

Foundations of Artificial Intelligence

44. Board Games: Monte-Carlo Tree Search Framework

Thomas Keller and Florian Pommerening

University of Basel

May 24, 2023

Board Games: Overview

chapter overview:

- 40. Introduction and State of the Art
- 41. Minimax Search and Evaluation Functions
- 42. Alpha-Beta Search
- 43. Stochastic Games
- 44. Monte-Carlo Tree Search Framework
- 45. Monte-Carlo Tree Search Configurations

Introduction

Monte-Carlo Tree Search

algorithms considered previously:

13	2	3	12
9	11	1	10
	6	4	14
15	8	7	5

systematic search:

- systematic exploration of search space
- computation of (state) quality follows performance metric



Monte-Carlo Tree Search

algorithms considered previously:

13	2	3	12
9	11	1	10
	6	4	14
15	8	7	5

systematic search:

- systematic exploration of search space
- computation of (state) quality follows performance metric



algorithms considered today:



search based on Monte-Carlo methods:

- sampling of game simulations
- estimation of (state) quality by averaging over simulation results



Game Applications

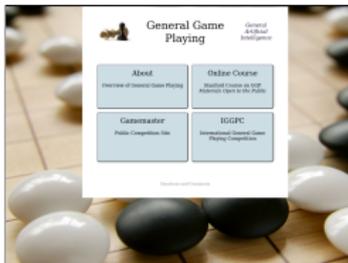
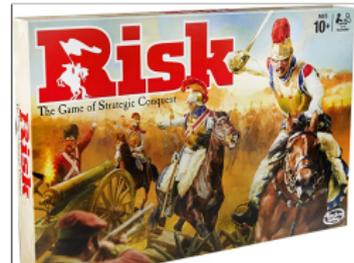
board games



hidden information games



stochastic games



general game playing



real-time strategy games



dynamic difficulty adjustment

Applications Beyond Games

story generation



chemical synthesis



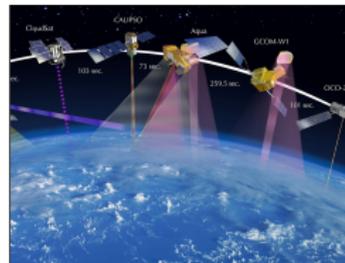
UAV routing



coast security



forest harvesting



Earth observation

MCTS Environments

MCTS environments cover **entire spectrum of properties**
↔ need some restrictions

MCTS Environments

MCTS environments cover [entire spectrum of properties](#)

↔ need some restrictions

we study MCTS under the [same restrictions as last week](#), i.e.,

- environment classification,
- problem solving method,
- objective of the agent and
- performance measure

are identical to last week

MCTS Environments

MCTS environments cover **entire spectrum of properties**

↪ need some restrictions

we study MCTS under the **same restrictions as last week**, i.e.,

- environment classification,
- problem solving method,
- objective of the agent and
- performance measure

are identical to last week

MCTS extensions exist that allow to **drop most restrictions**

Monte-Carlo Tree Search

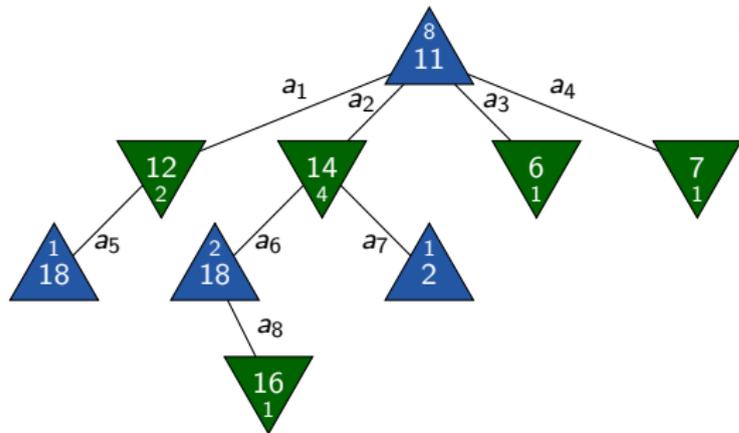
Data Structures

Monte-Carlo tree search

- is a **tree search** variant
 - ↪ **no closed list**
- iteratively performs **game simulations** from the initial position (called **trial** or **rollout**)
 - ↪ **no (explicit) open list**

