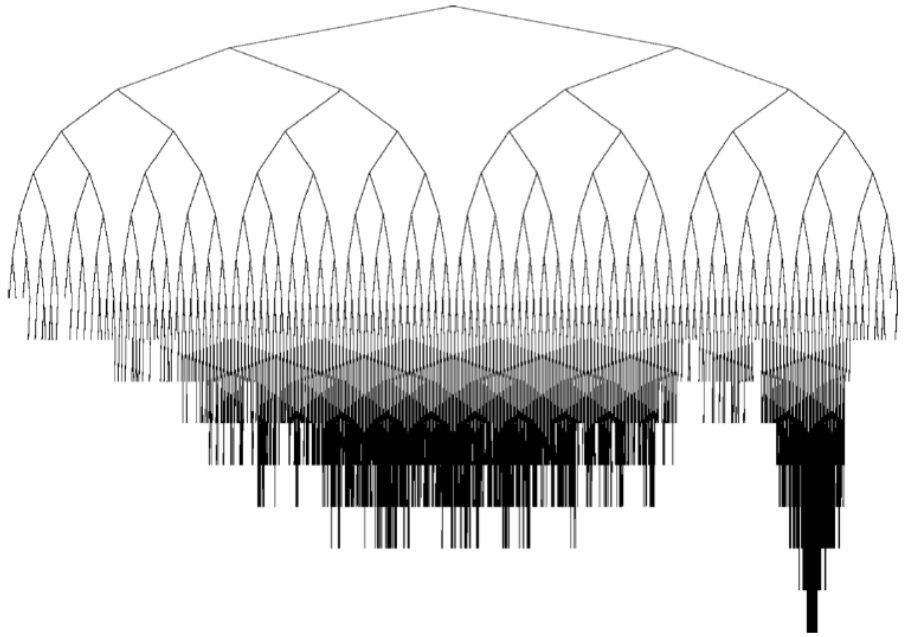
- Multiple agents coordinating their navigation to achieve a common objective
- Objective: explore areas with varying amounts of scientific interest.
  - Can be interpreted as a **reward**

This problem can naturally be described in the *Markov Decision Process* framework but:

- State space to plan in very large because:
  - Many moves per agent (b) *and* many agents (n)
  - Distributed reward requires substantial planning depth (d)

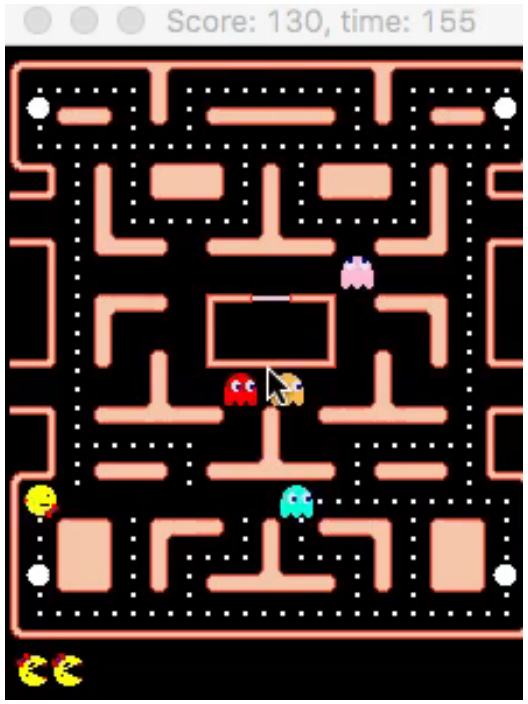
$$\mathrm{O}(b^{n\times d})$$

We will illustrate an algorithm, **Monte Carlo Tree Search**, (MCTS) which can help tackle these large dimensional problems.



Some examples of MCTS

Mario



Ms. Pacman

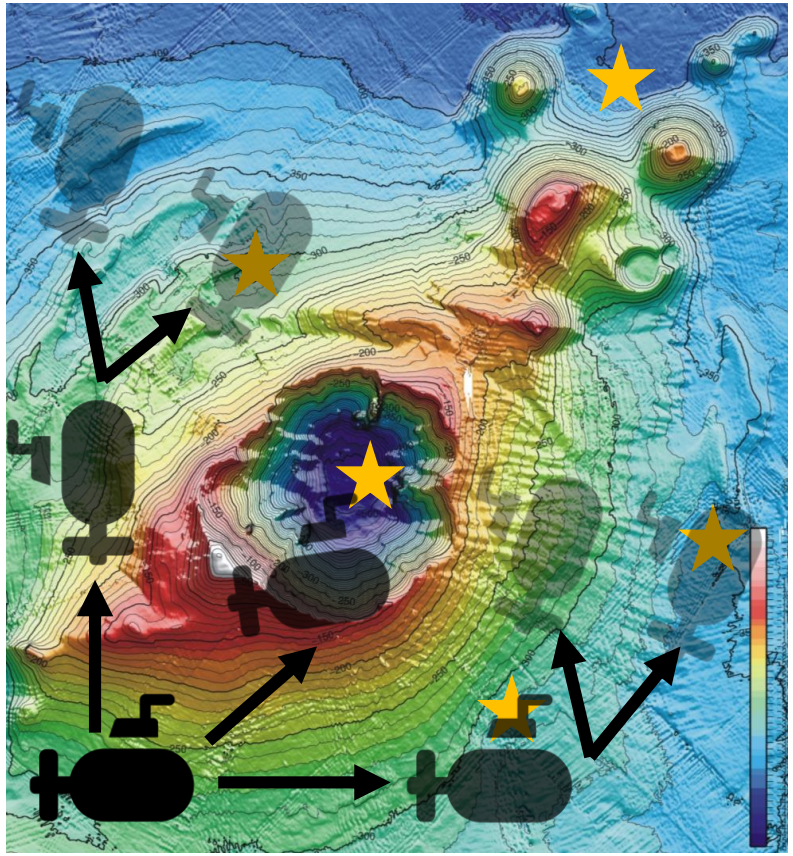
Rome: Total War

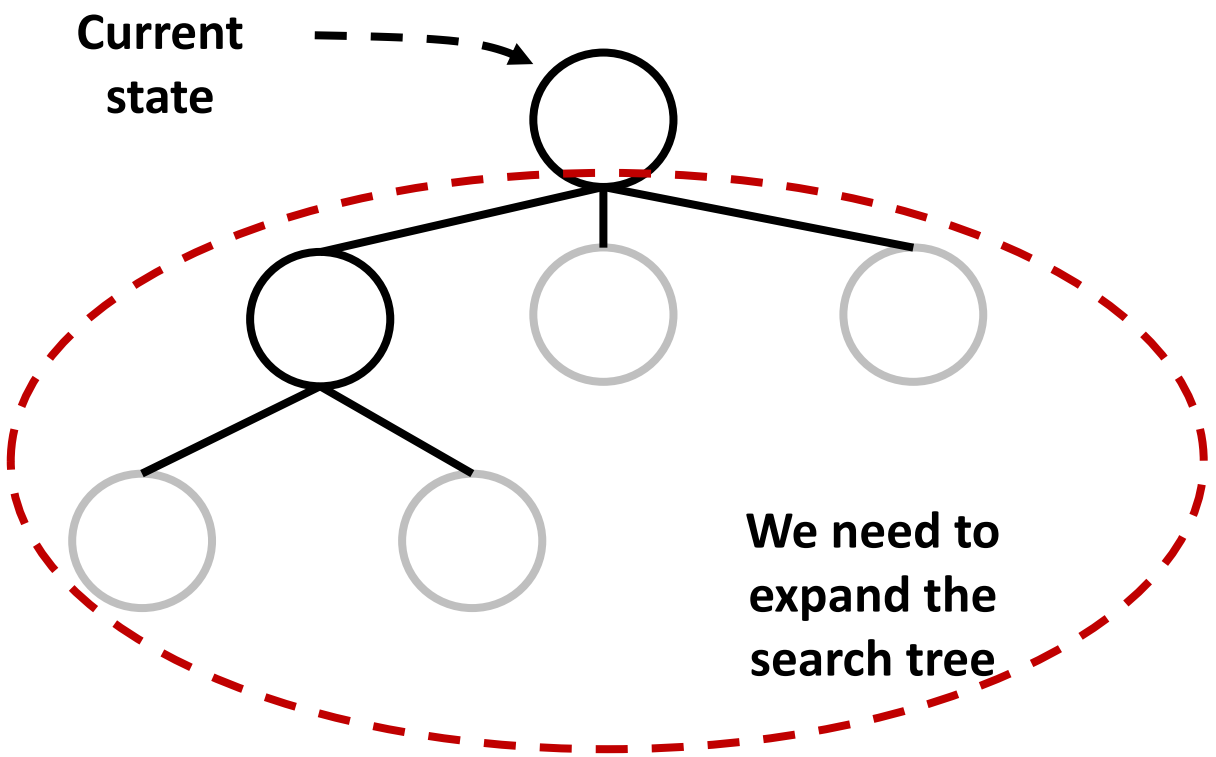
Outline

- Background
  - Monte Carlo Simulation
  - Upper Confidence Bounds
- Implementing MCTS with Upper Confidence Bounds for Trees
- Collaborative MCTS
  - Discuss Max-N, a variant of MCTS for multiple agents
  - Discuss how to decentralize MCTS
  - Discuss how to improve the default policy in decentralized MCTS

Path Planning Problem

- The problem of planning can be described in terms of a decision tree





Heuristic Search: A*

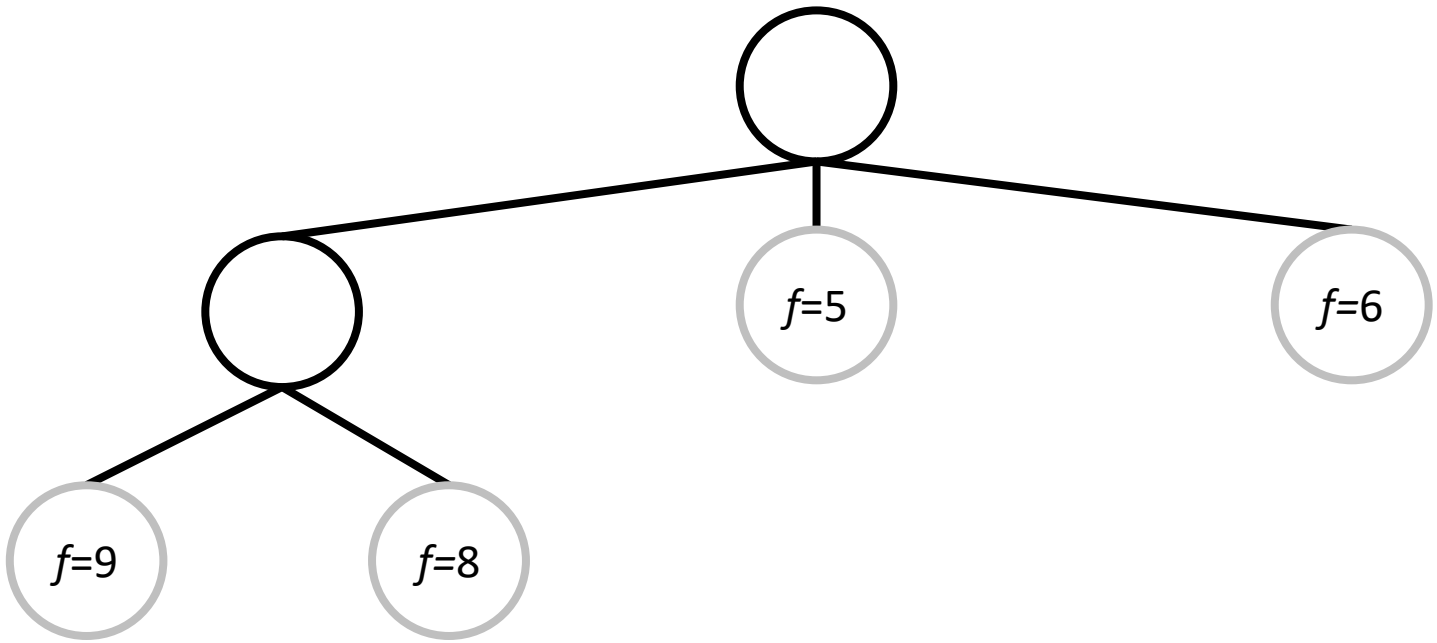
What happens if we try to solve a problem like this with a conventional heuristic search method: A*

Reminder: A* chooses a path to minimize the evaluation function:

$$f(x) = g(x) + h(x)$$

Cost so far — $g(x)$

Admissible heuristic — $h(x)$



Simply expand search tree for whatever action minimizes $f(x)$

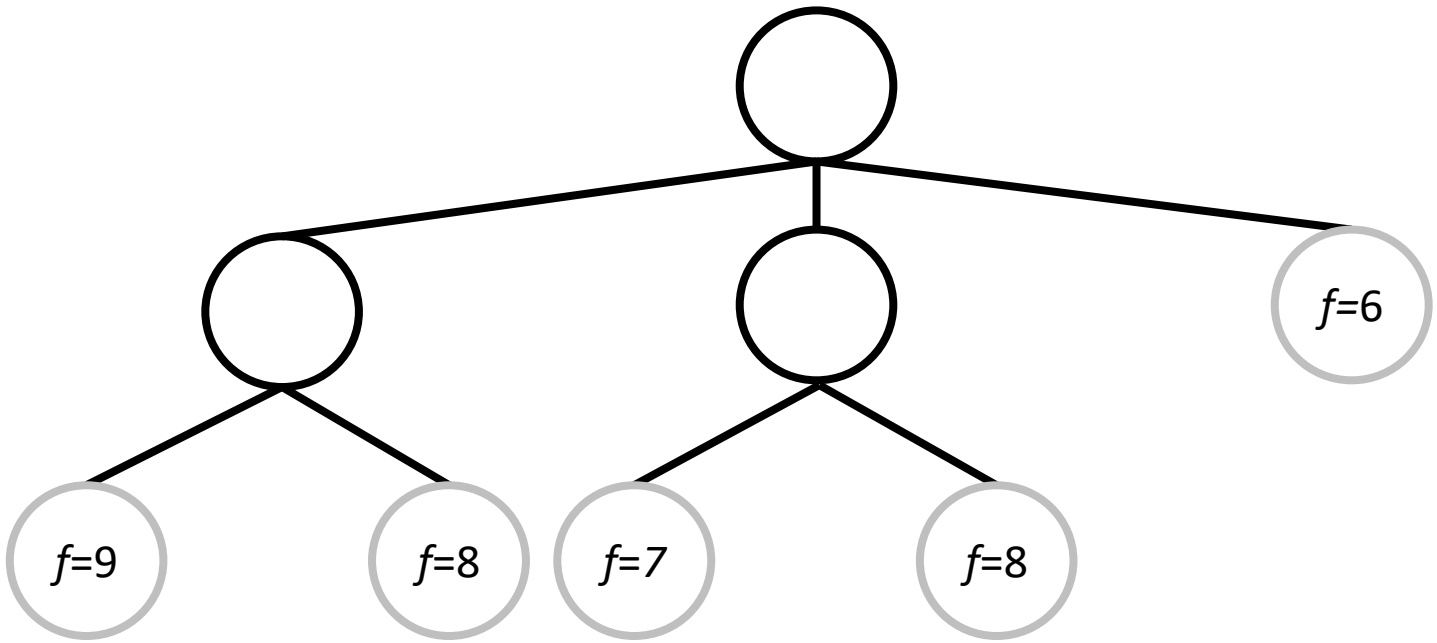
Heuristic Search: A*

What happens if we try to solve a problem like this with a conventional heuristic search method: A*

Reminder: A* chooses a path to minimize the evaluation function:

$$f(x) = g(x) + h(x)$$

Cost so far — **Admissible heuristic**



Simply expand search tree for whatever action minimizes $f(x)$

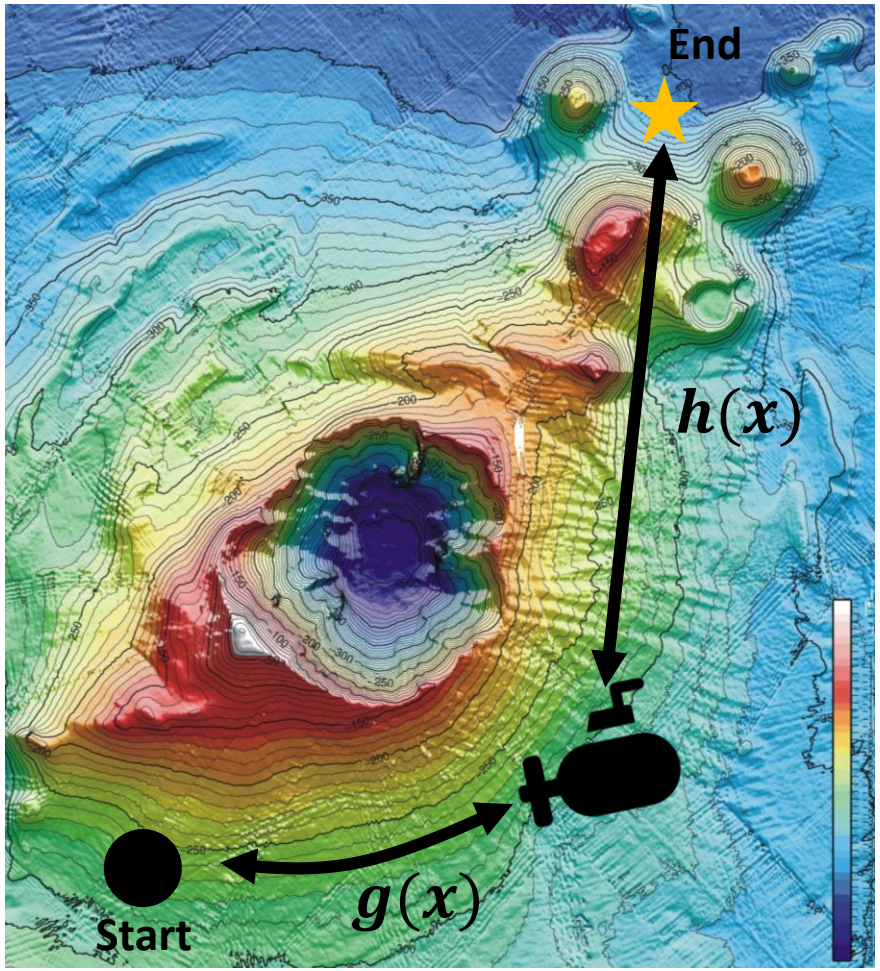
Heuristic Search: A*

$$f(x) = g(x) + h(x)$$

Cost so far — **Admissible heuristic**

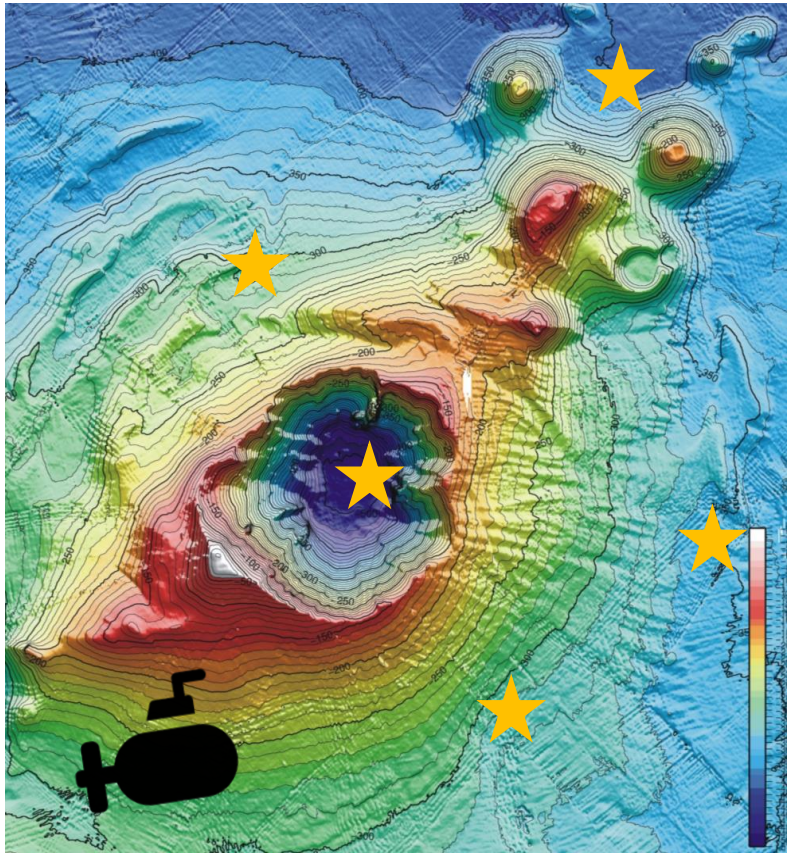
What heuristic ?

For a simple path planning problem one could use:
Euclidean distance to end goal



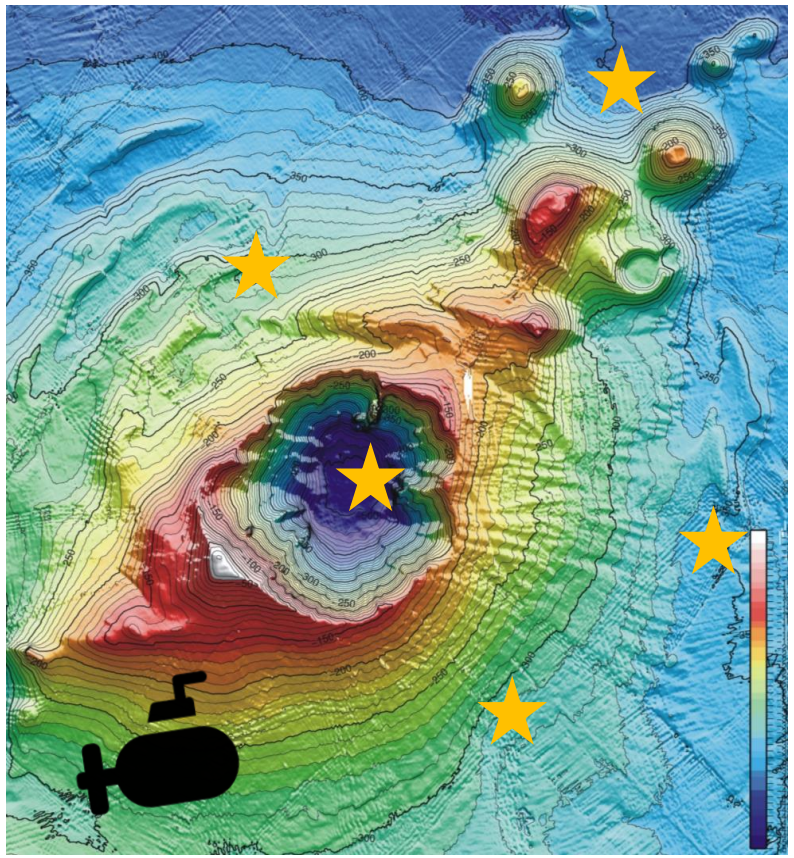
Heuristics

What heuristic should we use?



Heuristics

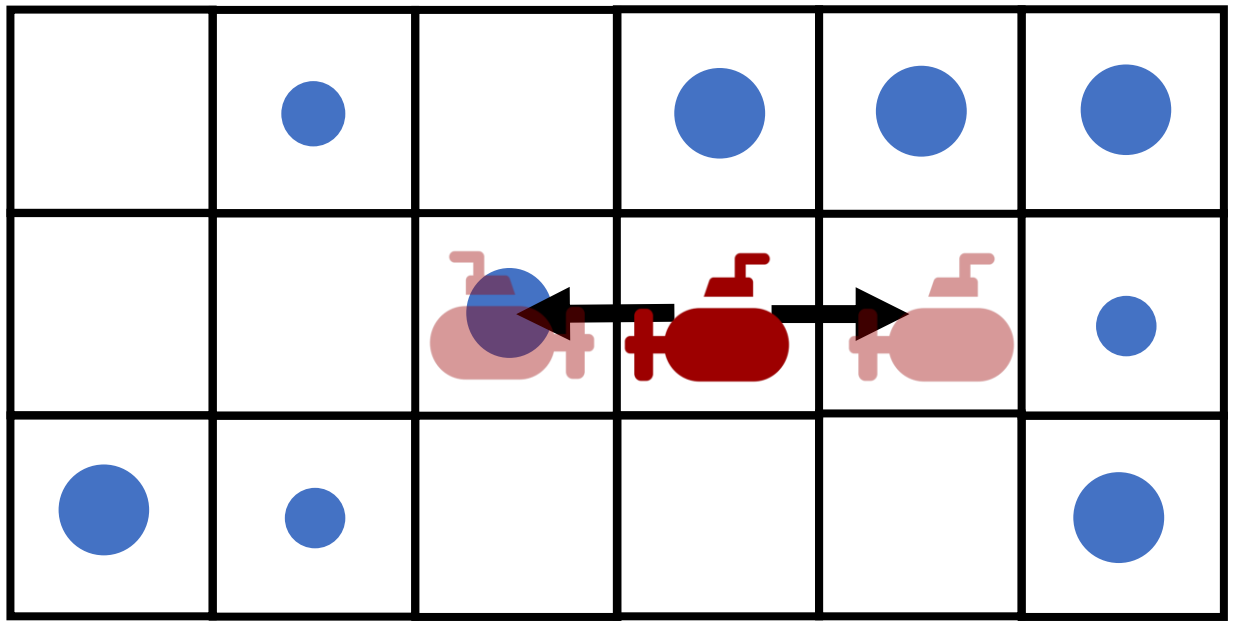
What heuristic should we use?



Finding a heuristic can be problematic!

Monte Carlo Simulation

Monte Carlo Methods: One can estimate the expected value of potential future reward from a state via random sampling of future moves.

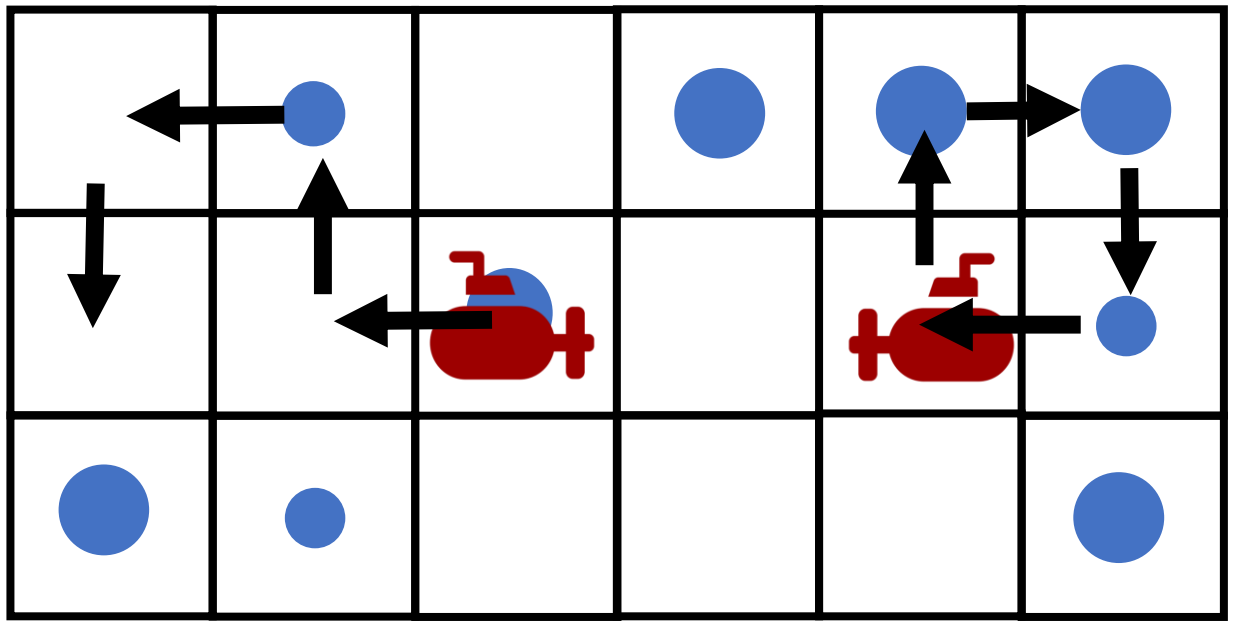


What move is a better strategy?

Monte Carlo Simulation

Monte Carlo Methods: One can estimate the expected value of potential future reward from a state via random sampling of future moves.

Random simulation can be used to ascertain the quality of a state: a heuristic!

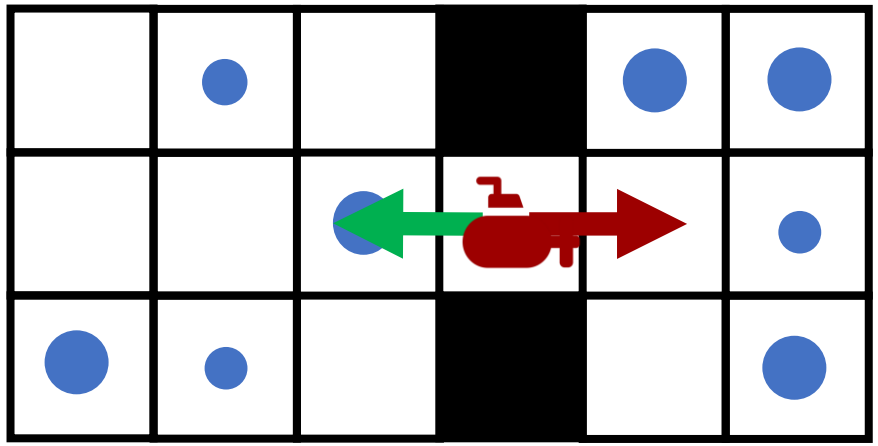


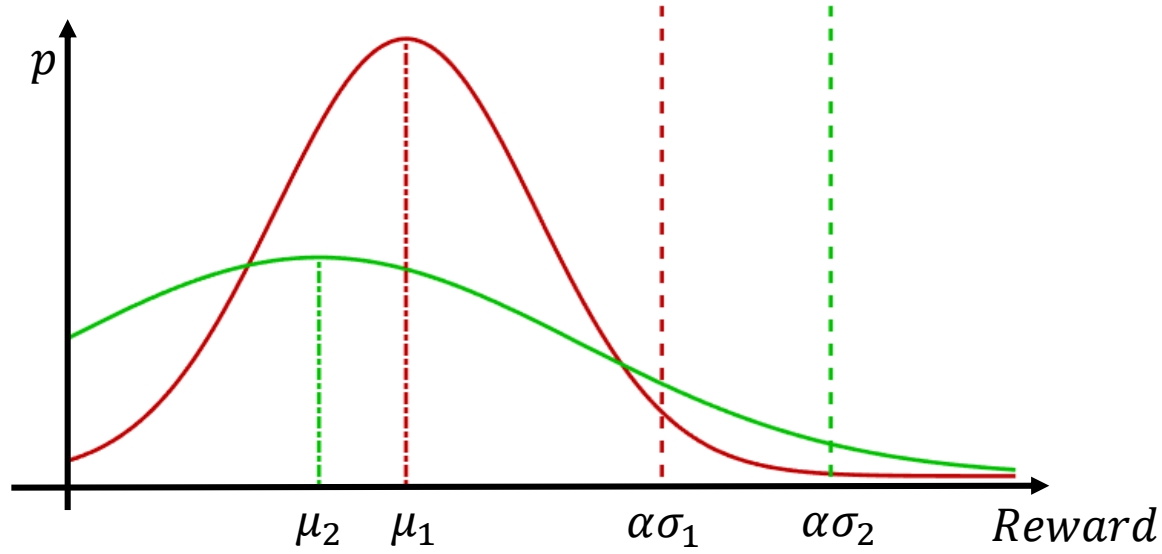
What move is a better strategy?

Uncertainty in Heuristics

But now random sampling introduces a new problem: Our sampled reward has uncertainty.
How do we pick the best way to expand the search tree?

Lets take the simplest case: two possible actions sampled with assumed Gaussian distribution.



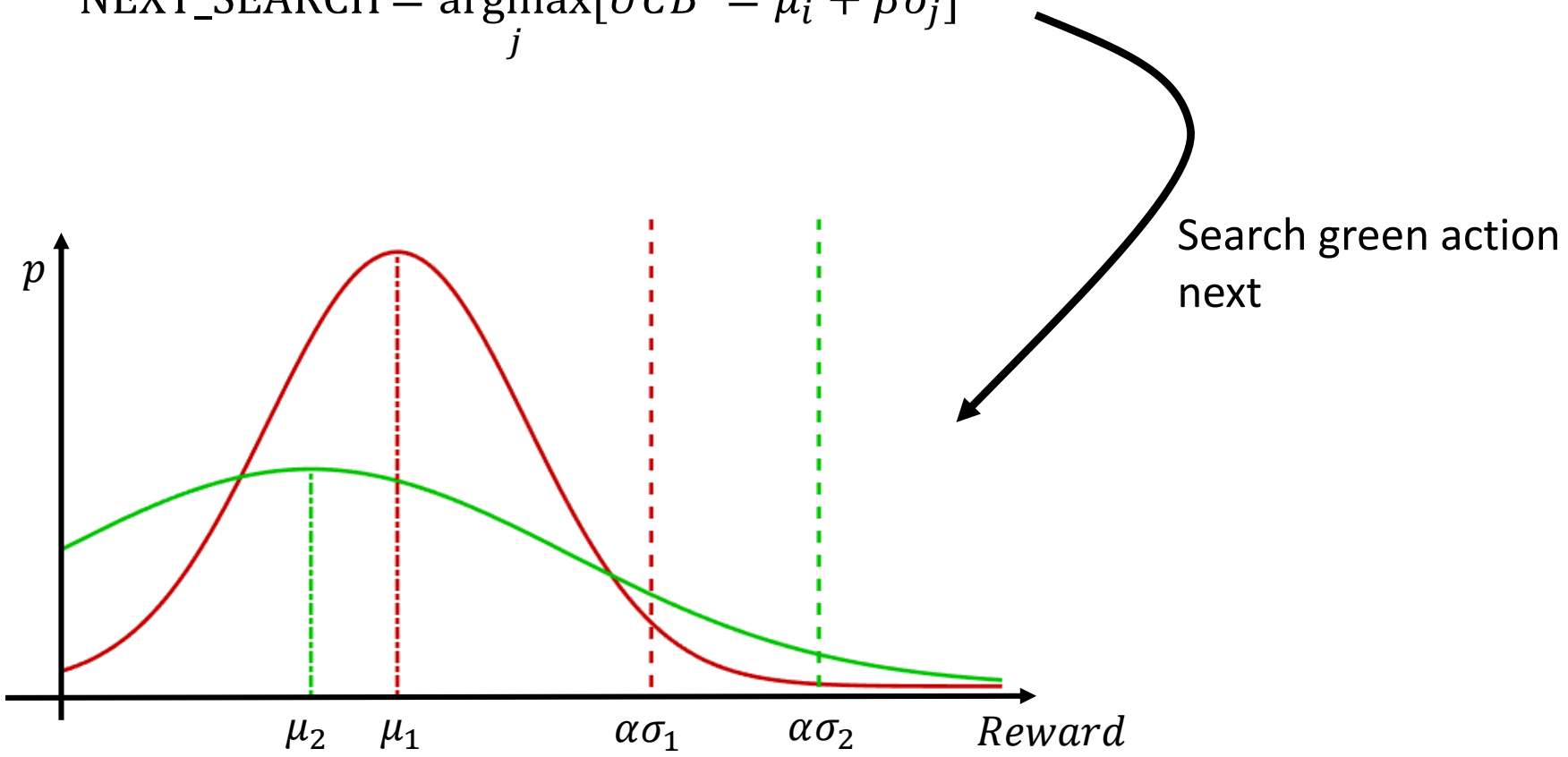


What should we sample next?
Balance searching in most promising direction (**Exploitation**) but also search where we are unsure (**Exploration**).

