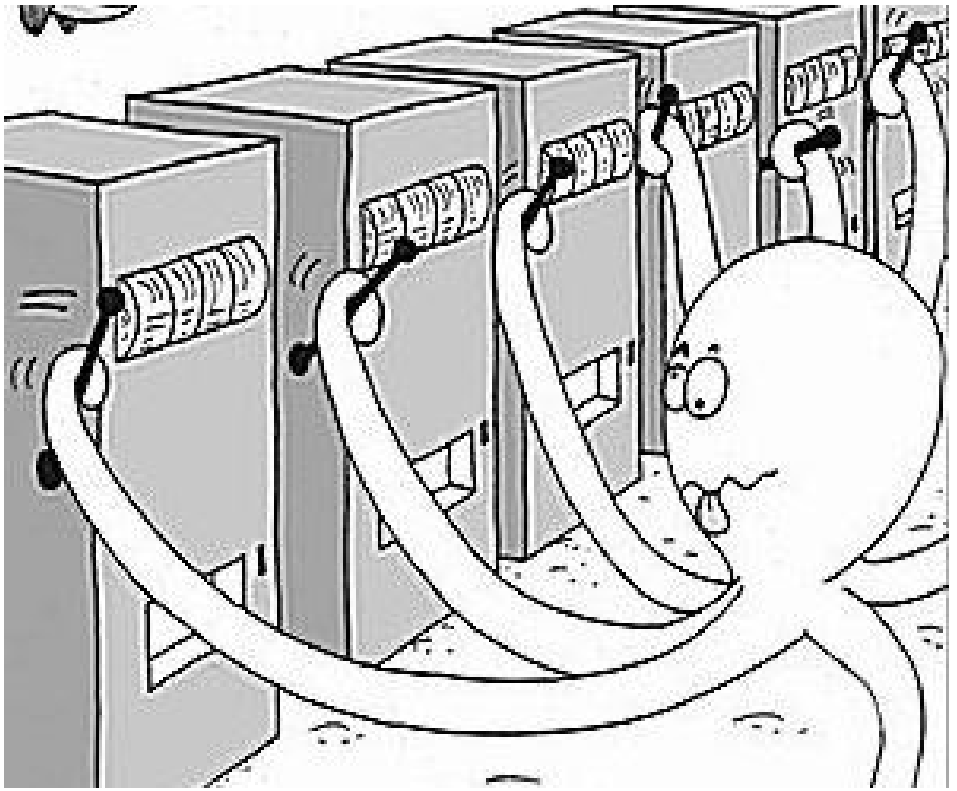
Reinforcement Learning

Kevin Spiteri
April 21, 2015

n-armed bandit



n-armed bandit

n-armed bandit

0.0 0.0 0.0 estimate

n-armed bandit

	0.9	0.5	0.1	
0	0.0	0.0	0.0	estimate
0	0	0	0	attempts
0	0	0	0	payoff

n-armed bandit

	0.9	0.5	0.1	
1.0	0.0	~~0.0~~ 1.0	0.0	estimate
1	0	~~0~~ 1	0	attempts
1	0	~~0~~ 1	0	payoff

n-armed bandit

	0.9	0.5	0.1	
0.5	0.0	~~1.0~~ 0.5	0.0	estimate
2	0	~~1~~ 2	0	attempts
1	0	~~1~~ 1	0	payoff

Exploration

	0.9	0.5	0.1	
0.67	~~0.0~~ 1.0	0.5	0.0	estimate
3	~~0~~ 1	2	0	attempts
2	~~0~~ 1	1	0	payoff

Going on …

	0.9	0.5	0.1	
0.86	0.9	0.5	0.1	estimate
300	280	10	10	attempts
258	252	5	1	payoff

Changing environment

	0.7	0.8	0.1	
0.86	0.9	0.5	0.1	estimate
300	280	10	10	attempts
258	252	5	1	payoff

Changing environment

	0.7	0.8	0.1	
0.77	0.8	0.65	0.1	estimate
600	560	20	20	attempts
463	448	13	2	payoff

Changing environment

	0.7	0.8	0.1	
0.72	0.74	0.74	0.1	estimate
1500	1400	50	50	attempts
1078	1036	37	5	payoff

n-armed bandit

- Optimal payoff **(0.82):**
  0.9 x 300 + 0.8 x 1200 = 1230

- Actual payoff **(0.72):**
  0.9 x 280 + 0.5 x 10 + 0.1 x 10 +
  0.7 x 1120 + 0.8 x 40 + 0.1 x 40 = 1078

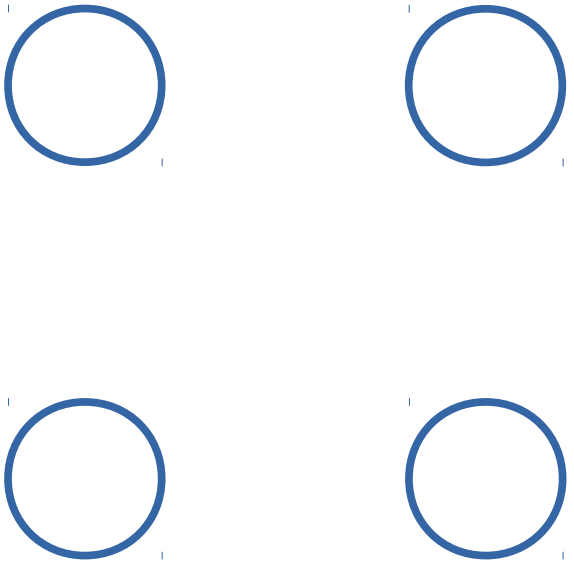
n-armed bandit

- Evaluation vs instruction.
- Discounting.
- Initial estimates.
- There is no best way or standard way.

Markov Decision Process (MDP)

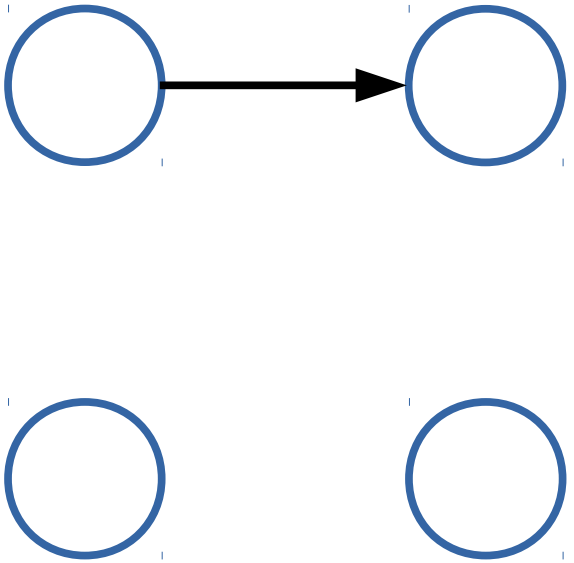
Markov Decision Process (MDP)

- States



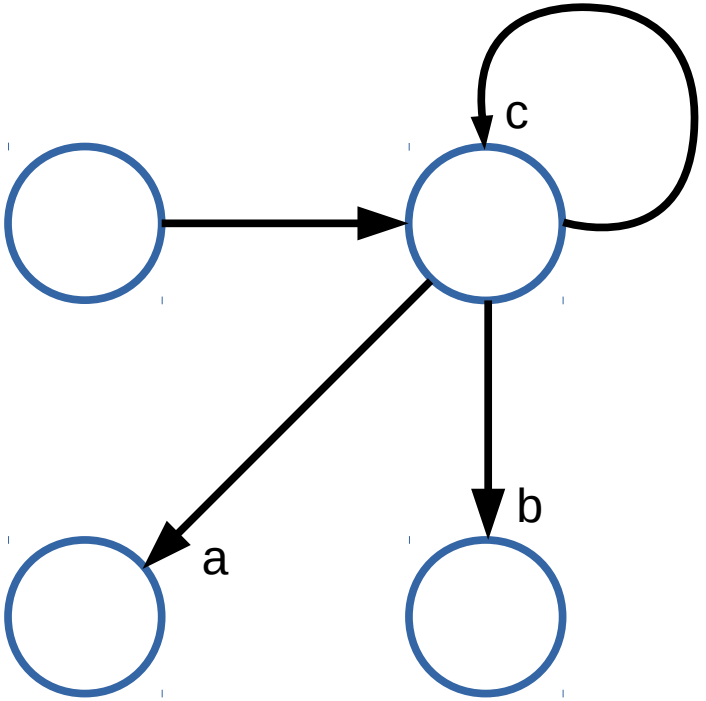
Markov Decision Process (MDP)

- States



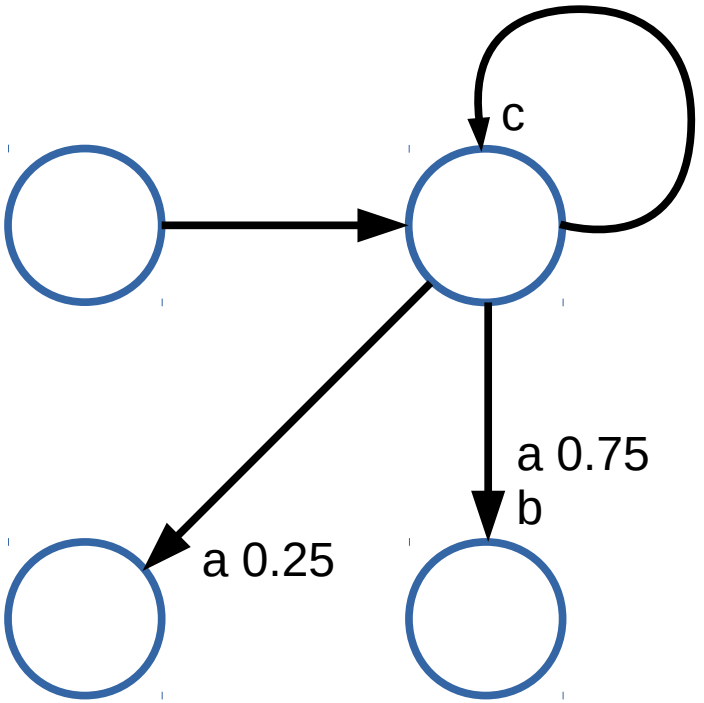
Markov Decision Process (MDP)

- States
- Actions



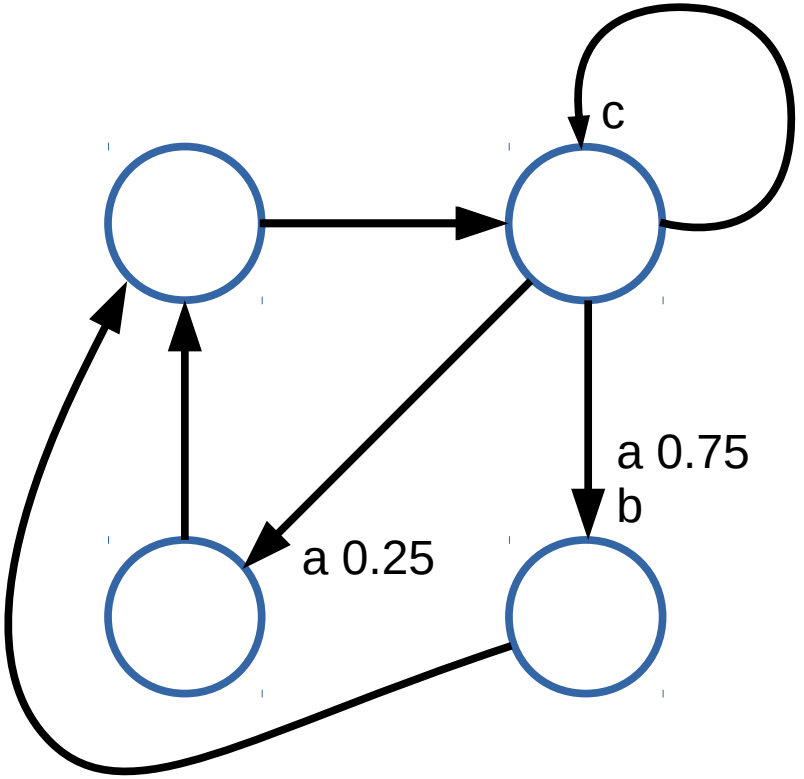
Markov Decision Process (MDP)

- States
- Actions
- Model



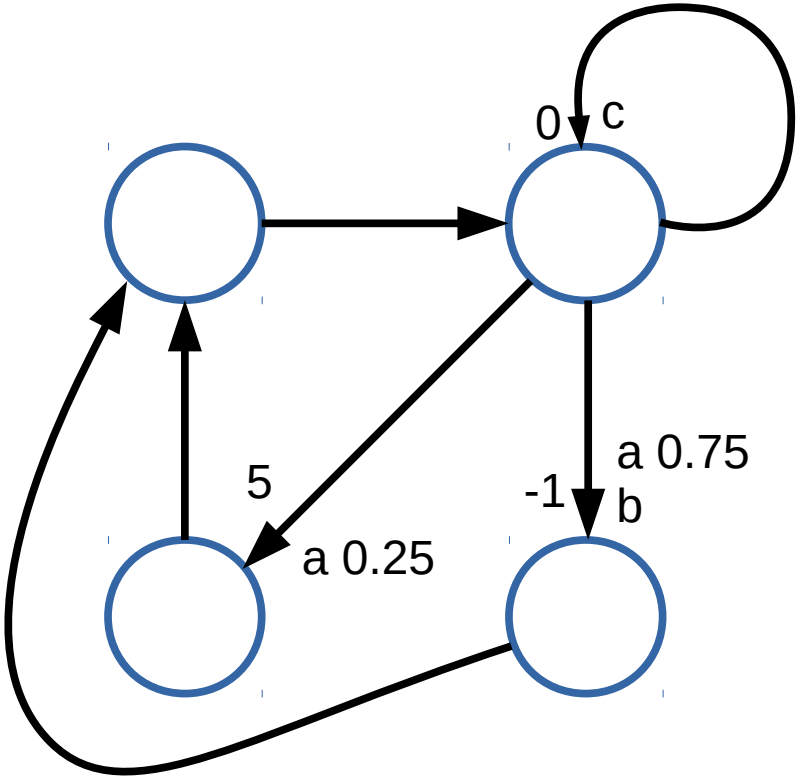
Markov Decision Process (MDP)

- States
- Actions
- Model



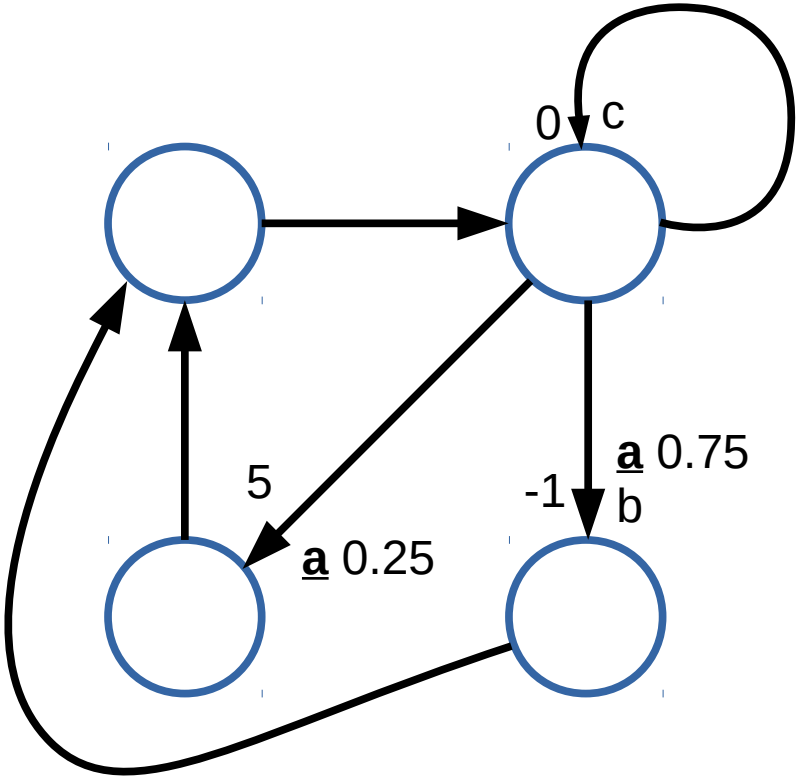
Markov Decision Process (MDP)