↪ **search nodes** are the only used data structure

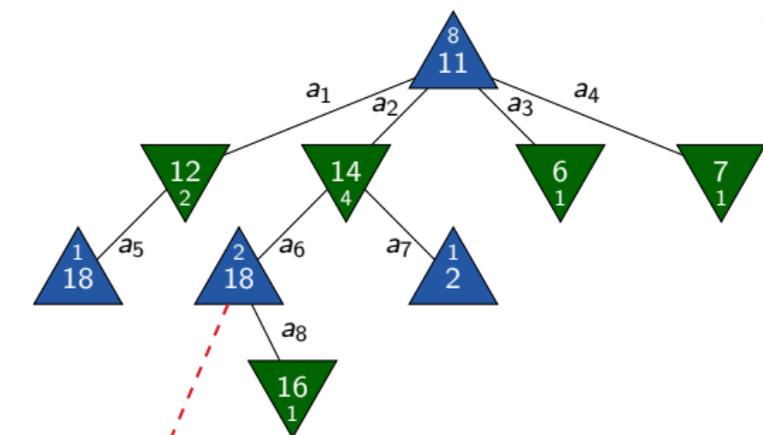
Data Structure: Search Nodes



search nodes store

- a reached **position**
- **how** it was reached
- its **successors**
- a **utility estimate** (\hat{v})
- a **visit counter** (N)
- possibly additional information

Data Structure: Search Nodes



search nodes store

- a reached **position**
- **how** it was reached
- its **successors**
- a **utility estimate** (\hat{v})
- a **visit counter** (N)
- possibly additional information

<i>position:</i>	not displayed
<i>move:</i>	a_6
<i>successors:</i>	[none , ]
\hat{v} :	18
N :	2
....:	...

Monte-Carlo Tree Search: Idea

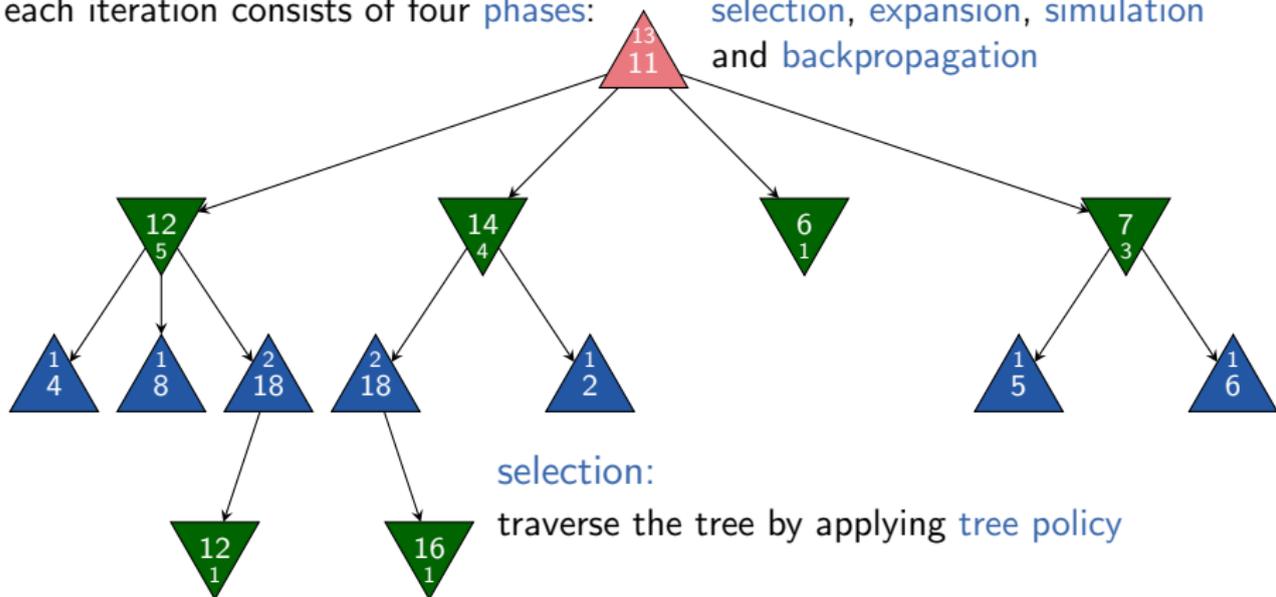
Monte-Carlo Tree Search (MCTS) ideas:

- build a partial game tree
- by performing trials as long as resources (deliberation time, memory) allow
- initially, the tree contains only the root node
- each trial adds (at most) one search node to the tree

after termination, play the associated move of a successor of the root node with highest utility estimate