Upper Confidence Bounds

Idea: lets continue searching the action with the highest upper confidence bound. For gaussian:

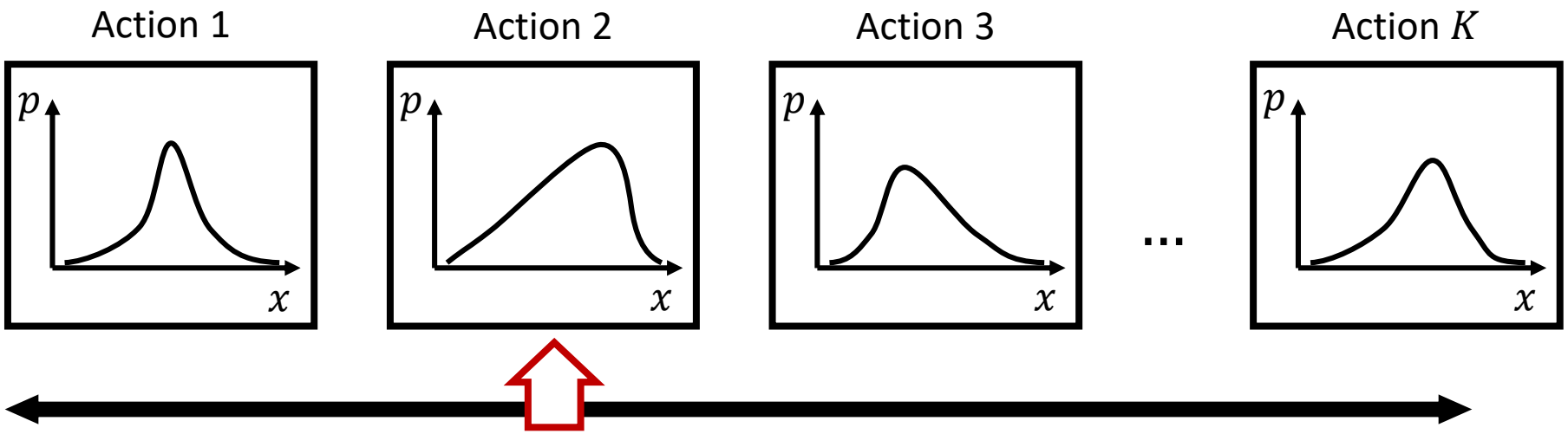
$$\text{NEXT\_SEARCH} = \underset{j}{\text{argmax}}[UCB = \mu_i + \beta\sigma_j]$$



Upper Confidence Bounds

- One can sequentially sample one of K actions which yield a reward

Underlying reward distributions unknown



- The policy should aim to minimize the *regret* after n searches:

$$R_N = \mu^\star n - \mu_j \sum_{j=1}^{K} \mathbb{E}[T_j(n)]$$

- It represents the loss from not choosing the best action.

Upper Confidence Bounds

Since the distributions aren't necessarily Gaussian we can use the Chernoff-Hoeffding inequality:

$$P\left[\mathbb{E}\left[X_j\right] > \bar{X}_{j,t} + u_j\right] \leq e^{-2n_j u_j^2}$$

Set to a small threshold which we decrease with each sample

subscript j : each possible action
X_j : Random variable for reward
u_j : Upper confidence bound
n_j : number of samples

$$u_j = \sqrt{\frac{2\ln n}{n_j}}$$

$$\text{UCB1} = \bar{X}_j + \sqrt{\frac{2\ln n}{n_j}}$$

P. Auer, N.Cesa-Bianchi, and P. Fischer. (2002). "Finite-time Analysis of the Multiarmed Bandit Problem". *Mach. Learn*

UCB1

Sample action which maximizes:

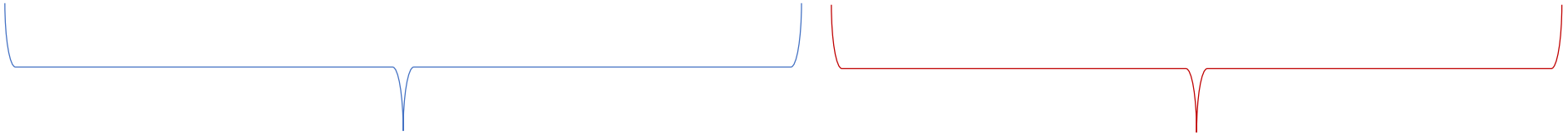
$$\text{UCB1} = \underbrace{\bar{X}_j}_{\text{Exploitation}} + \underbrace{\sqrt{\frac{2 \ln n}{n_j}}}_{\text{Exploration}}$$

Exploitation
Mean reward from that action

Exploration
Higher for poorly sampled actions

P. Auer, N.Cesa-Bianchi, and P. Fischer. (2002). "Finite-time Analysis of the Multiarmed Bandit Problem". *Mach. Learn*

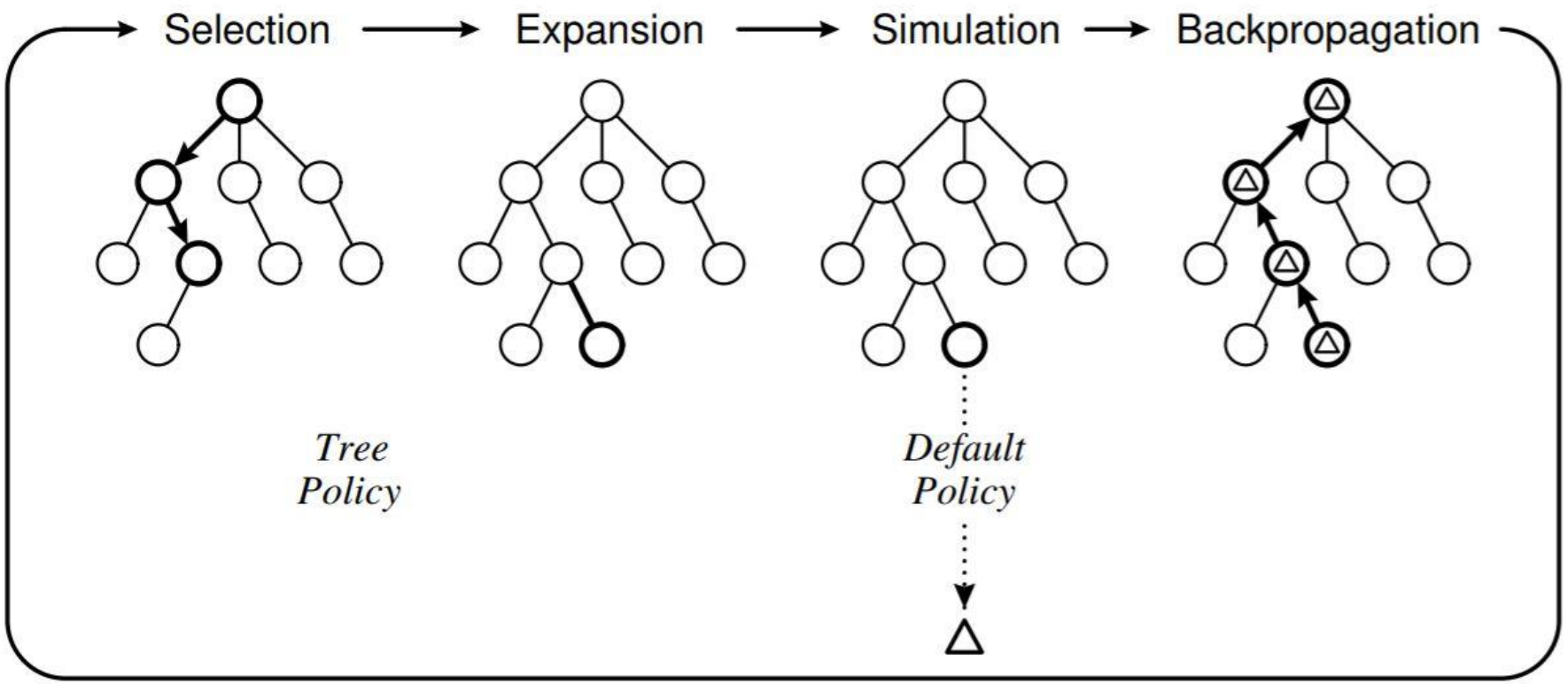
Monte Carlo Tree Search



Random Sampling

Decision Trees

Monte Carlo Tree Search



Tree Policy

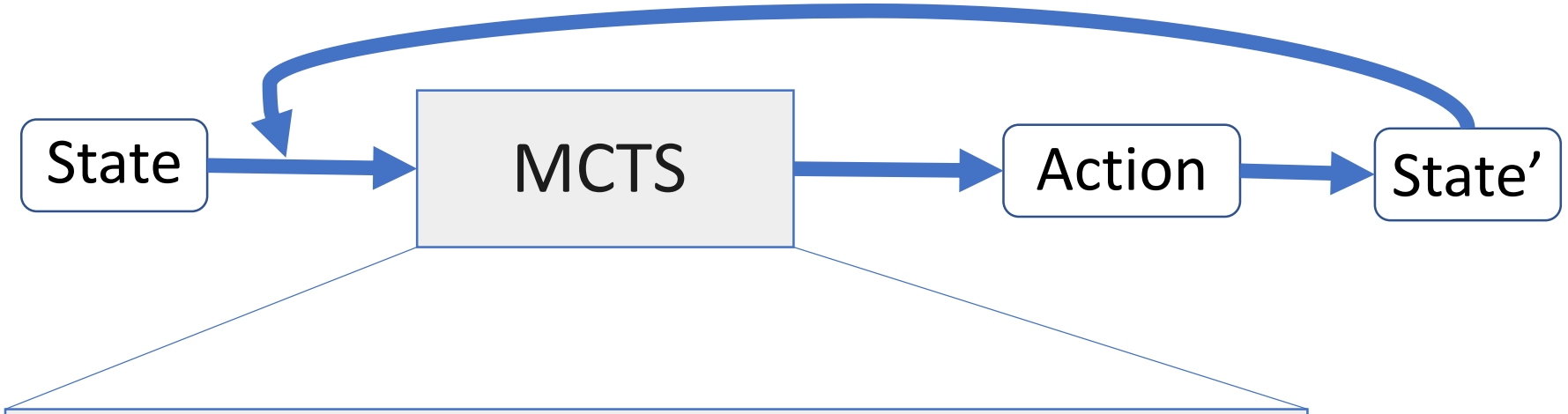
Selection Phase
Example: UCB

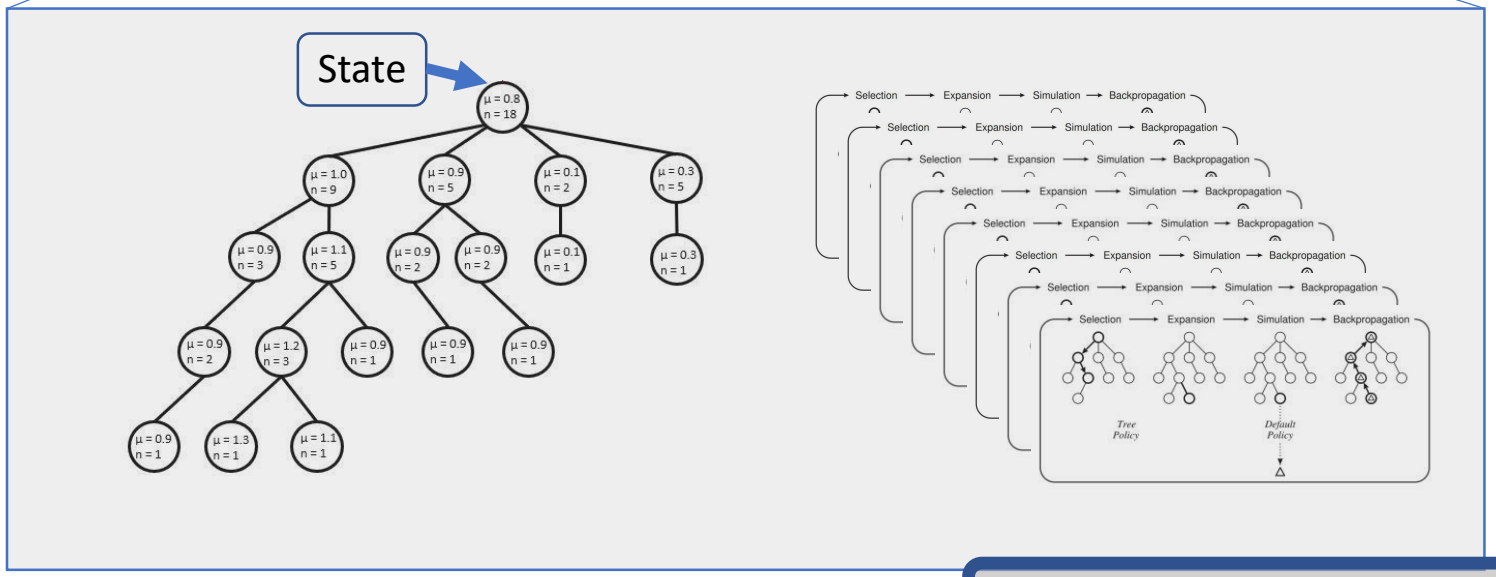
Default Policy

Simulation Phase
Example: Random Rollout

From *A Survey of Monte Carlo Tree Search Methods*

Planning and Acting with MCTS

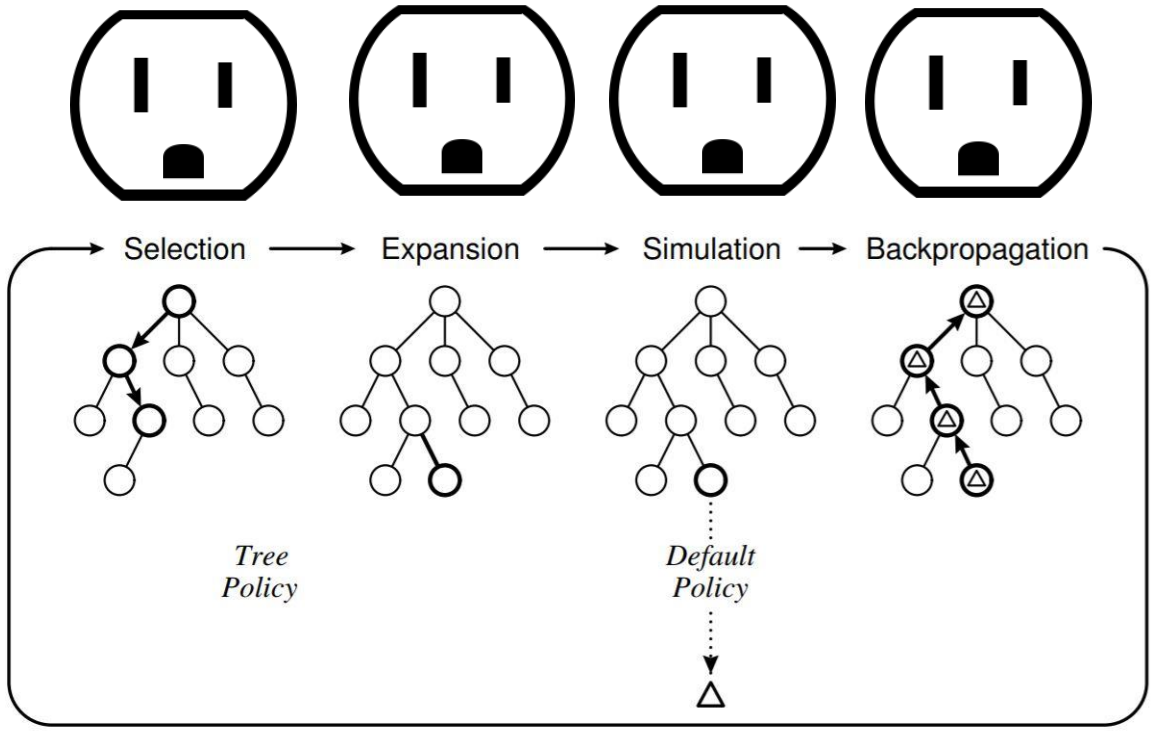


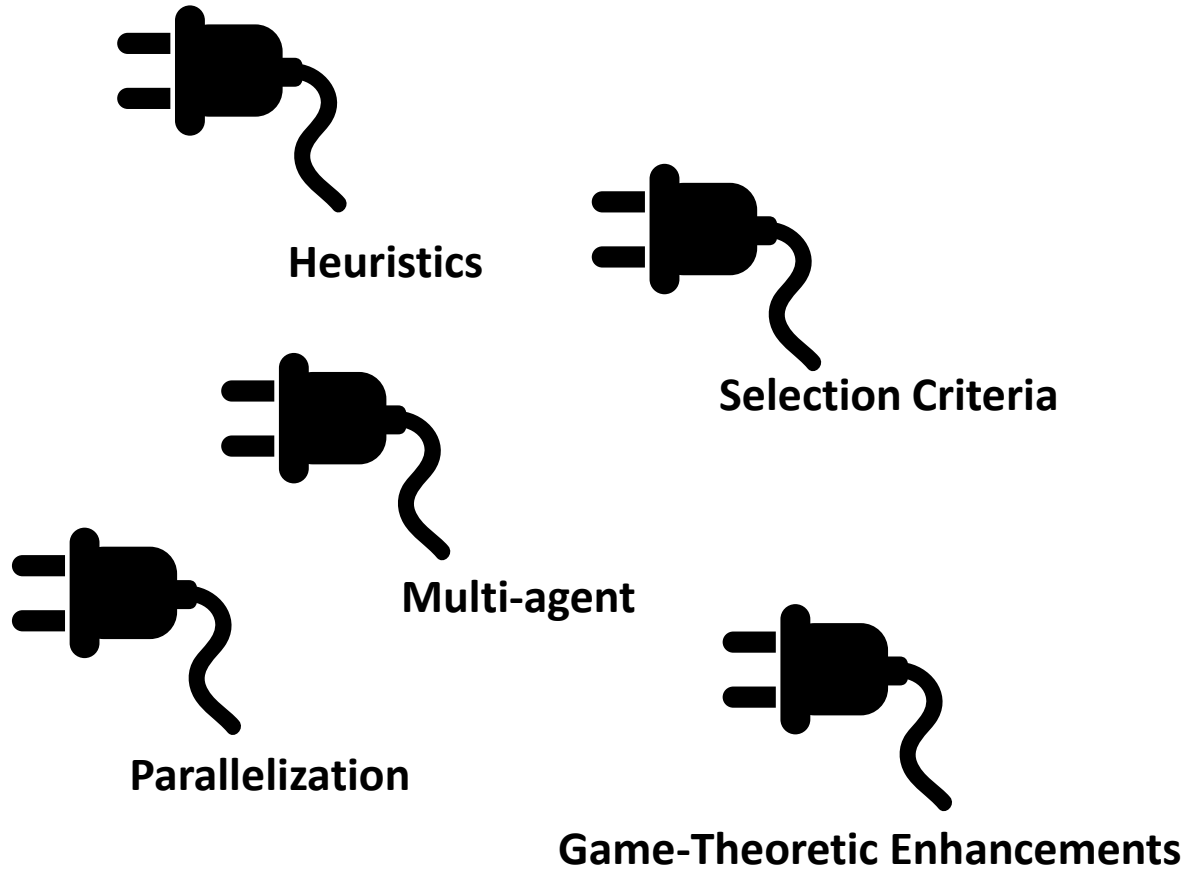


Stopping Criterion: Time Limit

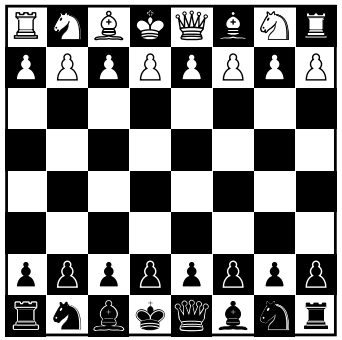
MCTS is Modular

MCTS is **Modular**





Combinatorial Games



Perfect Information ⟷ Partial Information

Deterministic ⟷ Non-deterministic

Sequential ⟷ Simultaneous

Discrete-Time ⟷ Continuous-Time

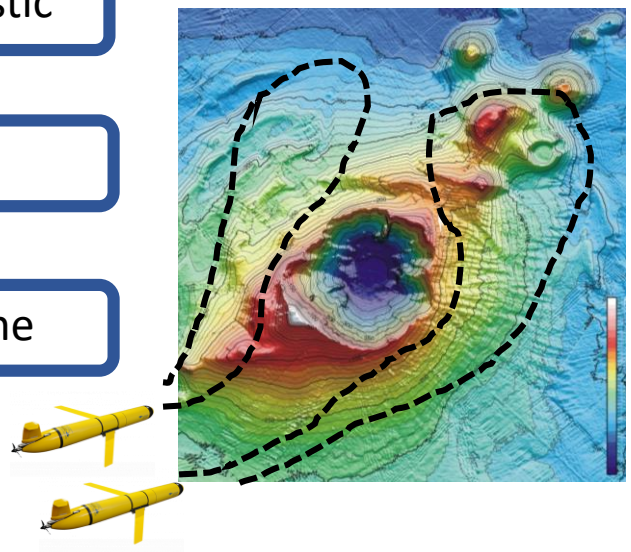
Our (Multi-Agent) Problem

Perfect Information ⟷ Partial Information

Deterministic ⟷ Non-deterministic

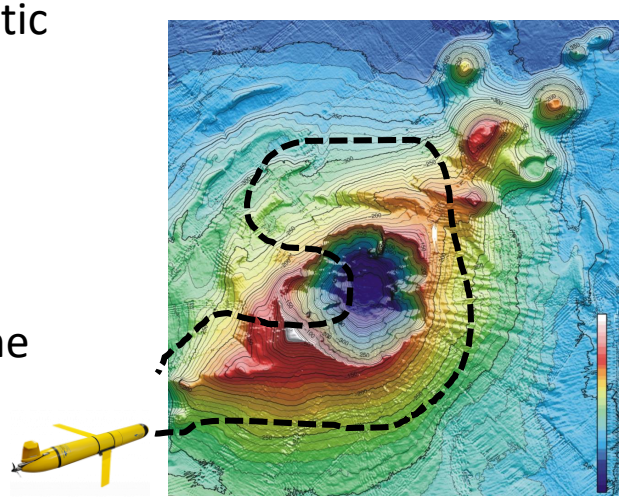
Sequential ⟷ Simultaneous

Discrete-Time ⟷ Continuous-Time



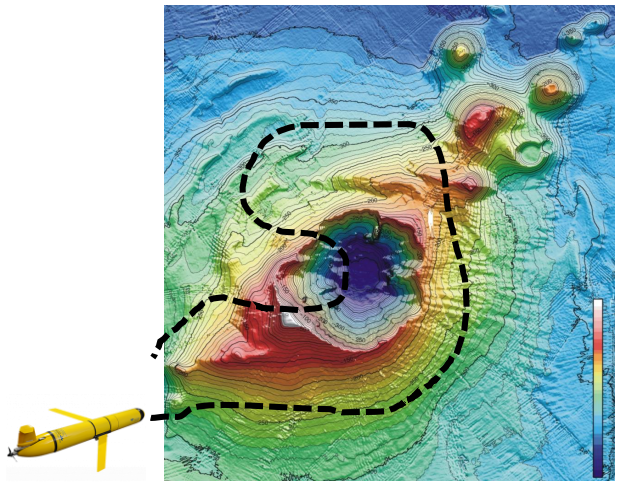
Our (Single-Agent) Problem

Perfect Information	Partial Information
Deterministic	Non-deterministic
Sequential	Simultaneous
Discrete-Time	Continuous-Time



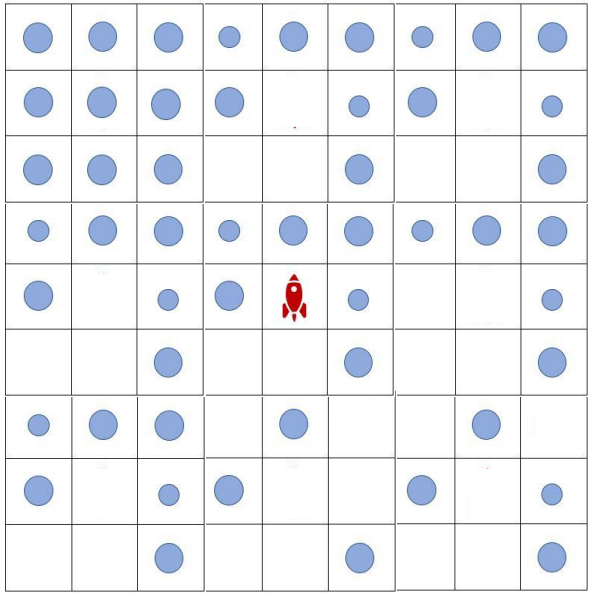
Our (Single-Agent) Problem

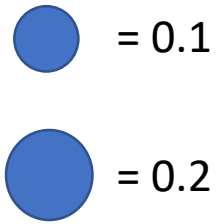
Investigating ~~collaborative~~ planning for AUVs in the Kolumbo crater.



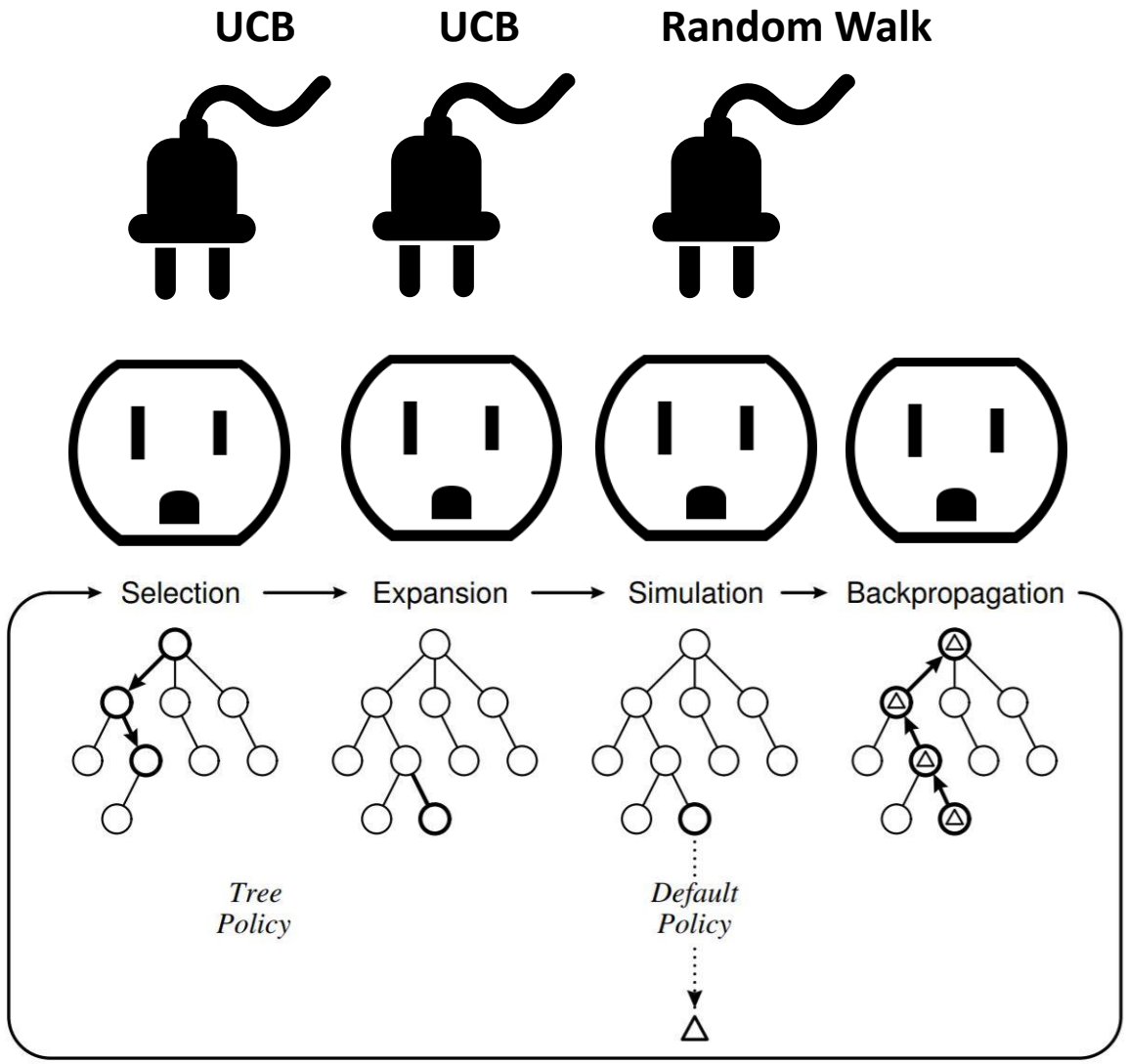
Simplified representation:

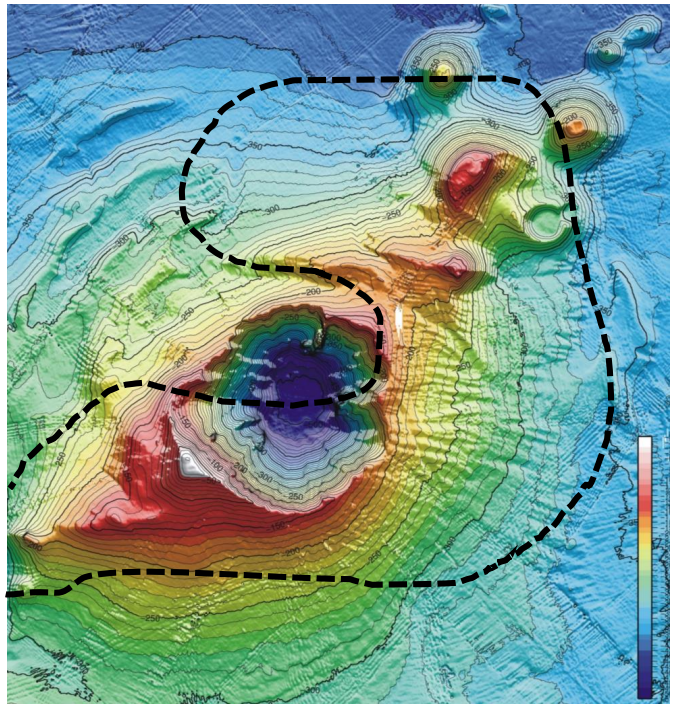
- Discretised map
- AUV can move (1 quadrant per time step)
- No dynamics
- Fixed mission duration
- **Single-Agent first: No collaboration**