- States
- Actions
- Model
- Reward



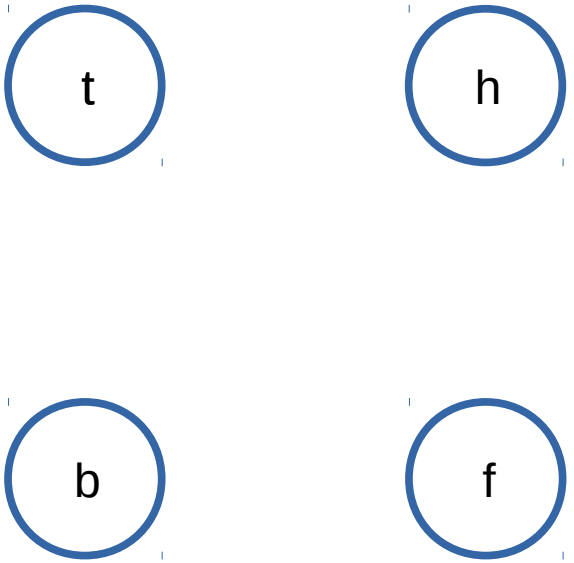
Markov Decision Process (MDP)

- States
- Actions
- Model
- Reward
- Policy



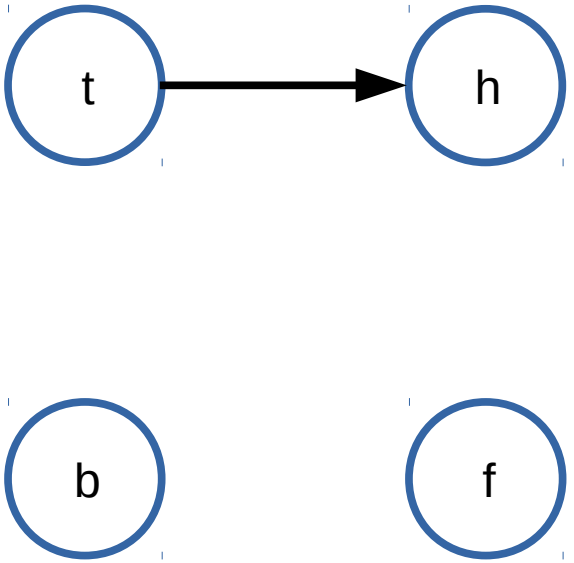
Markov Decision Process (MDP)

- States: ball
  table
  hand
  basket
  floor



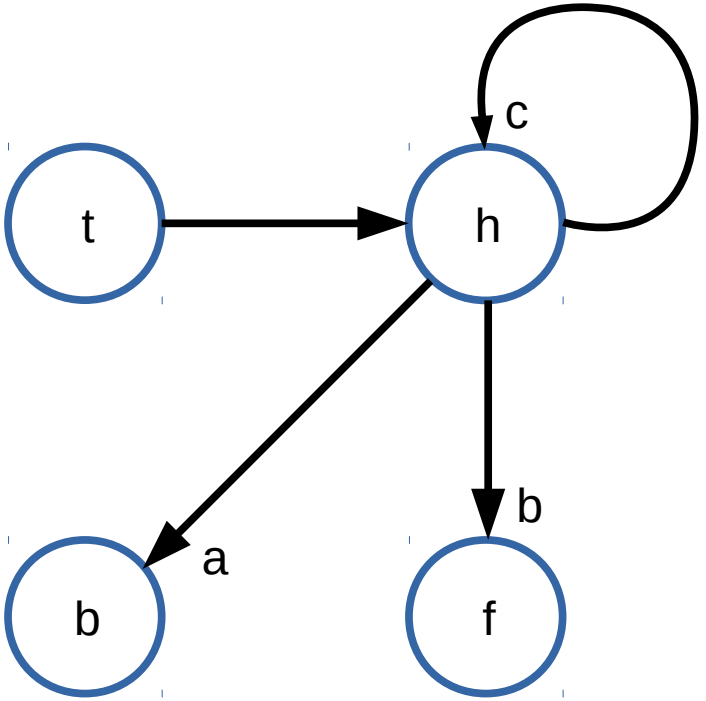
Markov Decision Process (MDP)

- States: ball
  table
  hand
  basket
  floor



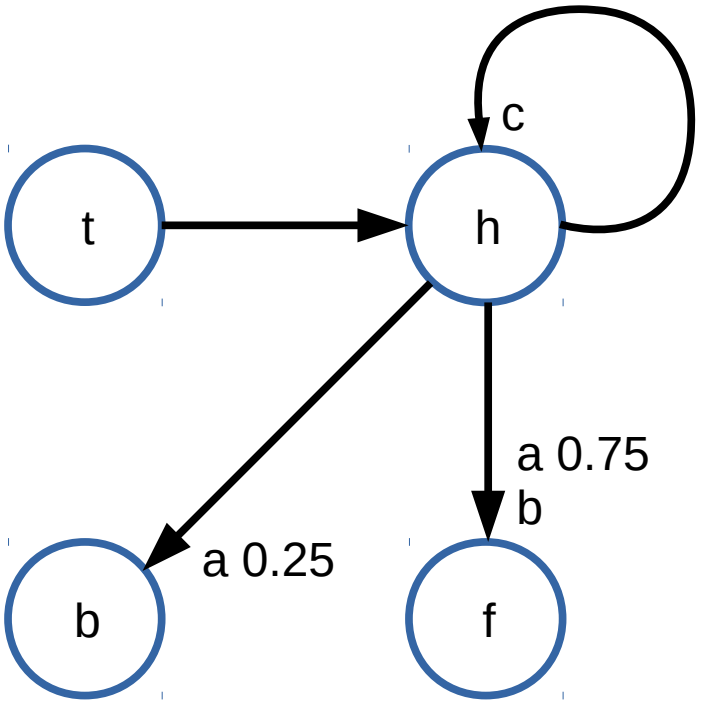
Markov Decision Process (MDP)

- States: ball
  table
  hand
  basket
  floor
- Actions:
  a) attempt
  b) drop
  c) wait



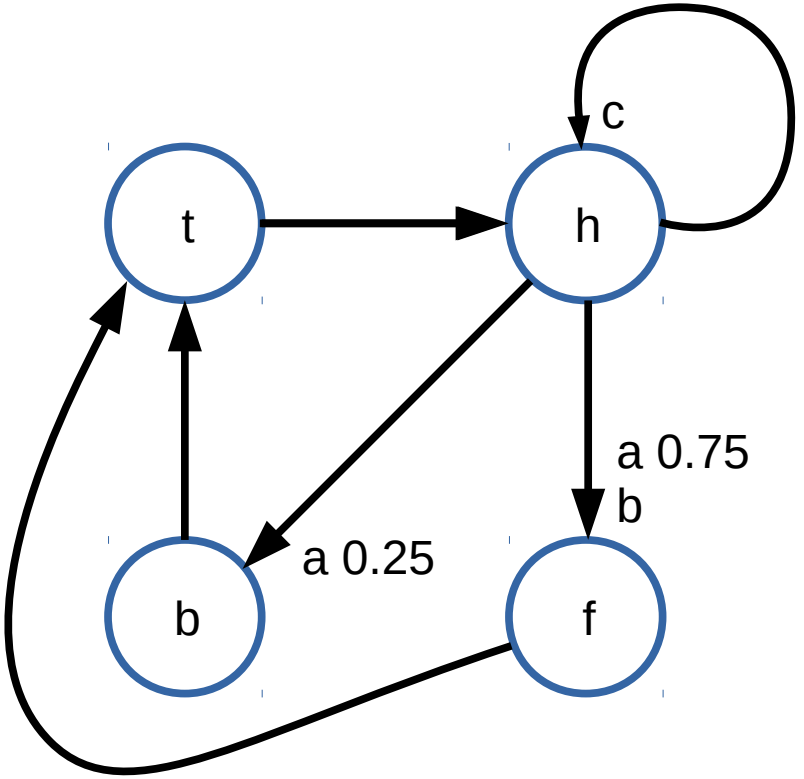
Markov Decision Process (MDP)

- States: ball
  table
  hand
  basket
  floor
- Actions:
  a) attempt
  b) drop
  c) wait



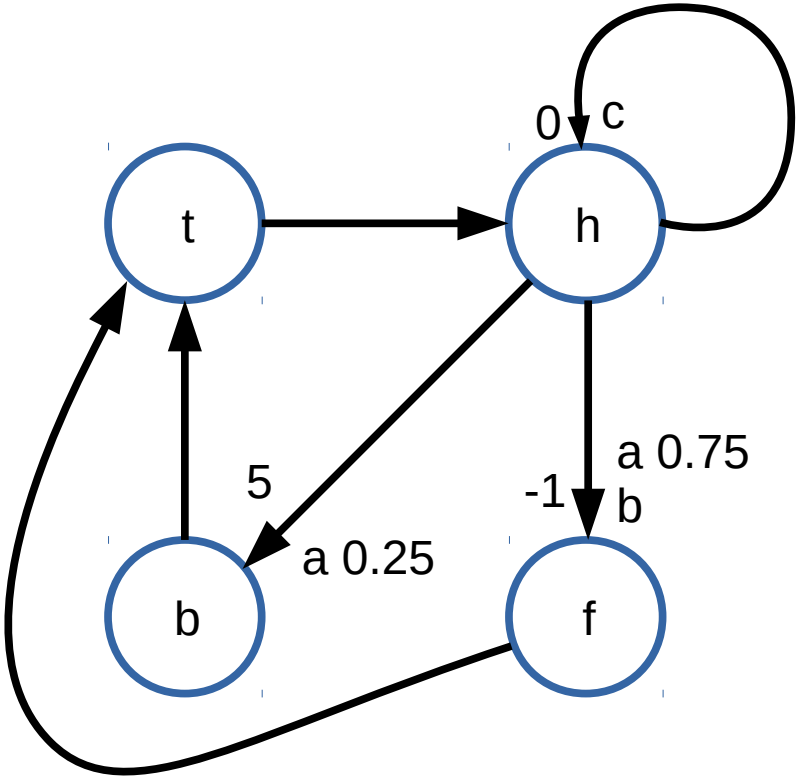
Markov Decision Process (MDP)

- States: ball
  table
  hand
  basket
  floor
- Actions:
  a) attempt
  b) drop
  c) wait



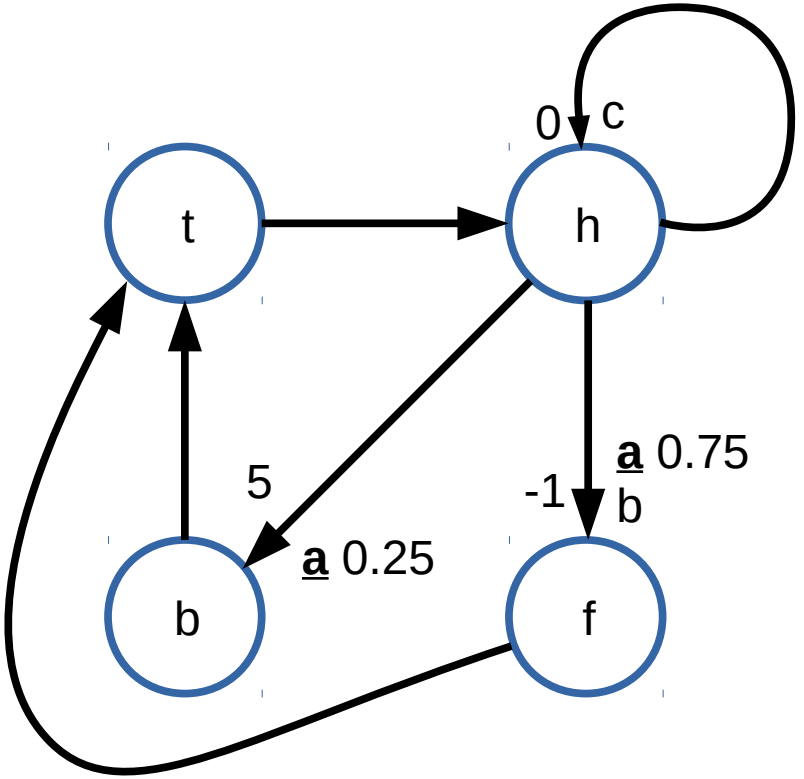
Markov Decision Process (MDP)

- States: ball
  table
  hand
  basket
  floor
- Actions:
  a) attempt
  b) drop
  c) wait



Markov Decision Process (MDP)

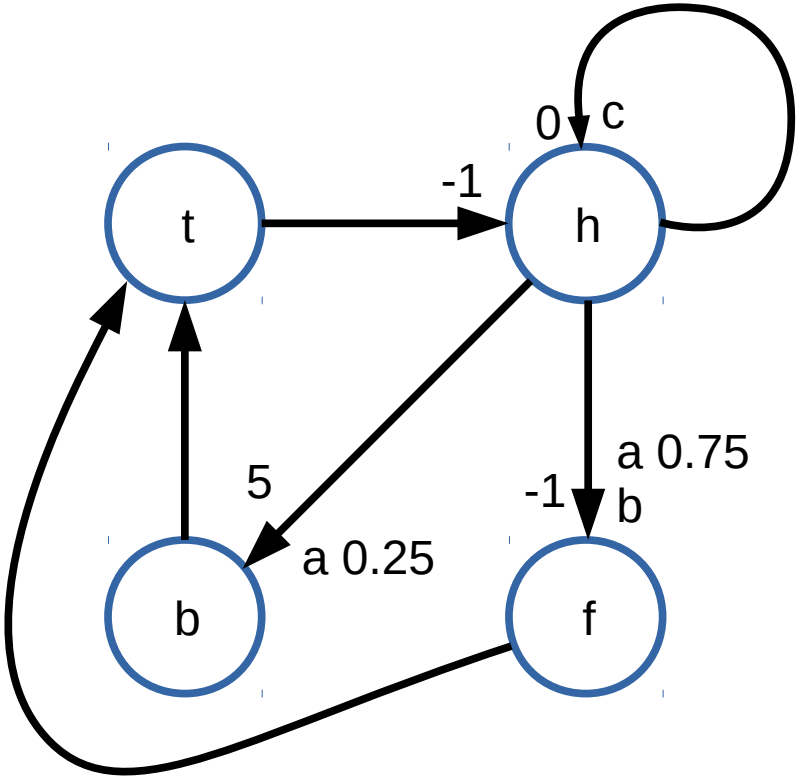
- States: ball
  table
  hand
  basket
  floor
- Actions:
  a) attempt
  b) drop
  c) wait



Expected reward per round:
0.25 x 5 + 0.75 x (-1) = 0.5

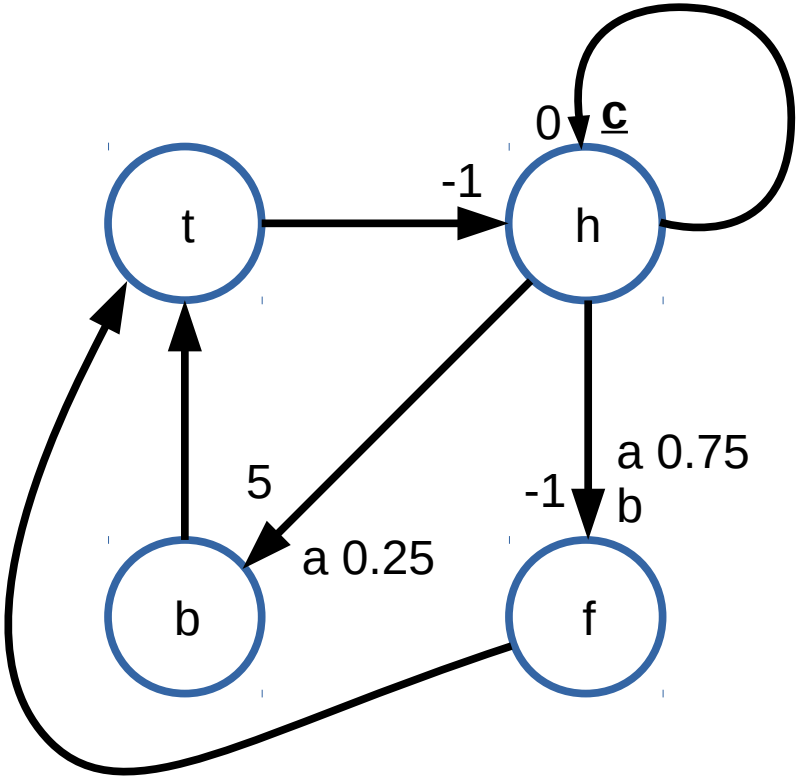
Markov Decision Process (MDP)