Idea and Example

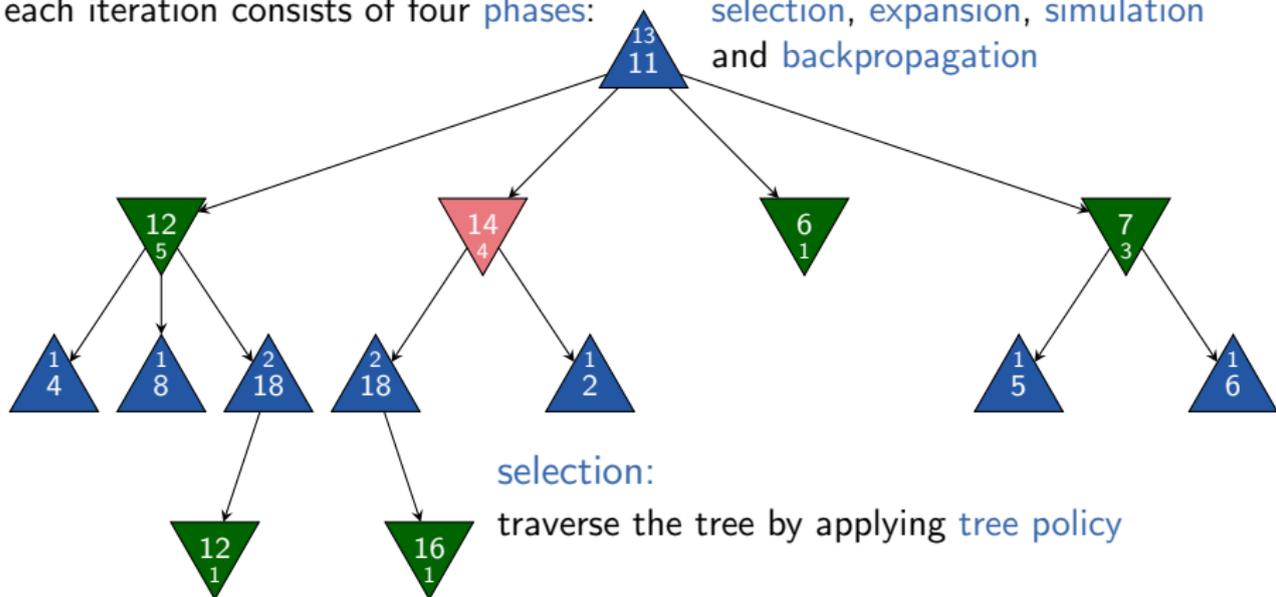
each iteration consists of four phases: selection, expansion, simulation and backpropagation



Idea and Example

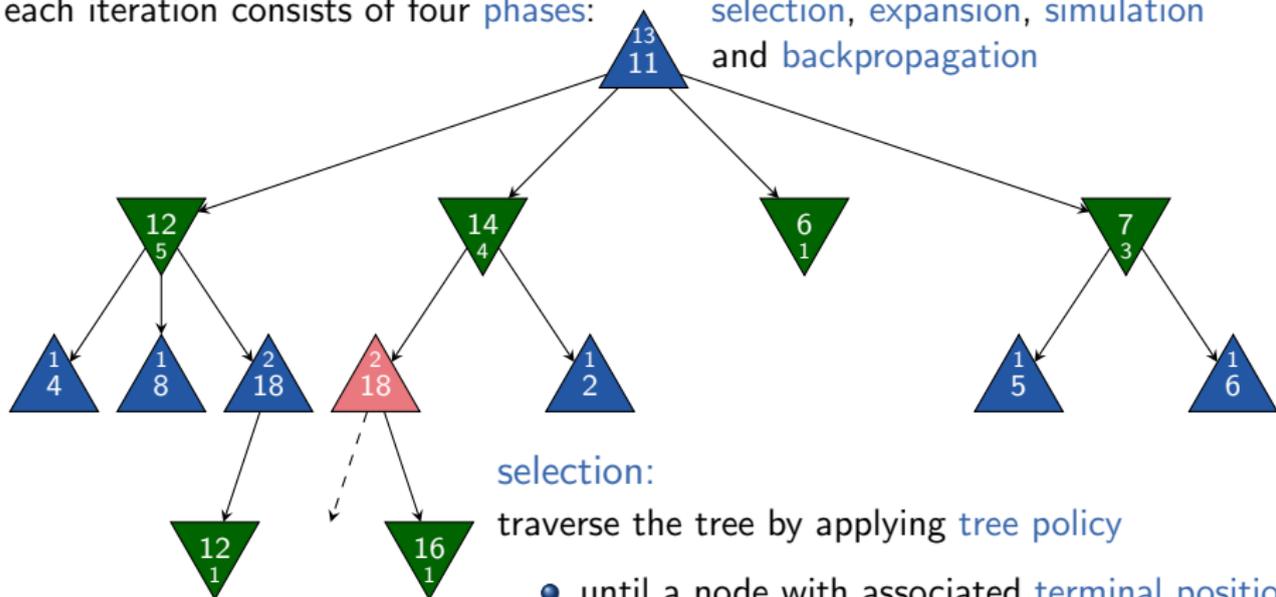
each iteration consists of four phases:

selection, expansion, simulation
and backpropagation



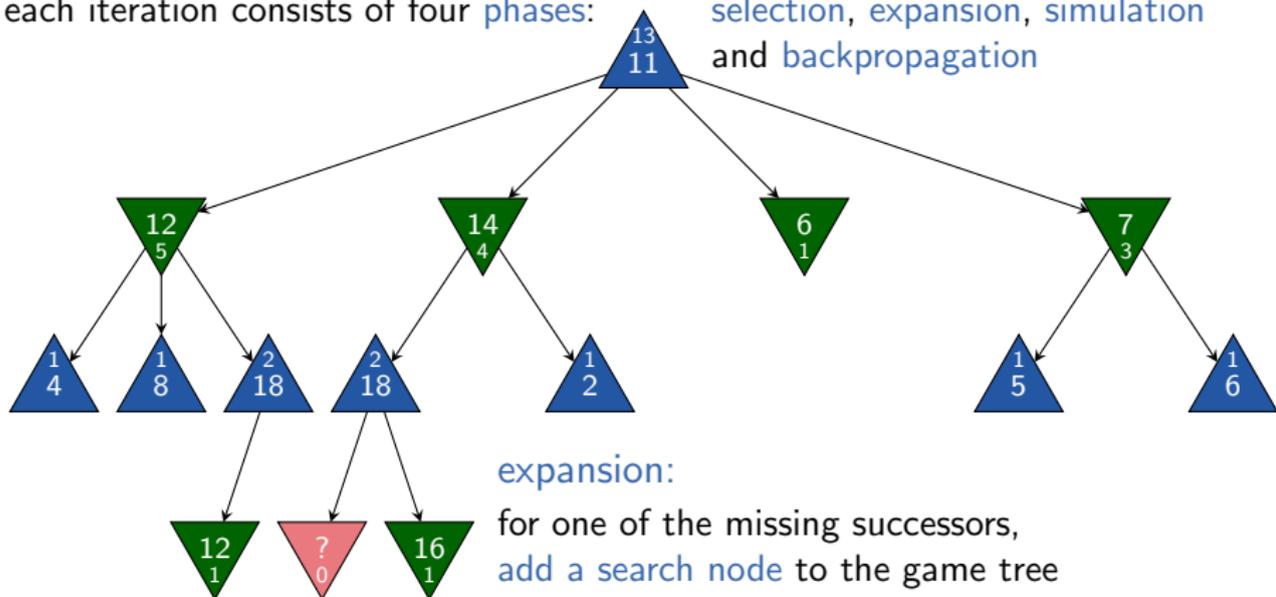
Idea and Example

each iteration consists of four phases: selection, expansion, simulation and backpropagation