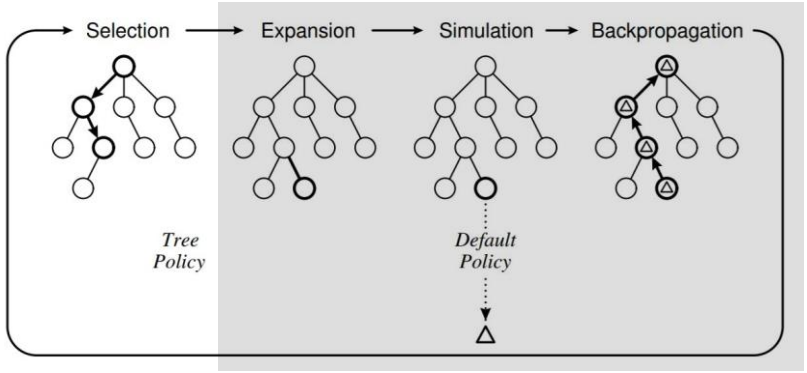


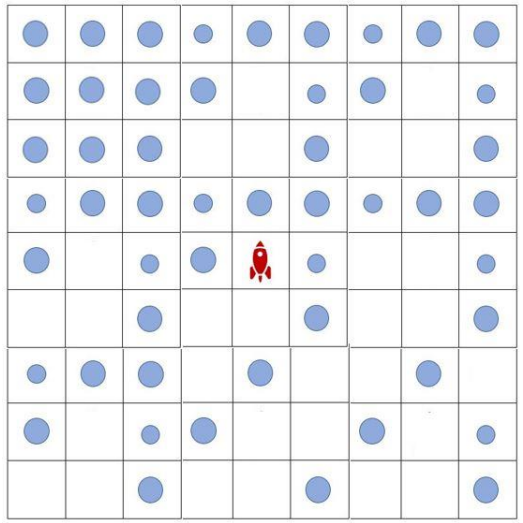
MCTS for Our Problem



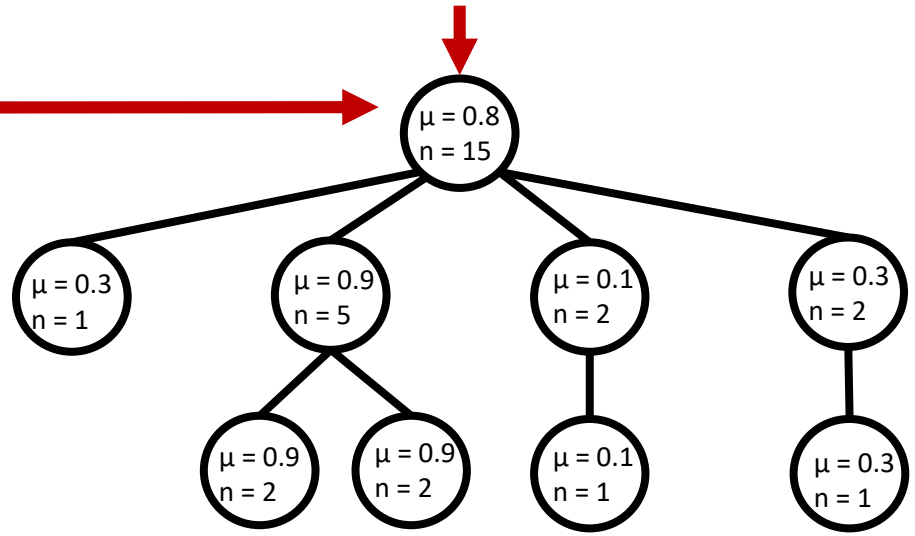


Selection

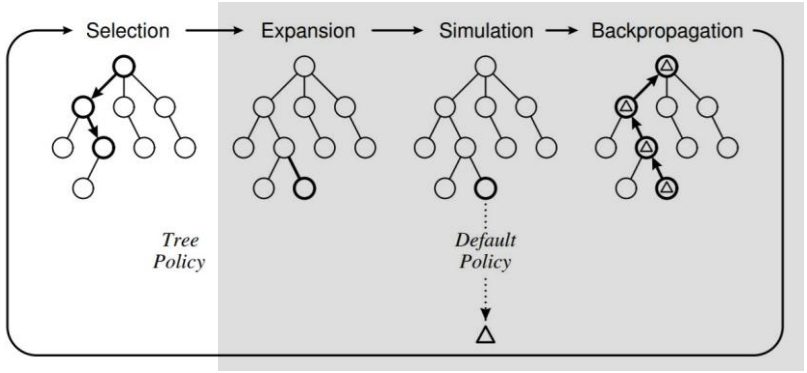




Search Tree

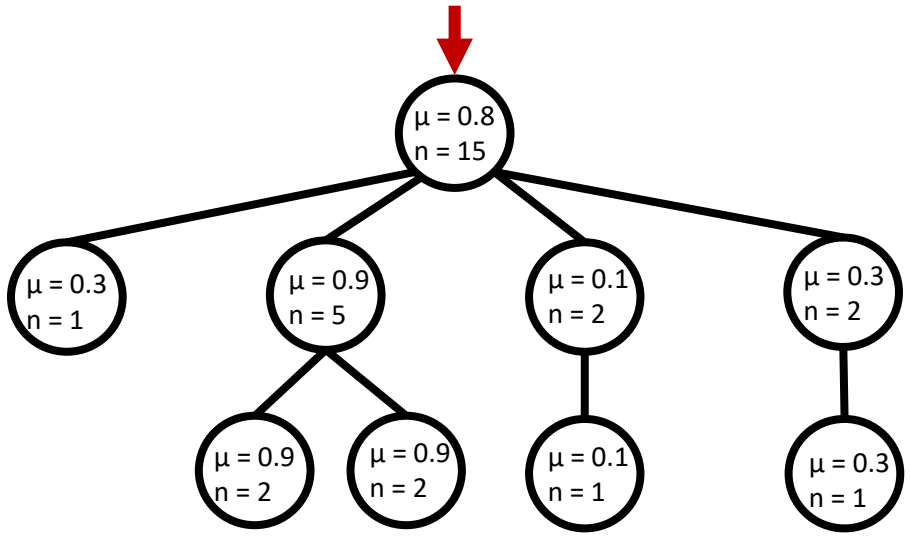


Selection

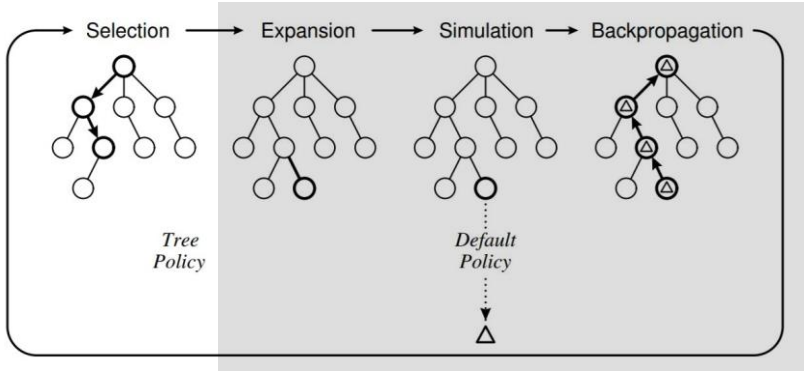


$$\text{UCB1} = \bar{X}_j + \sqrt{\frac{2 \ln n}{n_j}}$$

Search Tree

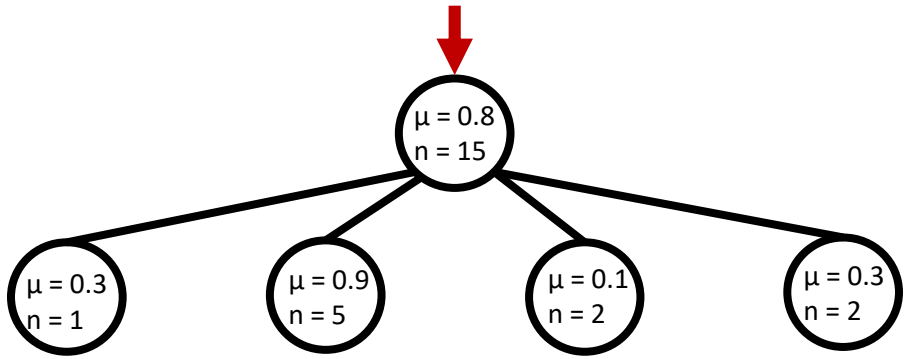


Selection



$$\text{UCB1} = \bar{X}_j + \sqrt{\frac{2 \ln n}{n_j}}$$

Search Tree

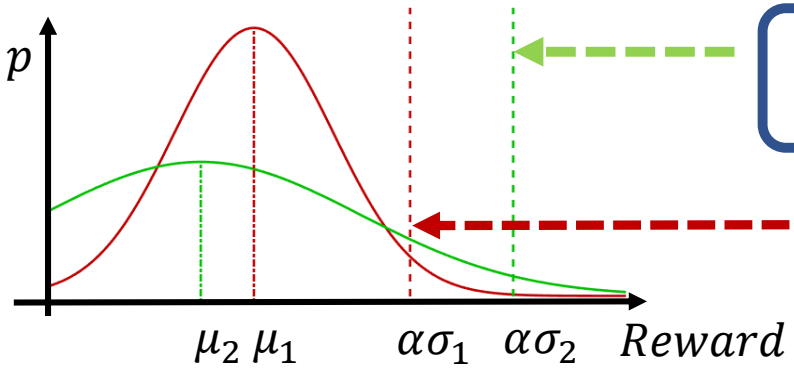


$$\text{UCB1} = 0.3 + \sqrt{\frac{2 \ln 15}{1}} = 2.63$$

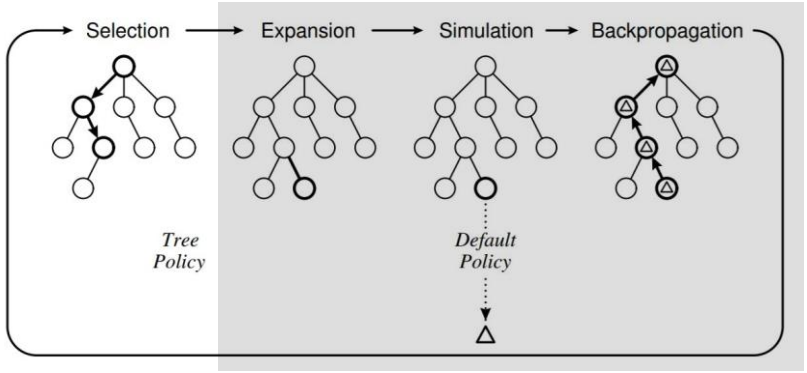
$$\text{UCB1} = 1.74$$

$$\text{UCB1} = 0.9 + \sqrt{\frac{2 \ln 15}{5}} = 1.94$$

$$\text{UCB1} = 1.95$$

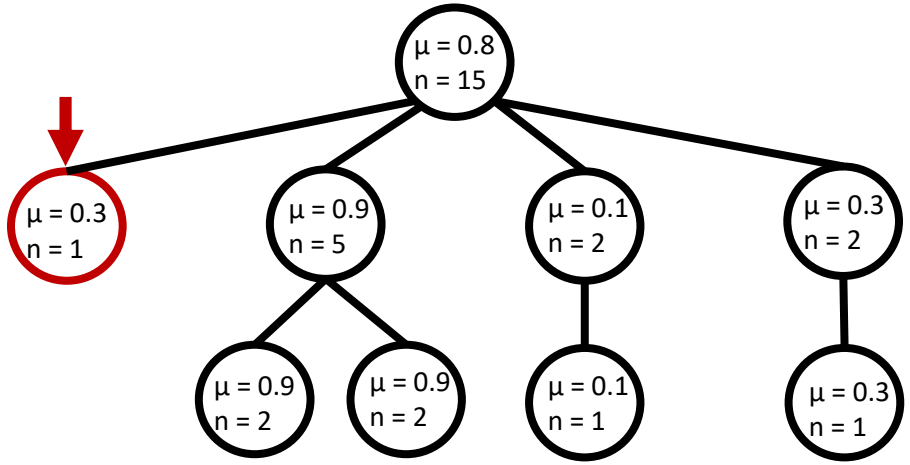


Selection

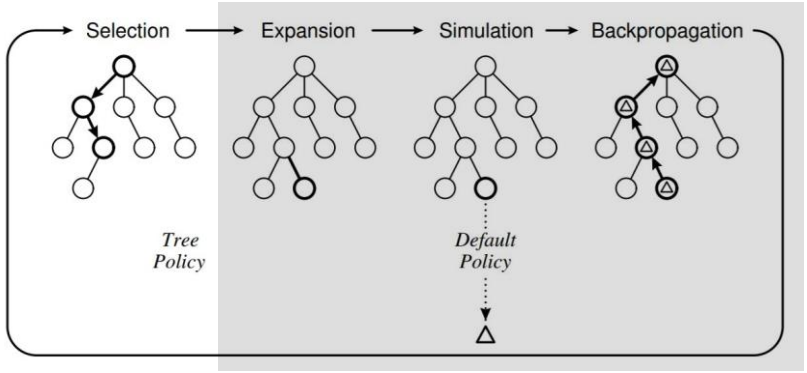


$$\text{UCB1} = \bar{X}_j + \sqrt{\frac{2 \ln n}{n_j}}$$

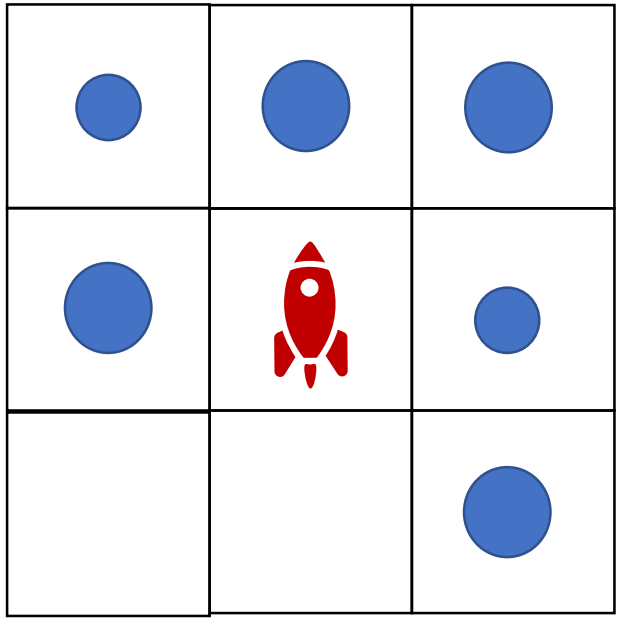
Search Tree



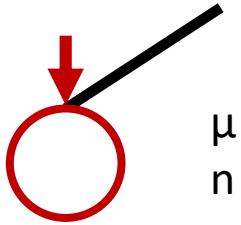
Selection



State

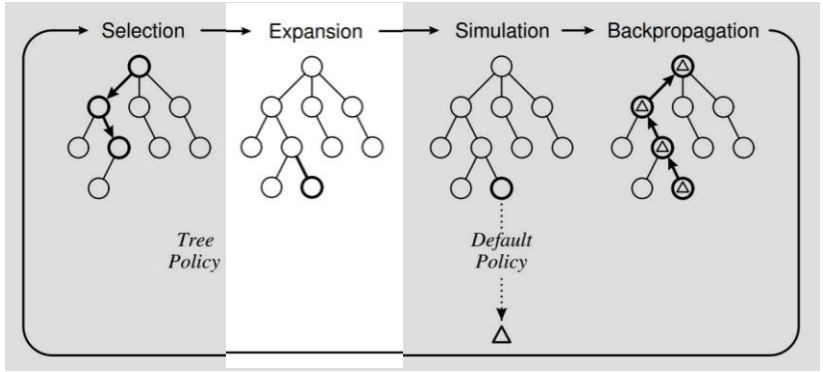


Search Tree

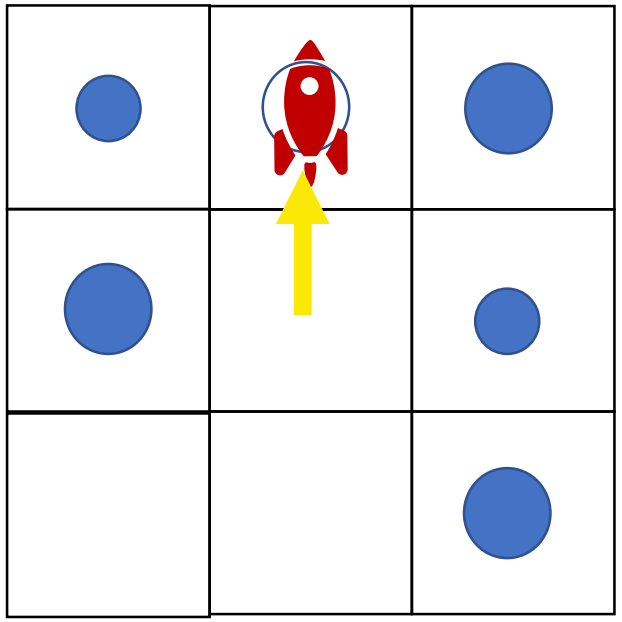


μ = 0.3
n = 1

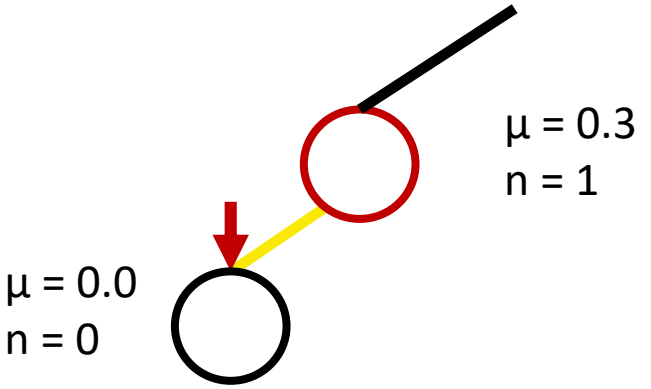
Expansion



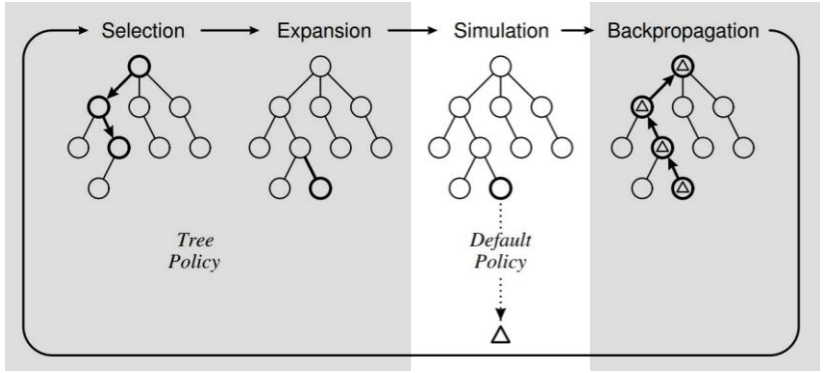
State



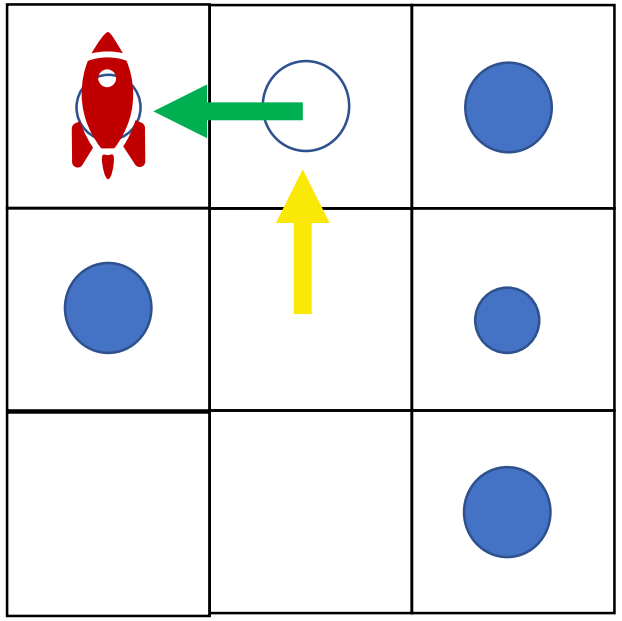
Search Tree



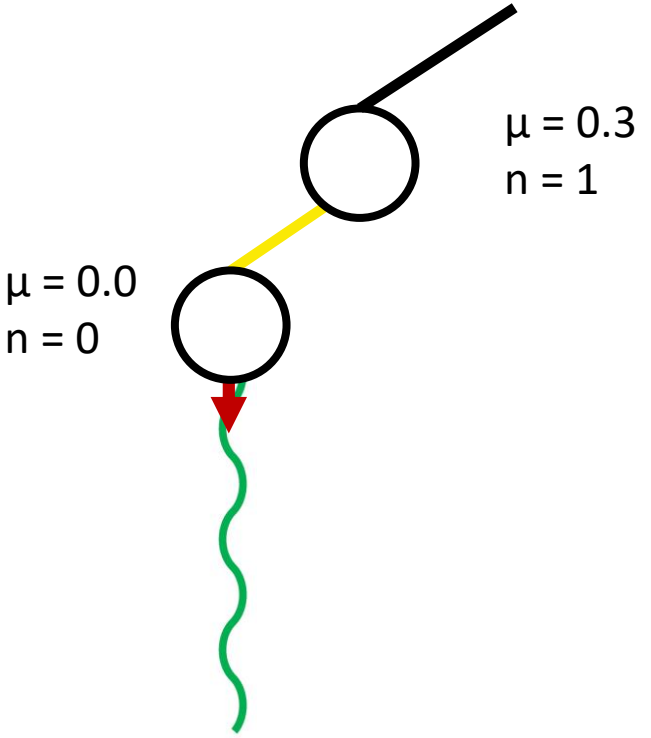
Simulation



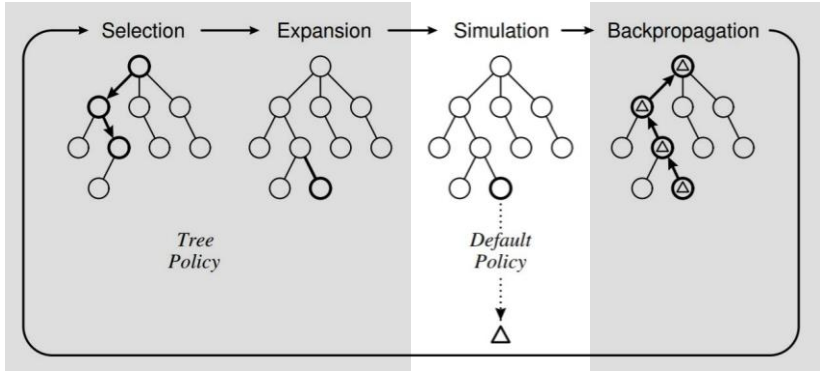
State



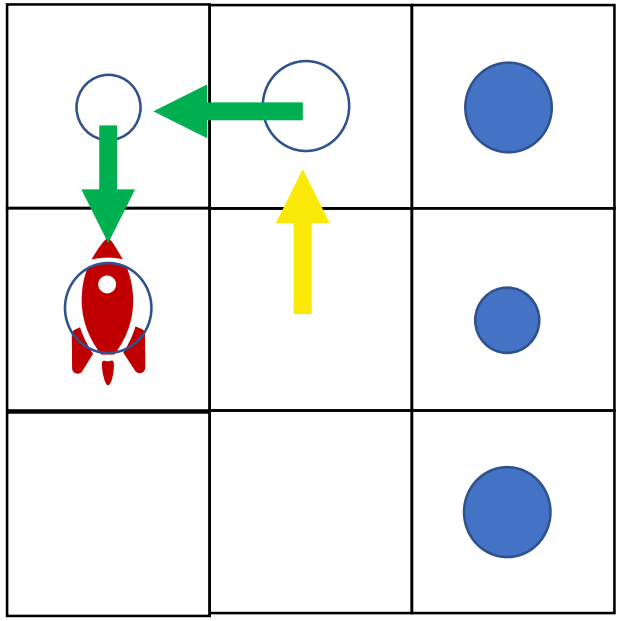
Search Tree



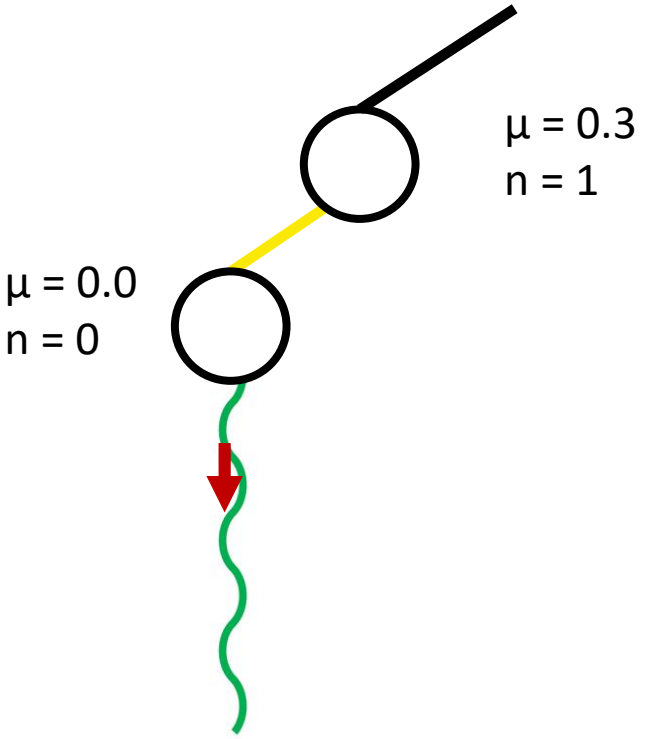
Simulation



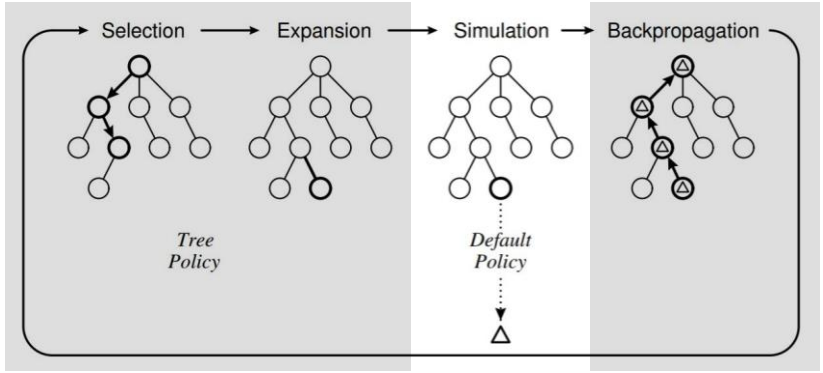
State



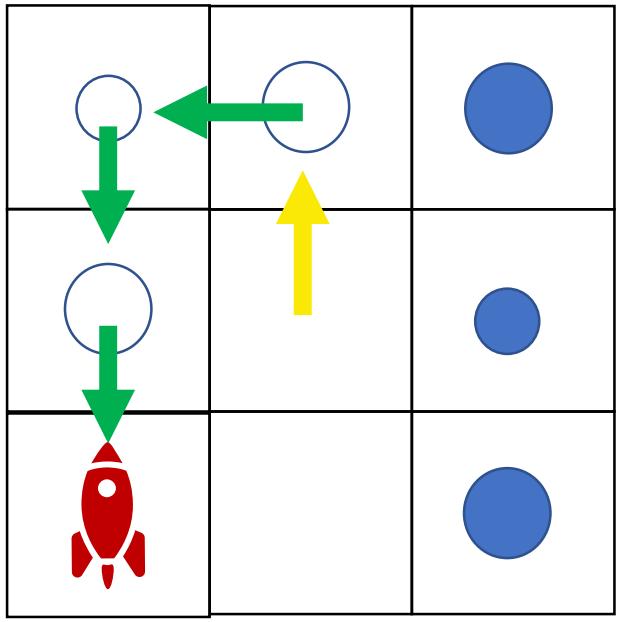
Search Tree