- States: ball
  table
  hand
  basket
  floor
- Actions:
  a) attempt
  b) drop
  c) wait



Markov Decision Process (MDP)

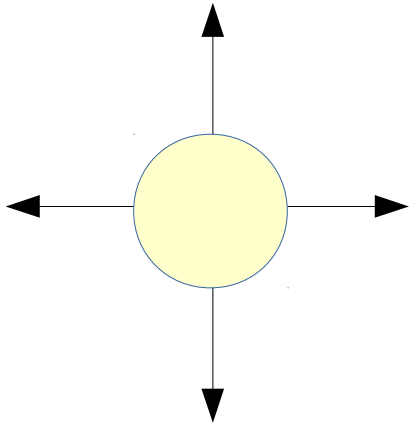
- States: ball
  table
  hand
  basket
  floor
- Actions:
  a) attempt
  b) drop
  c) wait

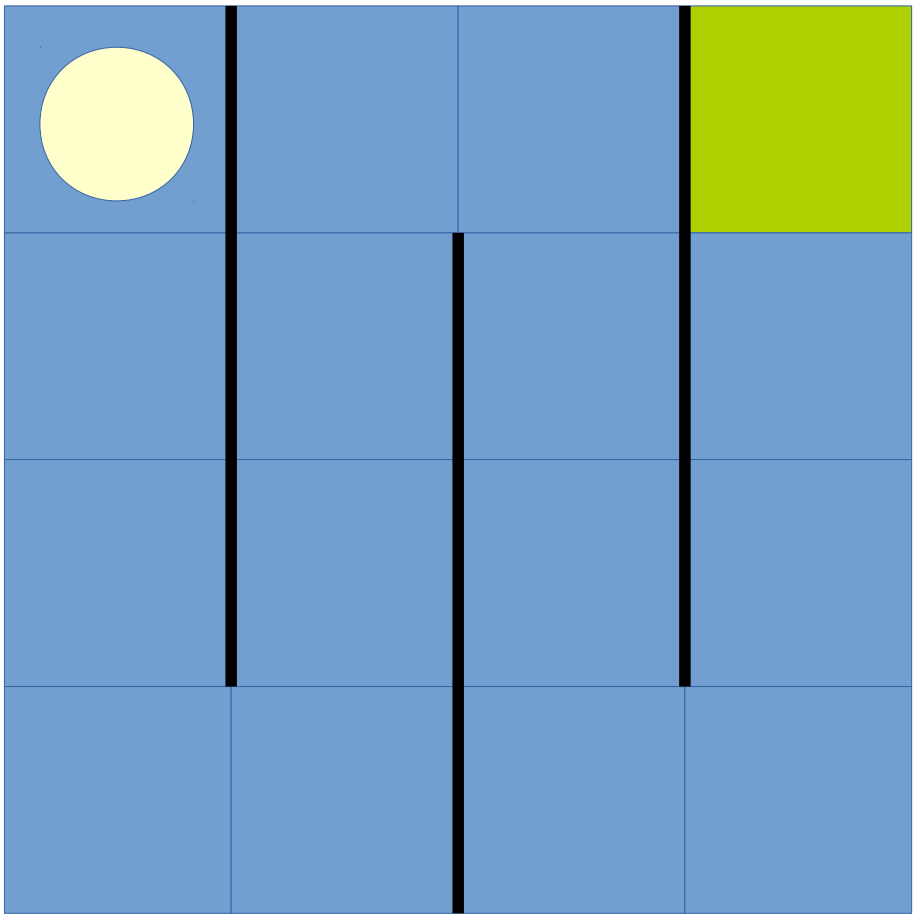


Reinforcement Learning Tools

- Dynamic Programming
- Monte Carlo Methods
- Temporal Difference Learning

Grid World





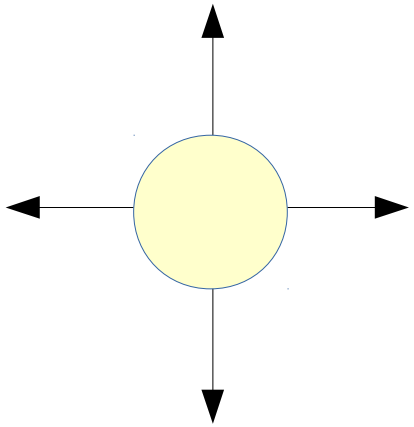
Reward:

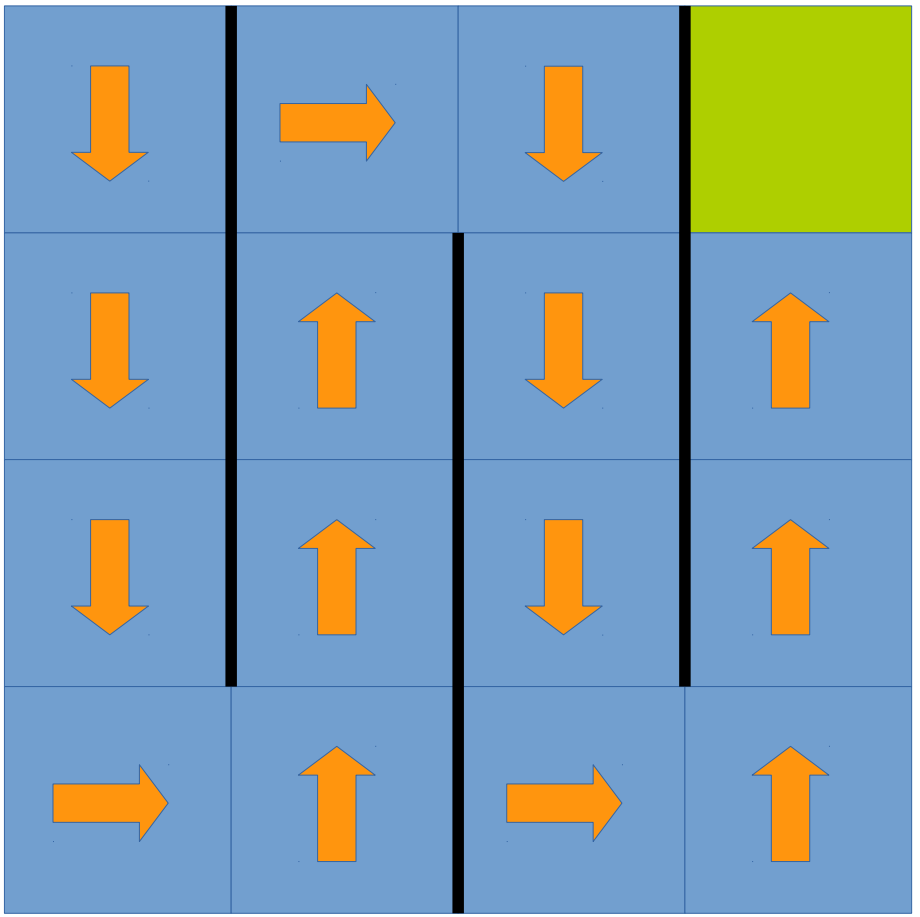
Normal move: -1

Over obstacle: -10

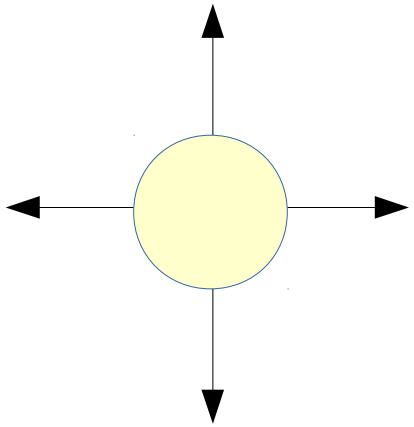
Best reward: -15

Optimal Policy



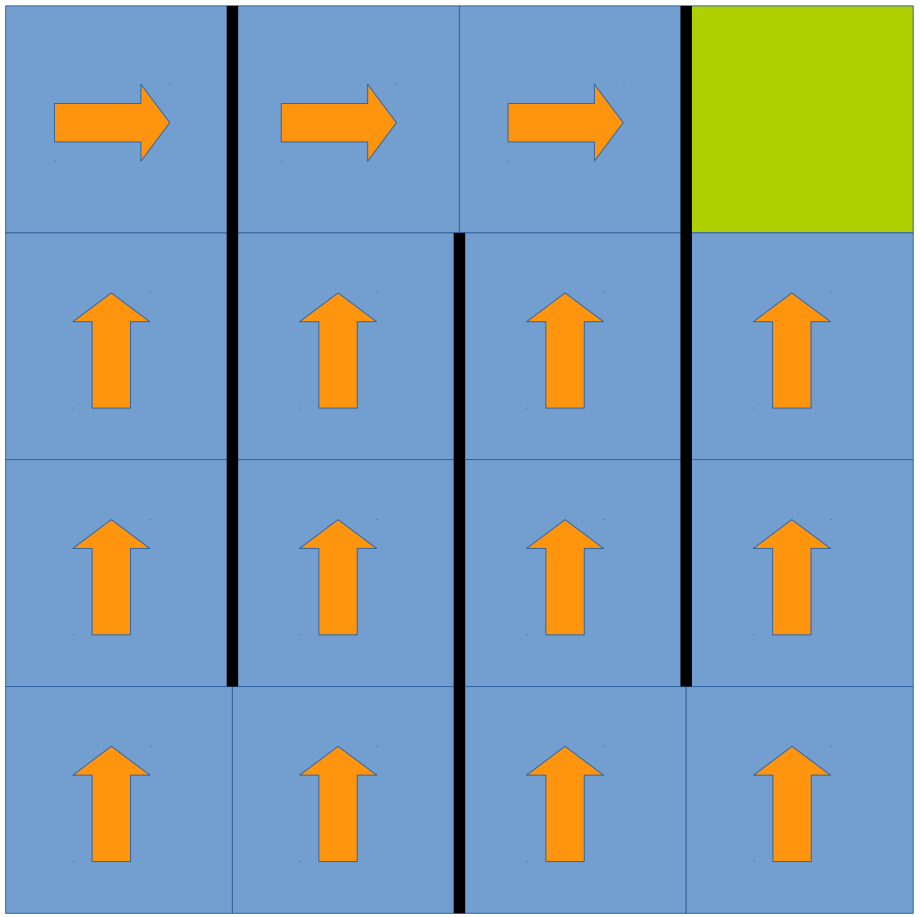


Value Function

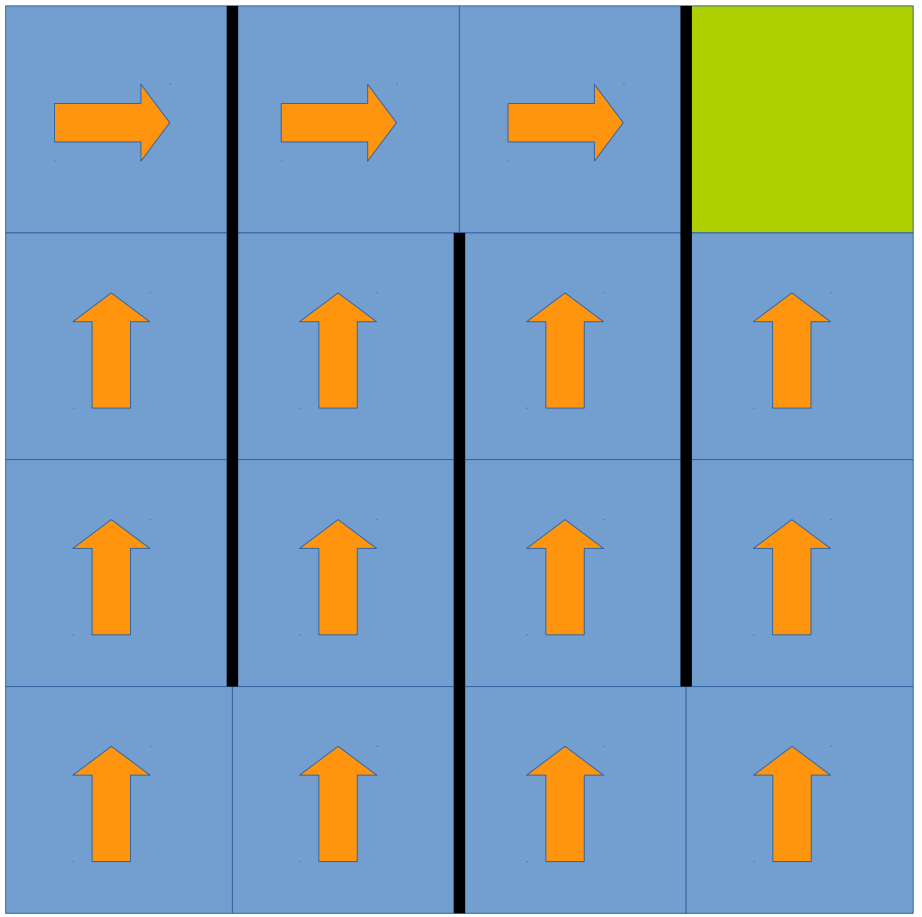


-15	-8	-7	0
-14	-9	-6	-1
-13	-10	-5	-2
-12	-11	-4	-3

Initial Policy

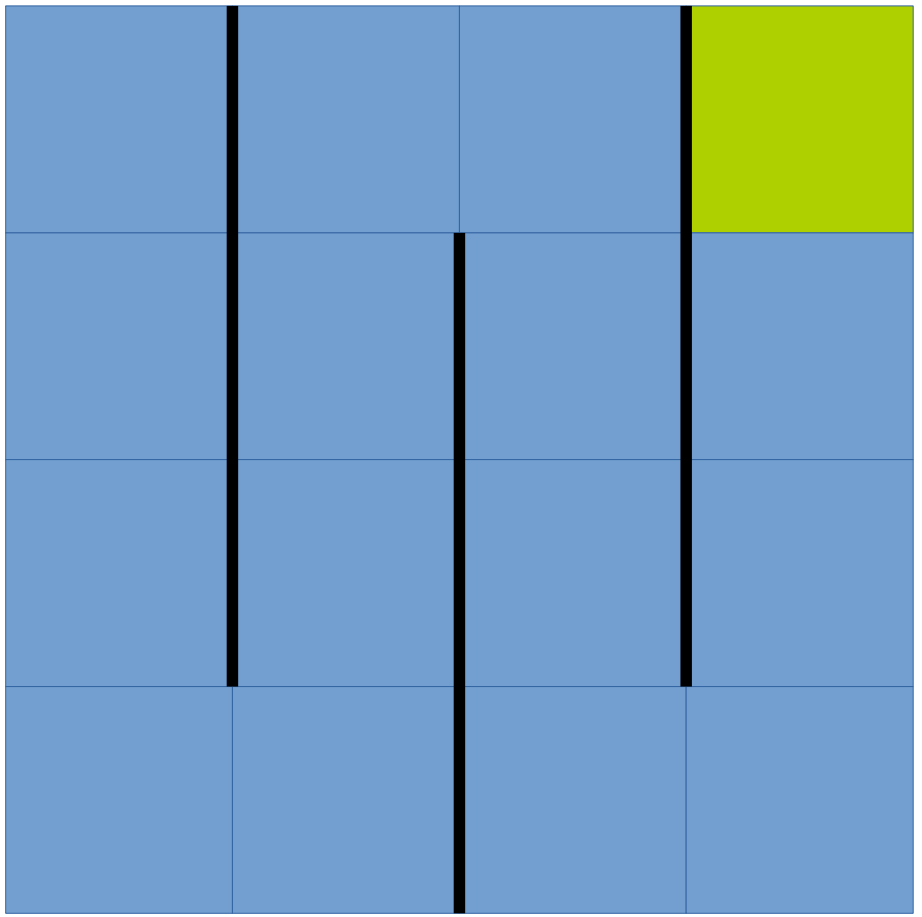


Policy Iteration



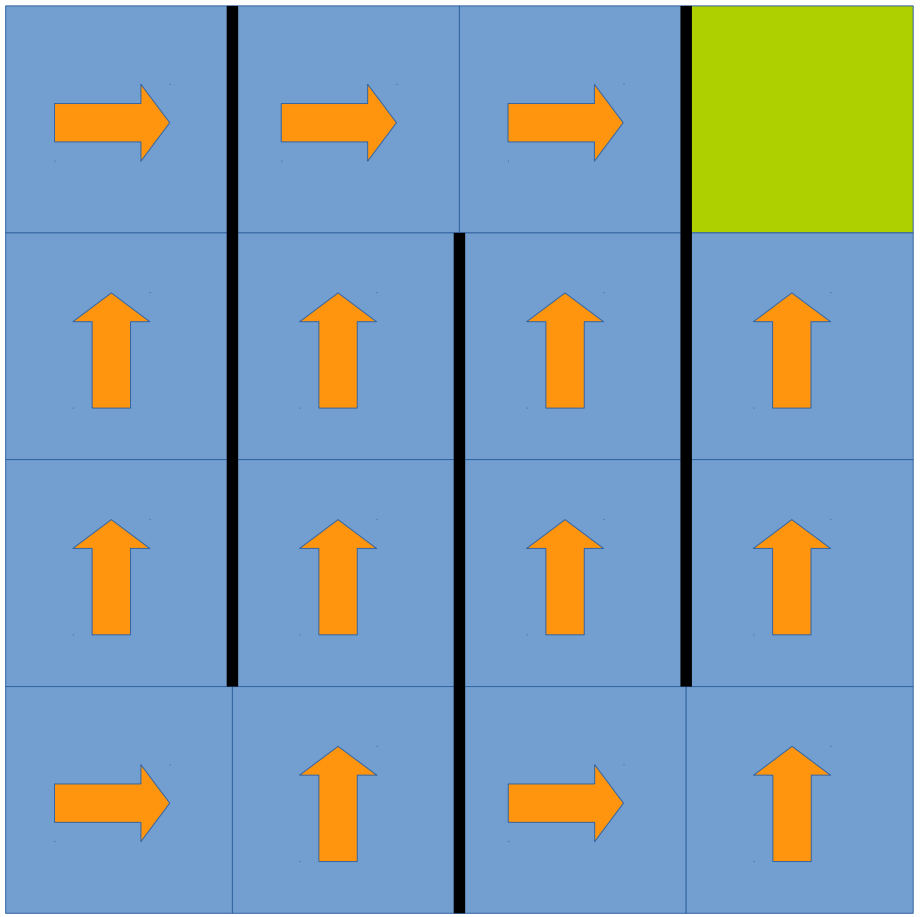
-21	-11	-10	0
-22	-12	-11	-1
-23	-13	-12	-2
-24	-14	-13	-3

Policy Iteration



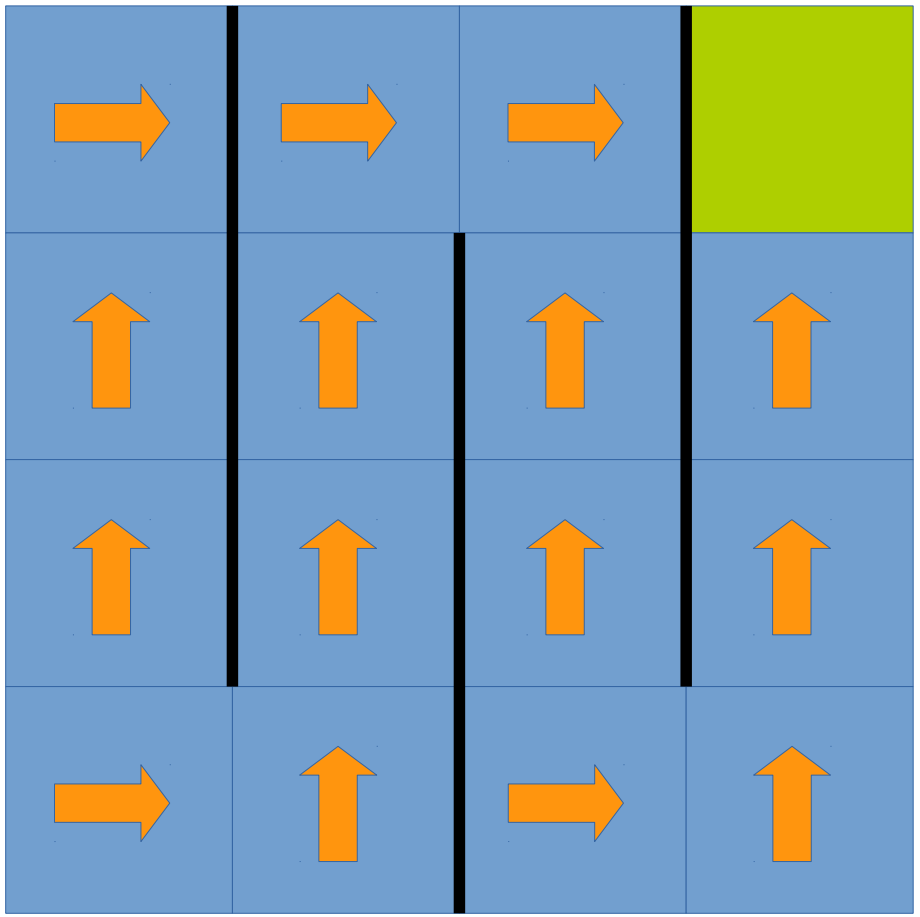
-21	-11	-10	0
-22	-12	-11	-1
-23	-13	-12	-2
-24	-14	-13	-3

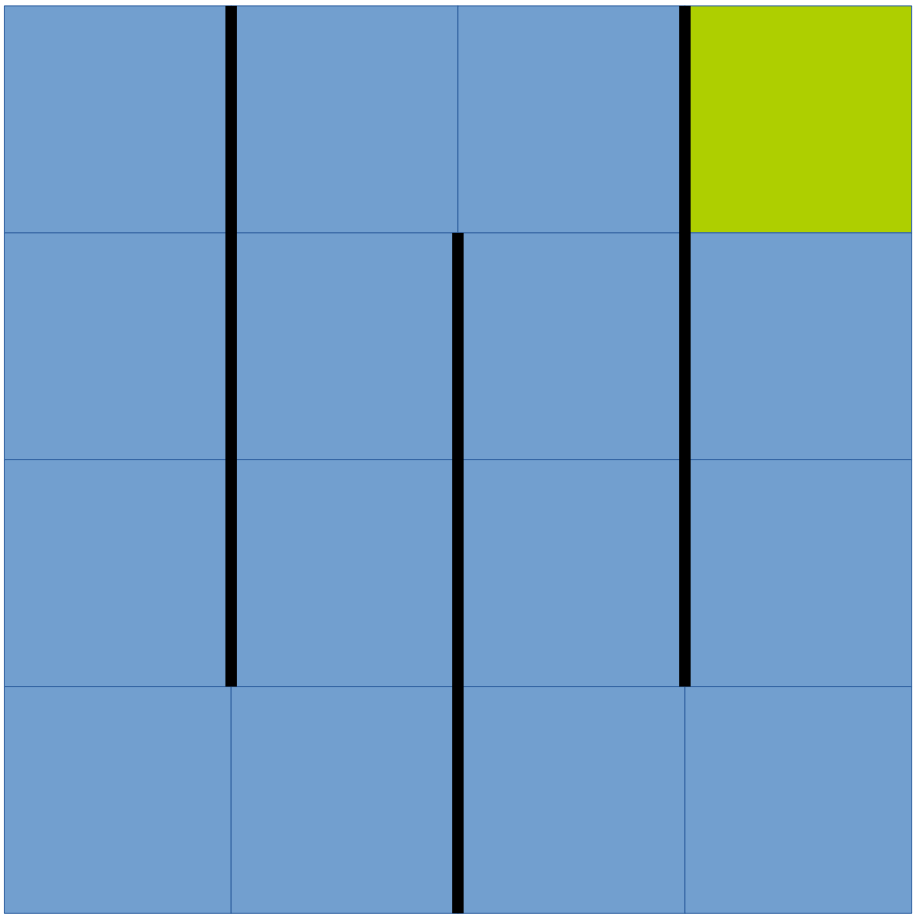
Policy Iteration



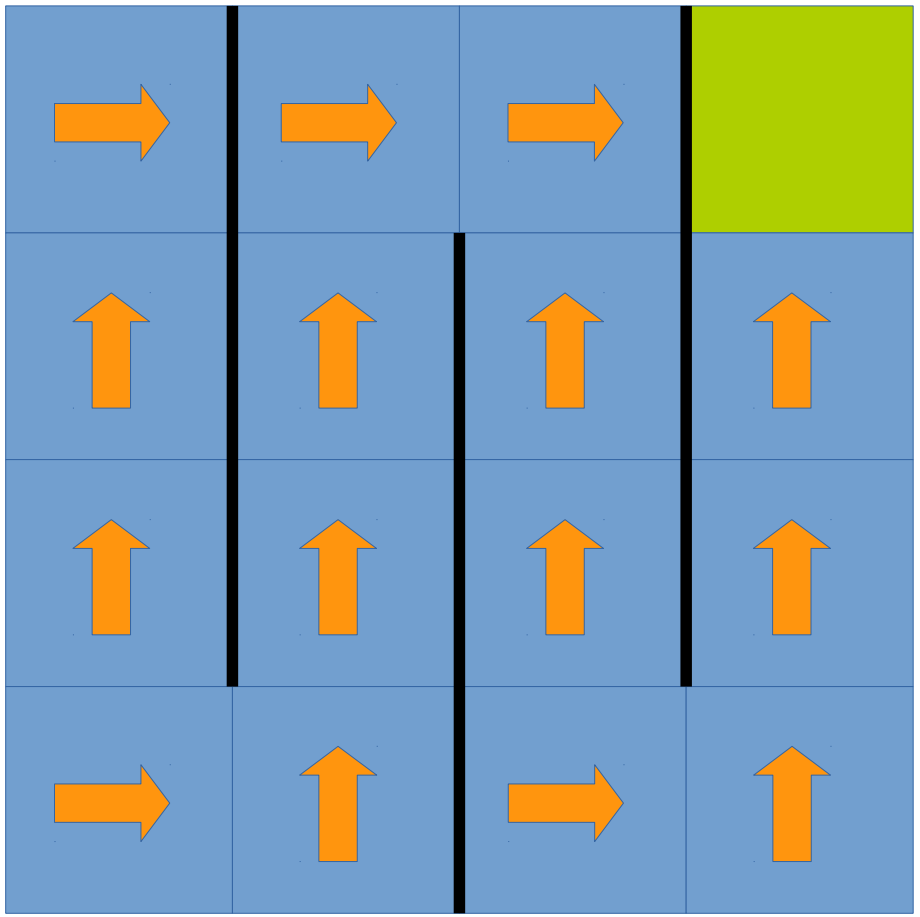
-21	-11	-10	0
-22	-12	-11	-1
-23	-13	-12	-2
-24	-14	-13	-3

Policy Iteration



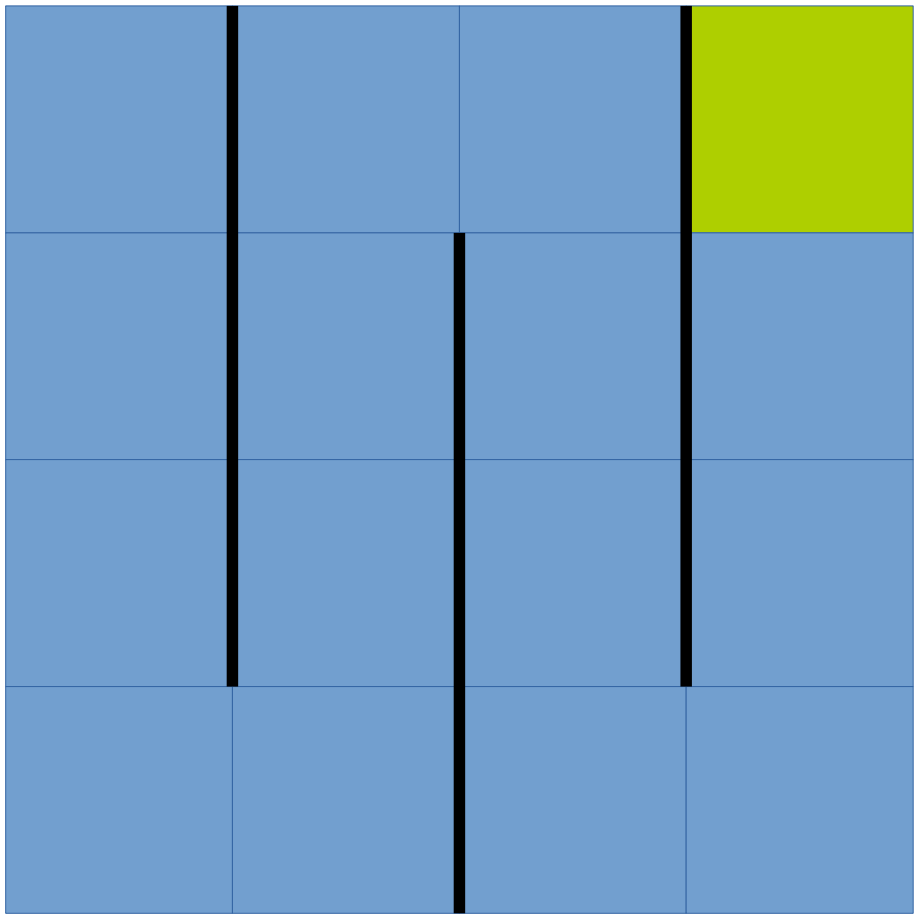


Policy Iteration



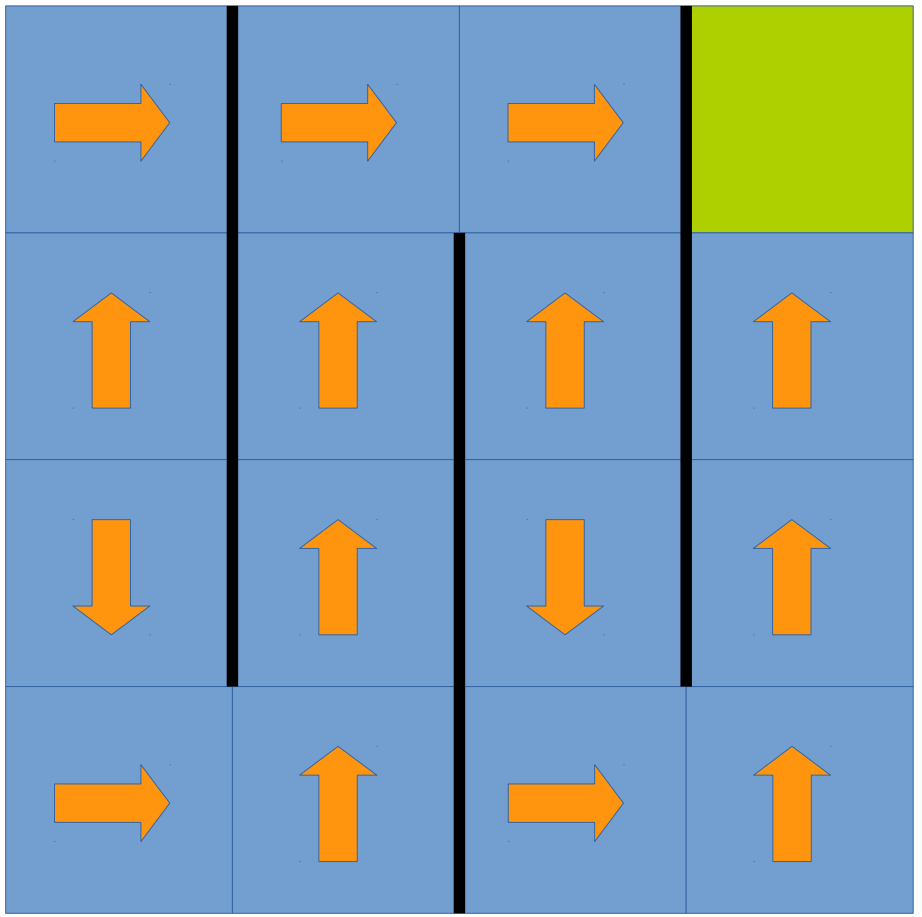
-21	-11	-10	0
-22	-12	-11	-1
-23	-13	-12	-2
-15	-14	-4	-3