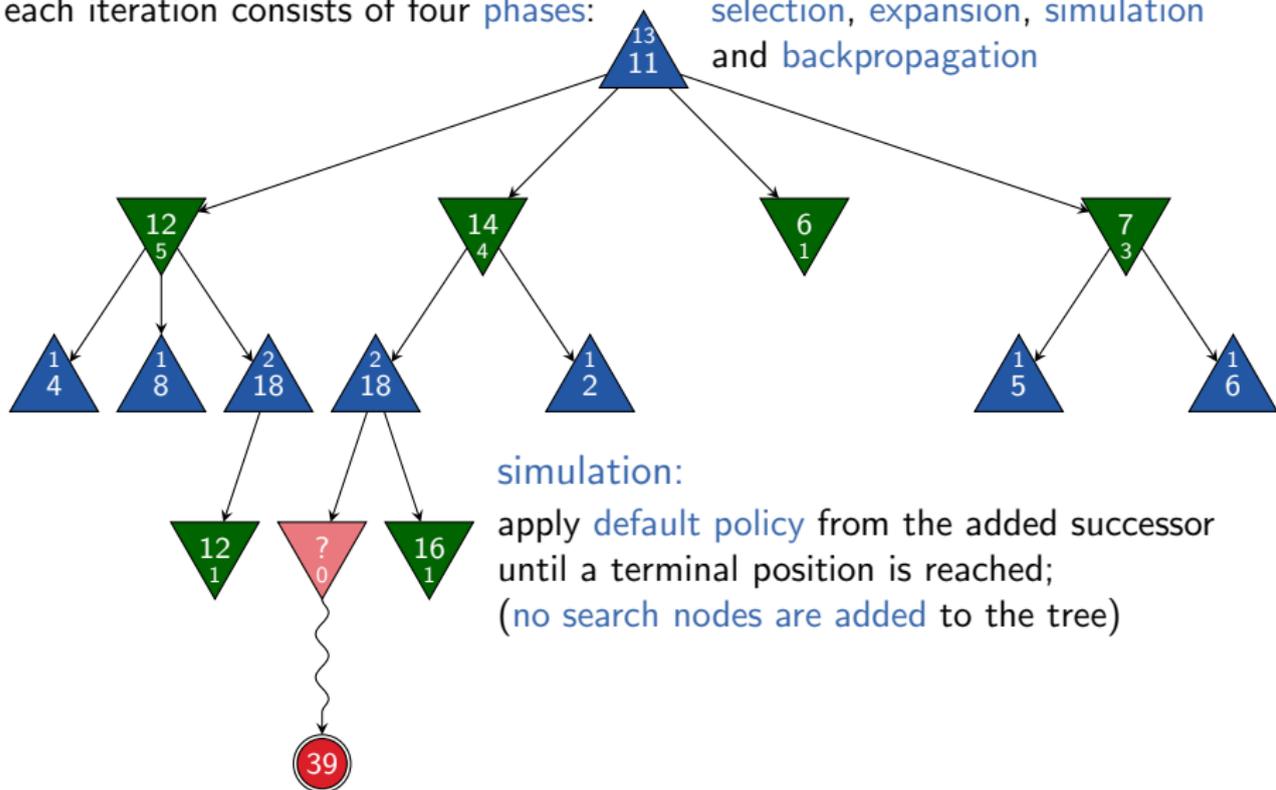
Idea and Example

each iteration consists of four phases: selection, expansion, simulation and backpropagation



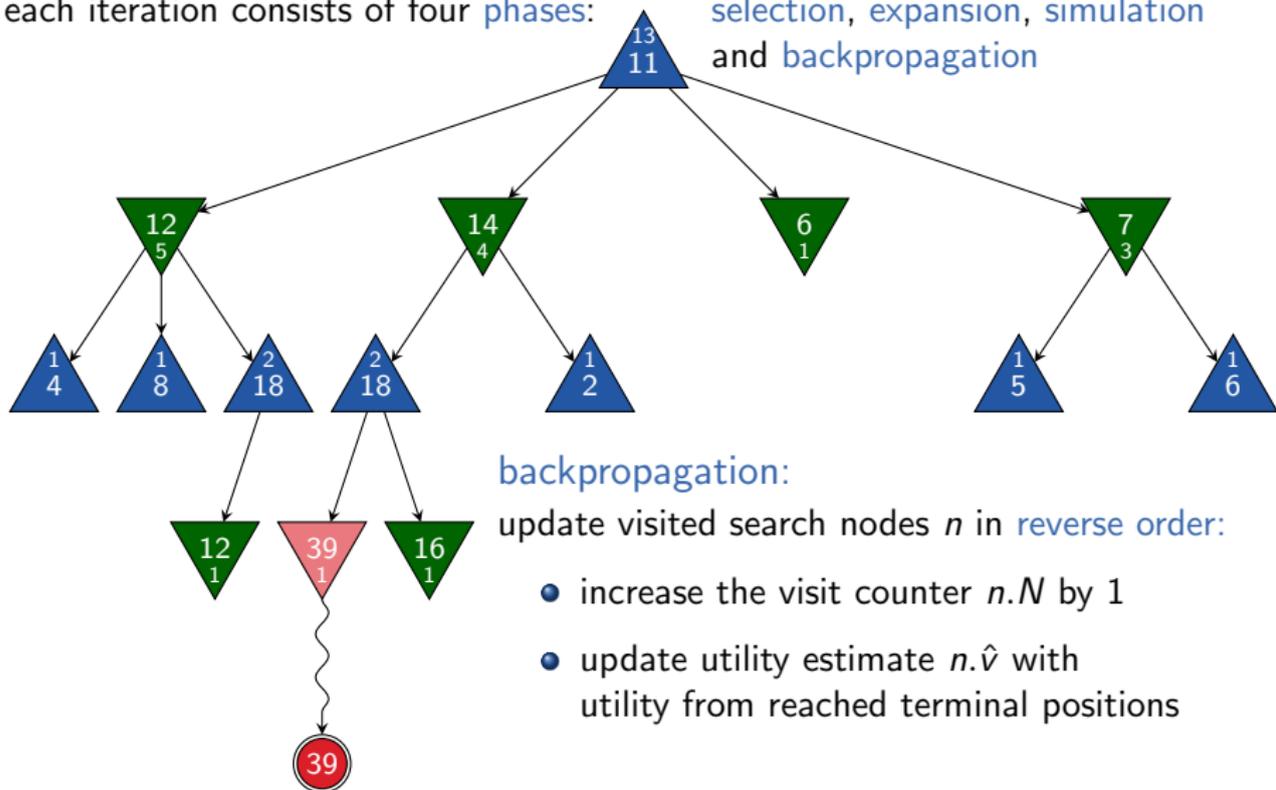
Idea and Example

each iteration consists of four phases: selection, expansion, simulation and backpropagation