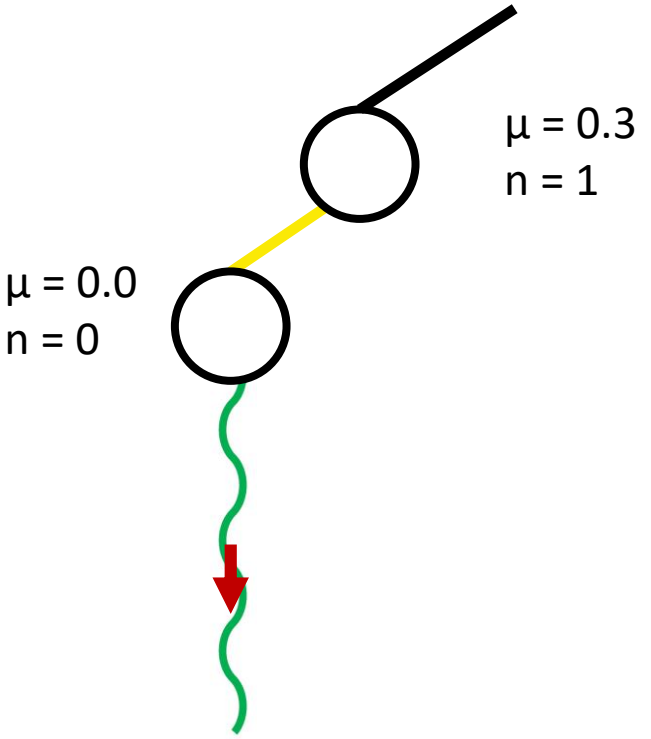
Simulation



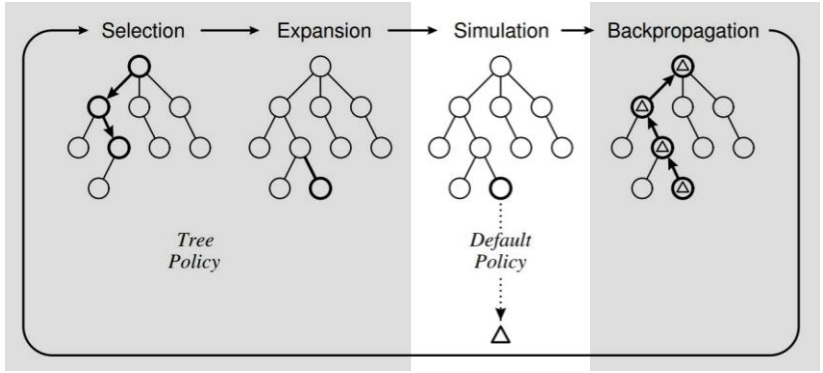
State



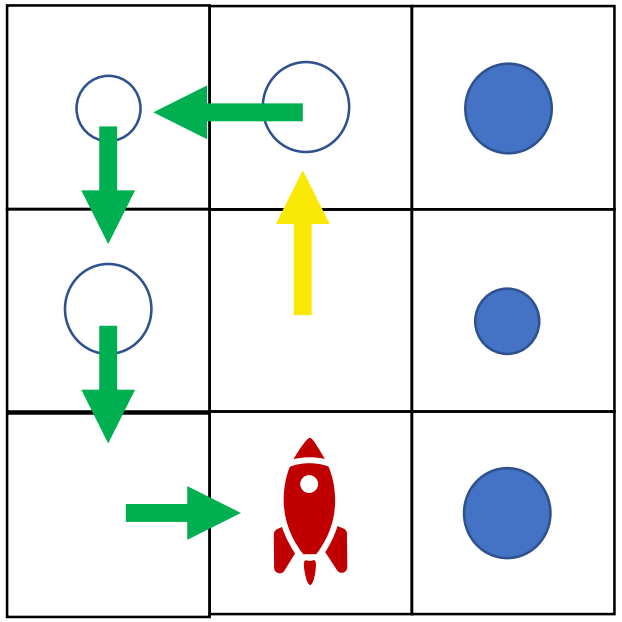
Search Tree



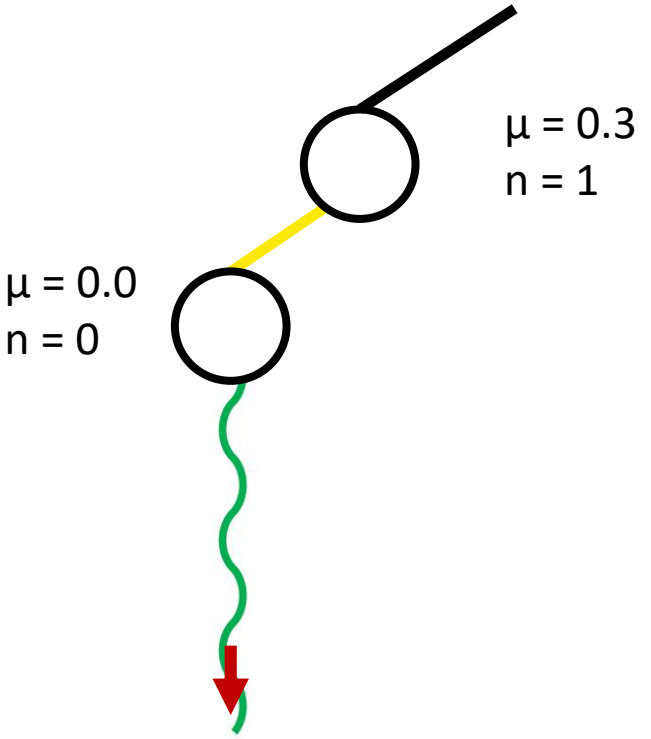
Simulation



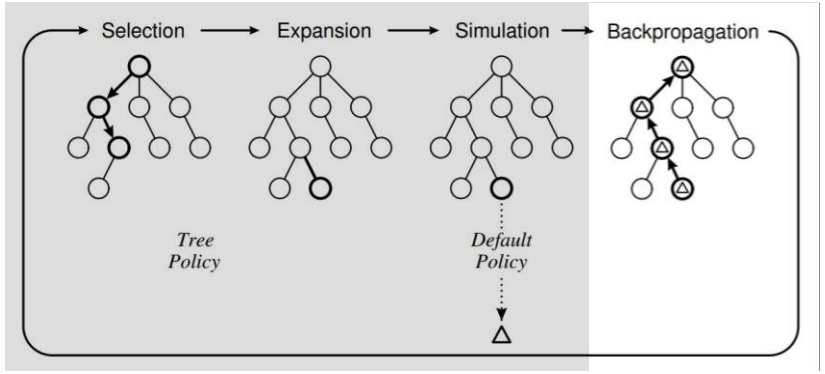
State



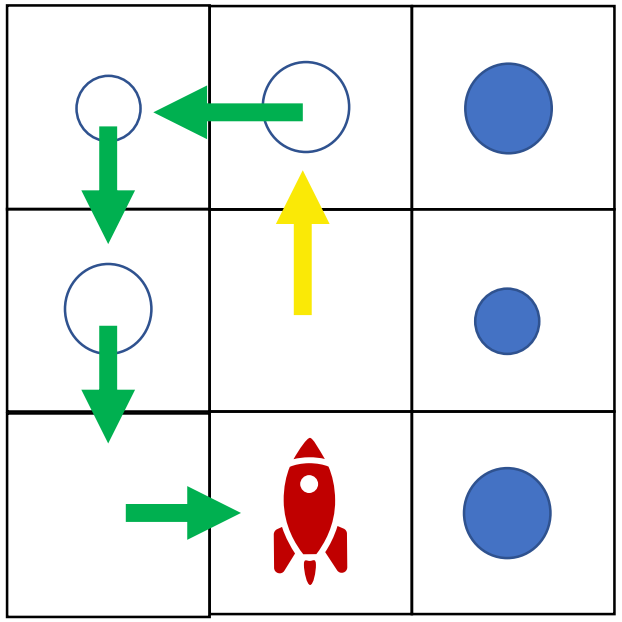
Search Tree



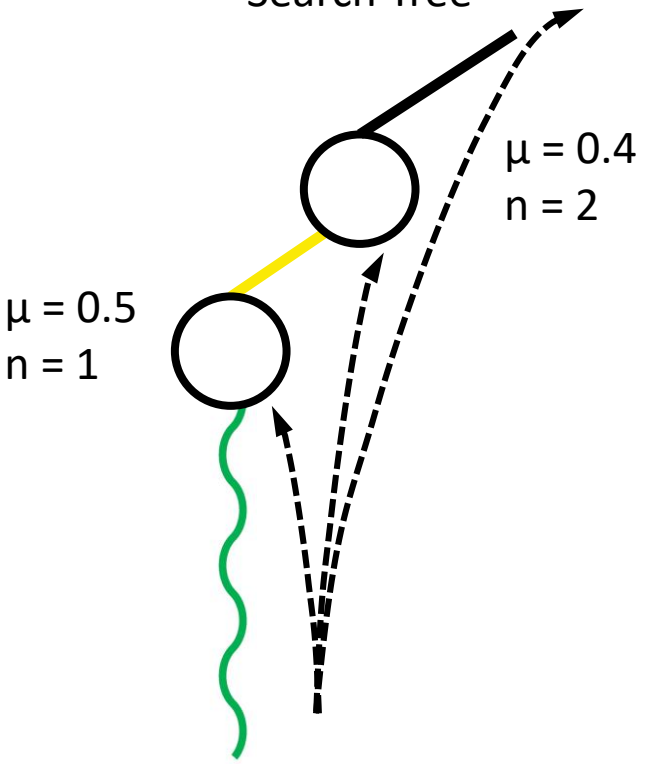
Backpropagation



State

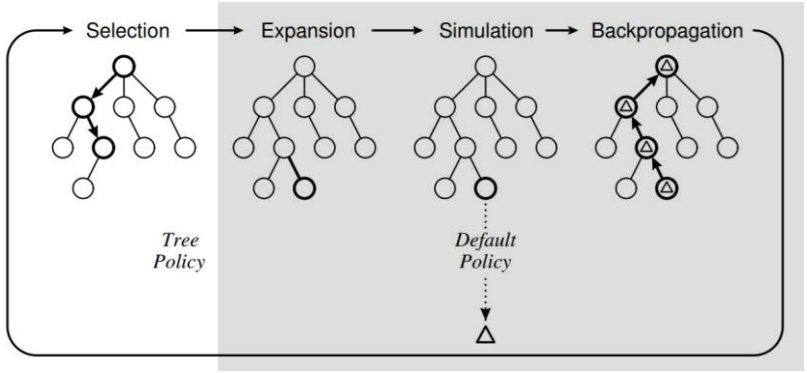


Search Tree

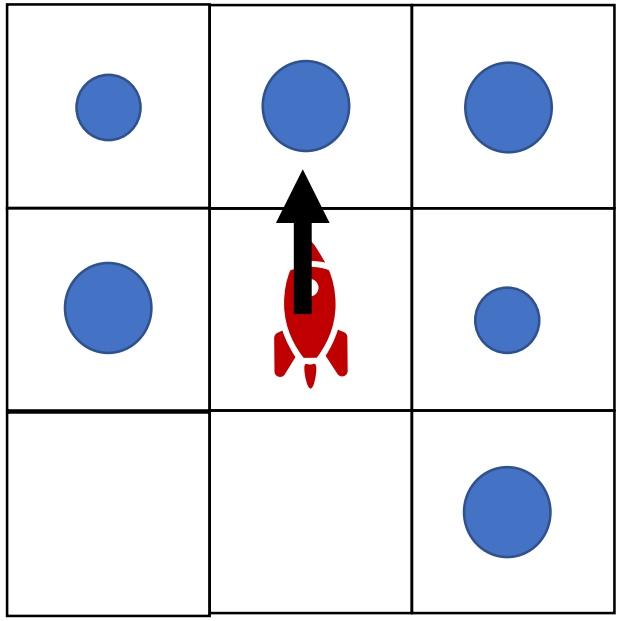


Reward: 0.5

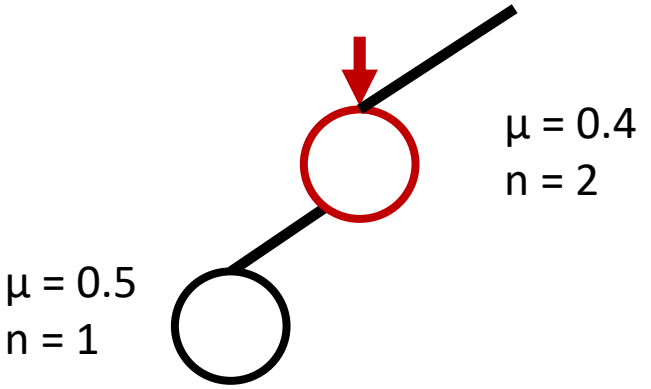
Selection



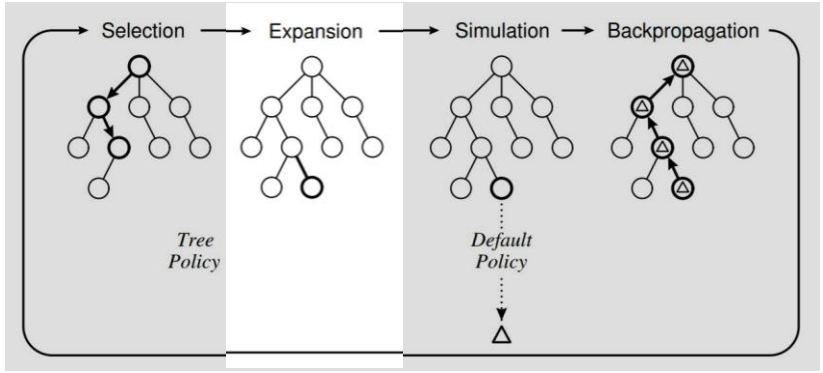
State



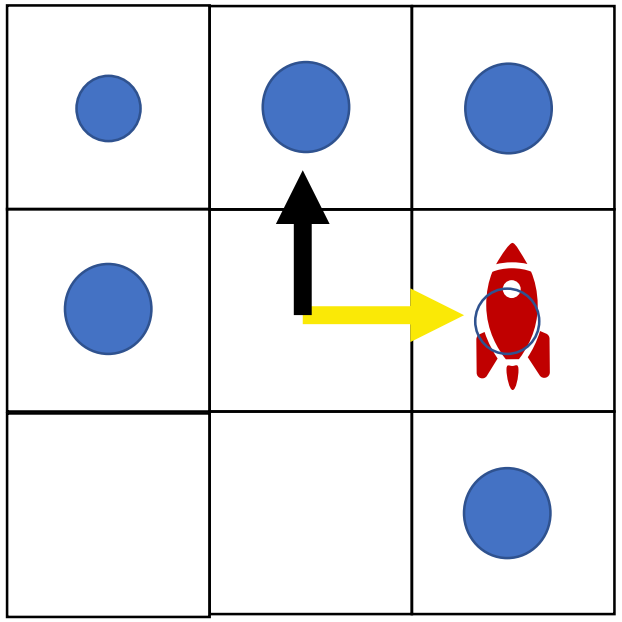
Search Tree



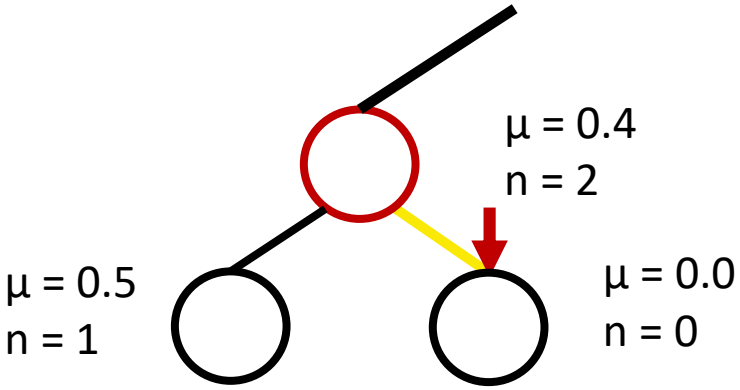
Expansion



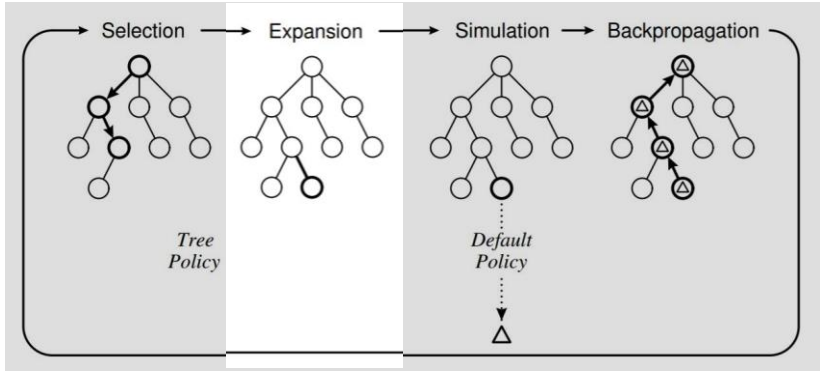
State



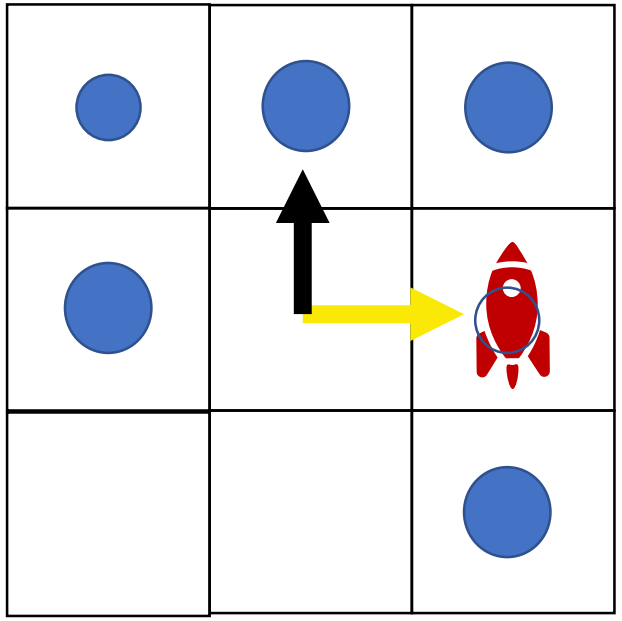
Search Tree



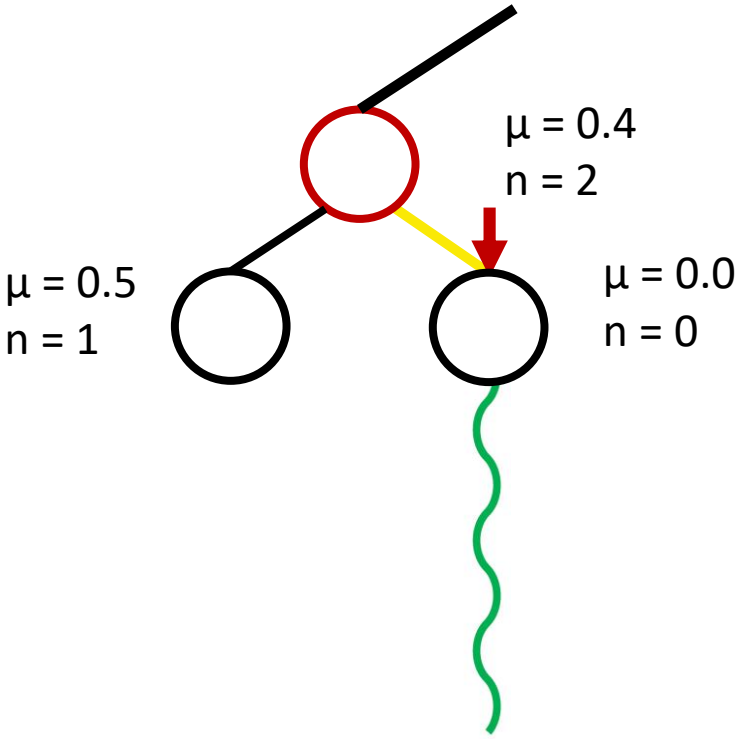
Simulation



State

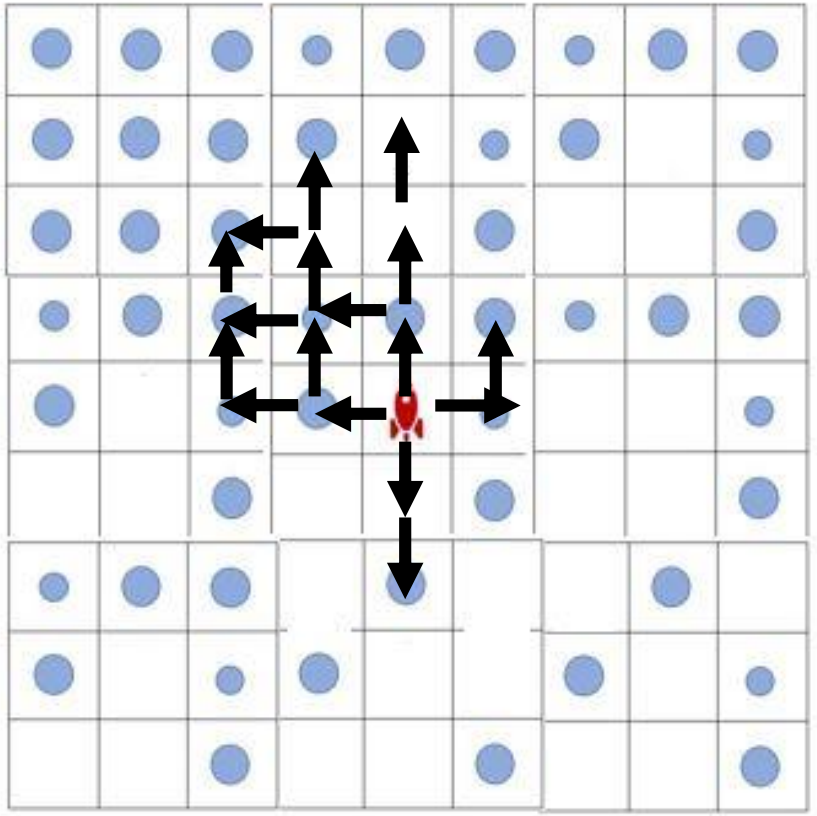


Search Tree



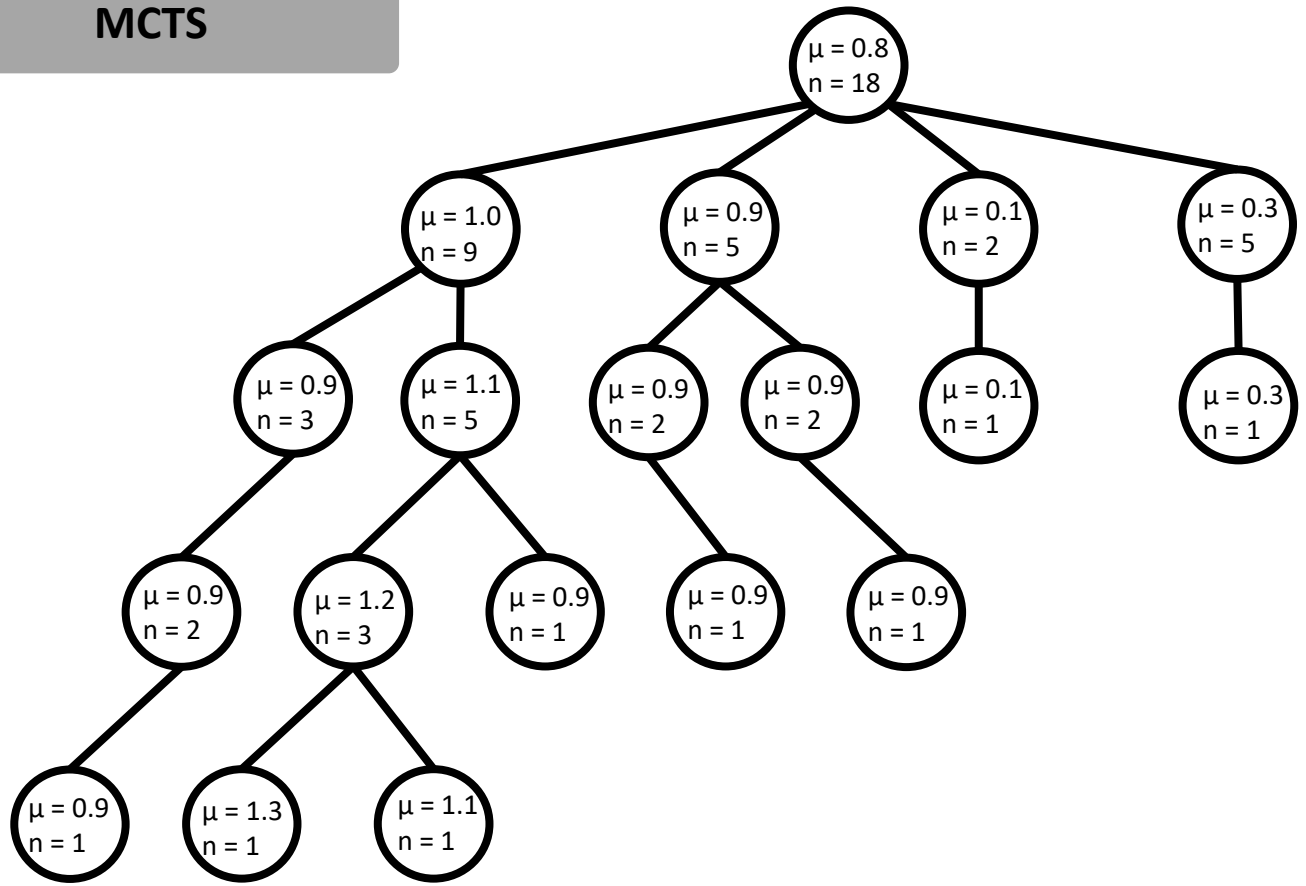
After Many Rounds…

State



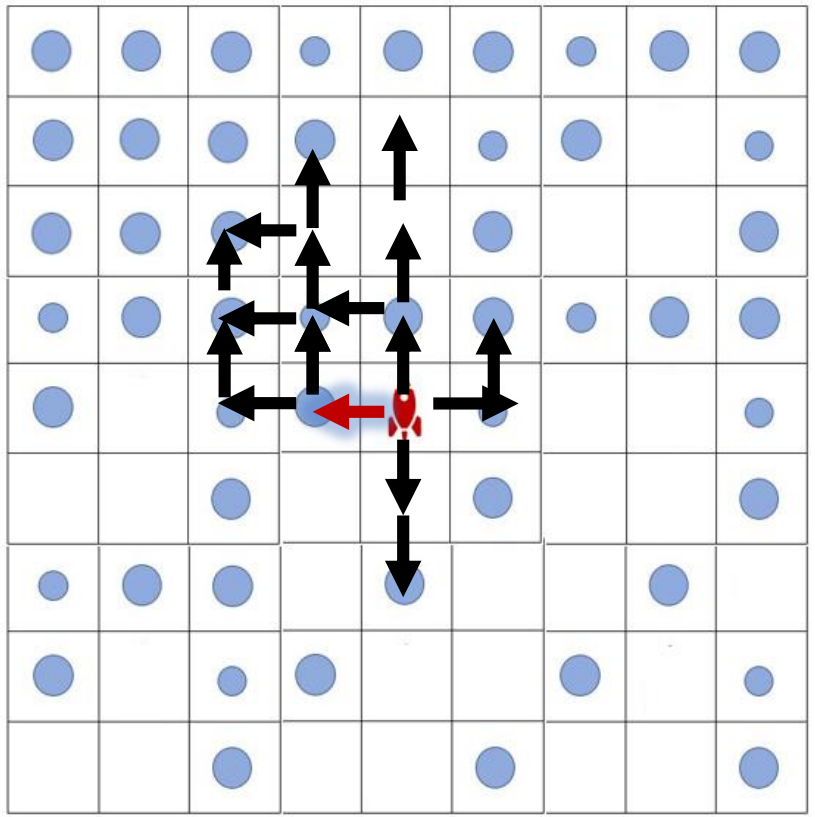
MCTS

Search Tree



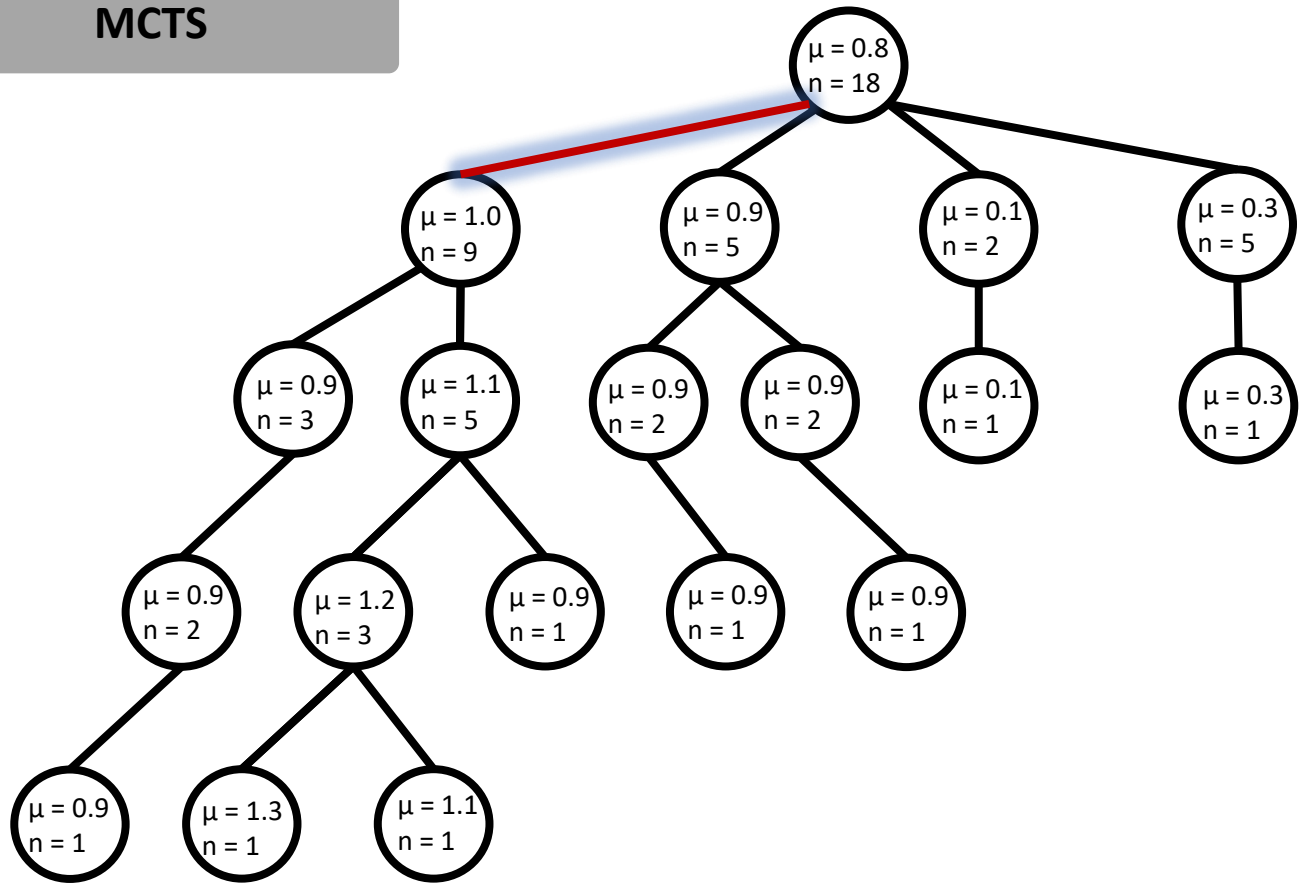
Execution

State



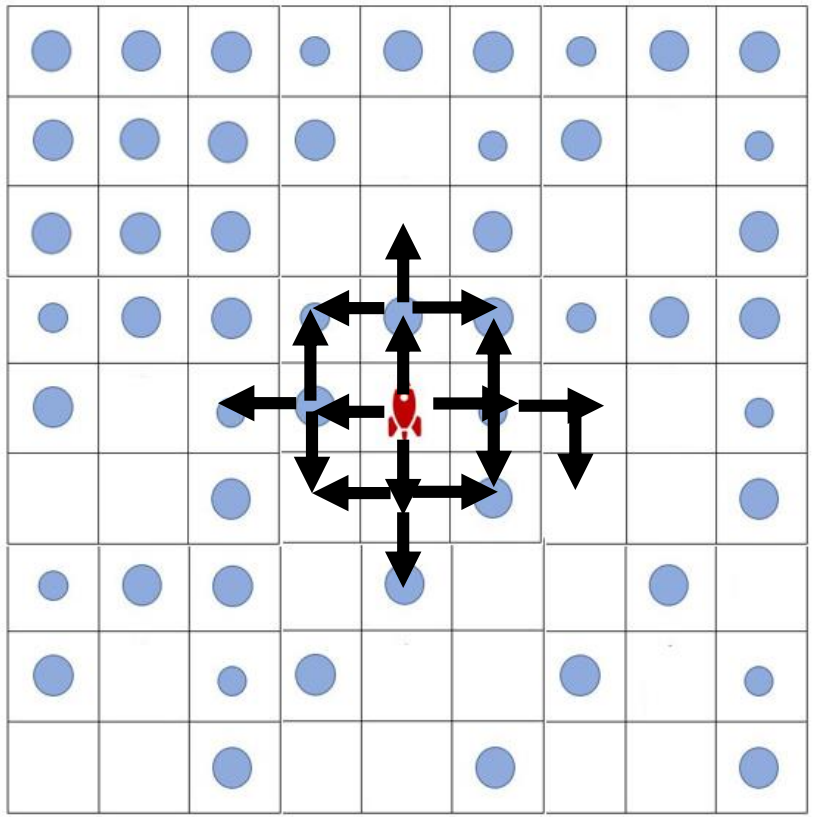
Search Tree

MCTS



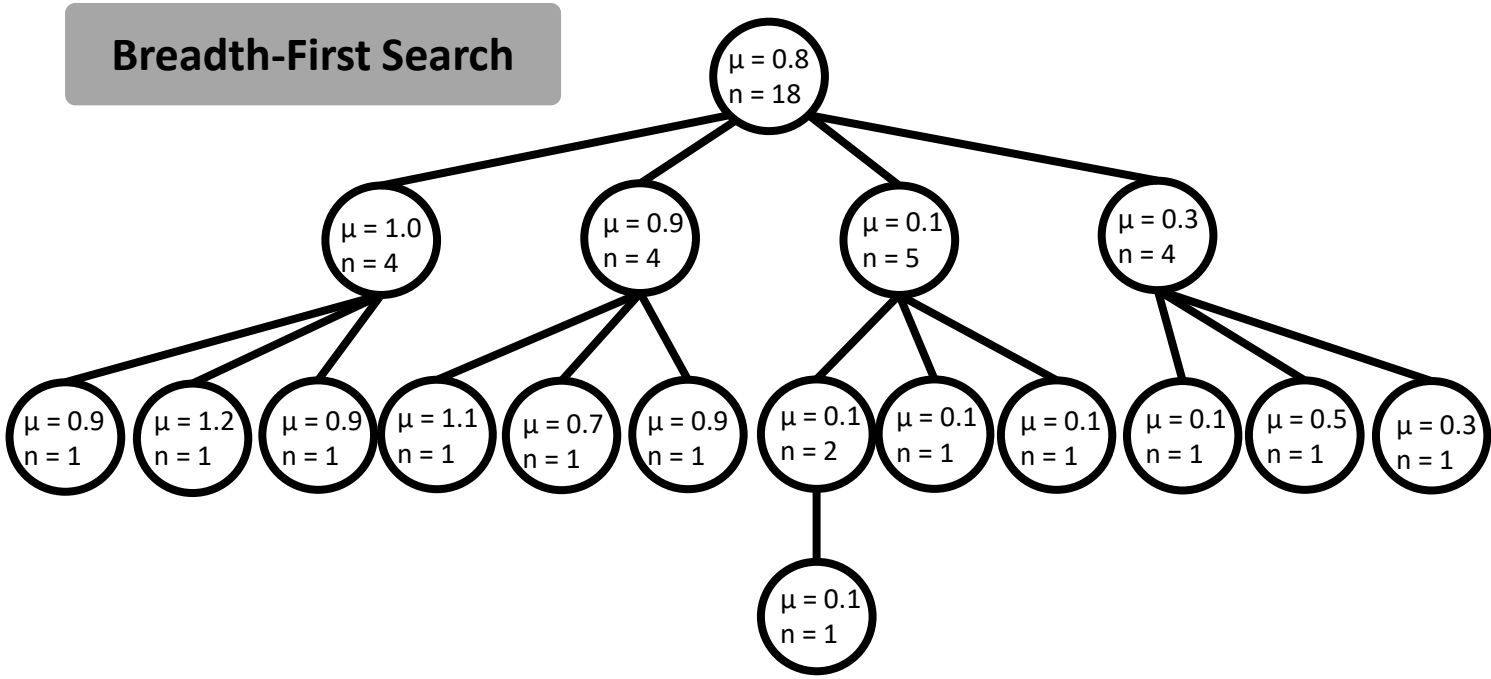
Tree Symmetry: Comparison to BFS

State

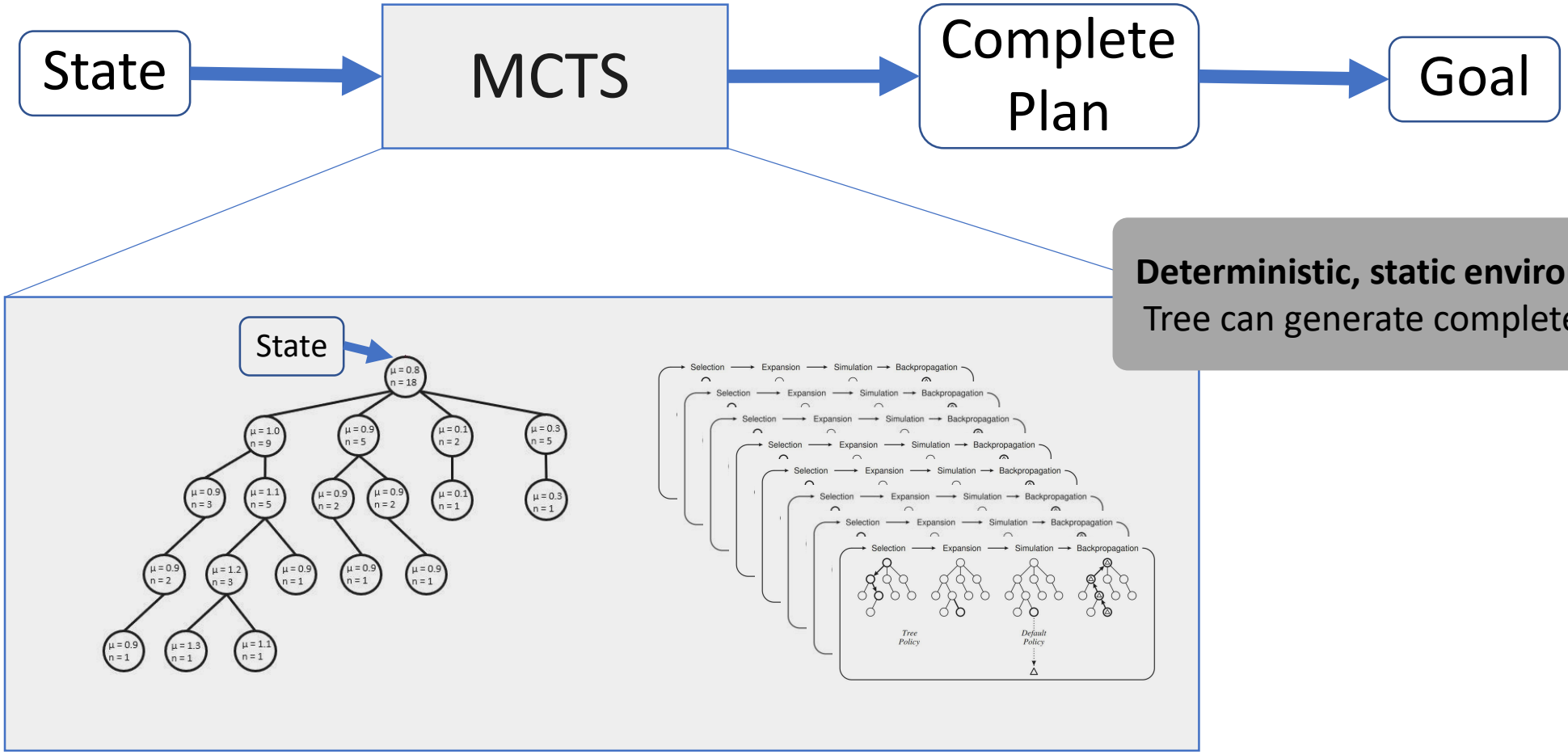


Search Tree

Breadth-First Search

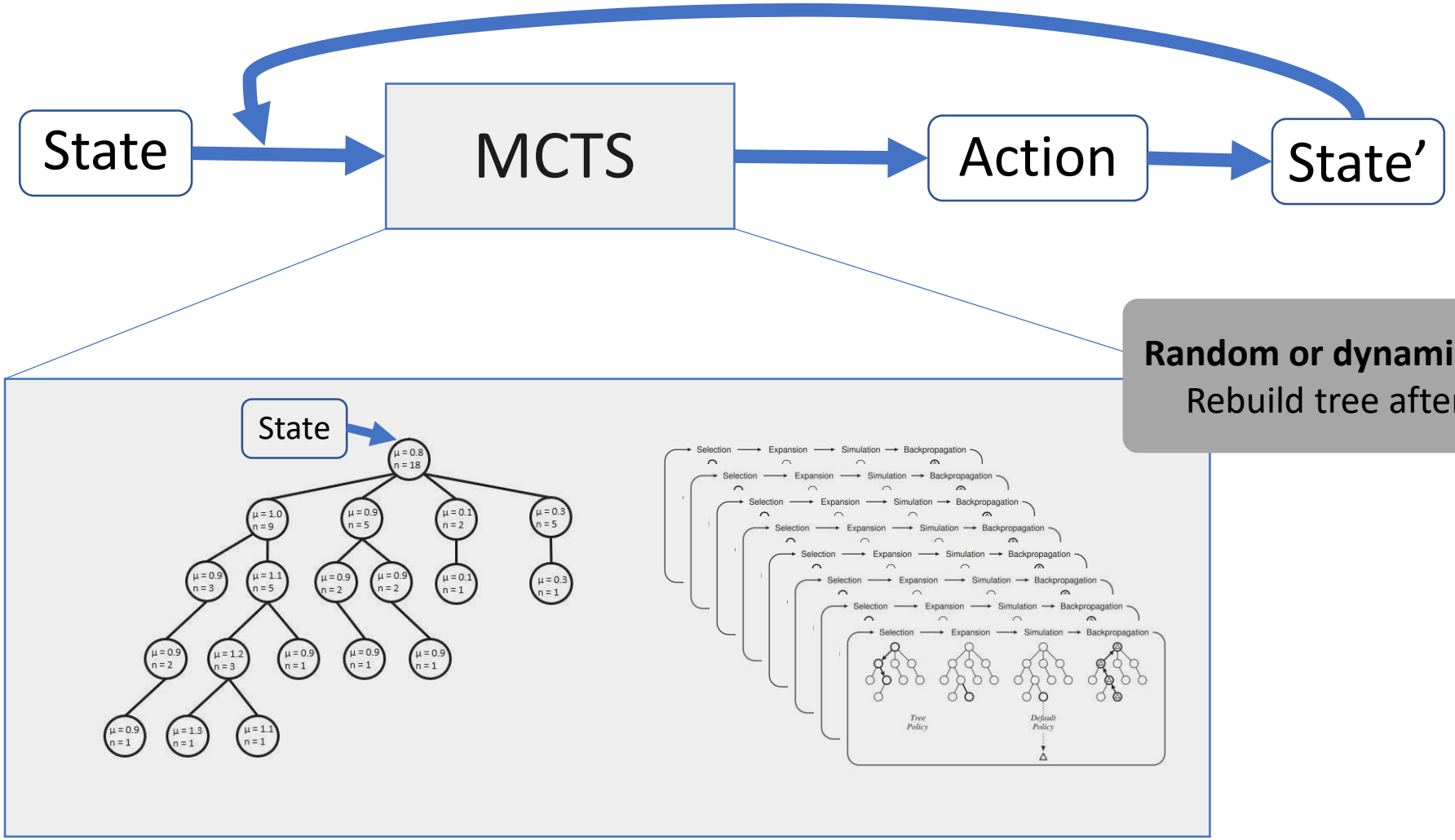


Planning and Acting with MCTS



Deterministic, static environment: Tree can generate complete plans

Planning and Acting with MCTS



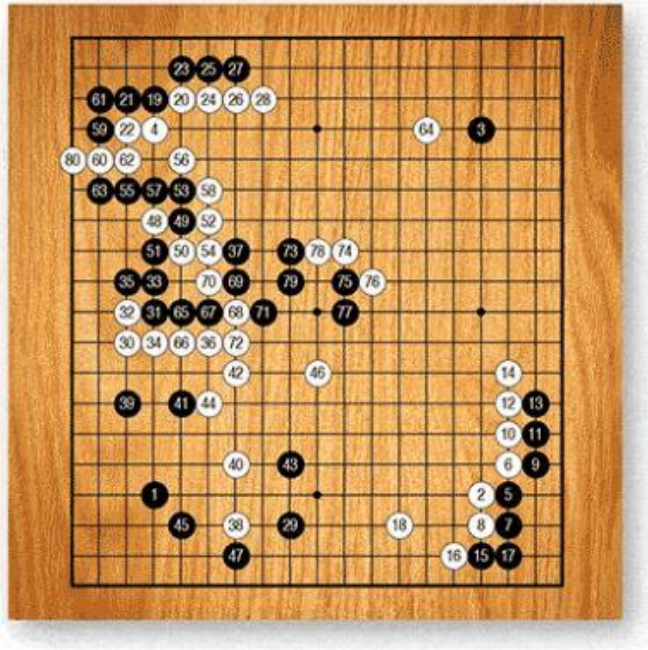
Random or dynamic environment:
Rebuild tree after each action