Policy Iteration



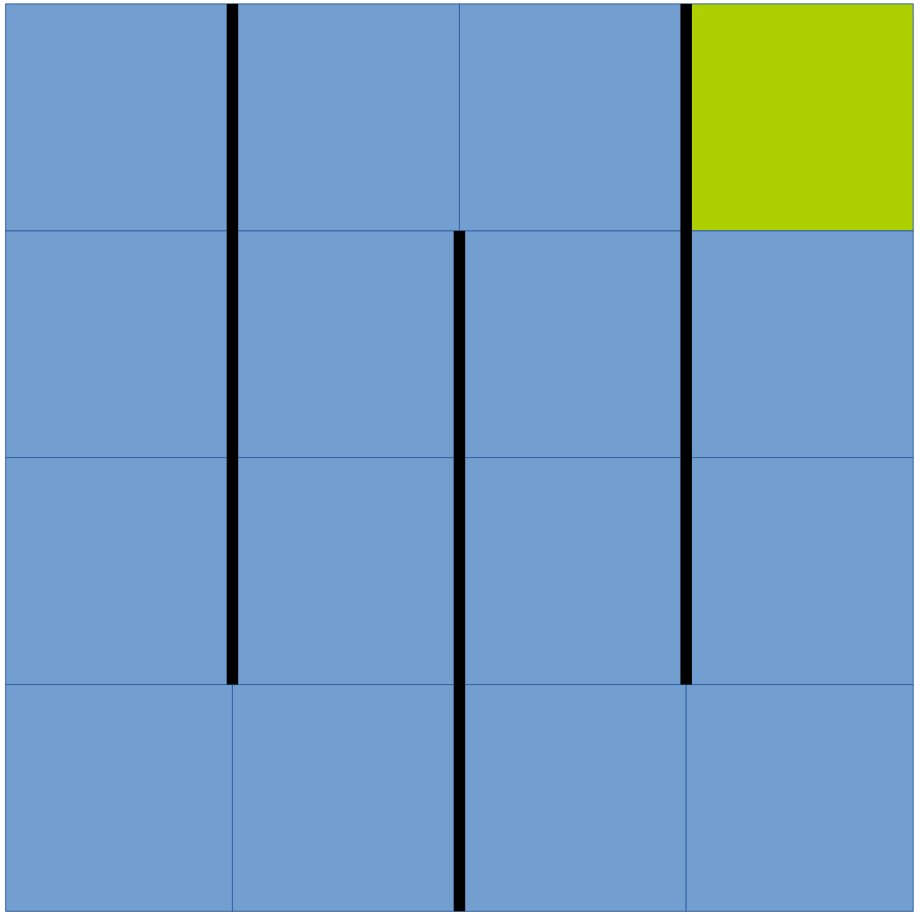
-21	-11	-10	0
-22	-12	-11	-1
-23	-13	-12	-2
-15	-14	-4	-3

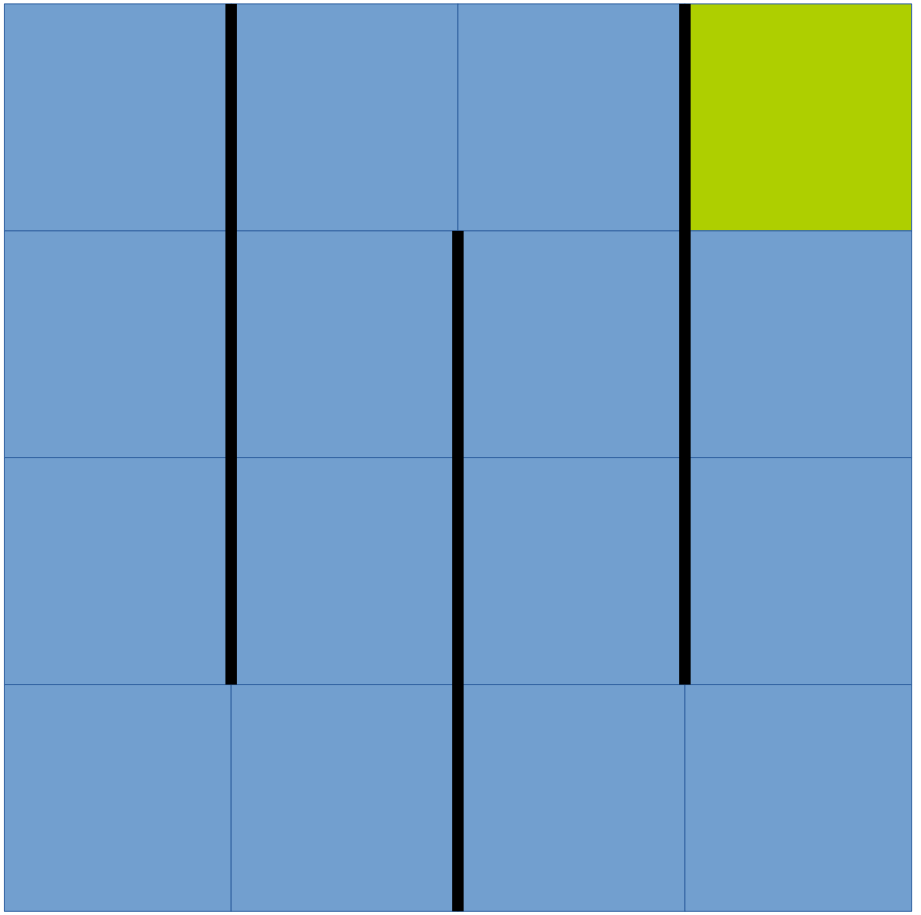
Policy Iteration



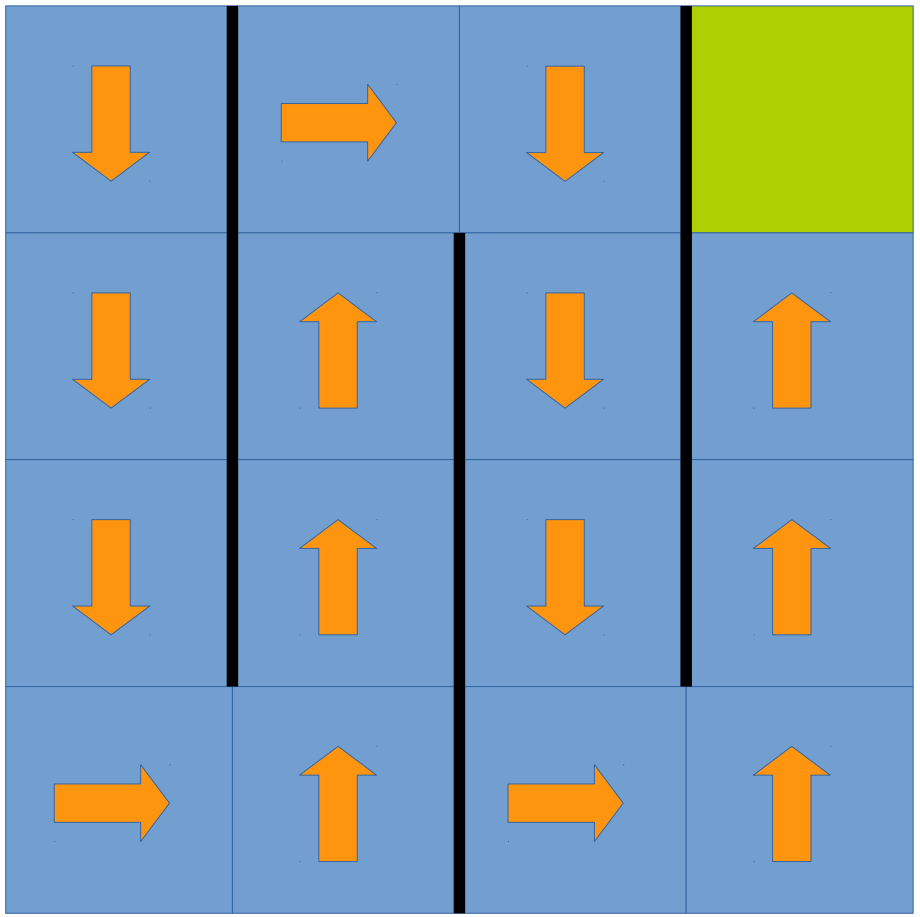
-21	-11	-10	0
-22	-12	-11	-1
-23	-13	-12	-2
-15	-14	-4	-3

Policy Iteration



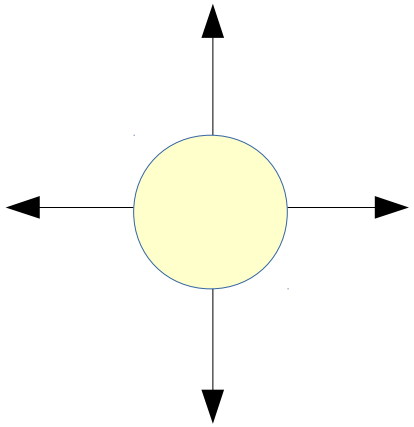


Policy Iteration



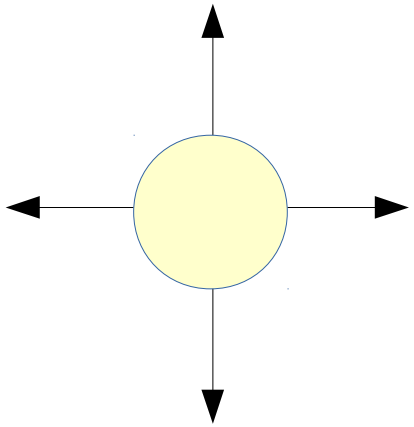
-15	-8	-7	0
-14	-9	-6	-1
-13	-10	-5	-2
-12	-11	-4	-3

Value Iteration



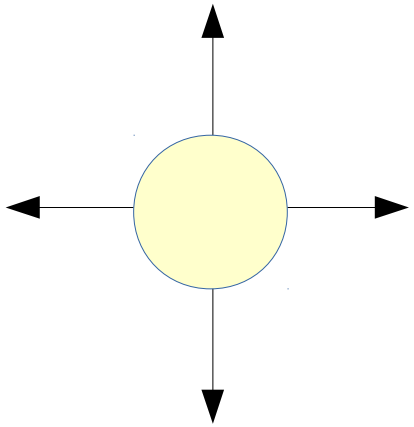
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Value Iteration



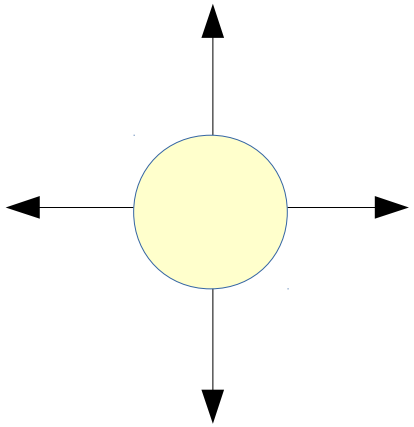
-1	-1	-1	0
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1

Value Iteration



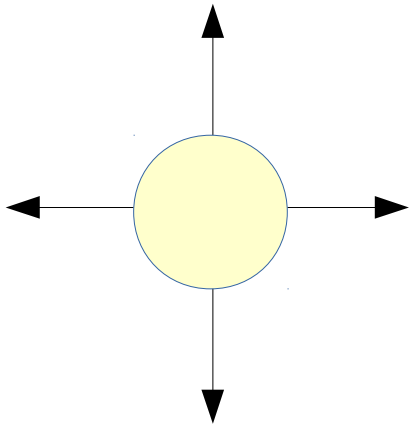
-2	-2	-2	0
-2	-2	-2	-1
-2	-2	-2	-2
-2	-2	-2	-2

Value Iteration



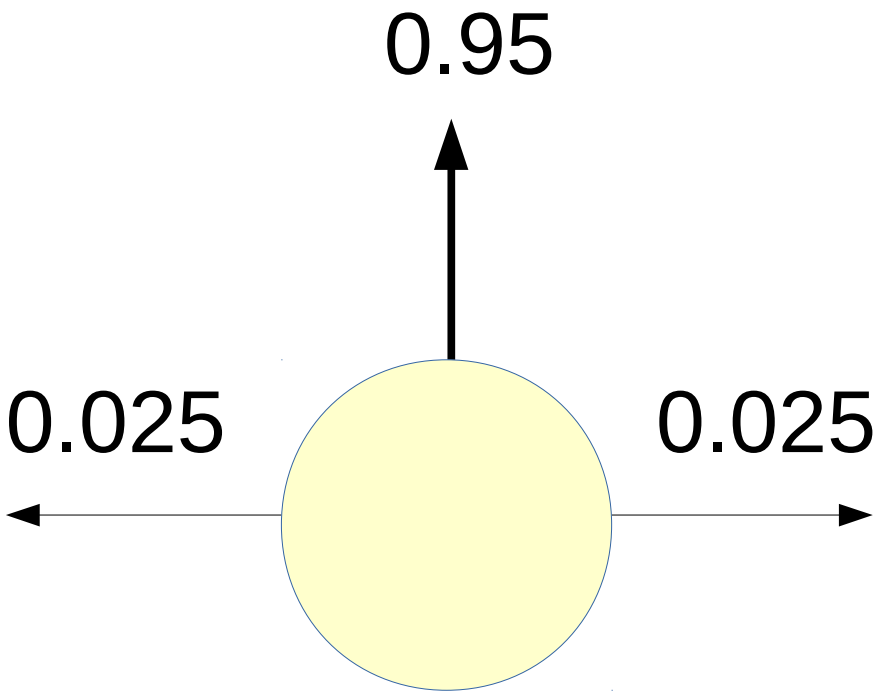
-3	-3	-3	0
-3	-3	-3	-1
-3	-3	-3	-2
-3	-3	-3	-3

Value Iteration

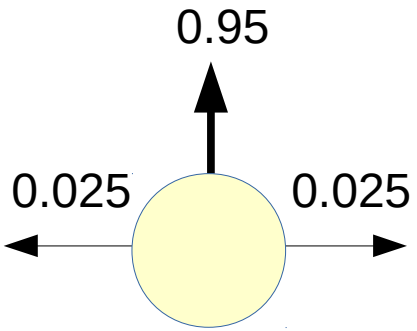


-15	-8	-7	0
-14	-9	-6	-1
-13	-10	-5	-2
-12	-11	-4	-3

Stochastic Model

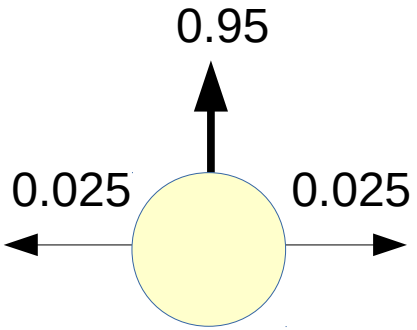


Value Iteration



-19.2	-10.4	-9.3	0
-18.1	-12.1	-8.2	-1.5
-17.0	-13.6	-6.7	-2.9
-15.7	-14.7	-5.1	-4.0

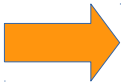
Value Iteration



-19.2	-10.4	-9.3	0
-18.1	-12.1	-8.2	-1.5
-17.0	-13.6	-6.7	-2.9
-15.7	-14.7	-5.1	-4.0

E.g. 13.6:

13.6 =
0.950 x 13.1 +
0.025 x 27.0 +
0.025 x 16.7



16.6 =
0.950 x 16.7 +
0.025 x 13.1 +
0.025 x 15.7

Richard Bellman

Bellman Equation

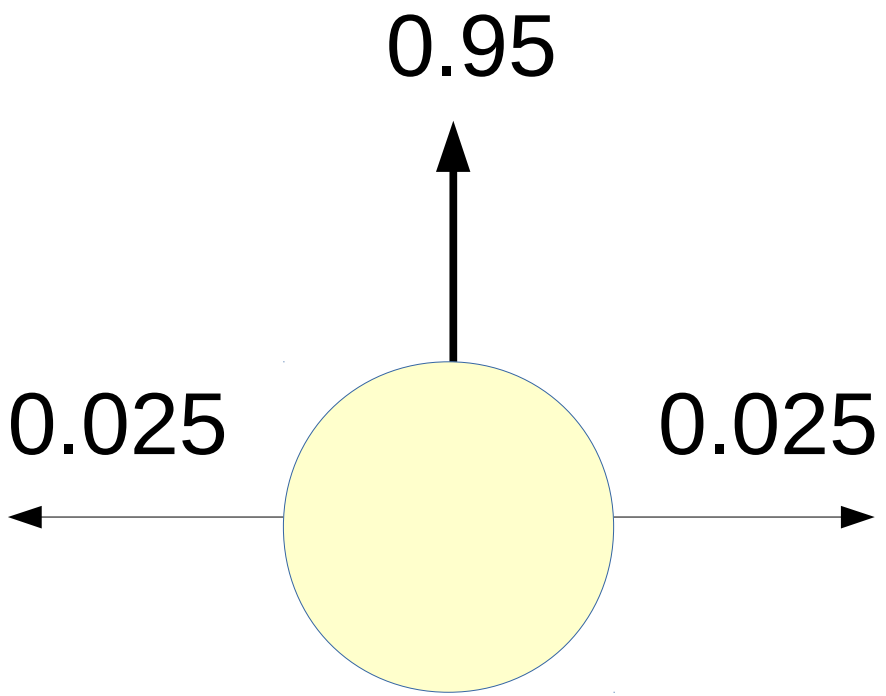
$$V^*(s) = \max_a \left(\mathcal{R}^a_{ss'} + \gamma V^*(s') \right)$$

$$V^*(s) = \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V^*(s') \right)$$

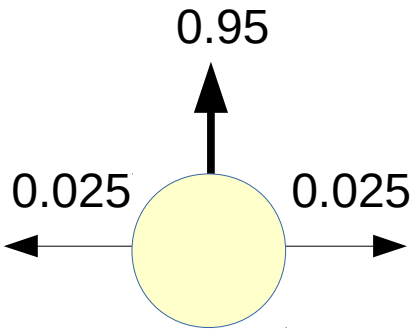
Reinforcement Learning Tools

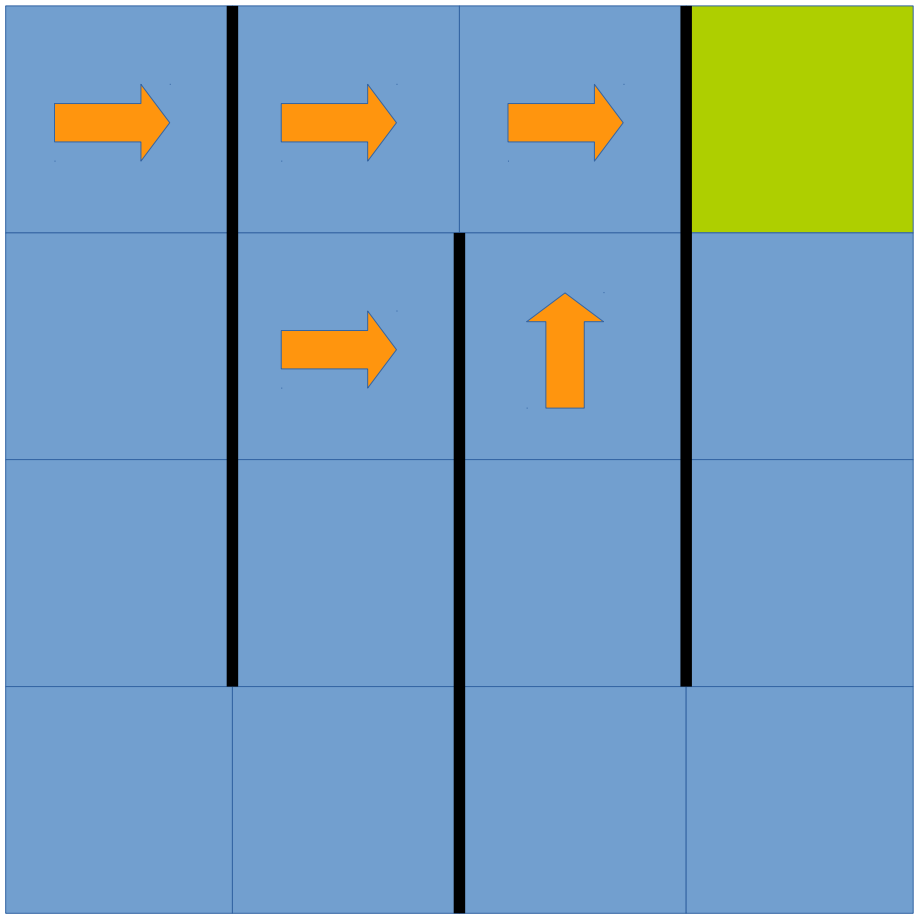
- Dynamic Programming
- Monte Carlo Methods
- Temporal Difference Learning