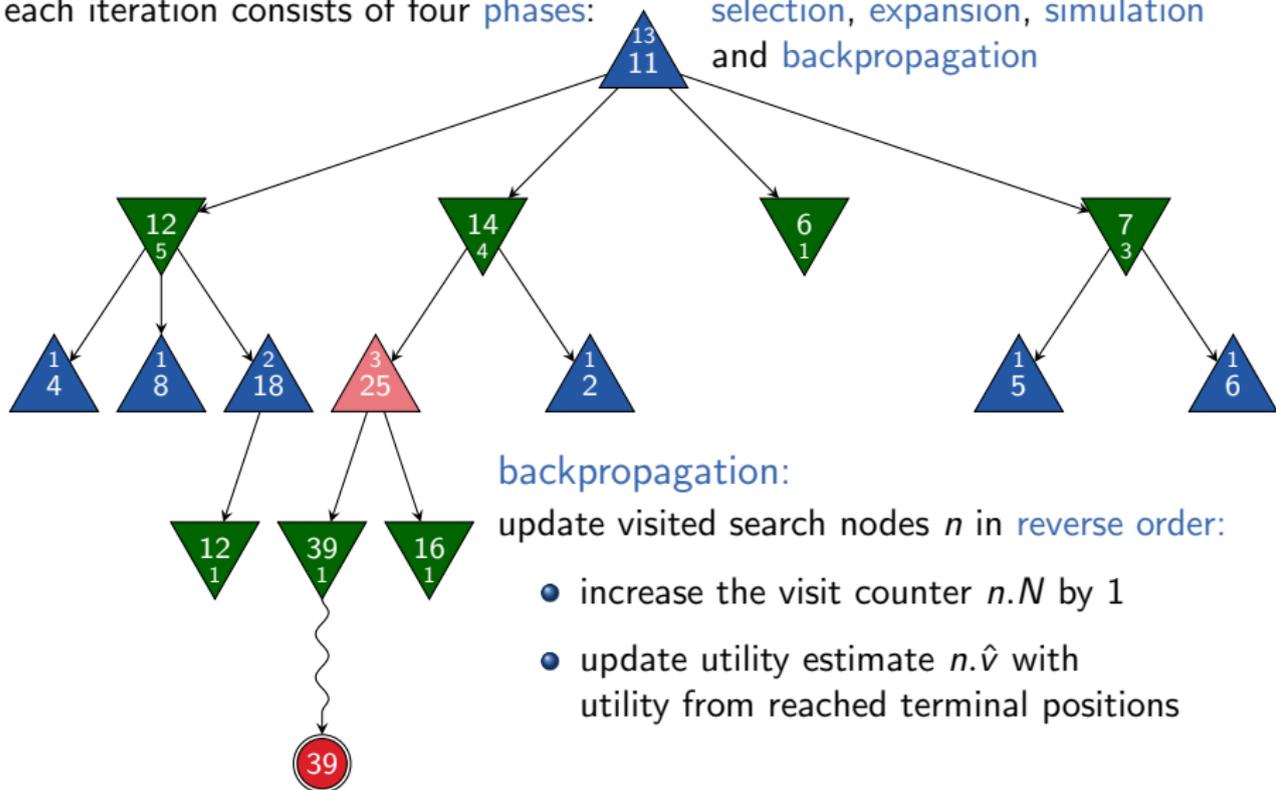
Idea and Example

each iteration consists of four phases: selection, expansion, simulation and backpropagation