Basic Characteristics of MCTS

Aheuristic	A heuristic is learned rather than hand-defined
Asymmetric	Search tree focuses more on interesting areas
Anytime	Proposes a complete policy at each step

MCTS is modular and different configurations may "tweak" these properties!

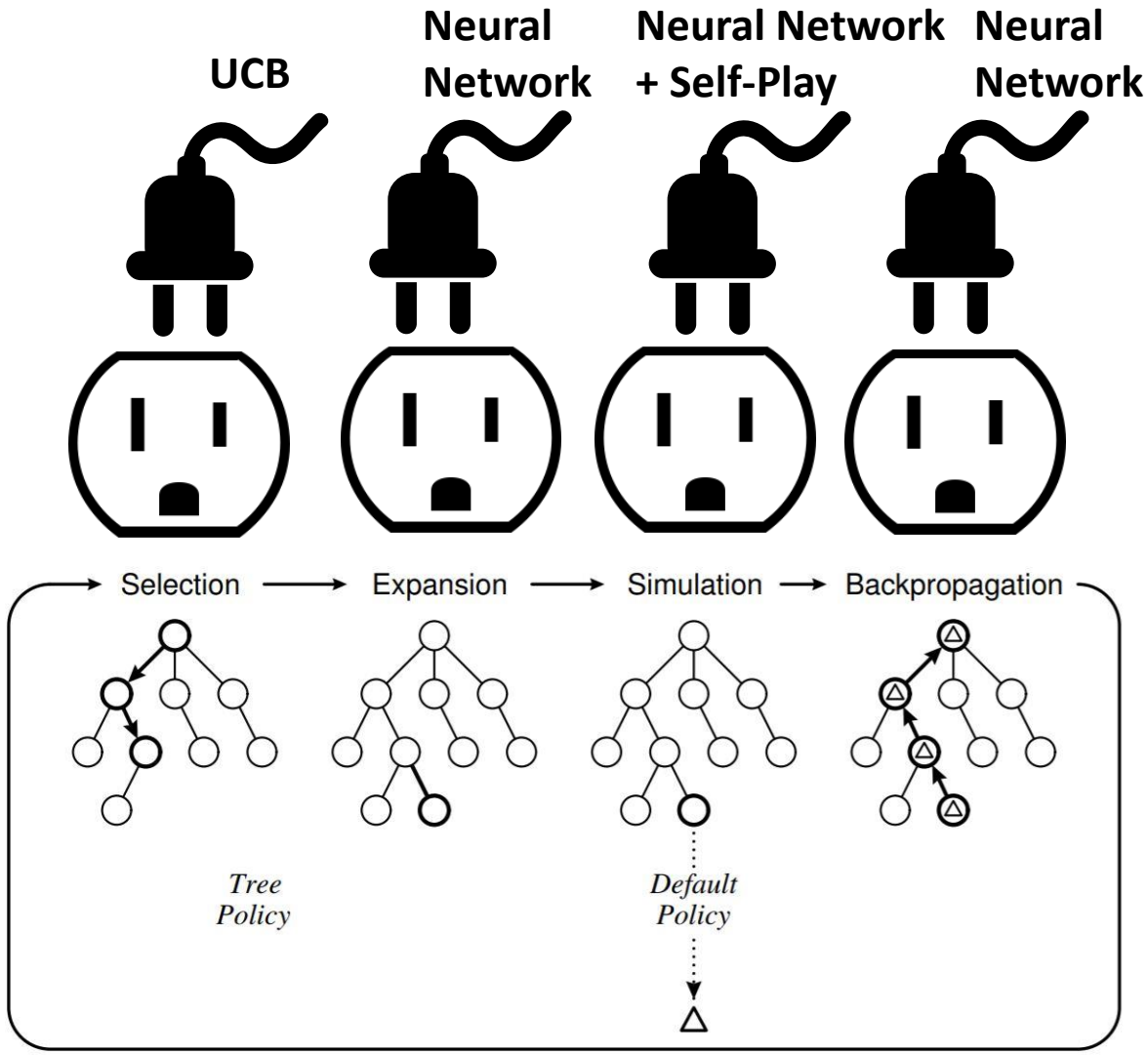
AlphaGo Zero

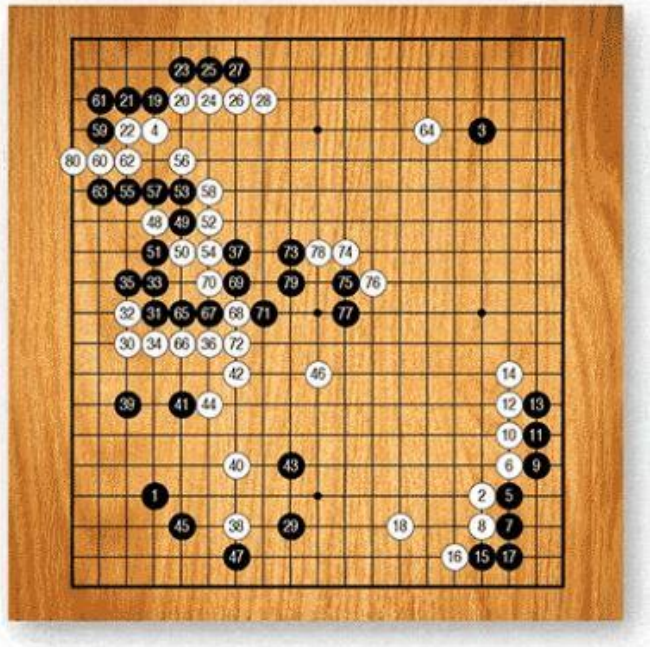


A game of Go.

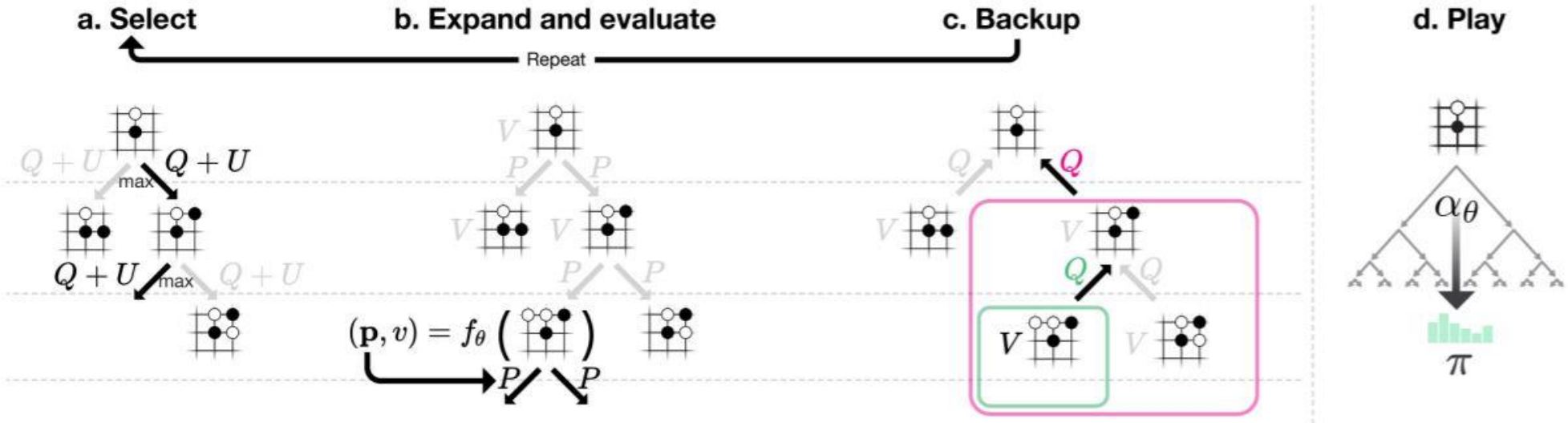
Images from DeepMind website

MCTS in AlphaGo Zero





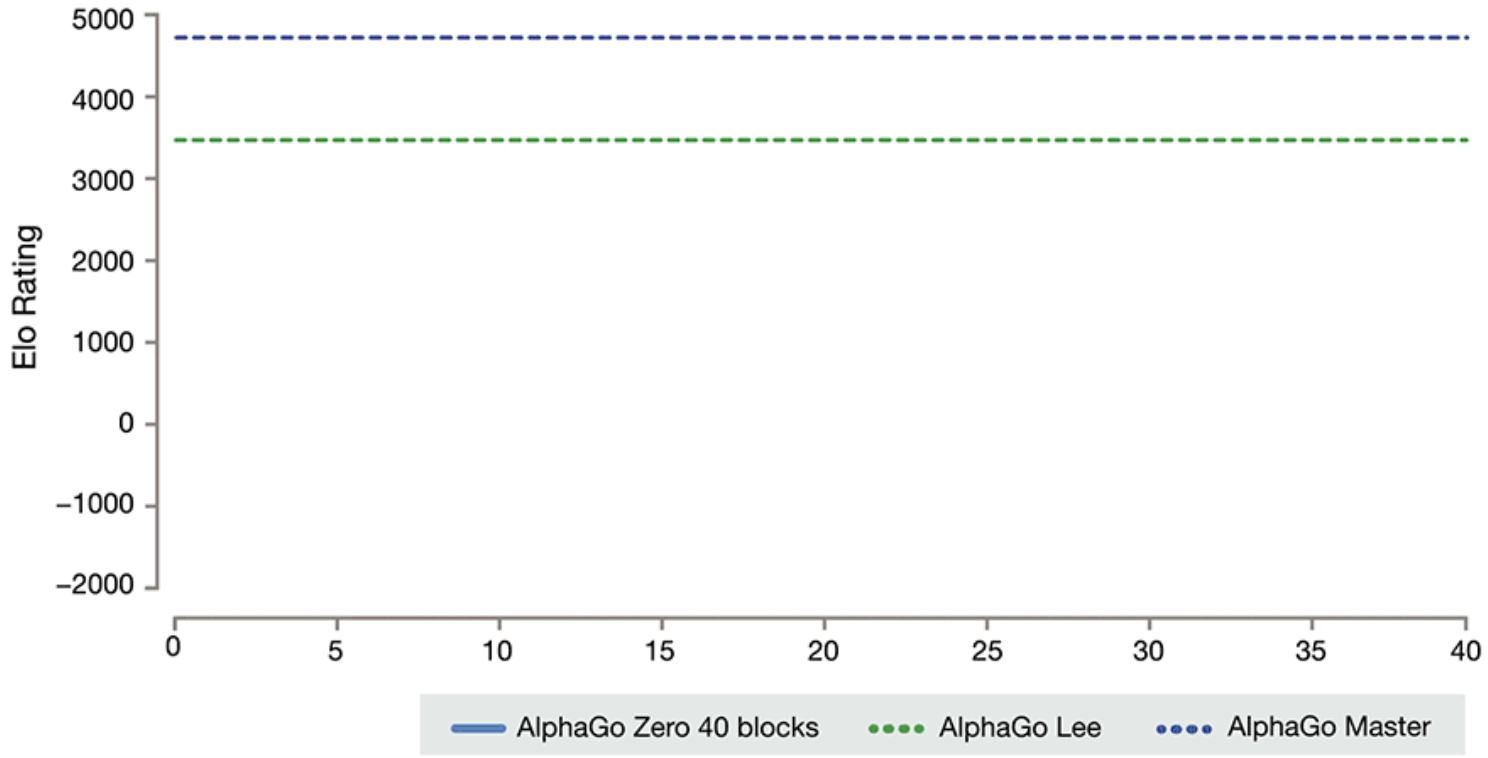
MCTS in AlphaGo Zero



MCTS in AlphaGo Zero

Images from DeepMind website

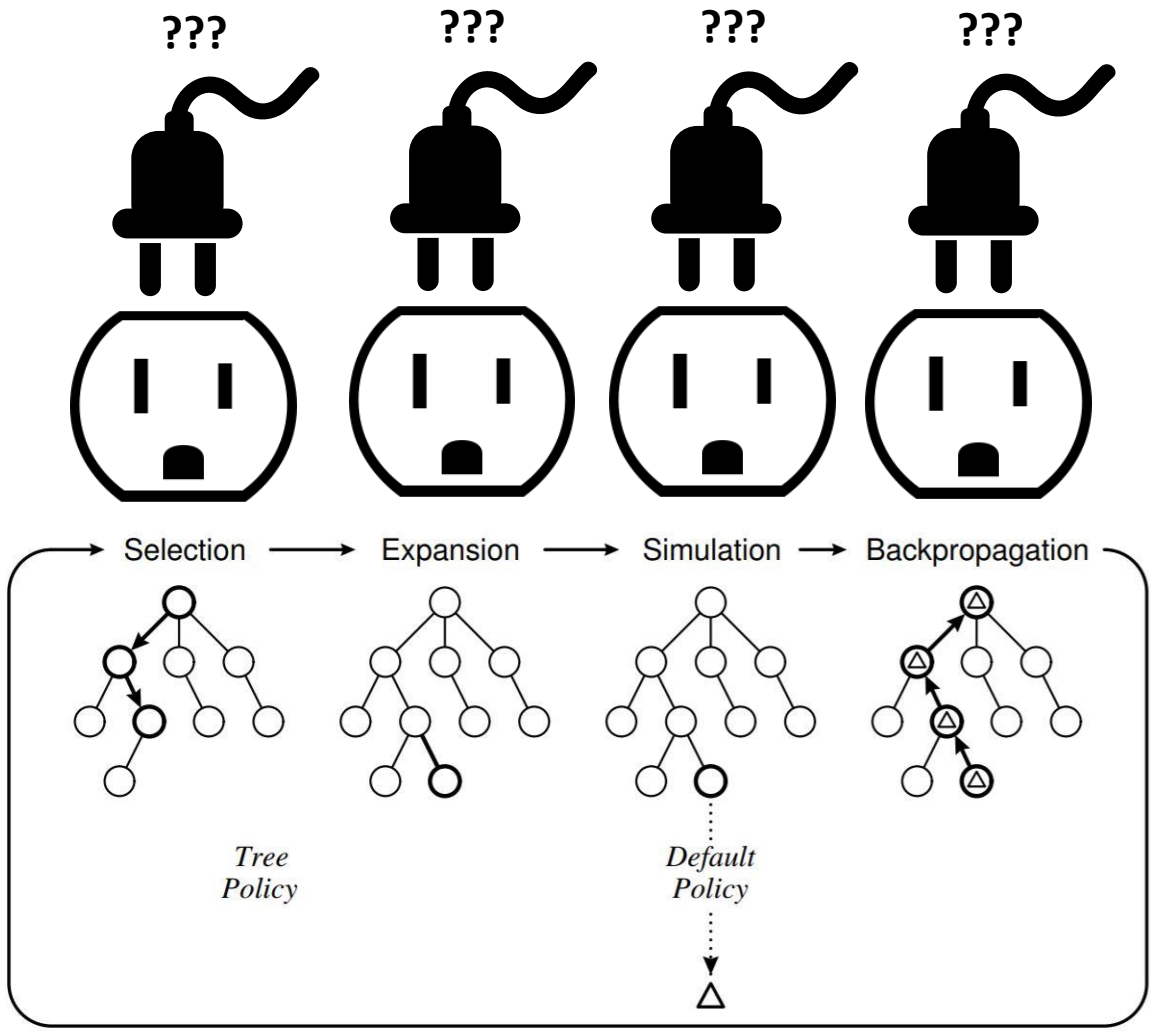
AlphaGo Zero Performance

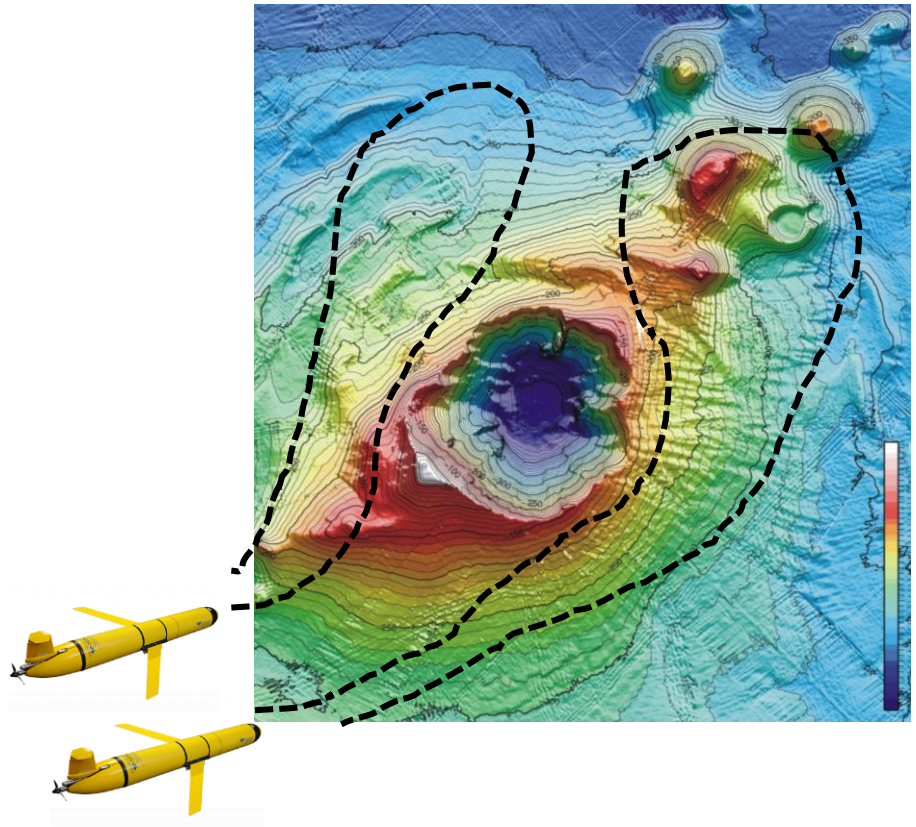


AlphaGo Zero performance.

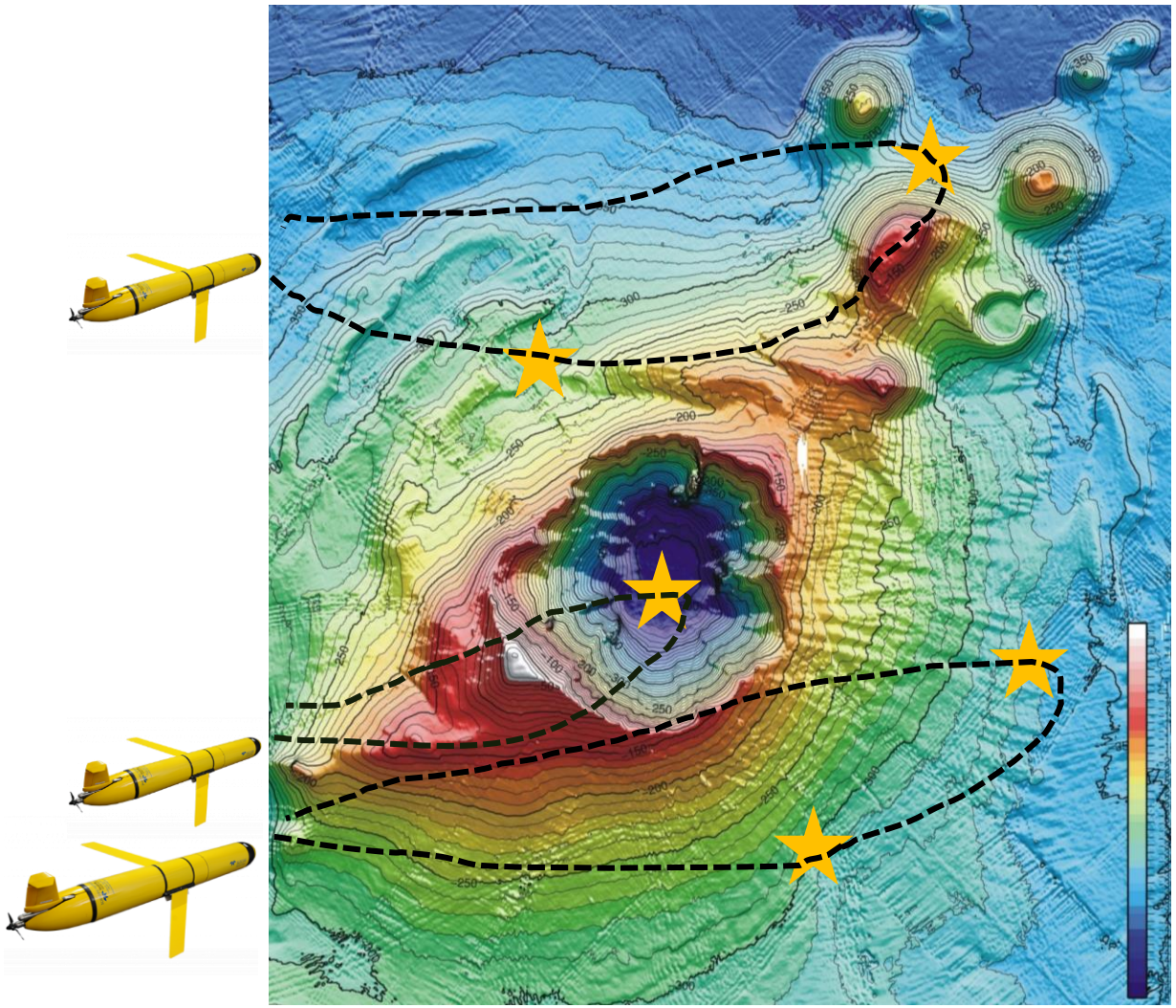
Images from DeepMind website

MCTS in Path Planning

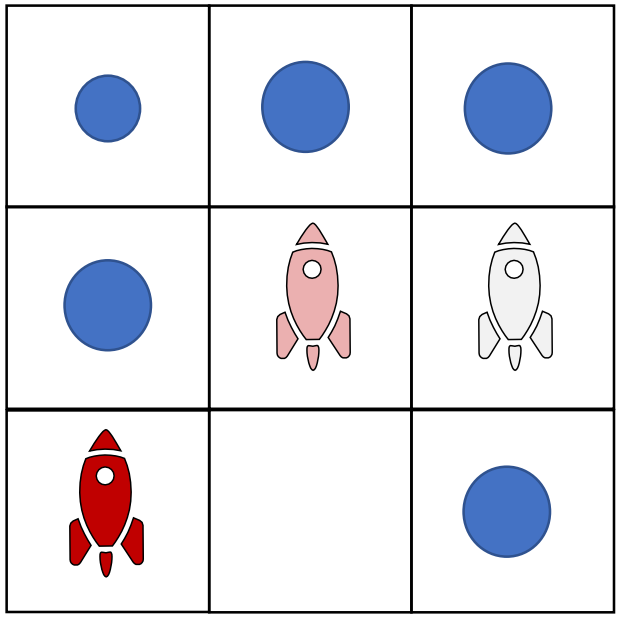




Extension to Collaboration

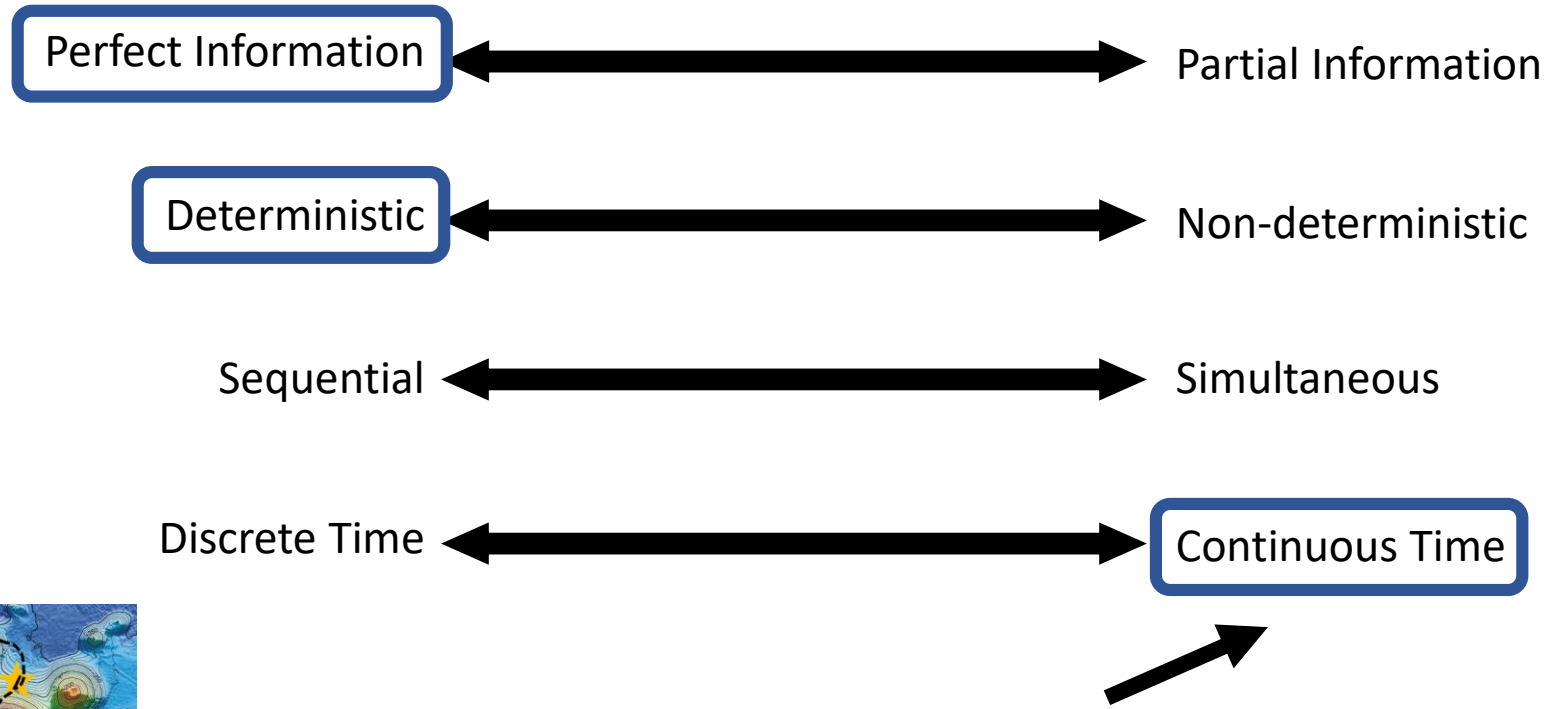


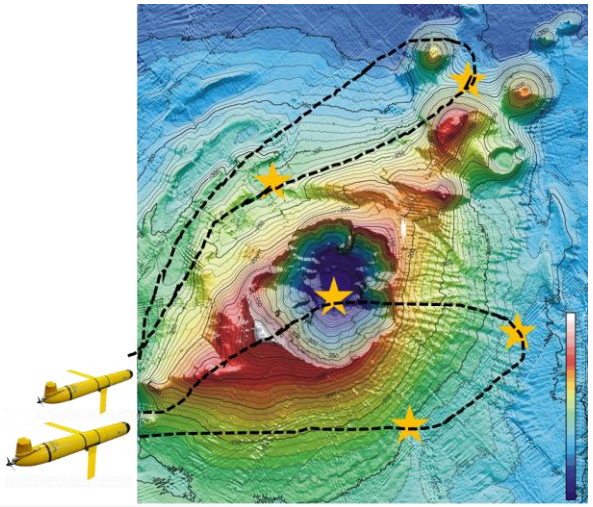
Discretized version with multiple gliders



Great. Solved, right?