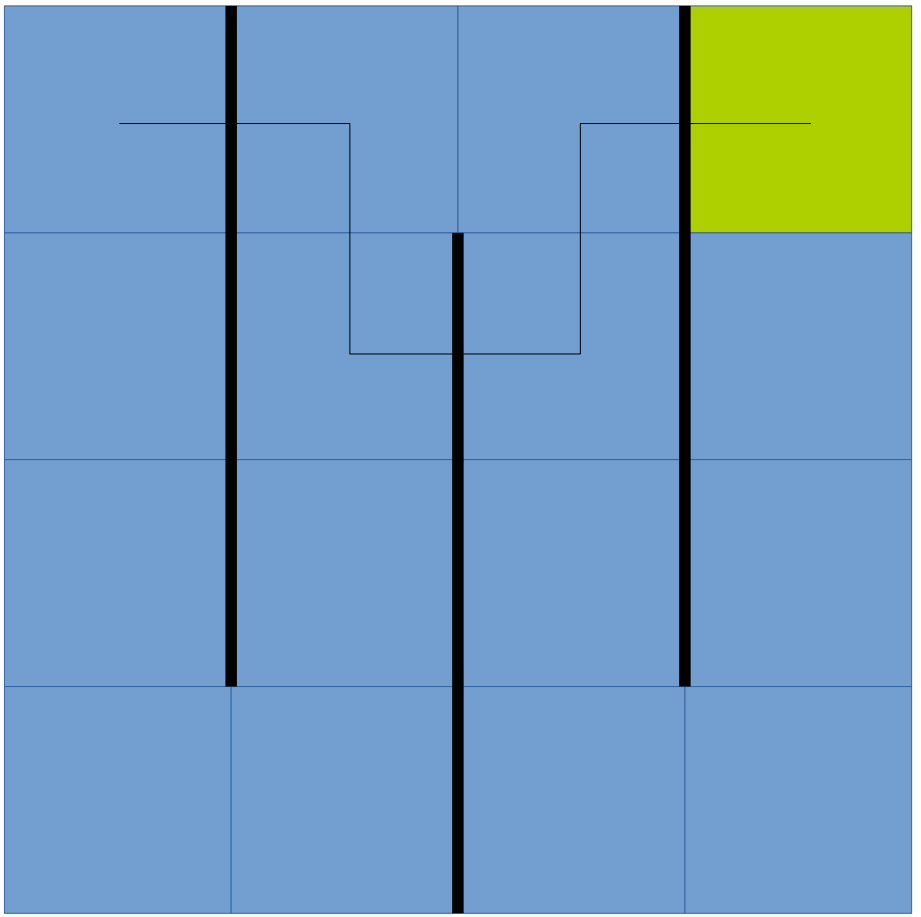
Monte Carlo Methods



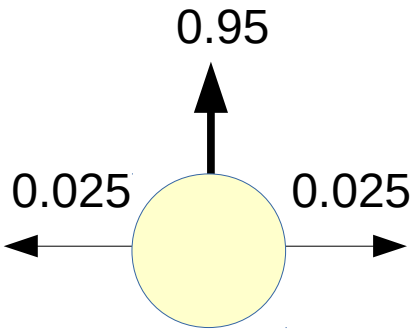
Monte Carlo Methods

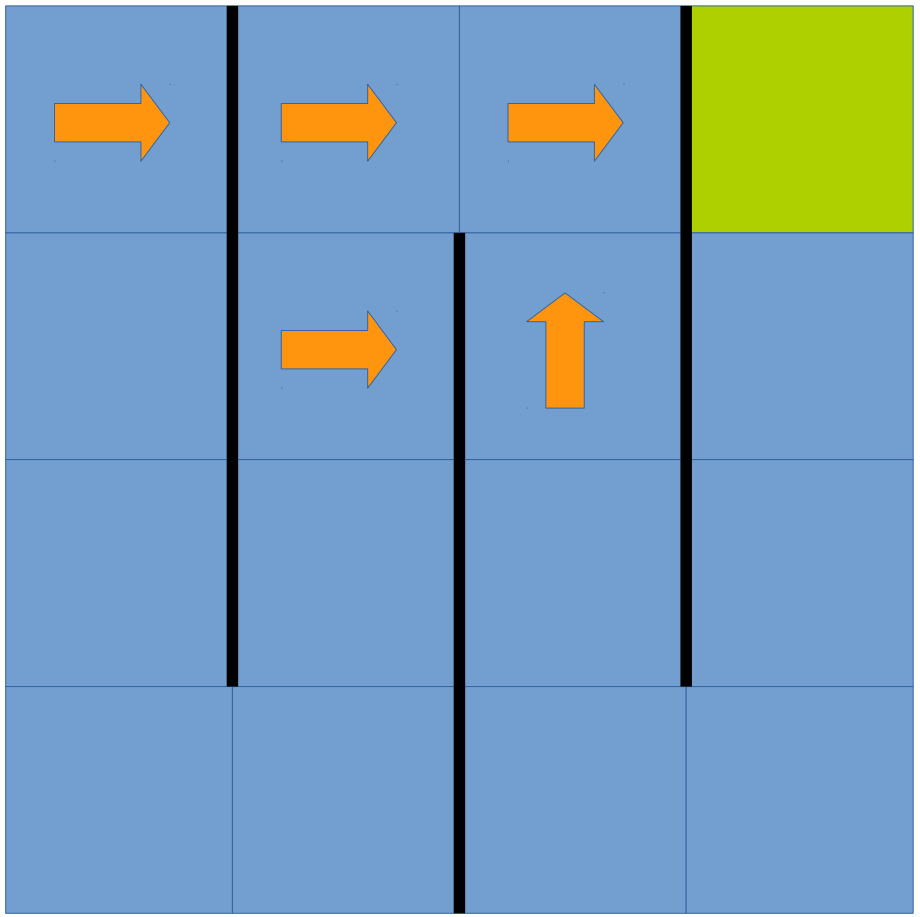


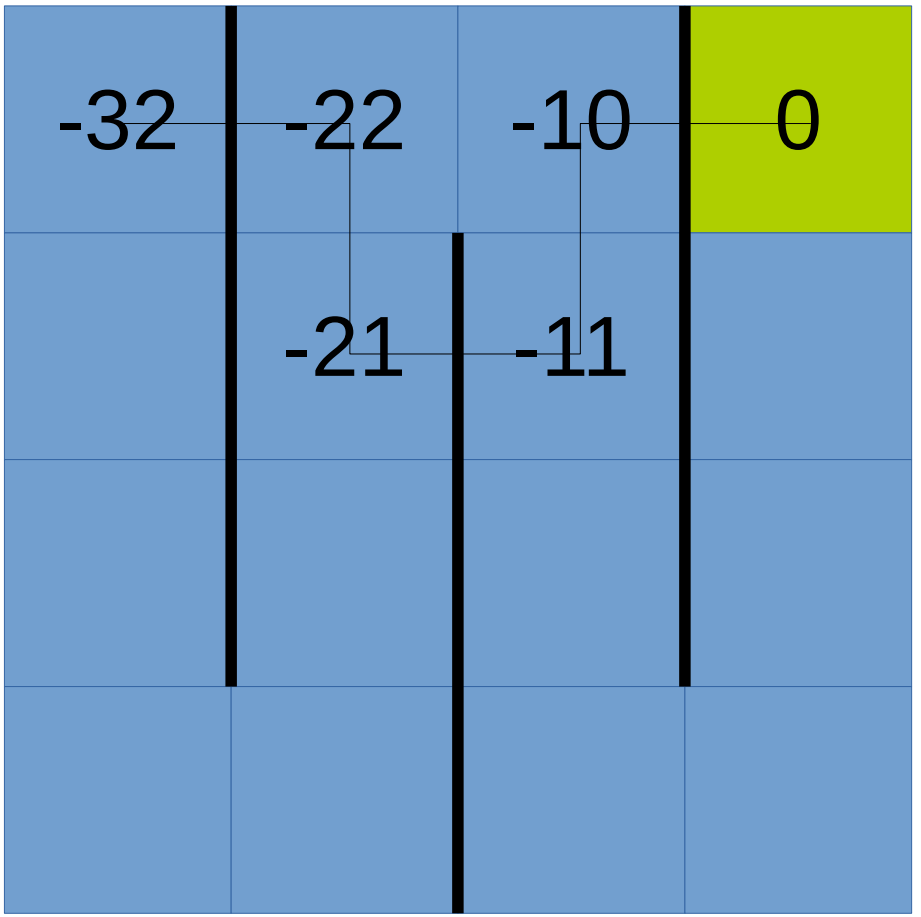




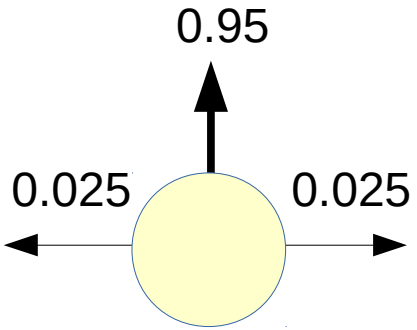
Monte Carlo Methods

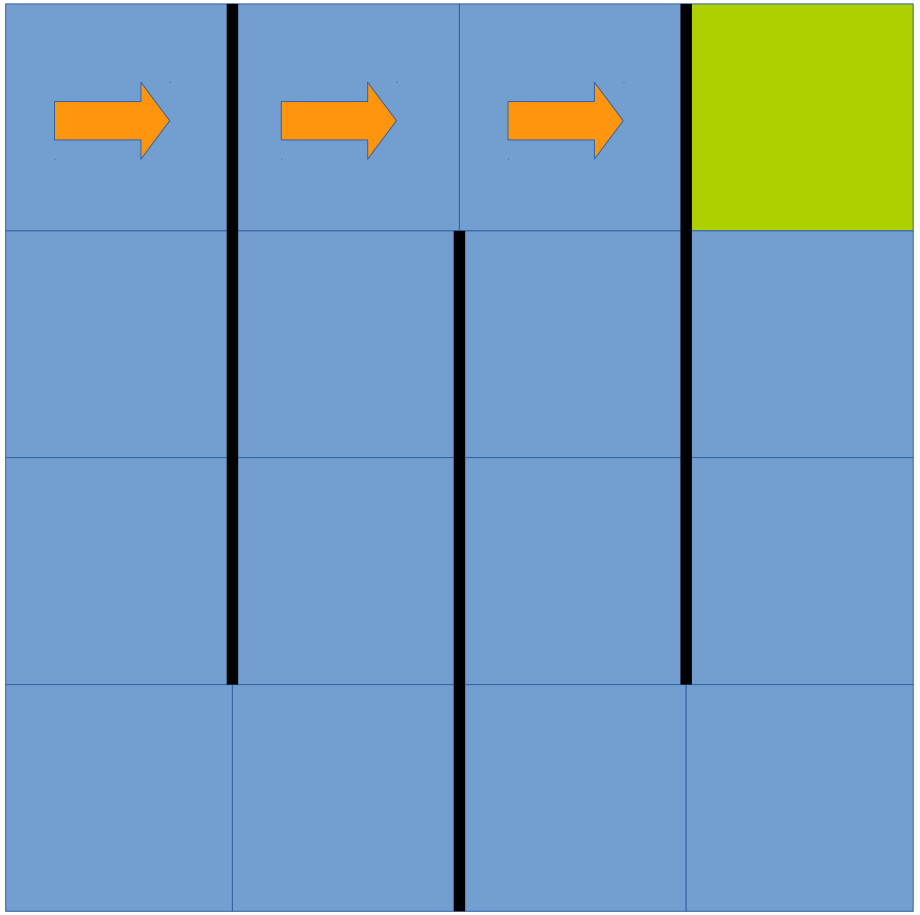


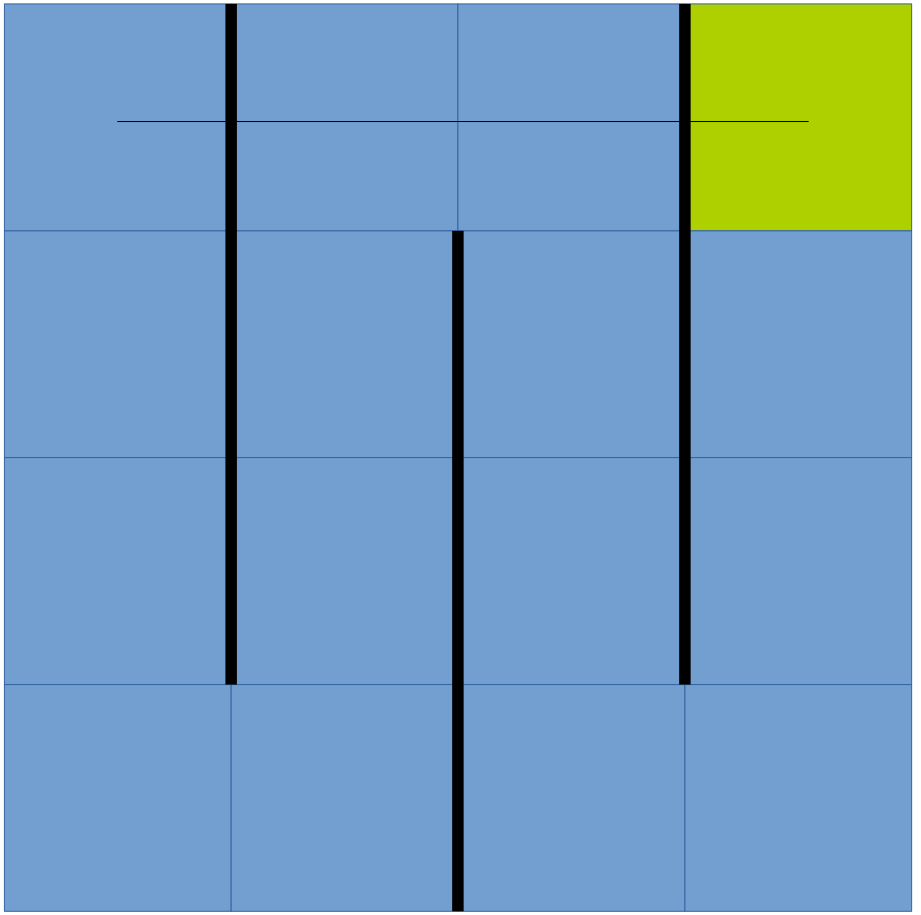




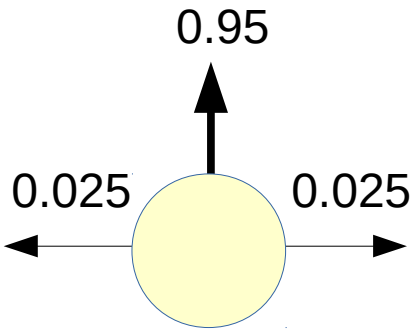
Monte Carlo Methods

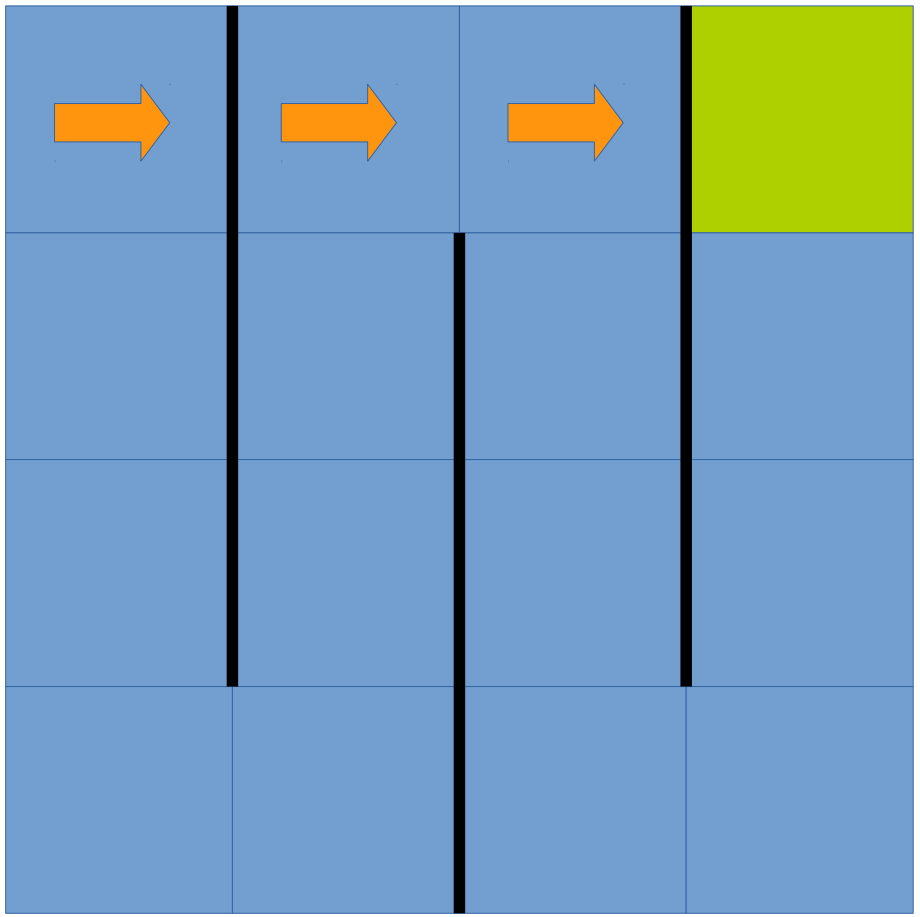


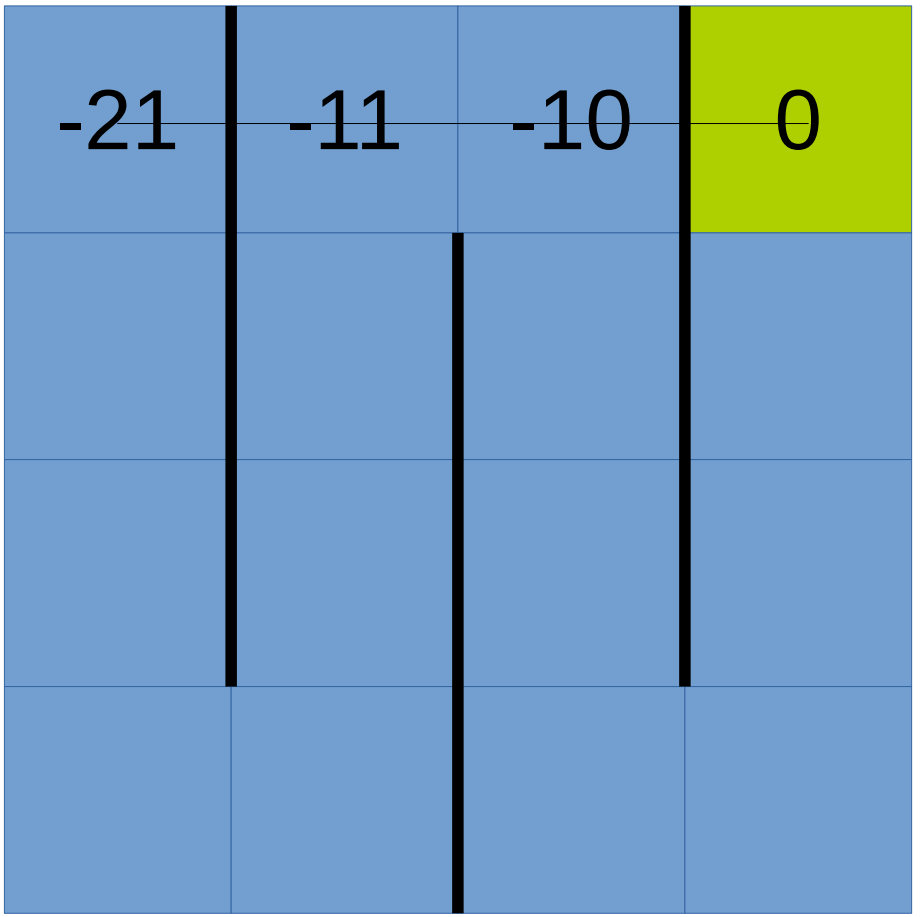




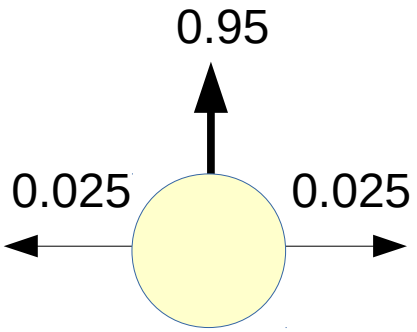
Monte Carlo Methods

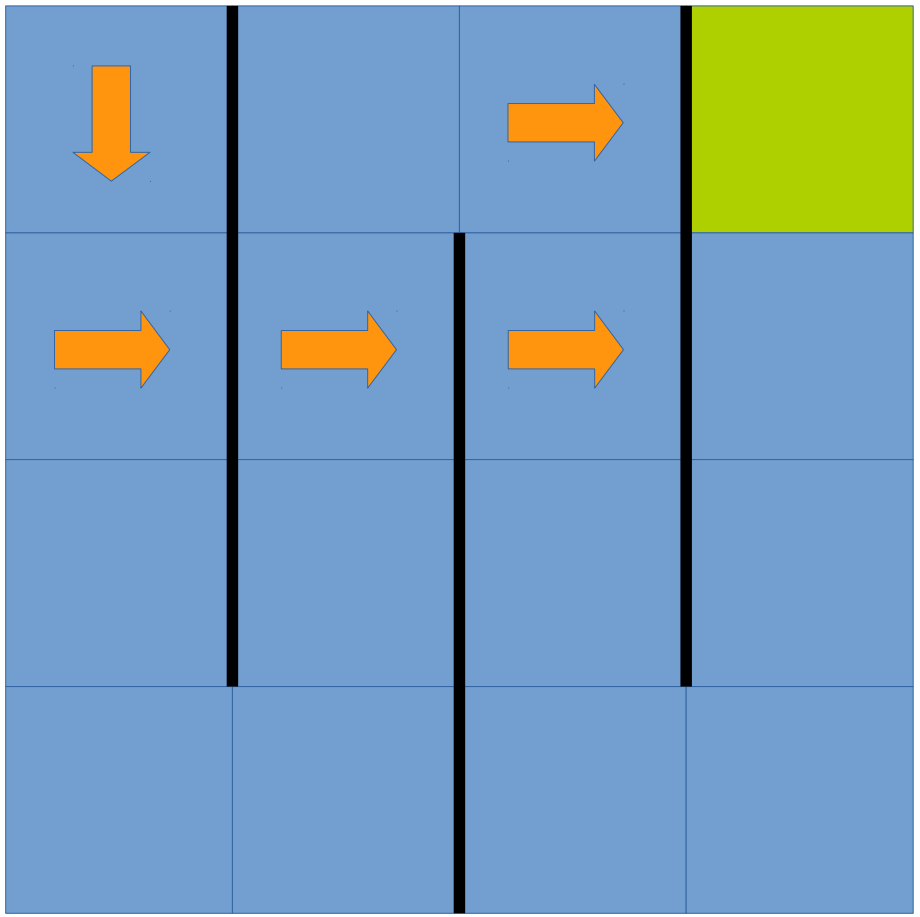


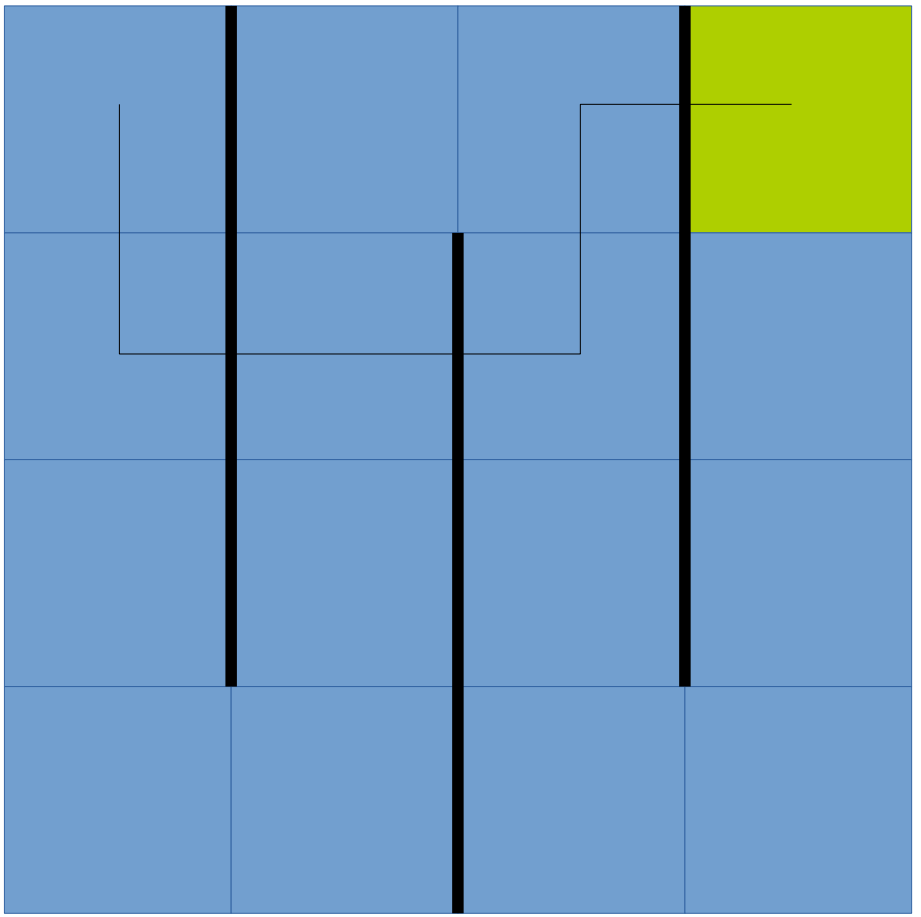




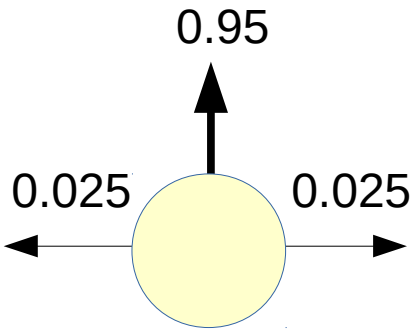
Monte Carlo Methods

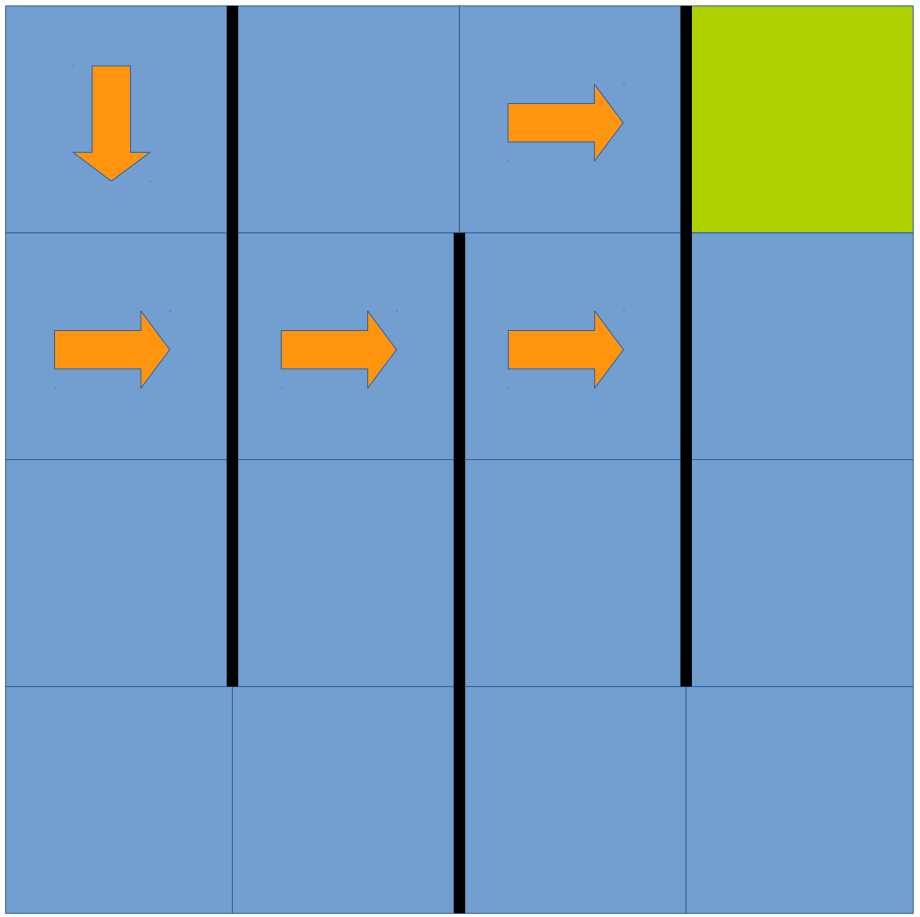


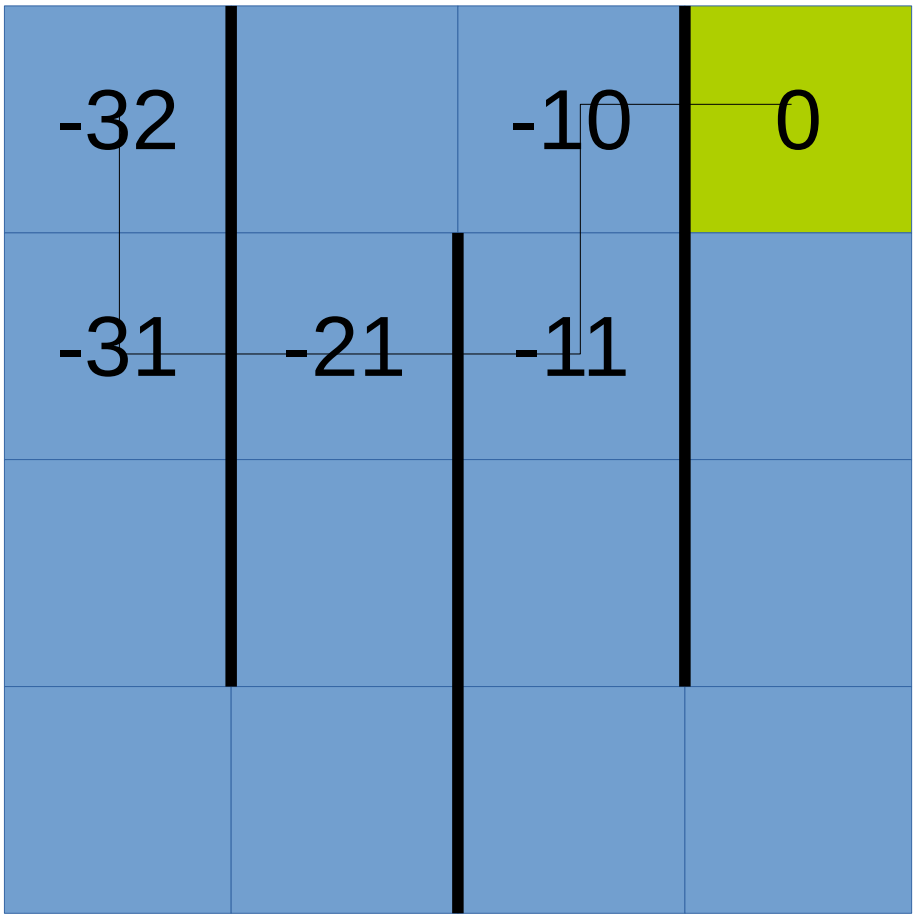




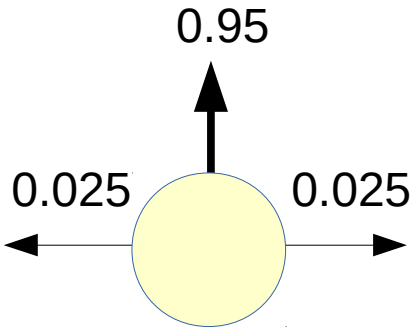
Monte Carlo Methods

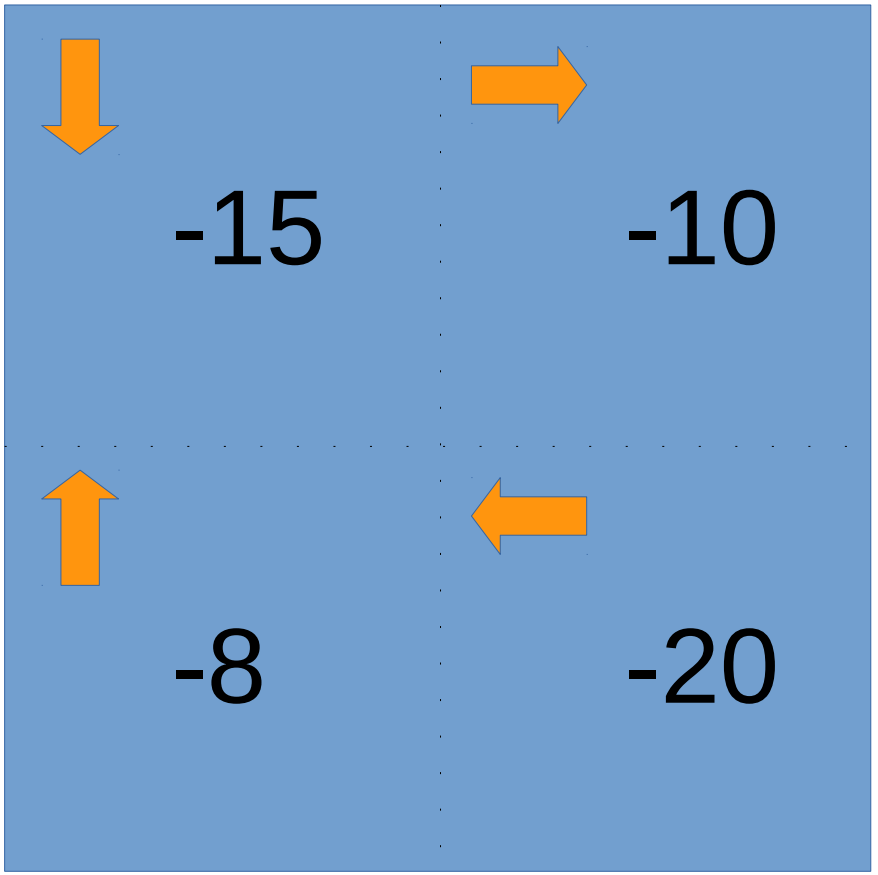






Q-Value

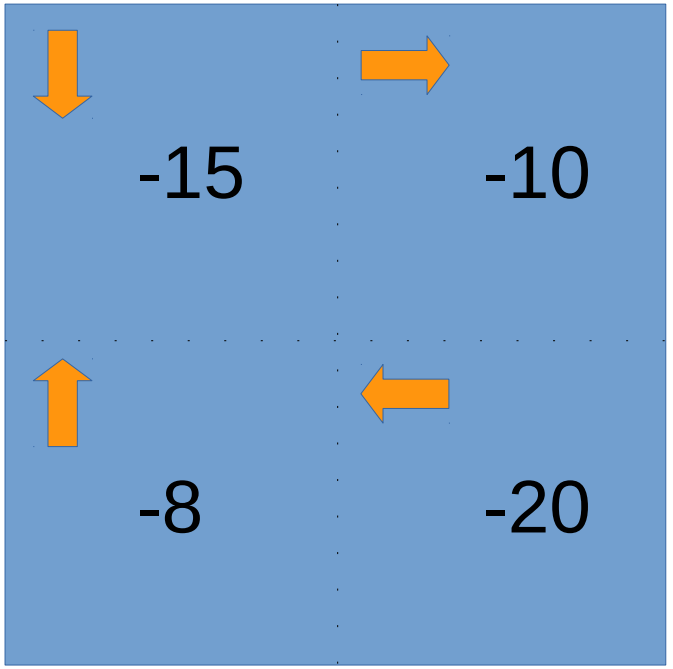




Bellman Equation

$$V^*(s) = \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V^*(s') \right)$$

$$Q^*(s,a) = \sum_{s'} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma \max_a Q^*(s',a') \right)$$