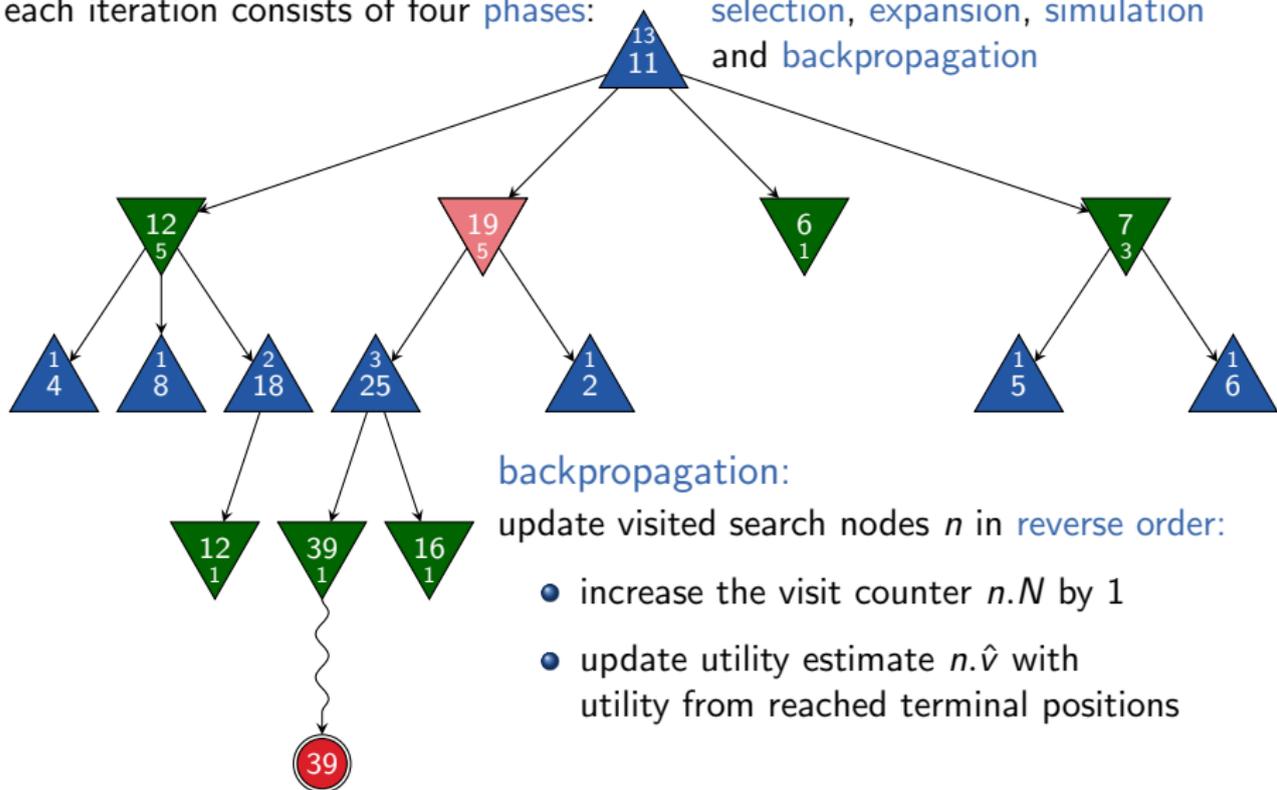
Idea and Example

each iteration consists of four phases: selection, expansion, simulation and backpropagation



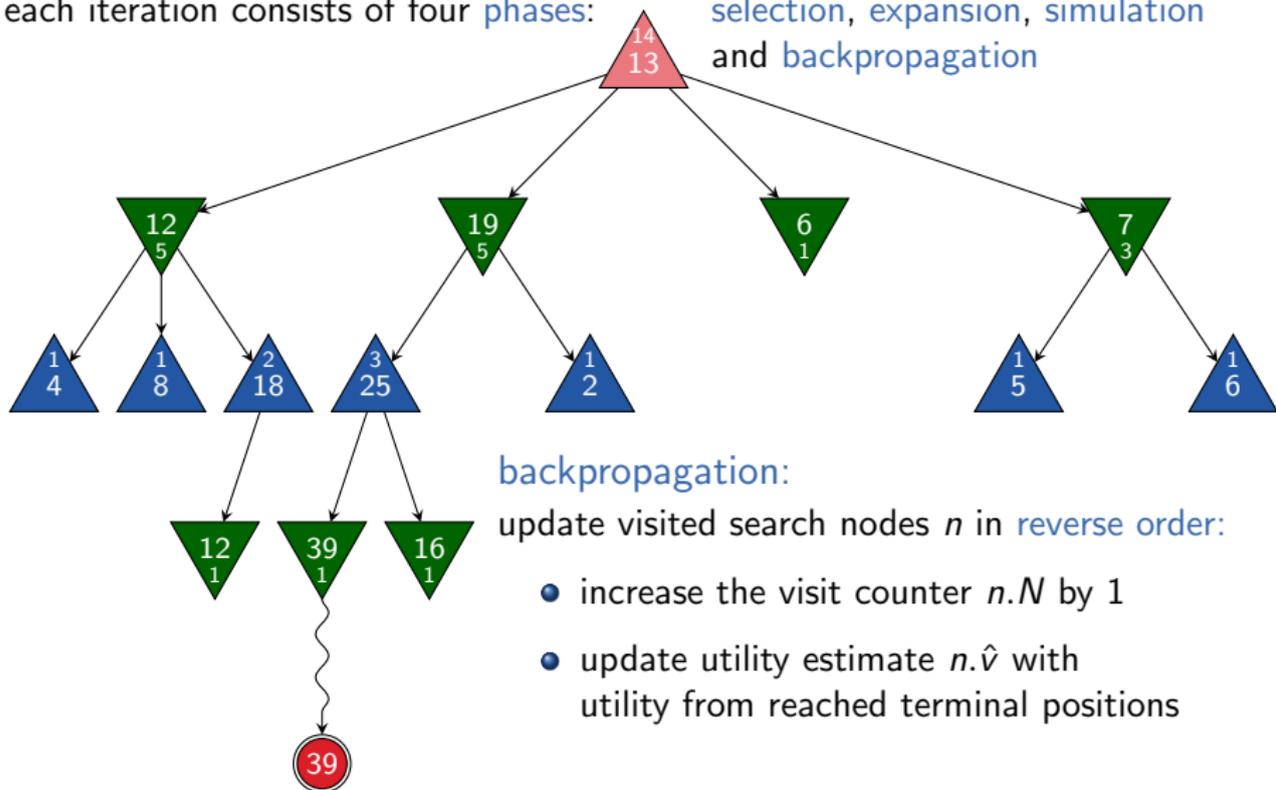
Idea and Example

each iteration consists of four phases: selection, expansion, simulation and backpropagation