Extension to Collaboration

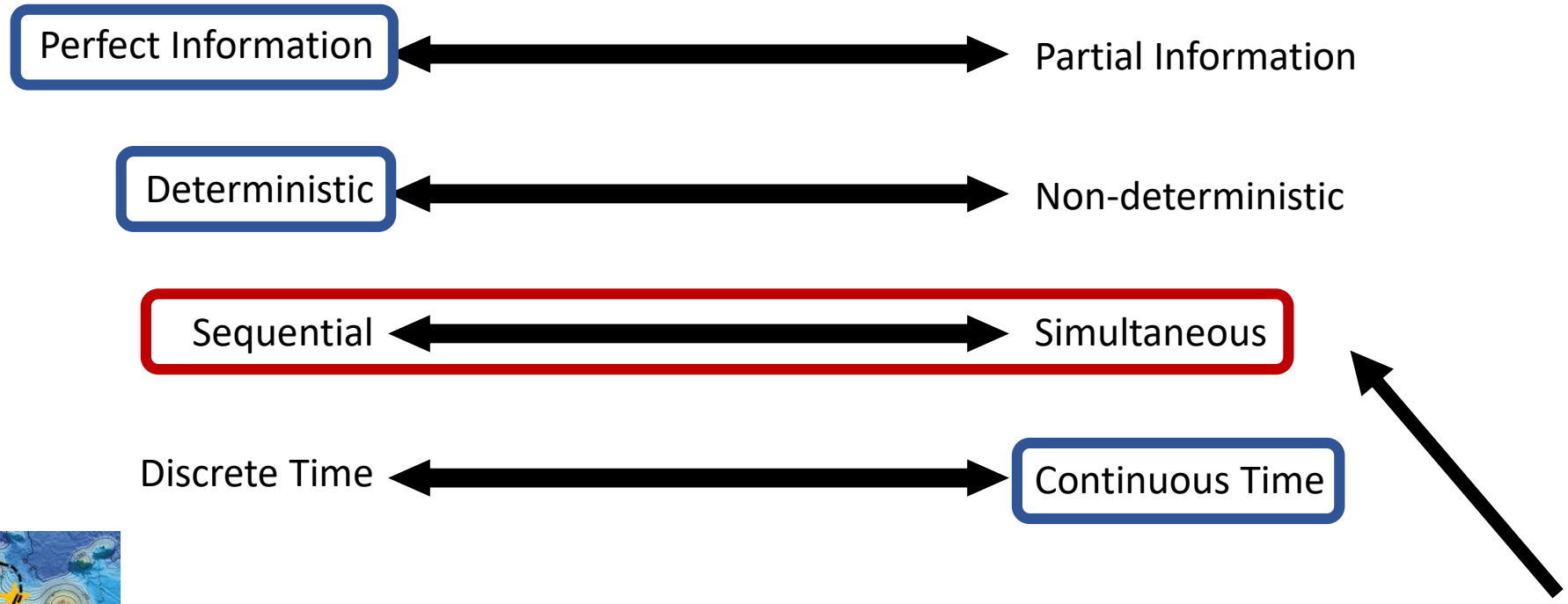


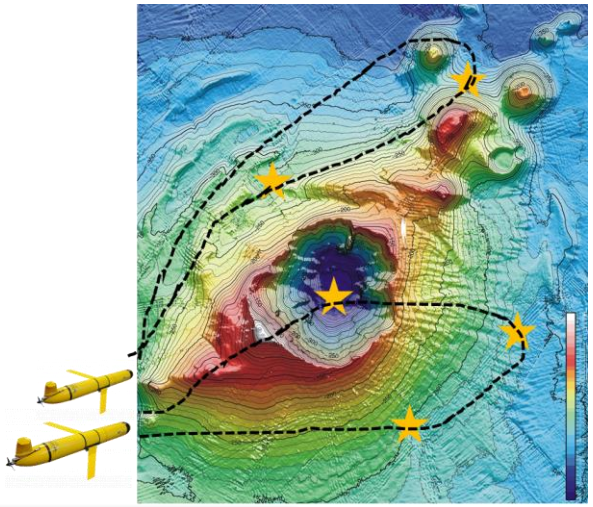


Because we have multiple agents acting, the world changes whether an individual agent takes a step or not. Thus our problem is continuous time. This results in:

- We must re-make the tree each time
- We have a limited amount of time to plan

Extension to Collaboration

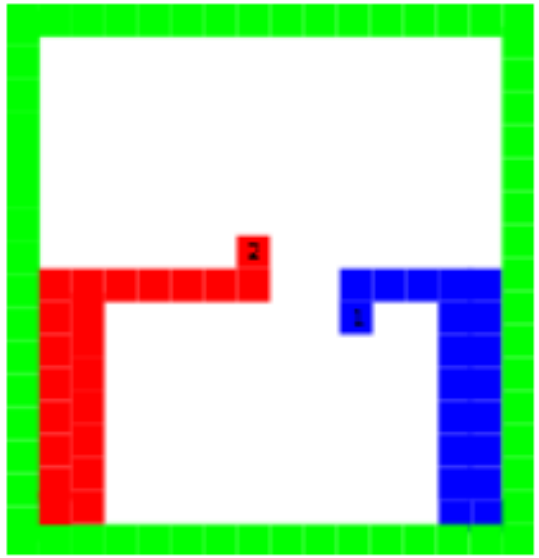




Unlike tic tac toe and the single agent application of MCTS to the search of the Kolumbo Volcano site, our multiple agents would **not act sequentially but rather in parallel.**

Does the non-sequential action of the agents matter?

Short answer: Probably not



Samothrakis et al. applied MCTS to the game TRON in 2010. They stated:

> "Although a simultaneous move game, it can be transformed into an alternate moves games by assuming that in each player's tree node the current player plays always first"

They pointed to the University of Waterloo Tron Competition, where the best versions of MCTS—employing TRON-specific leaf heuristics—performed comparably to other approaches.

S. Samothrakis, D. Robles, and S. M. Lucas, "A uct agent for tron: Initial investigations," in Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games, 2010, pp. 365–371. DOI: 10.1

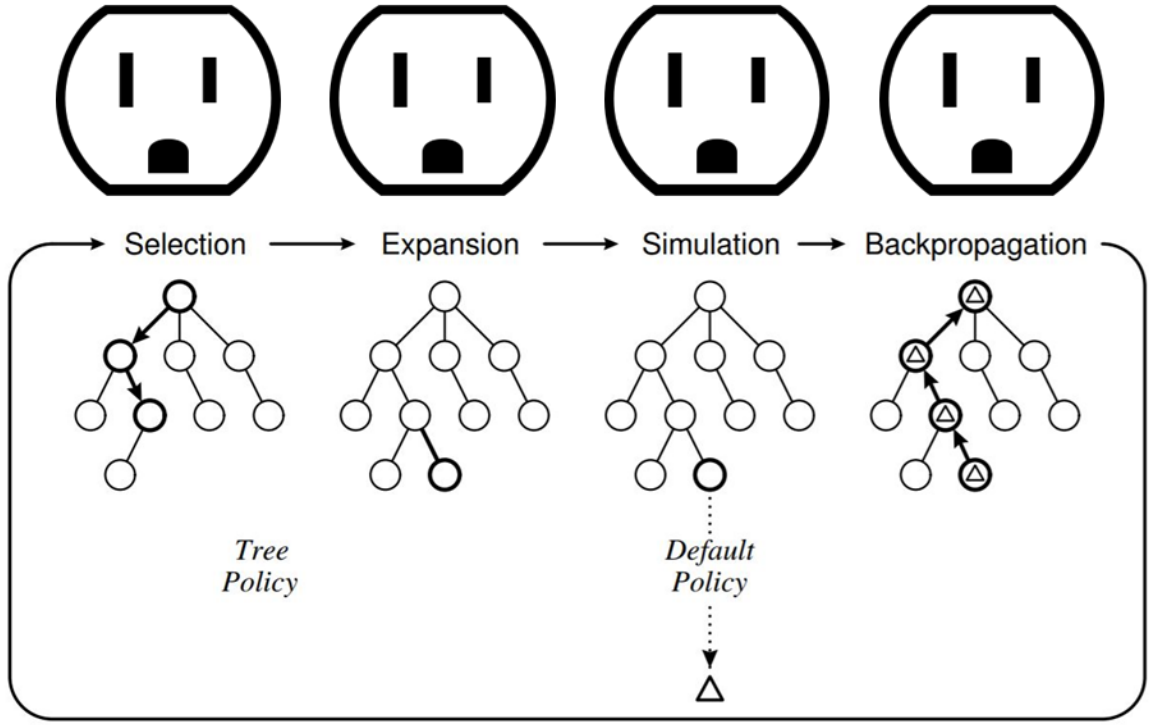
The guts of MCTS will remain the same

Perfect Information ⟷ Partial Information

Deterministic ⟷ Non-deterministic

Sequential ⟷ Simultaneous

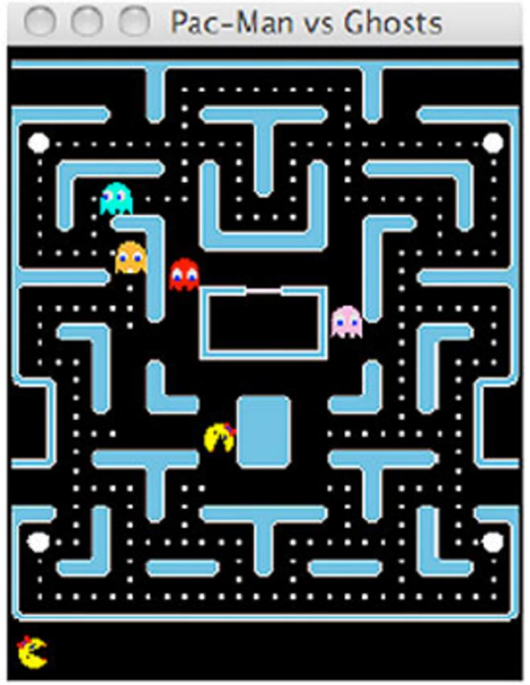
Discrete Time ⟷ Continuous Time



Outline

- Background
  - Monte Carlo Simulation
  - Upper Confidence Bounds
- Implementing MCTS with Upper Confidence Bounds for Trees
- Collaborative MCTS
  - Discuss Max-N, a variant of MCTS for multiple agents
  - Discuss how to decentralize MCTS
  - Discuss how to improve the default policy in decentralized MCTS

Ms. Pacman Competition (SM Lucas, 2007)



Team 1

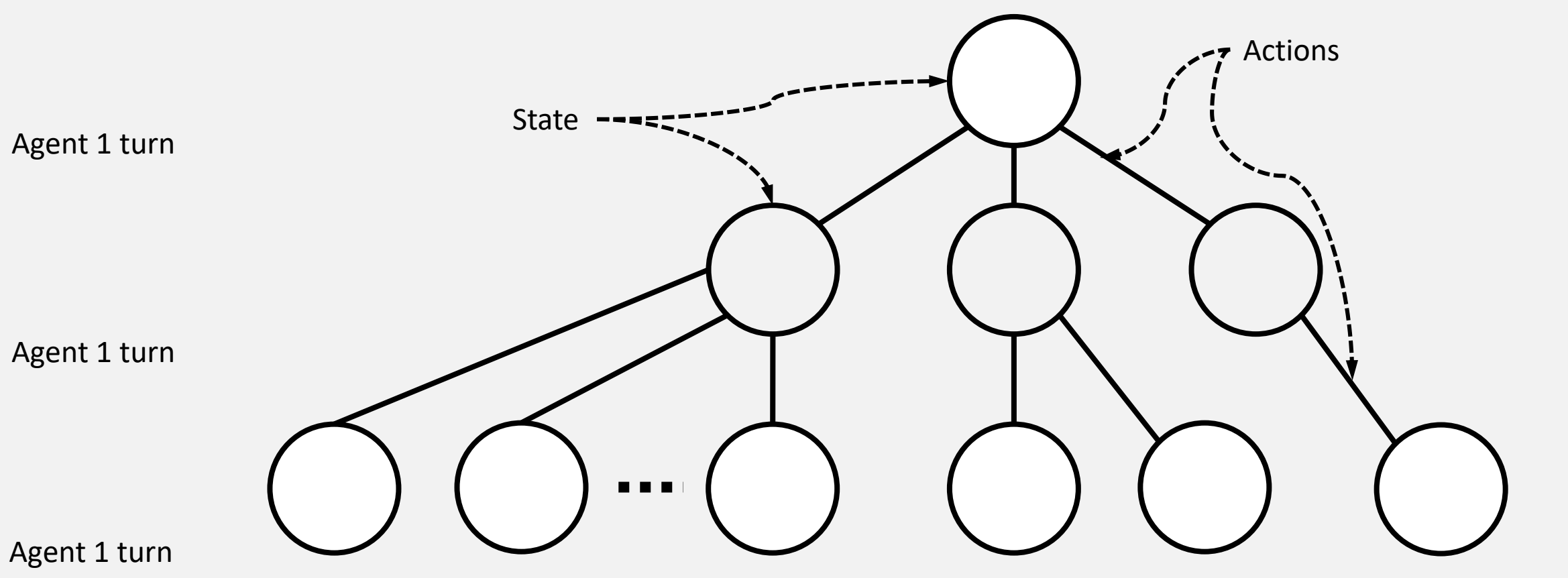
Team 2

- Perfect Information ⟷ Partial Information
- Deterministic ⟷ Non-deterministic
- Sequential ⟷ Simultaneous
- Discrete Time ⟷ Continuous Time

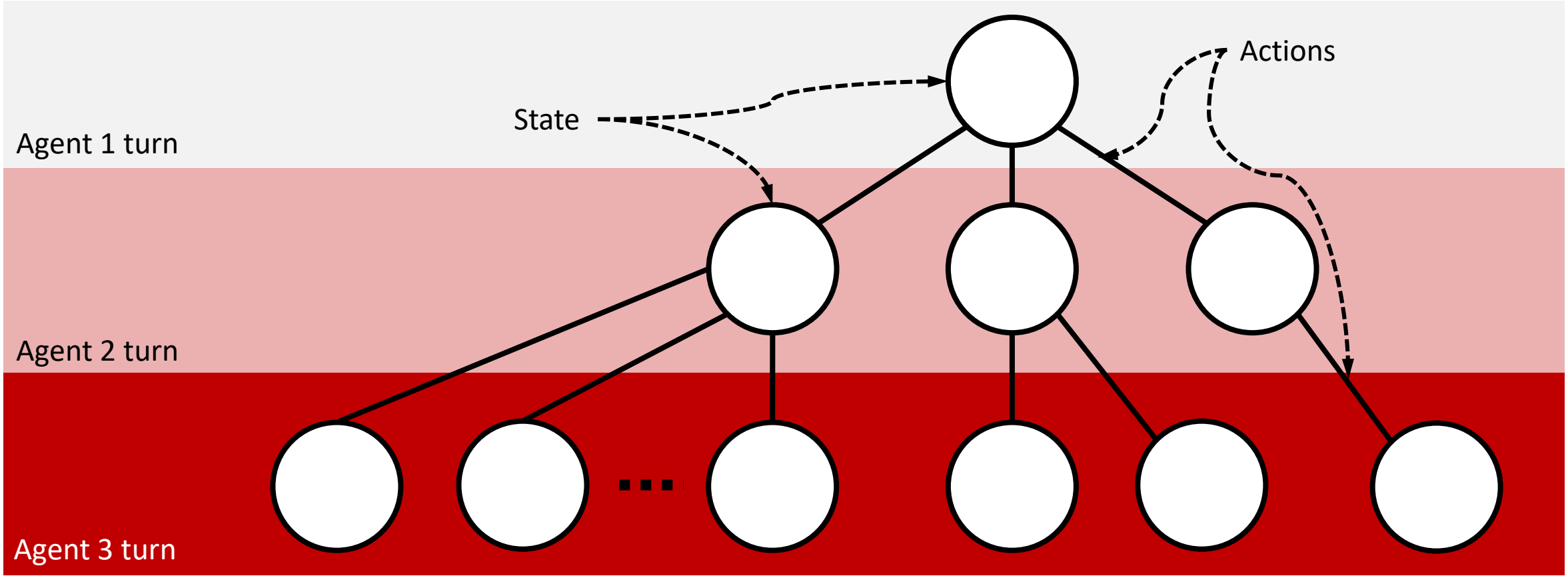
Ghosts work collaboratively to eat Ms. Pacman

S. Samothrakis, D. Robles and S. Lucas, "Fast Approximate Max-n Monte Carlo Tree Search for Ms Pac-Man," in *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 2, pp. 142-154, June 2011.doi: 10.1109/TCIAIG.2011.2144597

Collaboration through Max-N: Original Tree Structure

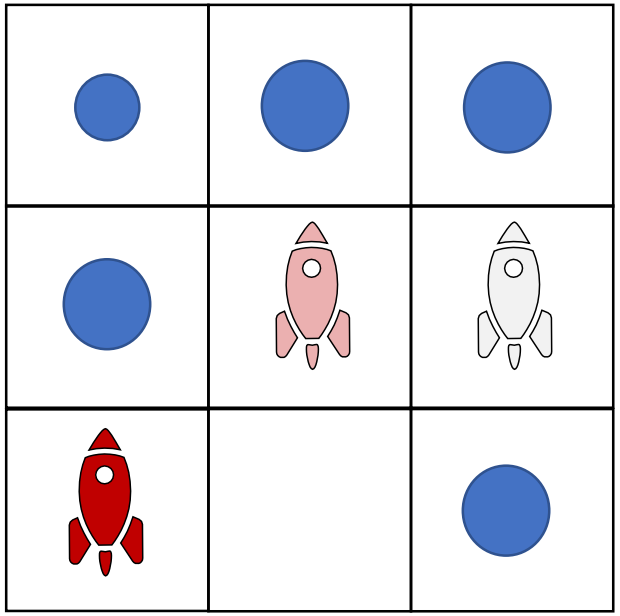


Collaboration through Max-N: Modification for Max-n

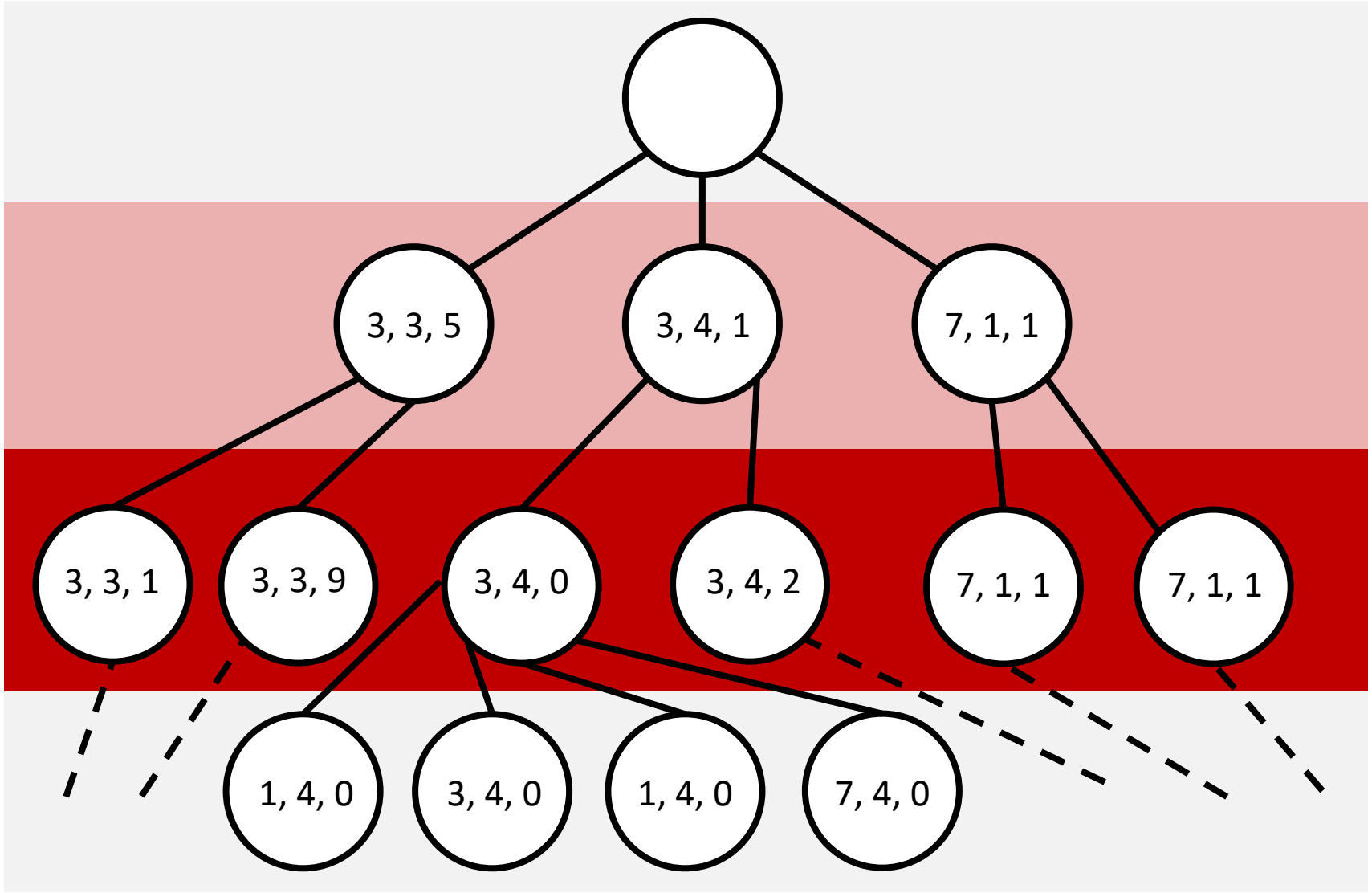


Leverage the finding from the TRON paper (Samothrakis 2010)
to assign the agents' moves sequentially

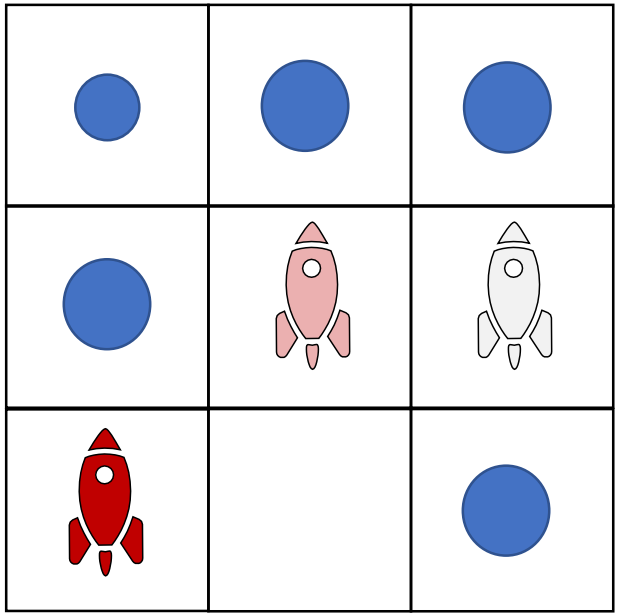
Collaboration through Max-N: Modification for Max-n



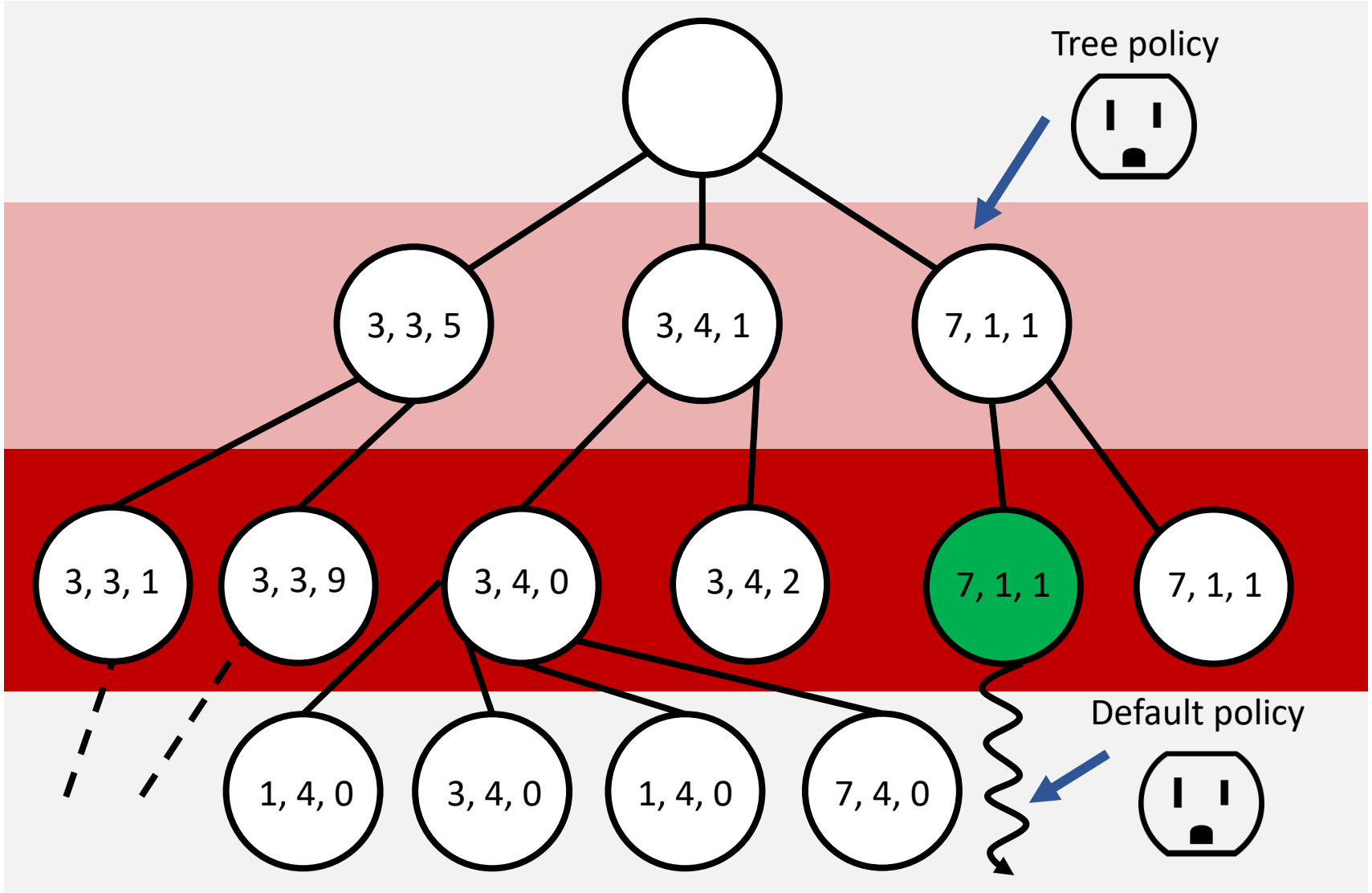
Instead of returning a single value for the playout, we return an N-tuple for N agents



Collaboration through Max-N: Modification for Max-n



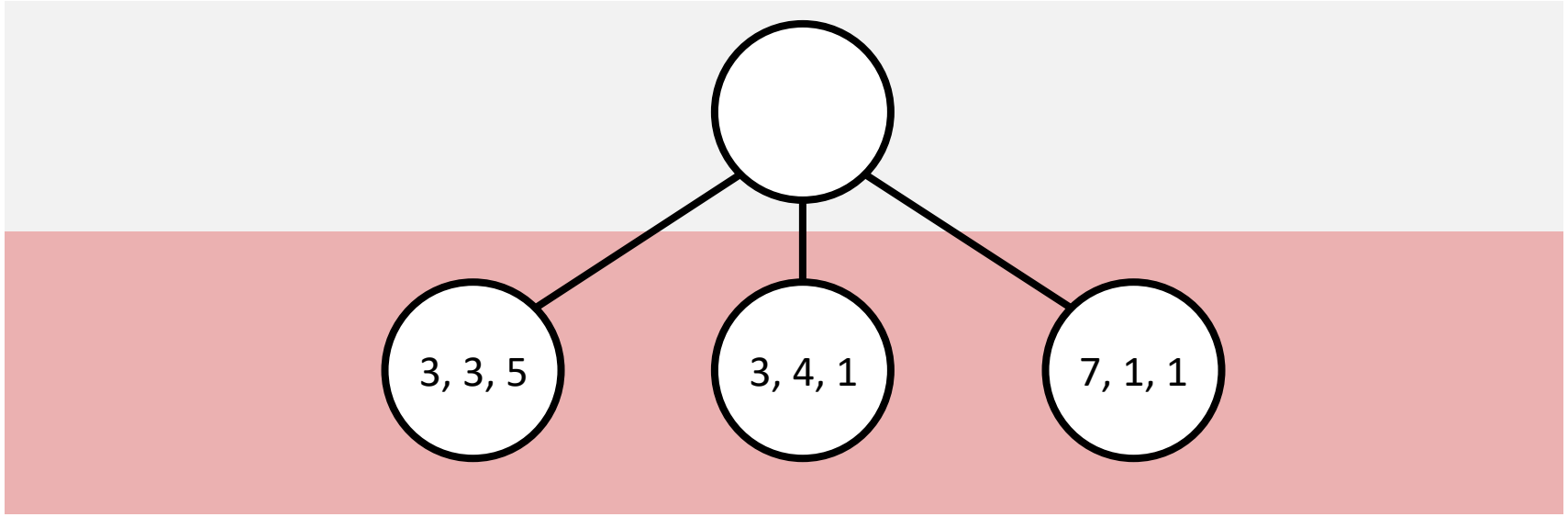
Instead of returning a single value for the playout, we return an N-tuple for N agents



Collaboration through Max-N: Action choice

Canonical Max-N:
Players are adversarial

- All players against you
  - Paranoid UCT
- Fixed coalitions
  - UCT with Alliances
- Flexible coalitions
  - Confident UCT



→

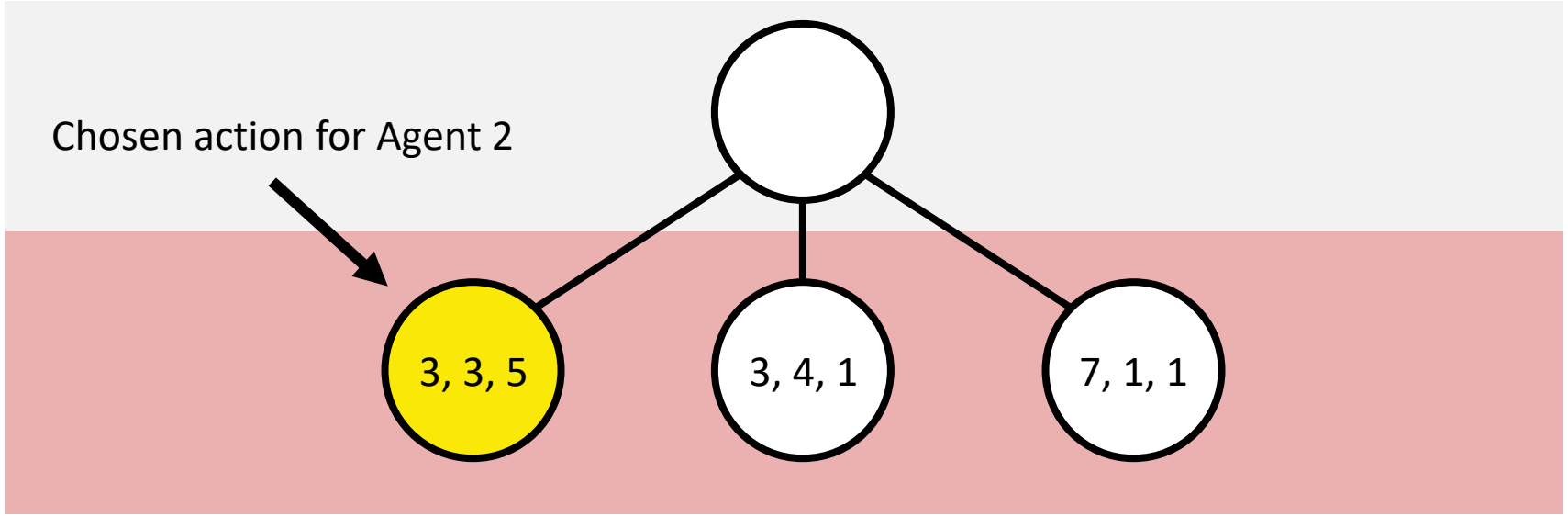
Because our agents are collaborative, we can choose the action that maximizes sum of the scores

Browne, C. *et al.* A survey of Monte-Carlo tree search methods. *IEEE Trans. Comput. Intell. AI in Games* **4**, 1–43 (2012)

Collaboration through Max-N: Action choice

Canonical Max-N:
Players are adversarial

- All players against you
  - Paranoid UCT
- Fixed coalitions
  - UCT with Alliances
- Flexible coalitions
  - Confident UCT



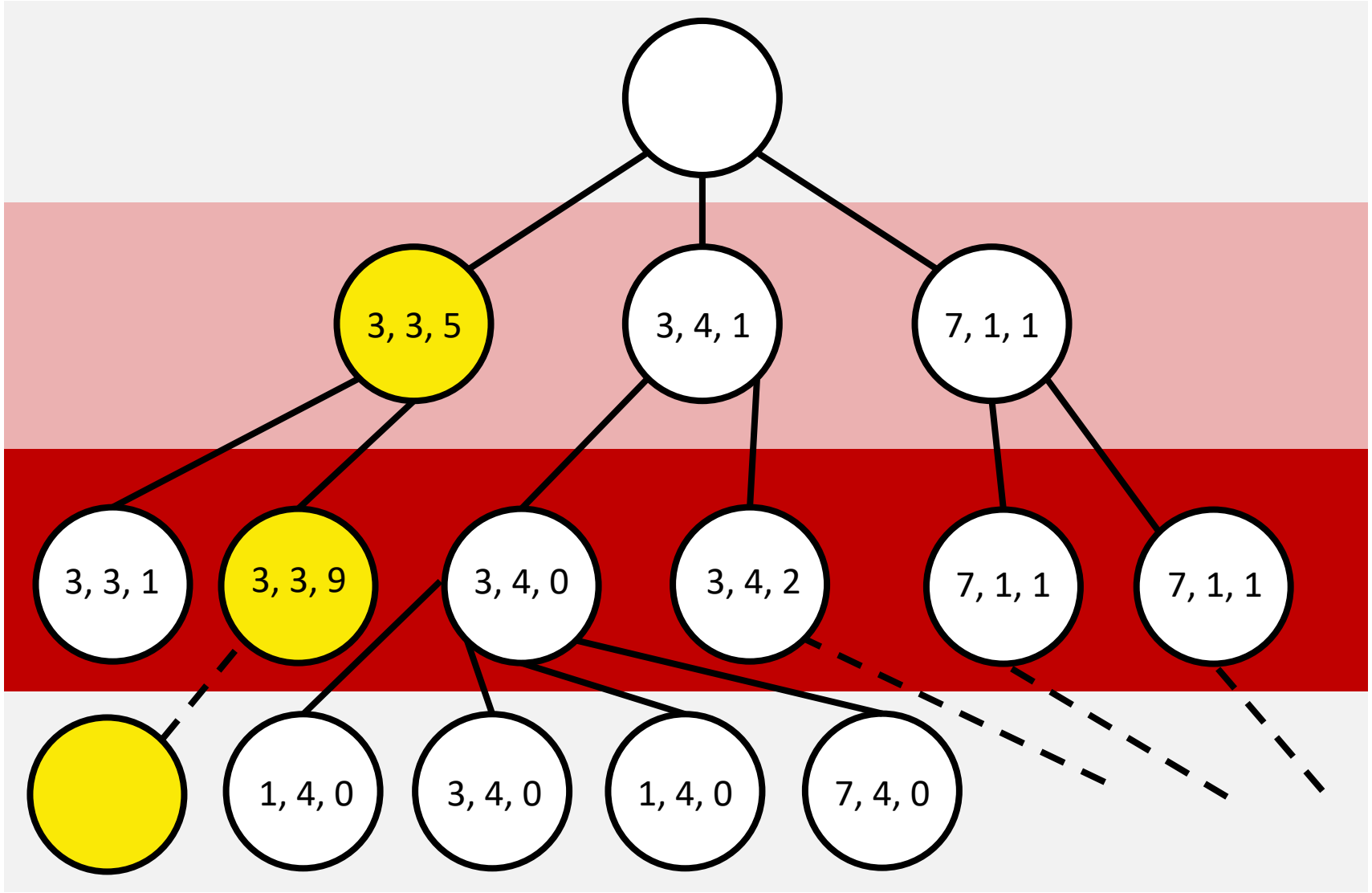
→

Because our agents are collaborative, we can choose the action that maximizes sum of the scores

Bilal Kartal, Ernesto Nunes, Julio Godoy, and Maria Gini. Monte carlo tree search for multi-robot task allocation. In Thirtieth AAAI Conference on Artificial Intelligence, pages 4222–4223, 2016.