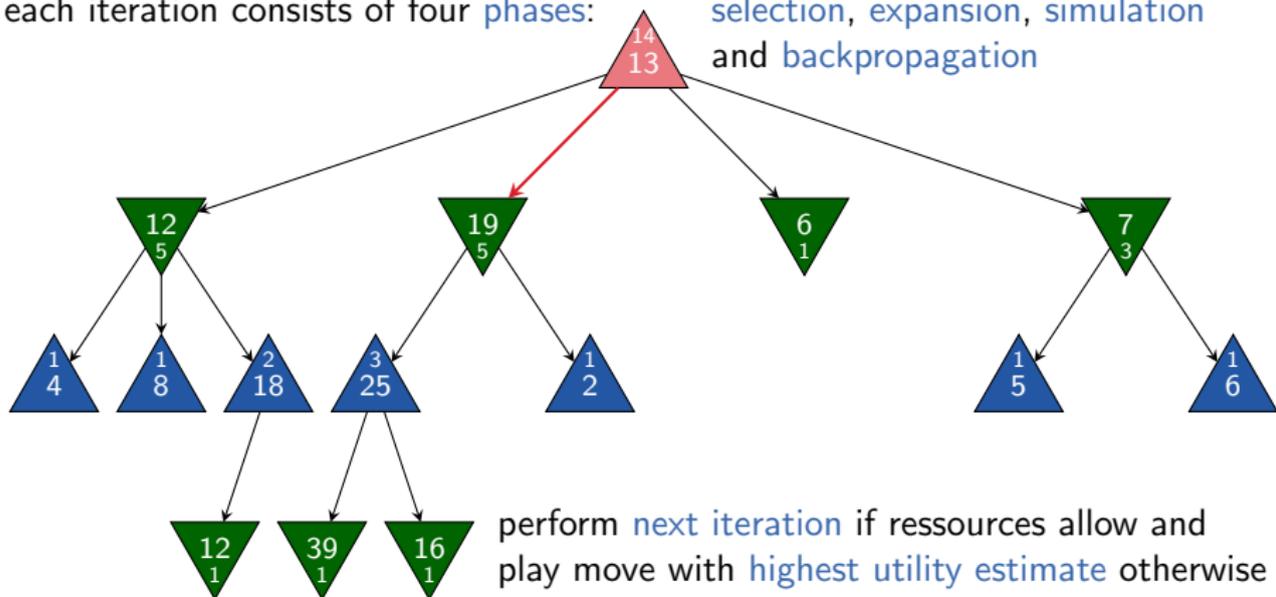
Idea and Example

each iteration consists of four phases: selection, expansion, simulation and backpropagation



Idea and Example

each iteration consists of four phases: selection, expansion, simulation and backpropagation



Monte-Carlo Tree Search: Pseudo-Code

Monte-Carlo Tree Search

```
 $n_0 := \text{create\_root\_node}()$   
while time_allows():  
    visit_node( $n_0$ )  
 $n_{\text{best}} := \arg \max_{n \in \text{succ}(n_0)} n \cdot \hat{v}$   
return  $n_{\text{best}}.\text{move}$ 
```

Monte-Carlo Tree Search: Pseudo-Code

```
function visit_node(n)
```

```
if is_terminal(n.position):
```

```
    utility := utility(n.position)
```

```
else:
```

```
    s := n.get_unvisited_successor()
```

```
    if s is none:
```

```
        n' := apply_tree_policy(n)
```

```
        utility := visit_node(n')
```

```
    else:
```

```
        utility := simulate_game(s)
```

```
        n.add_and_initialize_child_node(s, utility)
```

```
n.N := n.N + 1
```

```
n. $\hat{v}$  := n. $\hat{v}$  +  $\frac{utility - n.\hat{v}}{n.N}$ 
```

```
return utility
```

Summary

Summary

- Monte-Carlo methods compute **averages** over a number of random **samples**.
- **Monte-Carlo Tree Search (MCTS)** algorithms **simulate** a playout of the game
- and iteratively build a search tree, adding (at most) one node in each iteration.
- MCTS is parameterized by a **tree policy** and a **default policy**.