Collaboration through Max-N: Simultaneous Execution

Simultaneous

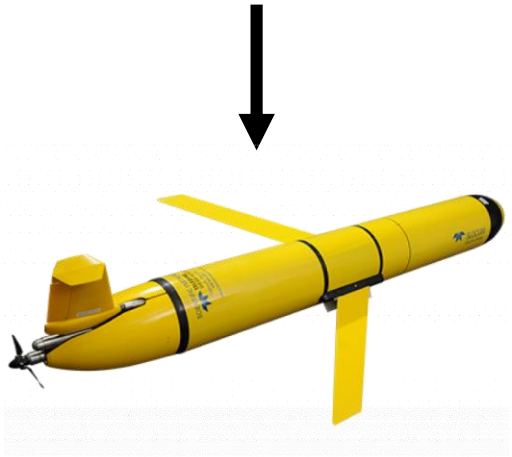


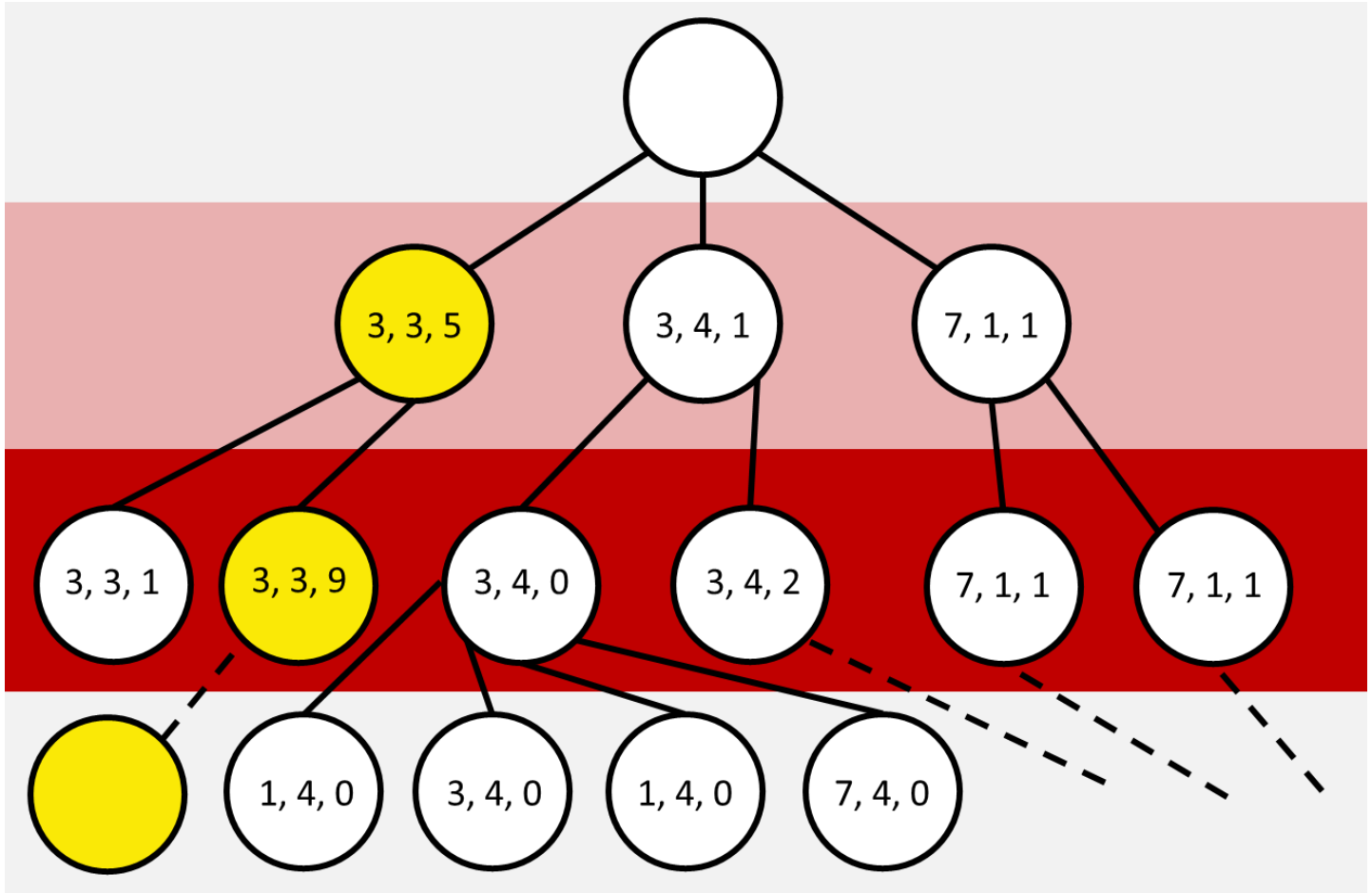
Bilal Kartal, Ernesto Nunes, Julio Godoy, and Maria Gini. Monte carlo tree search for multi-robot task allocation. In Thirtieth AAAI Conference on Artificial Intelligence, pages 4222–4223, 2016.

Collaboration through Max-N

Benefits
- Simple, modular

Disadvantages
- Requires robust communication between agents and master





Non-robust Communication → Decentralization

Central planner has authority

Desires frequent communication for state updates

Central planner

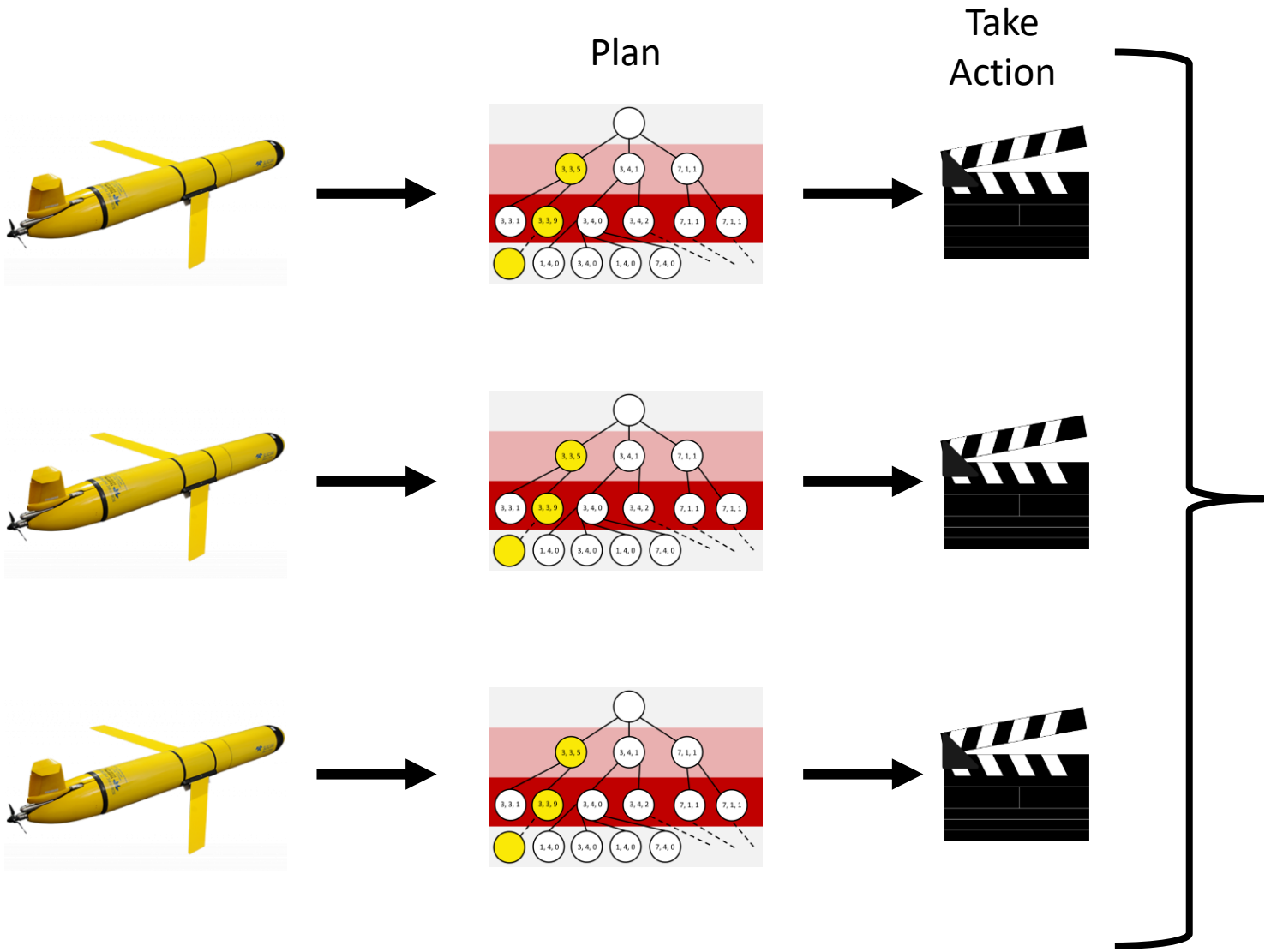
Communication may be unreliable

Hand authority over to agents

Allows individual agents to plan independently

Central planner

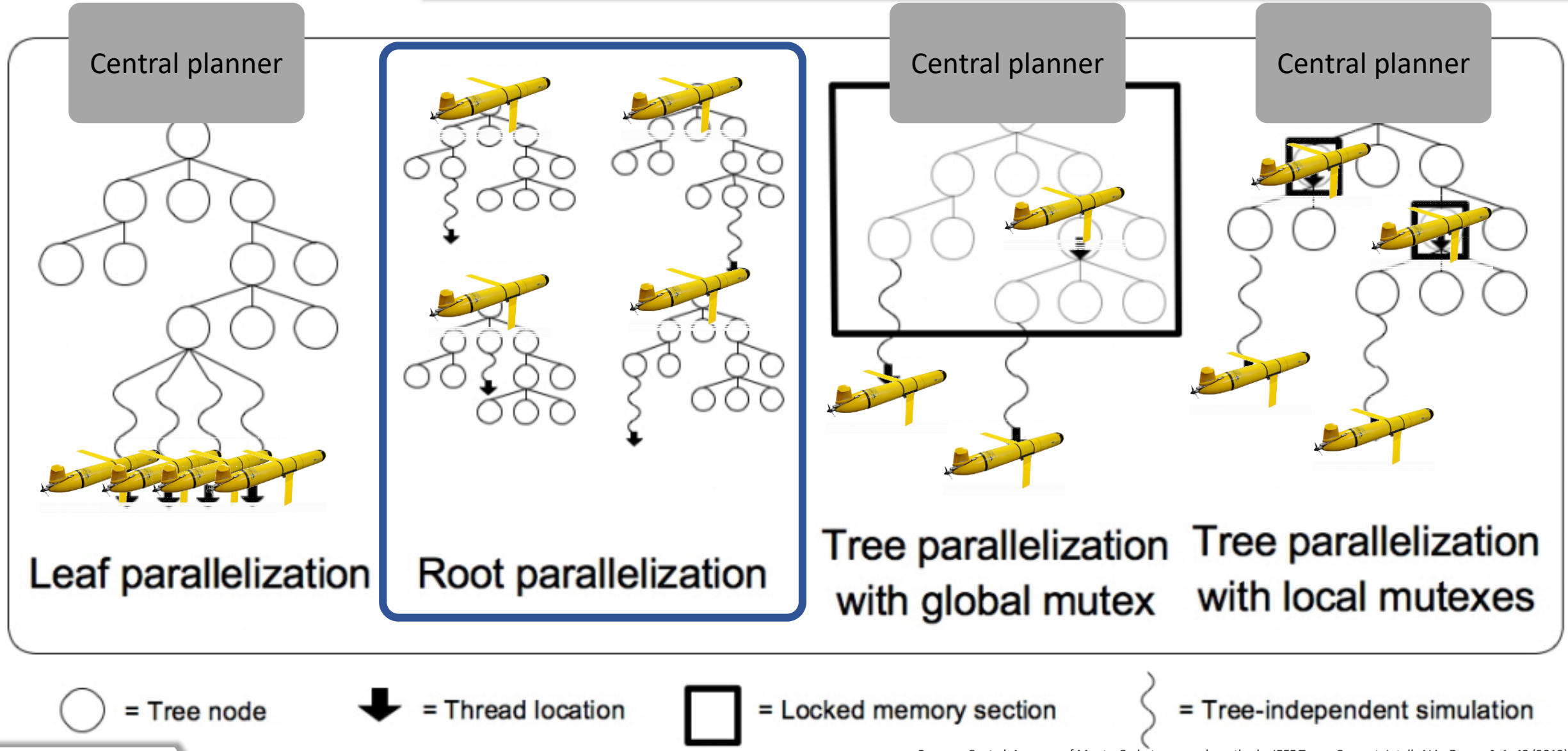
Handing over authority to the agents



- Each agent builds a multi-agent tree
- Keep tree policy as UCB
- Keep default policy as random simulation for each agent

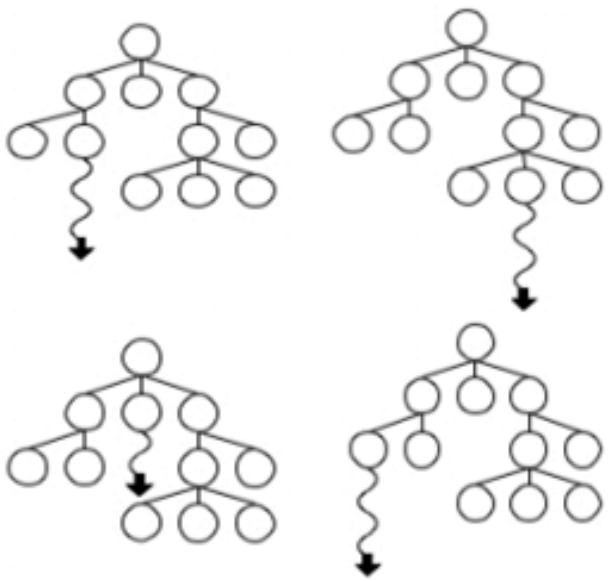
What kind of parallelization is this?

Brief aside on types of parallelization within MCTS



Browne, C. *et al.* A survey of Monte-Carlo tree search methods. *IEEE Trans. Comput. Intell. AI in Games* **4**, 1–43 (2012)

Additional step in Root Parallelization



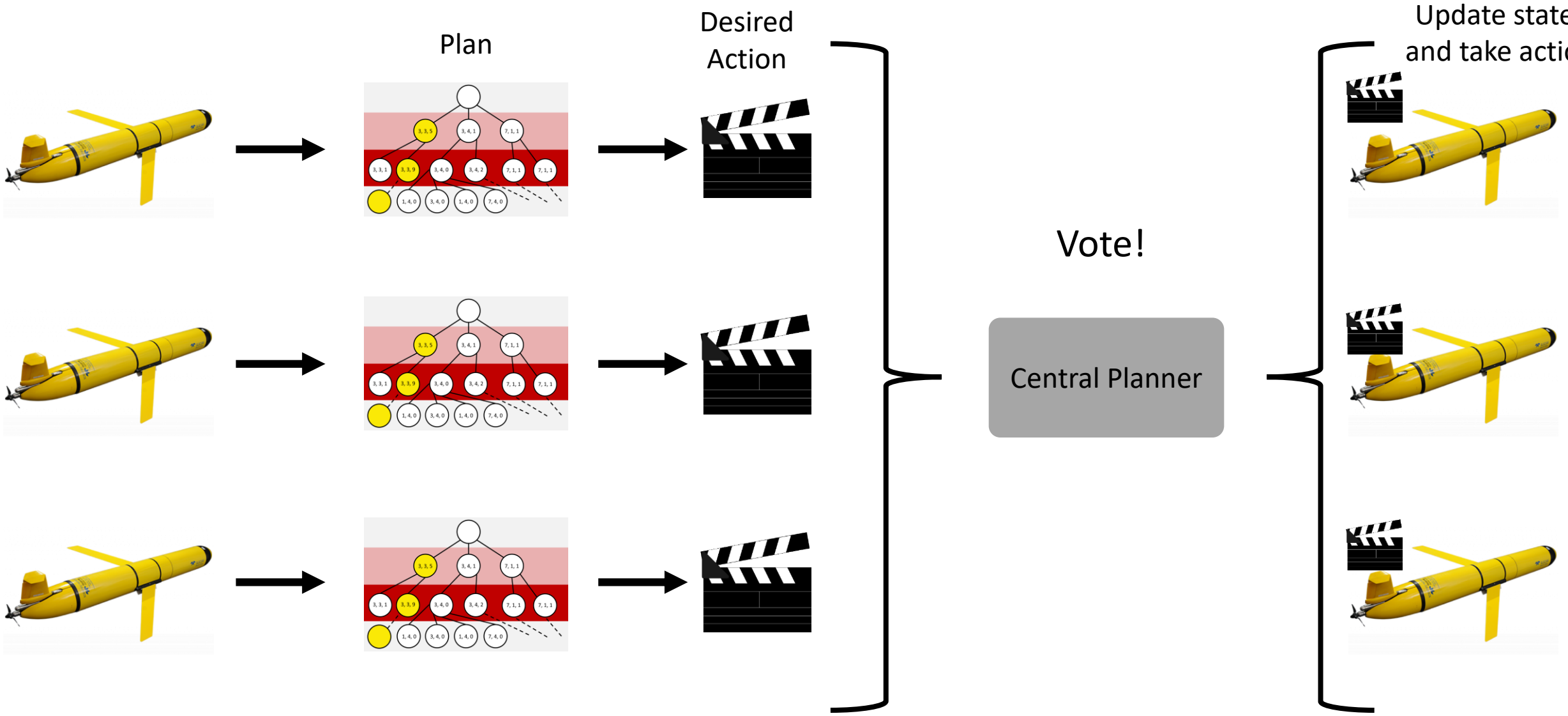
Root parallelization

- Each tree votes on an action sequence
- Action sequence with largest number of votes is taken
- Soejima et al. showed evidence that a majority voting scheme gives better performance than conventional approaches

They hypothesized that this is because root parallelization, which utilizes unique random seeds for each tree, escapes local optima better.

Browne, C. *et al.* A survey of Monte-Carlo tree search methods. *IEEE Trans. Comput. Intell. AI in Games* **4**, 1–43 (2012)

Root parallelization for our agents



Browne, C. *et al.* A survey of Monte-Carlo tree search methods. *IEEE Trans. Comput. Intell. AI in Games* **4**, 1–43 (2012)

Passing action sequences: Ms Pacman ghosts

1. We are still uses random playouts for a default policy. We can do better.

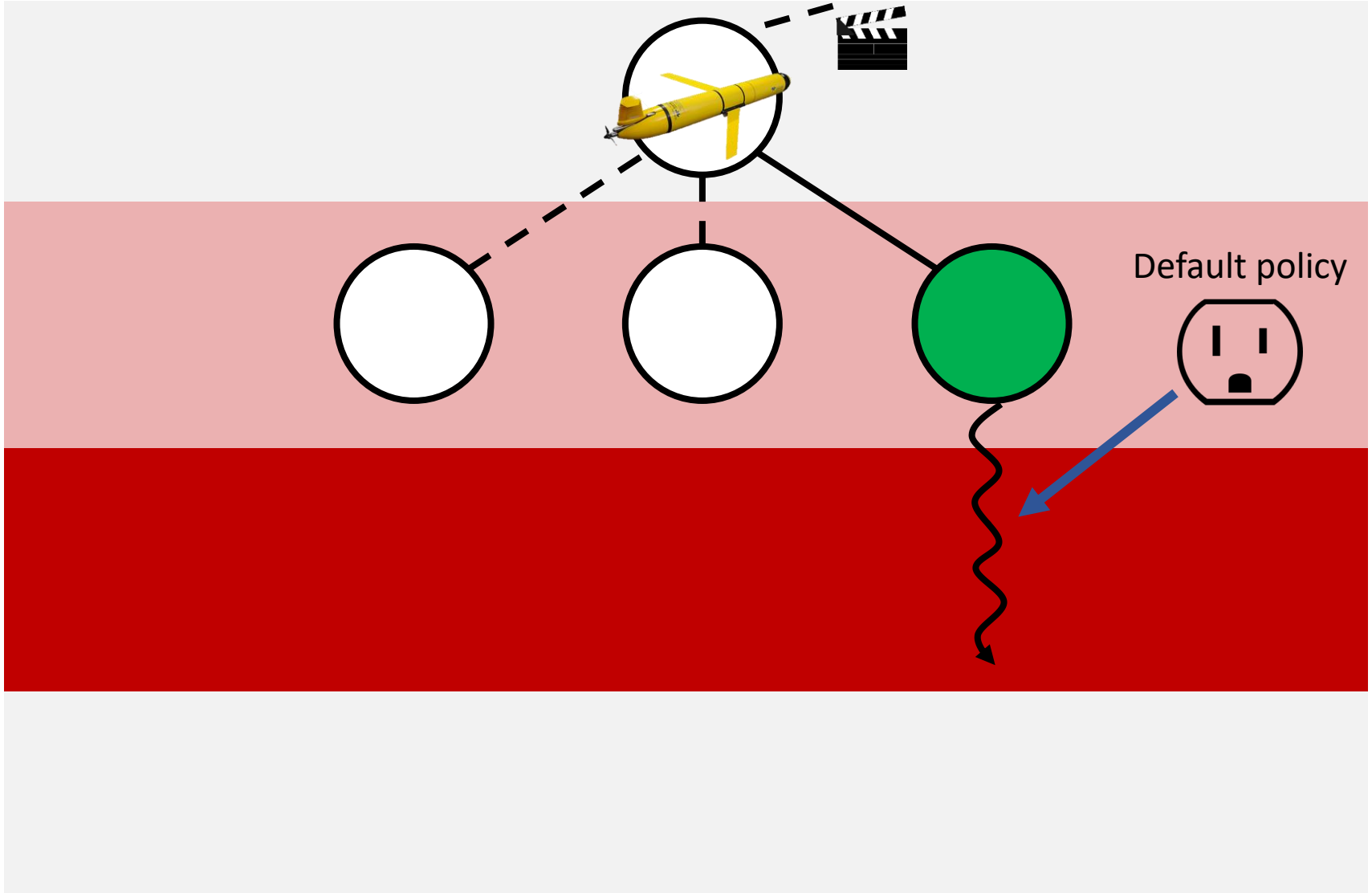
2. We have seemingly reintroduced the need for robust communication between the agents.

To improve the default policy, we can pass action sequences from the other agents to each agent

Nguyen, Kien & Thawonmas, Ruck. (2013). Monte Carlo Tree Search for Collaboration Control of Ghosts in Ms. Pac-Man. Computational Intelligence and AI in Games, IEEE Transactions on. 5. 57-68. 10.1109/TCIAIG.2012.2214776.

Passing action sequences detailed look

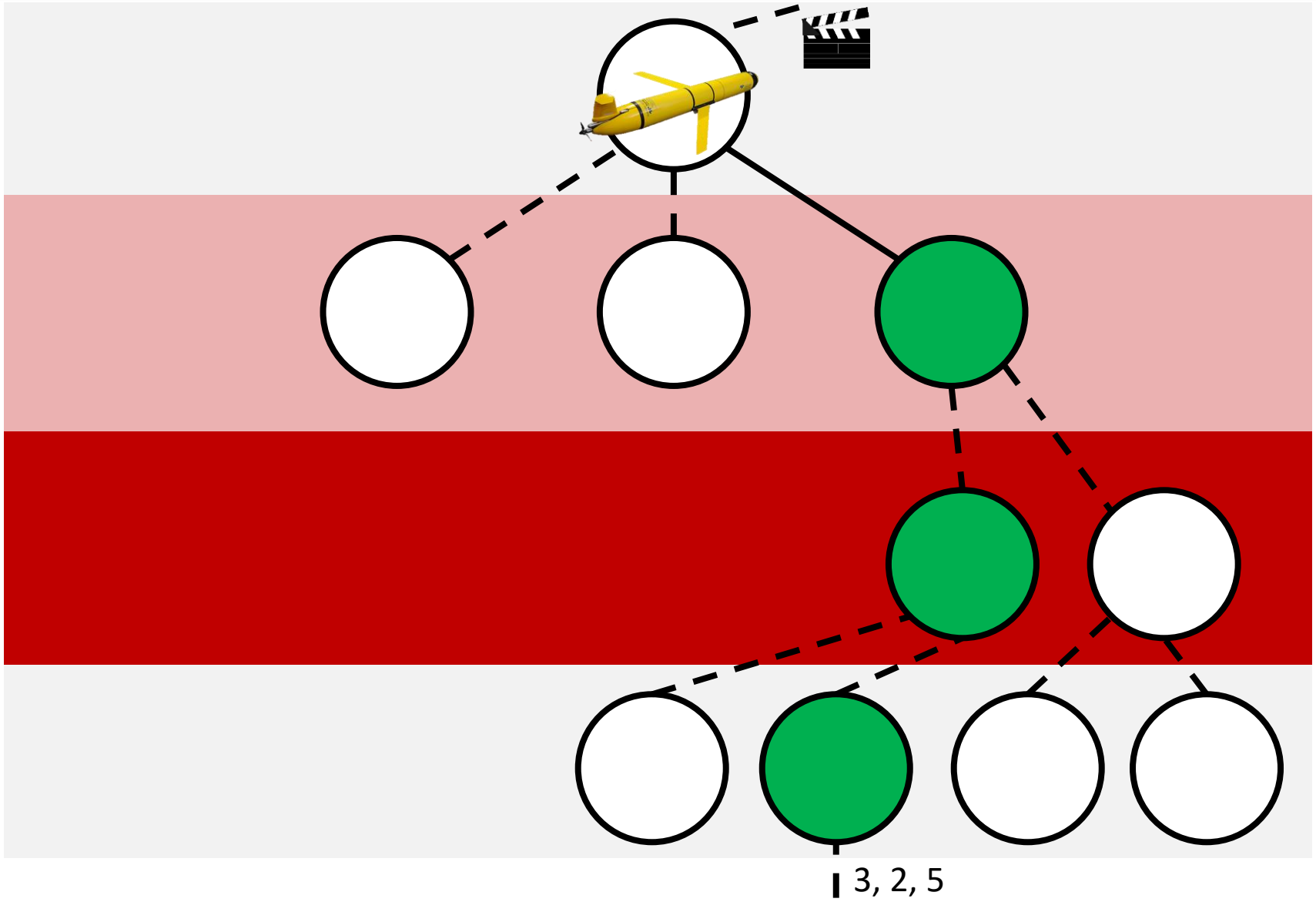
- Our original default policy was to simulate with random moves



Nguyen, Kien & Thawonmas, Ruck. (2013). Monte Carlo Tree Search for Collaboration Control of Ghosts in Ms. Pac-Man. Computational Intelligence and AI in Games, IEEE Transactions on. 5. 57-68. 10.1109/TCIAIG.2012.2214776.

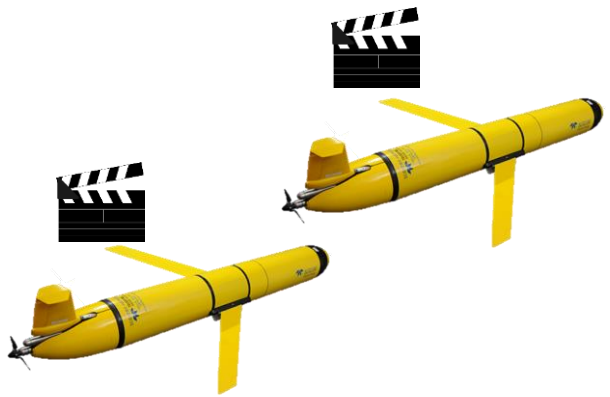
Passing action sequences: Ms Pacman ghosts

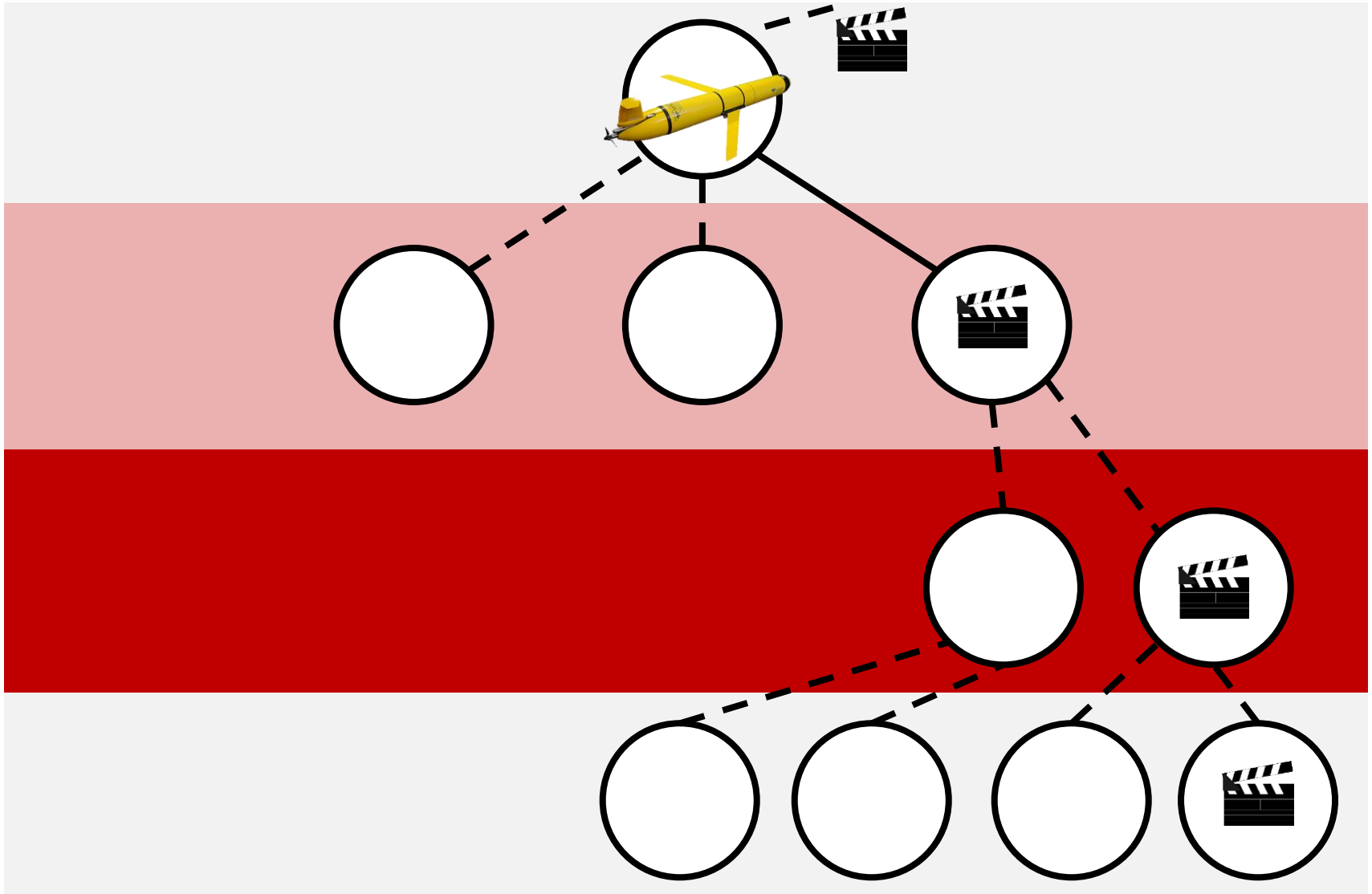
- Our original default policy was to simulate with random moves



Passing action sequences: Ms Pacman ghosts

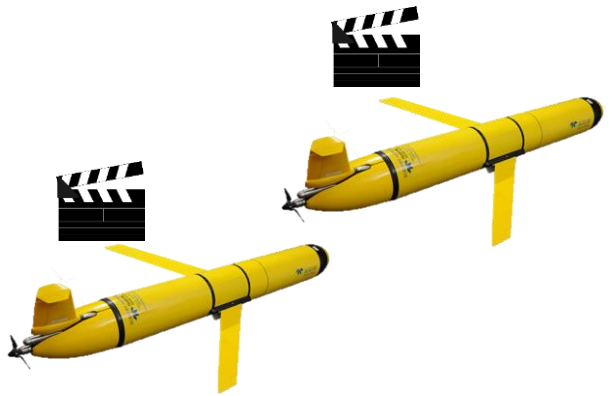
- Our original default policy was to simulate with random moves
- Instead, if our agent had a good guess on what all the other agents were going to do, we could improve our simulation

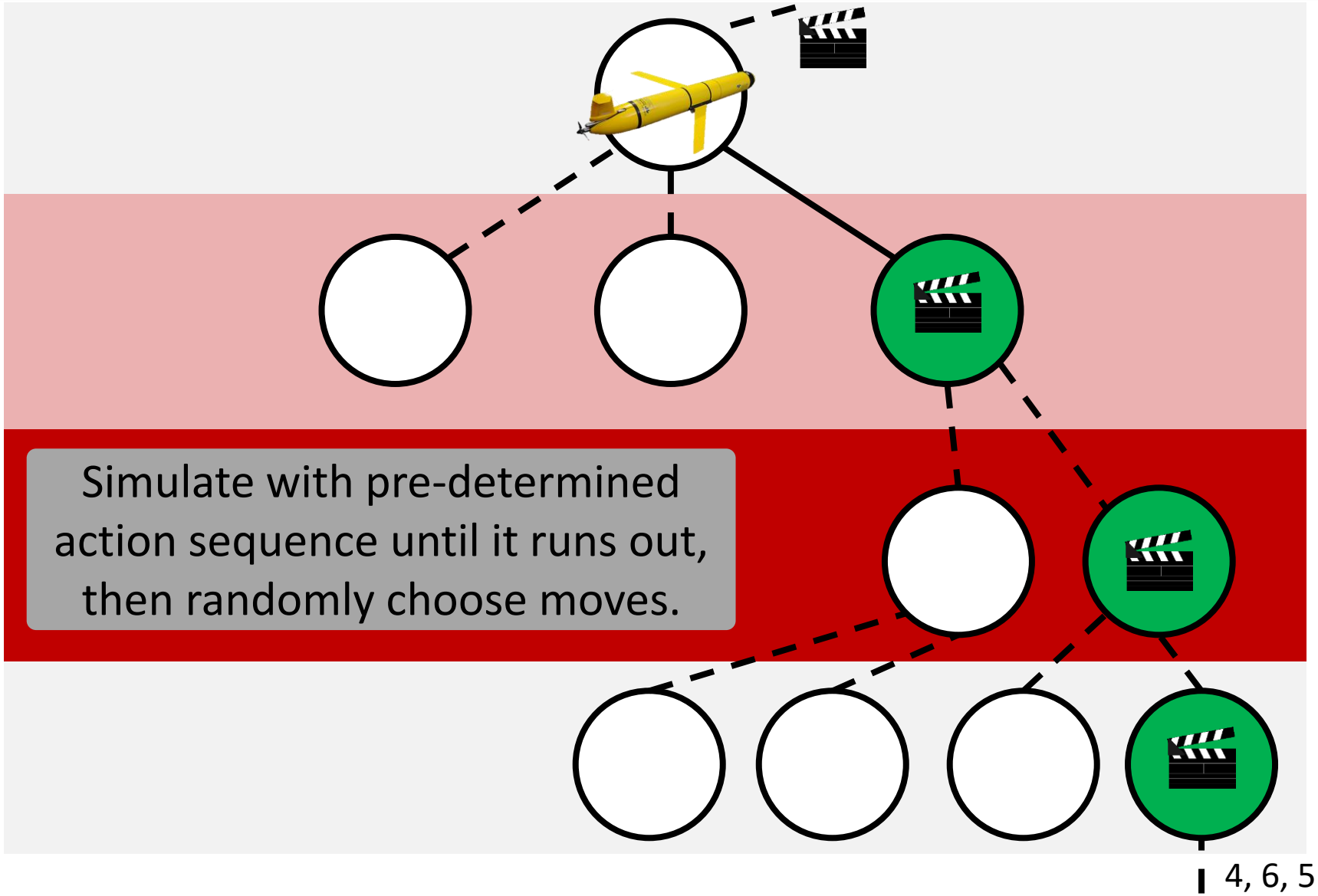




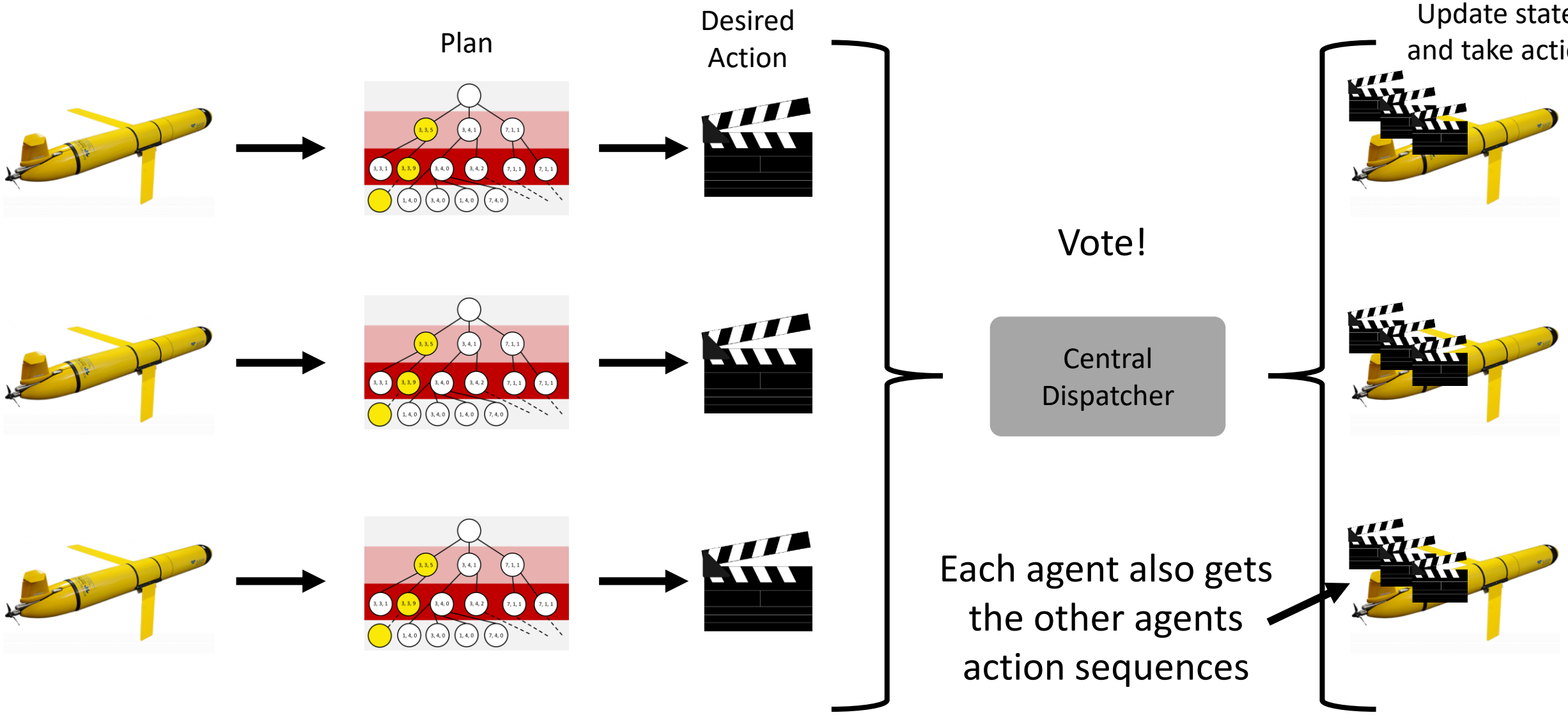
Passing action sequences: Ms Pacman ghosts

- Our original default policy was to simulate with random moves
- Instead, if our agent had a good guess on what all the other agents were going to do, we could improve our simulation



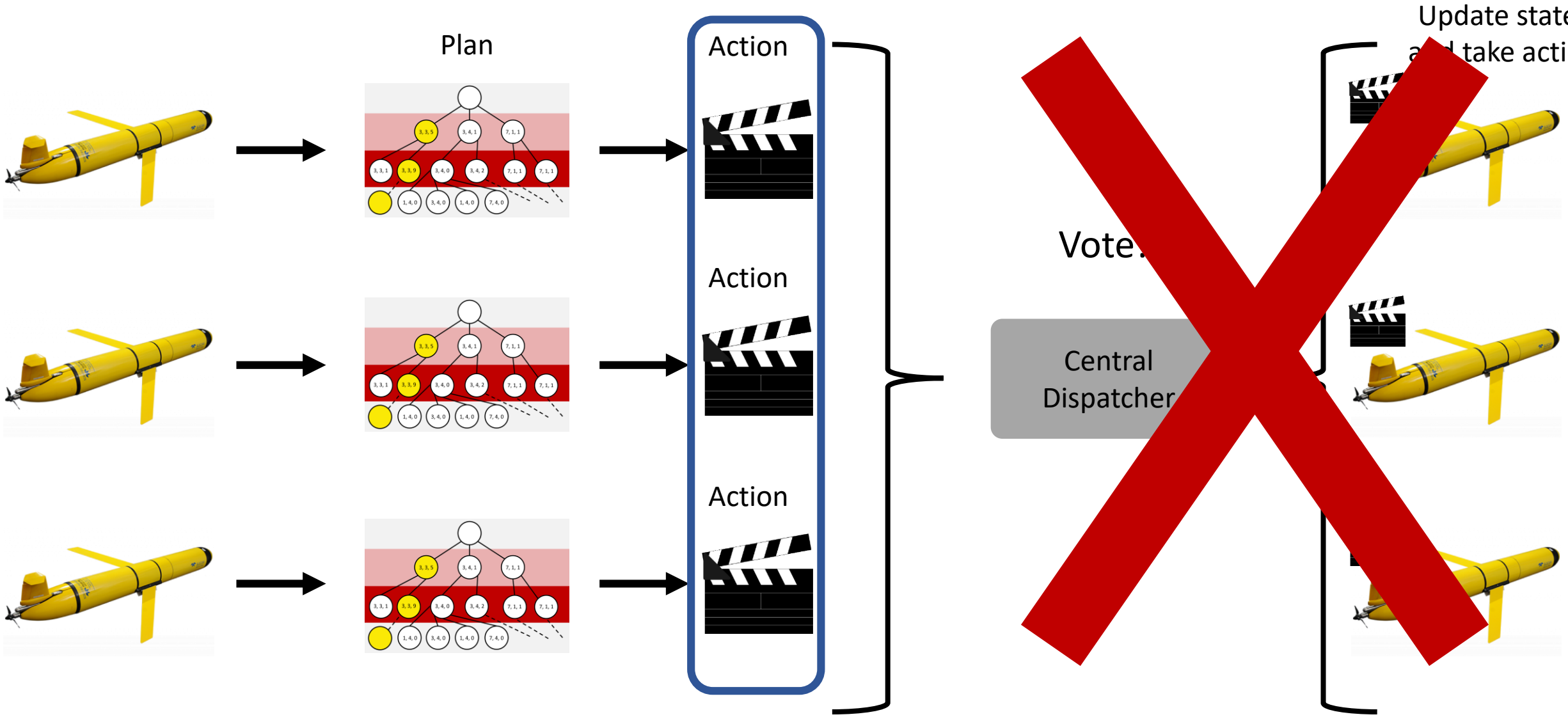


Root parallelization for our agents



Browne, C. *et al.* A survey of Monte-Carlo tree search methods. *IEEE Trans. Comput. Intell. AI in Games* **4**, 1–43 (2012)

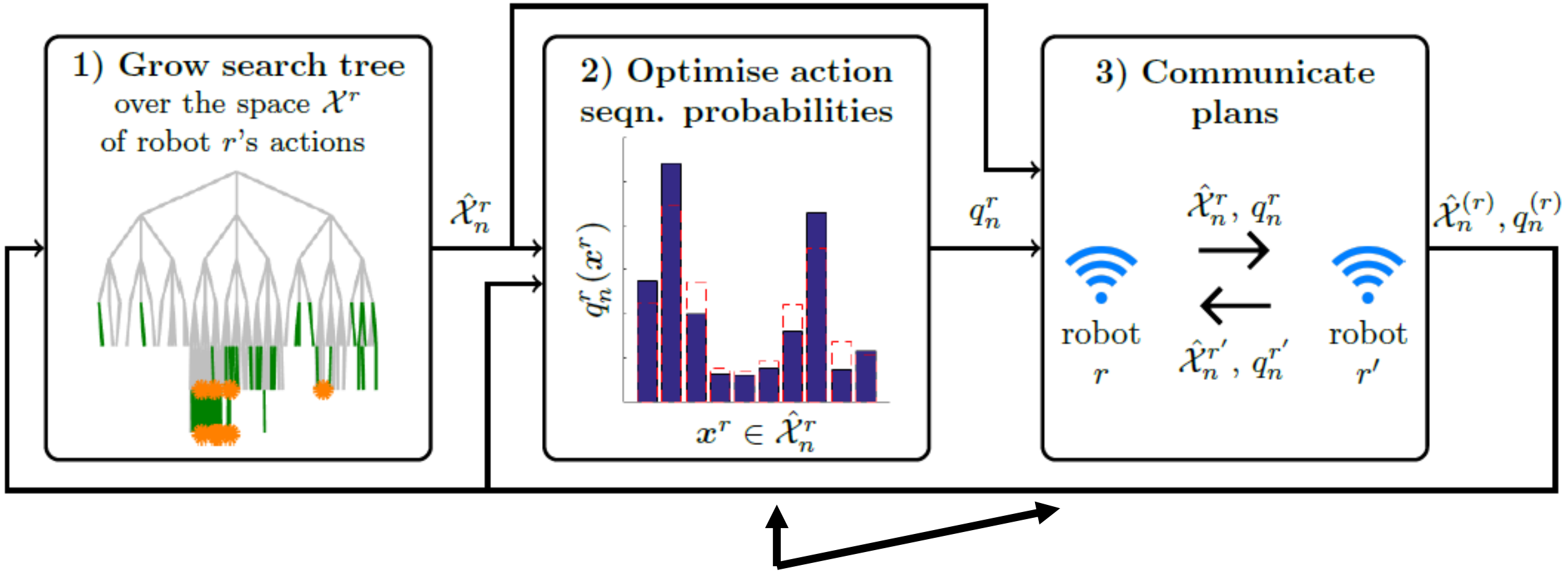
What if/when communication fails?



Browne, C. *et al.* A survey of Monte-Carlo tree search methods. *IEEE Trans. Comput. Intell. AI in Games* **4**, 1–43 (2012)

What happens if/when communication fails?

Best et al. (2018) investigated this with a spoofed up version of MCTS

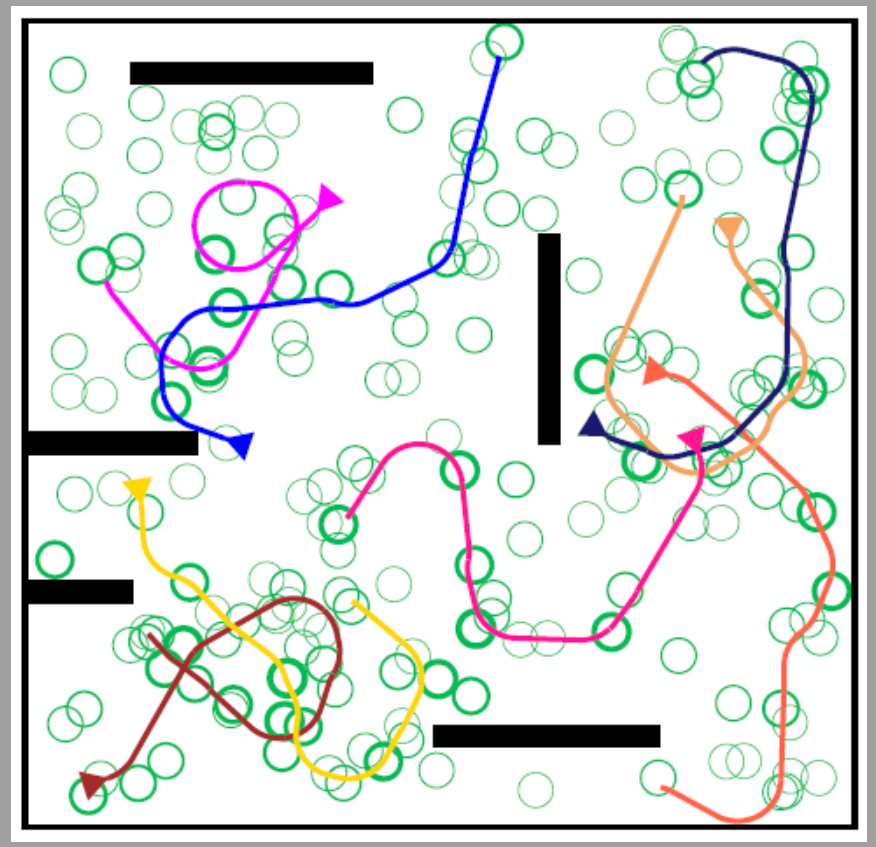


Developed and passed probabilistic action sequences between robots instead of deterministic

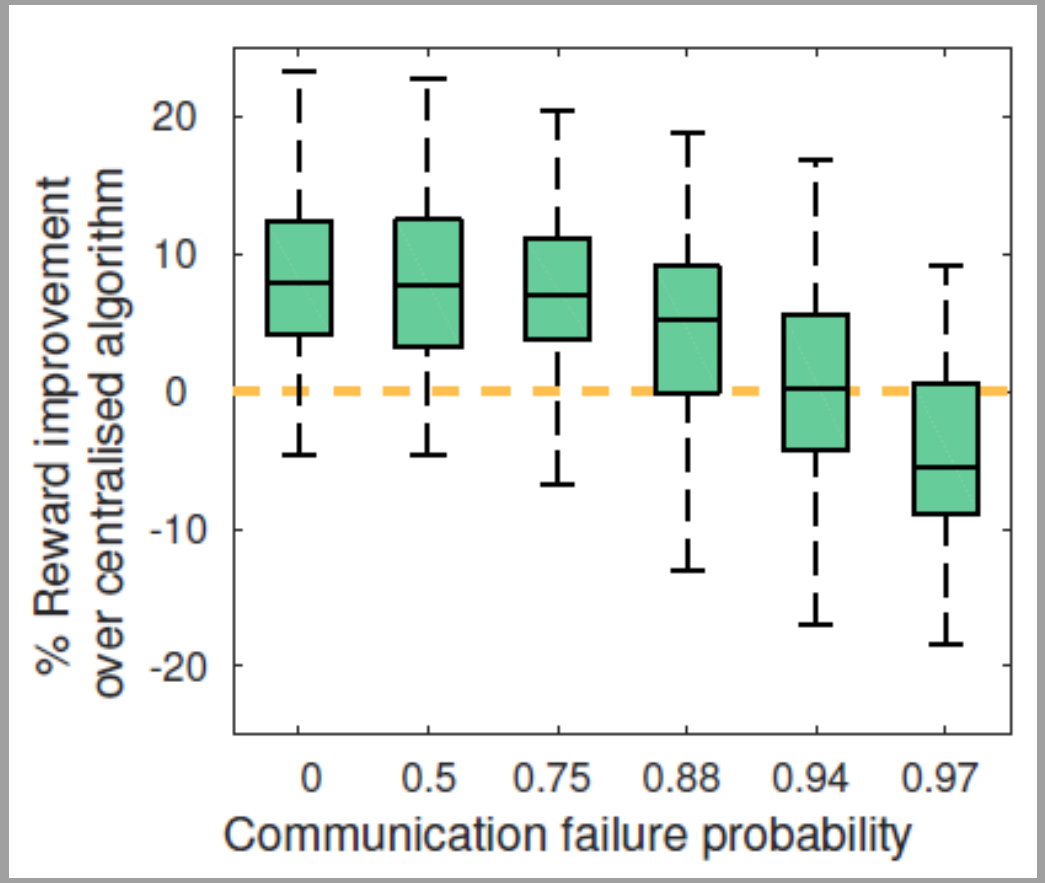
Best, Graeme & Cliff, Oliver & Patten, Timothy & R Mettu, Ramgopal & Fitch, Robert. (2018). Dec-MCTS: Decentralized planning for multi-robot active perception. The International Journal of Robotics Research. 027836491875592. 10.1177/0278364918755924.

Decentralized MCTS did well despite bad communication

Problem environment

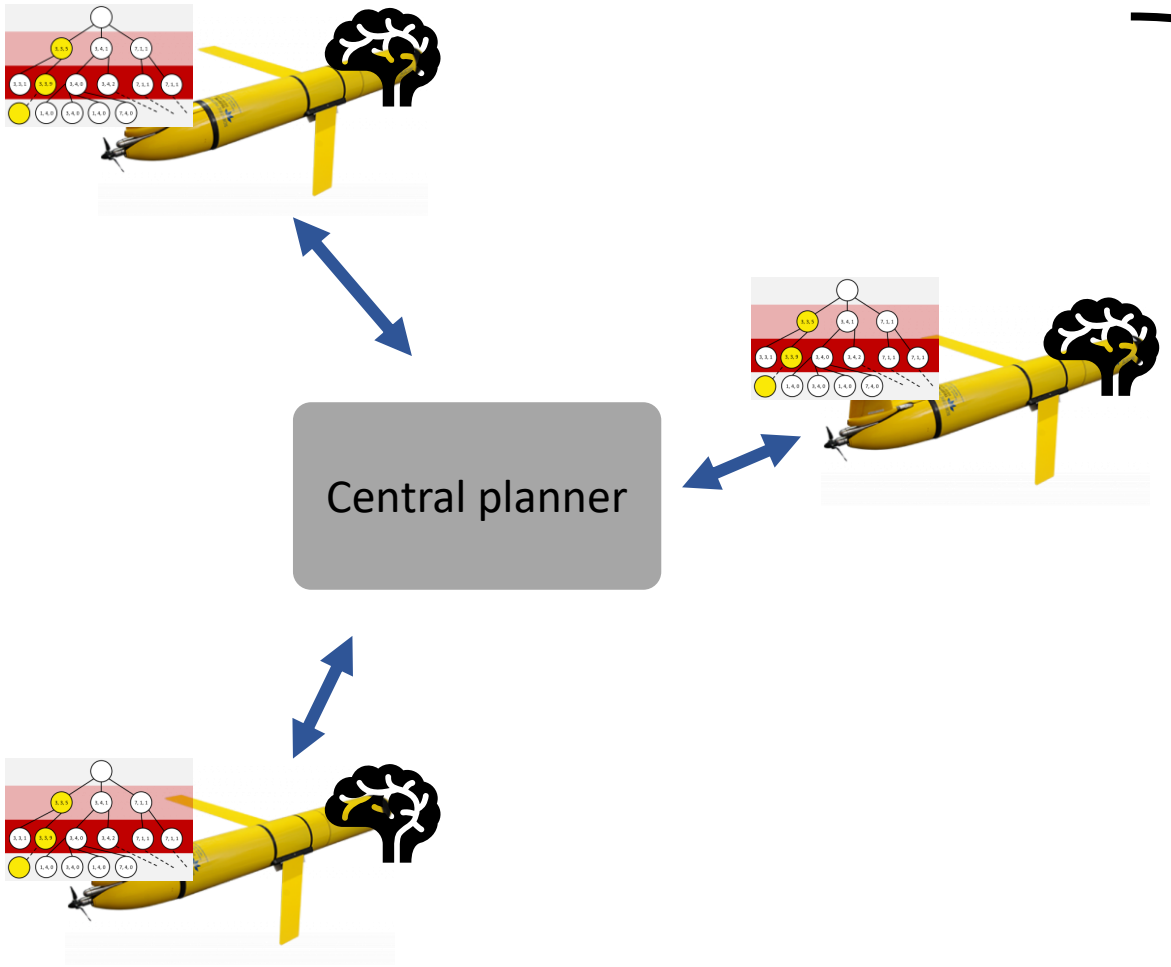


Communication Failure vs Reward Improvement



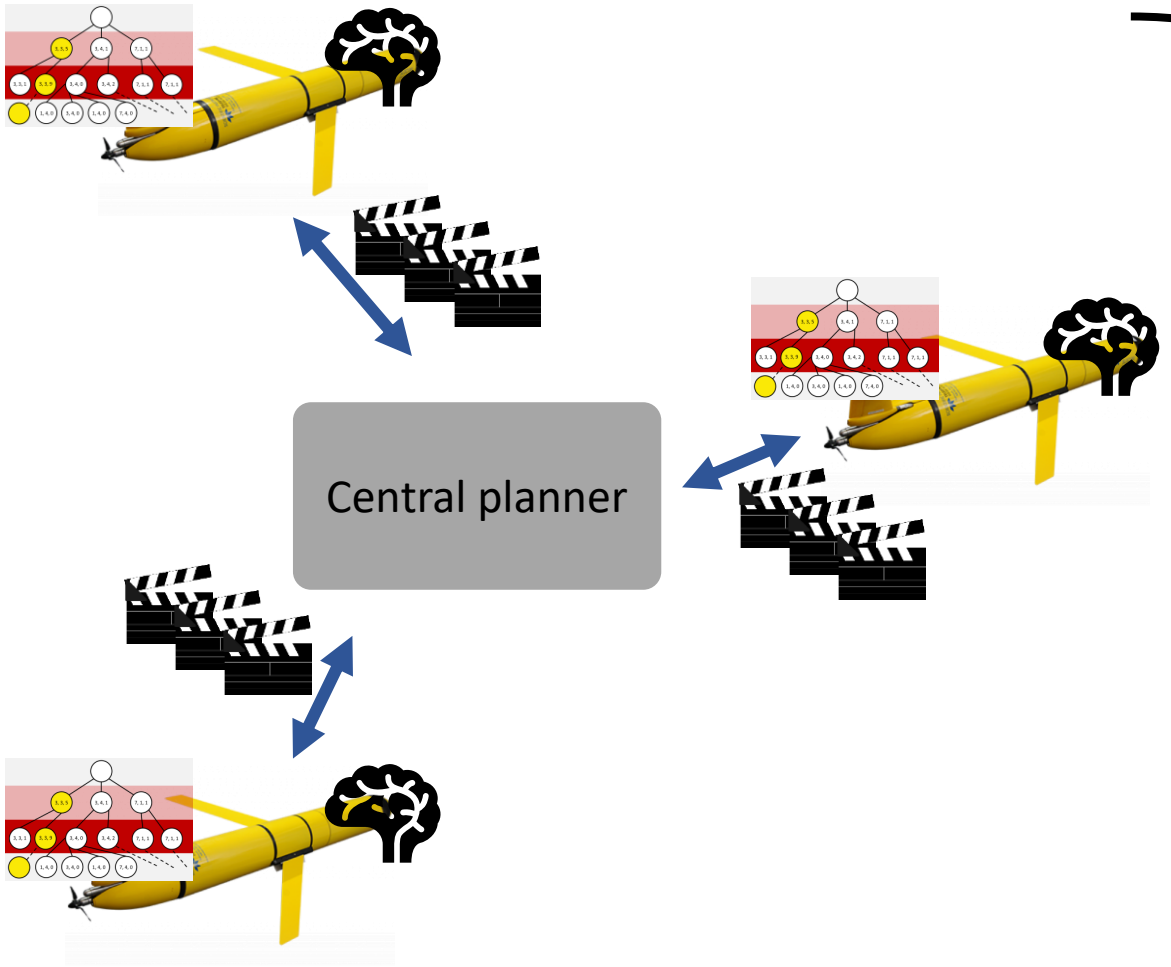
Best, Graeme & Cliff, Oliver & Patten, Timothy & R Mettu, Ramgopal & Fitch, Robert. (2018). Dec-MCTS: Decentralized planning for multi-robot active perception. The International Journal of Robotics Research. 027836491875592. 10.1177/0278364918755924.

MCTS for Collaboration



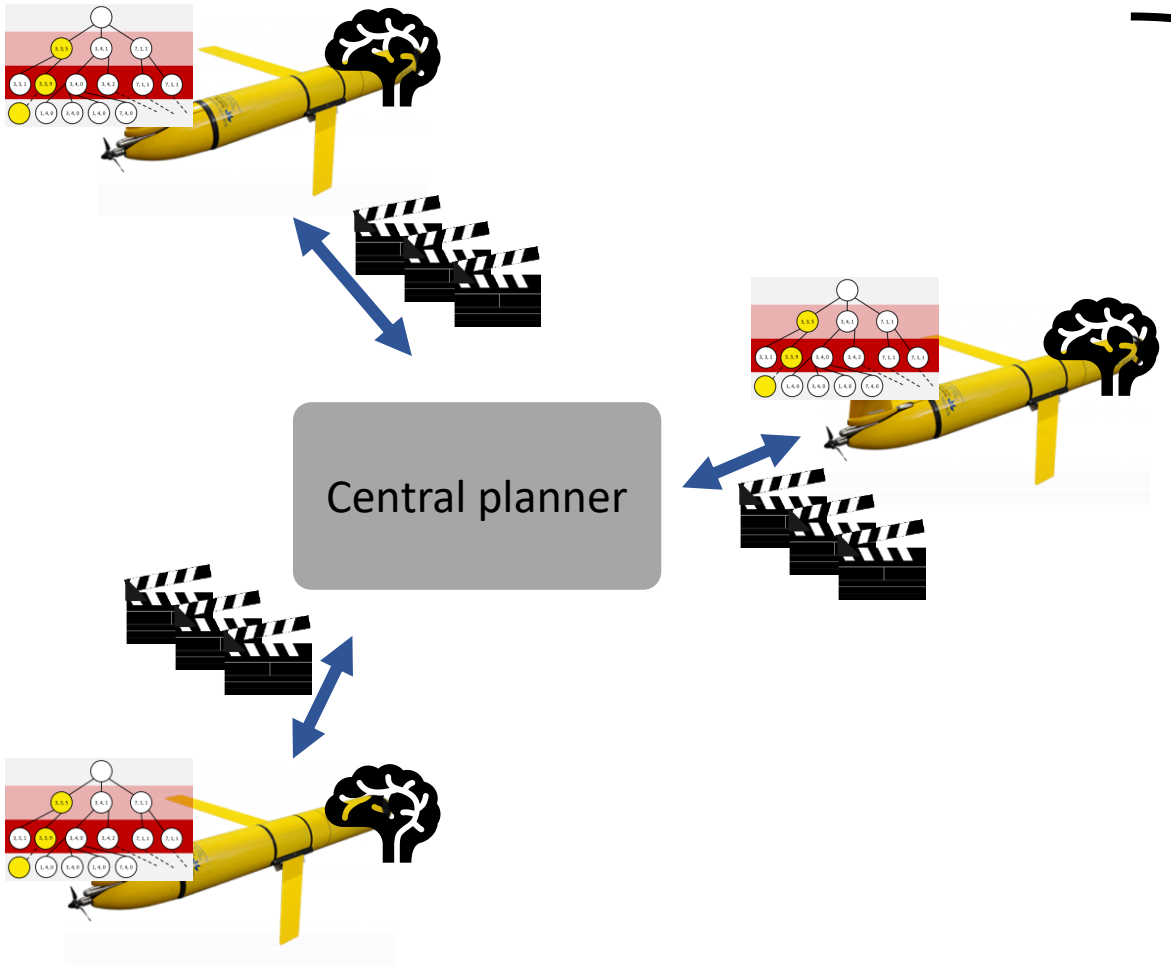
- Each agent runs own MCTS, incorporating heuristics and other performance boosting modifications

MCTS for Collaboration



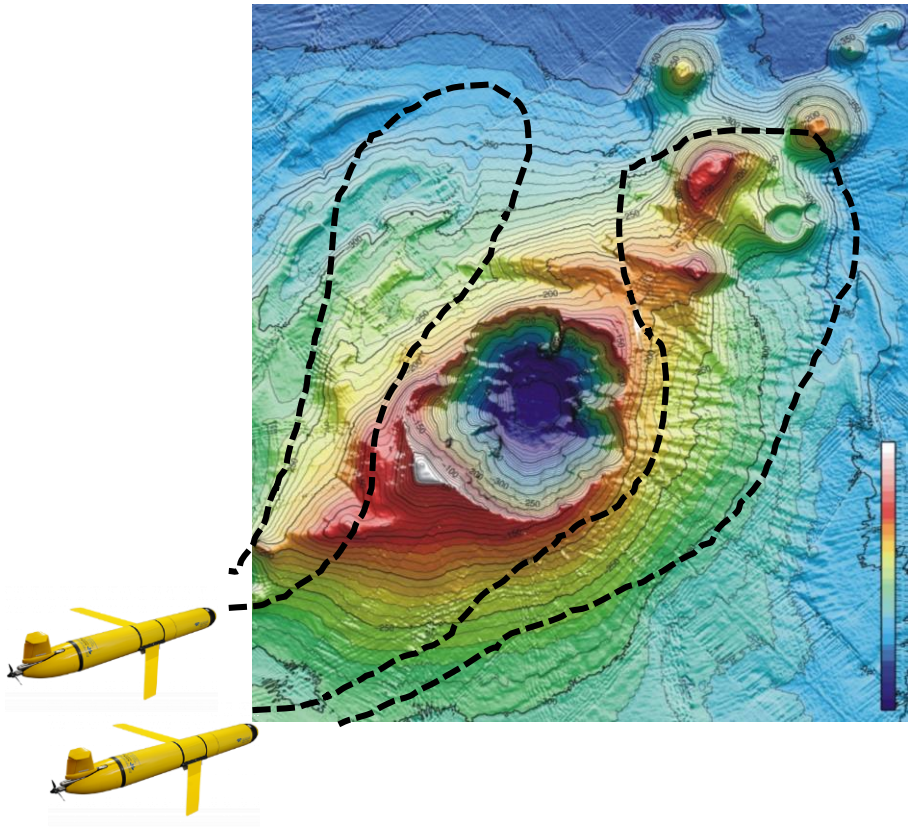
- Each agent runs own MCTS, incorporating heuristics and other performance boosting modifications
- When communication allows, action sequences are passed between the gliders and the central planner

MCTS for Collaboration



- Each agent runs own MCTS, incorporating heuristics and other performance boosting modifications
- When communication allows, action sequences are passed between the gliders and the central planner
- These action sequences are then incorporated into each local agents MCTS

MCTS for Collaboration



Questions?