Learning Rate

- We do not replace an old Q value with a new one.
- We update at a designed learning rate.
- Learning rate too small: slow to converge.
- Learning rate too large: unstable.
- Will Dabney PhD Thesis:
  Adaptive Step-Sizes for Reinforcement Learning.

Reinforcement Learning Tools

- Dynamic Programming
- Monte Carlo Methods
- Temporal Difference Learning

Richard Sutton

Temporal Difference Learning

- **Dynamic Programming:**
  Learn a guess from other guesses *(Bootstrapping).*

- **Monte Carlo Methods:**
  Learn without knowing model.

Temporal Difference Learning

Temporal Difference:

- Learn a guess from other guesses *(Bootstrapping).*
- Learn without knowing model.
- Works with longer episodes than Monte Carlo methods.

Temporal Difference Learning

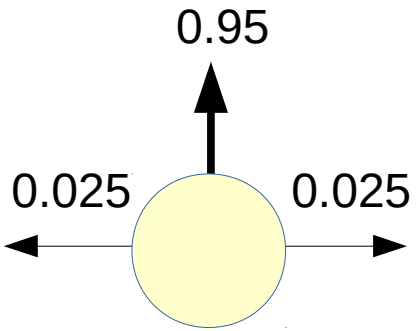
Monte Carlo Methods:

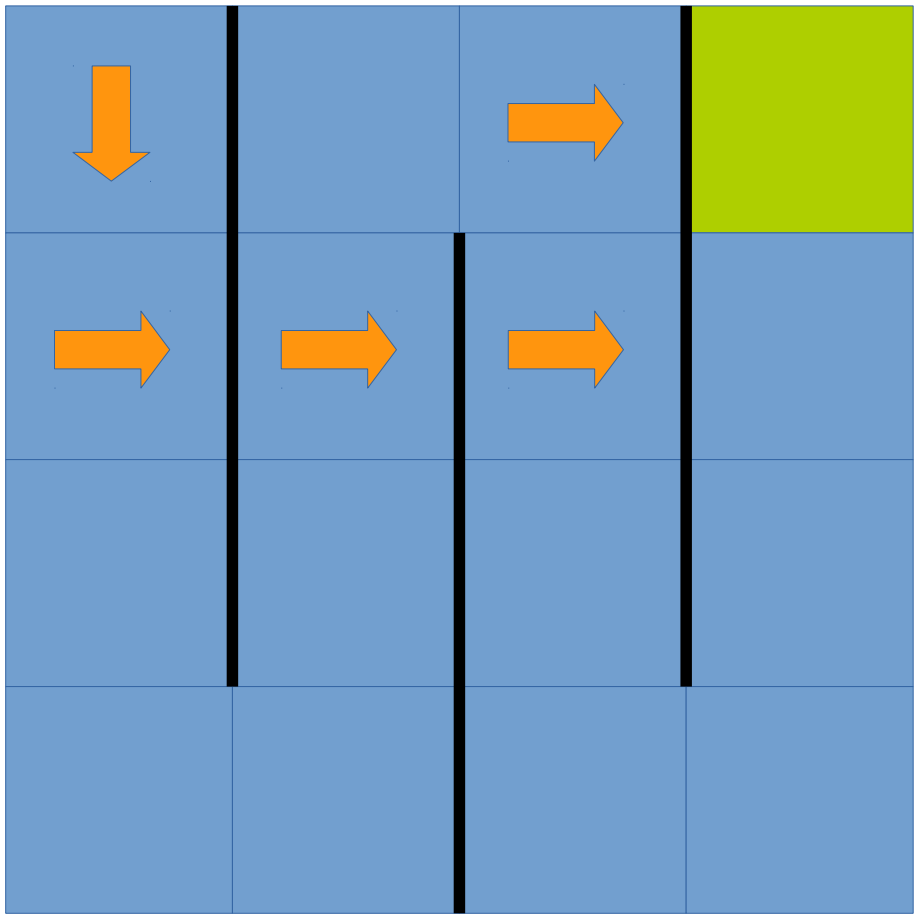
- First run through whole episode.
- Update states at end.

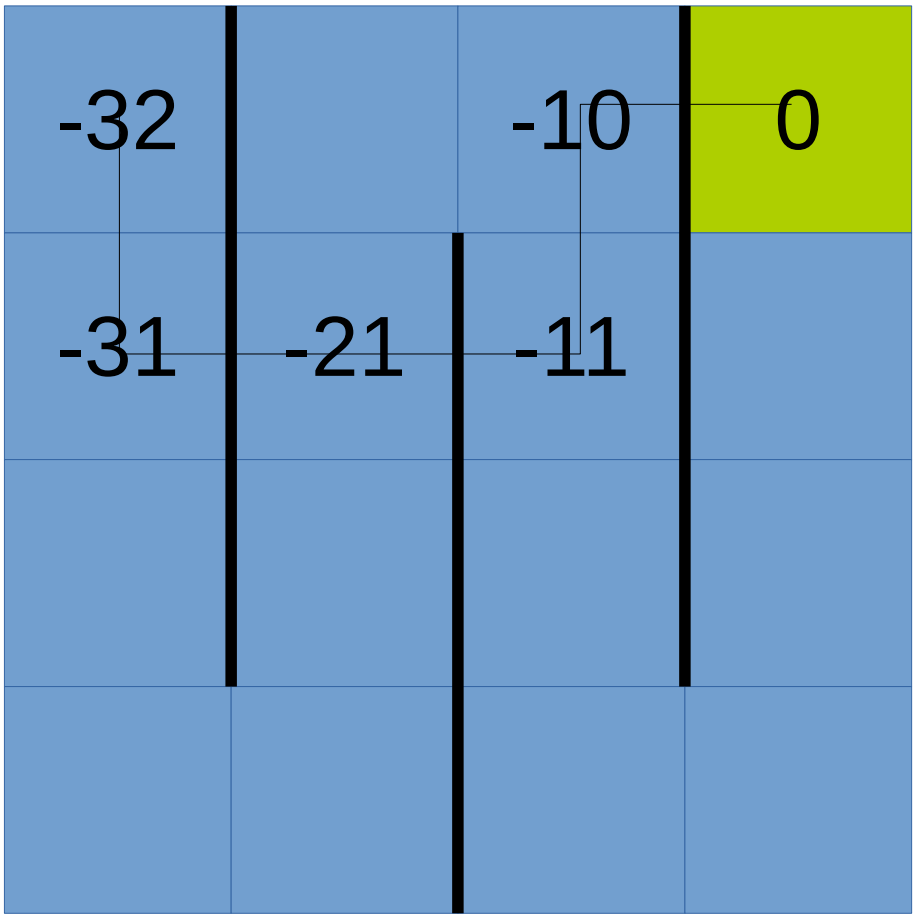
Temporal Difference Learning:

- Update state at each step using earlier guesses.

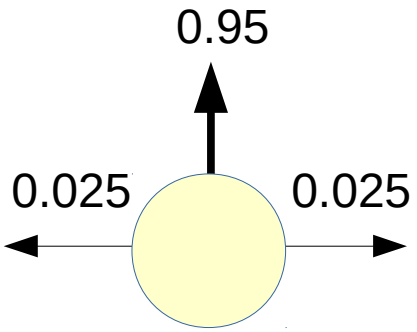
Monte Carlo Methods

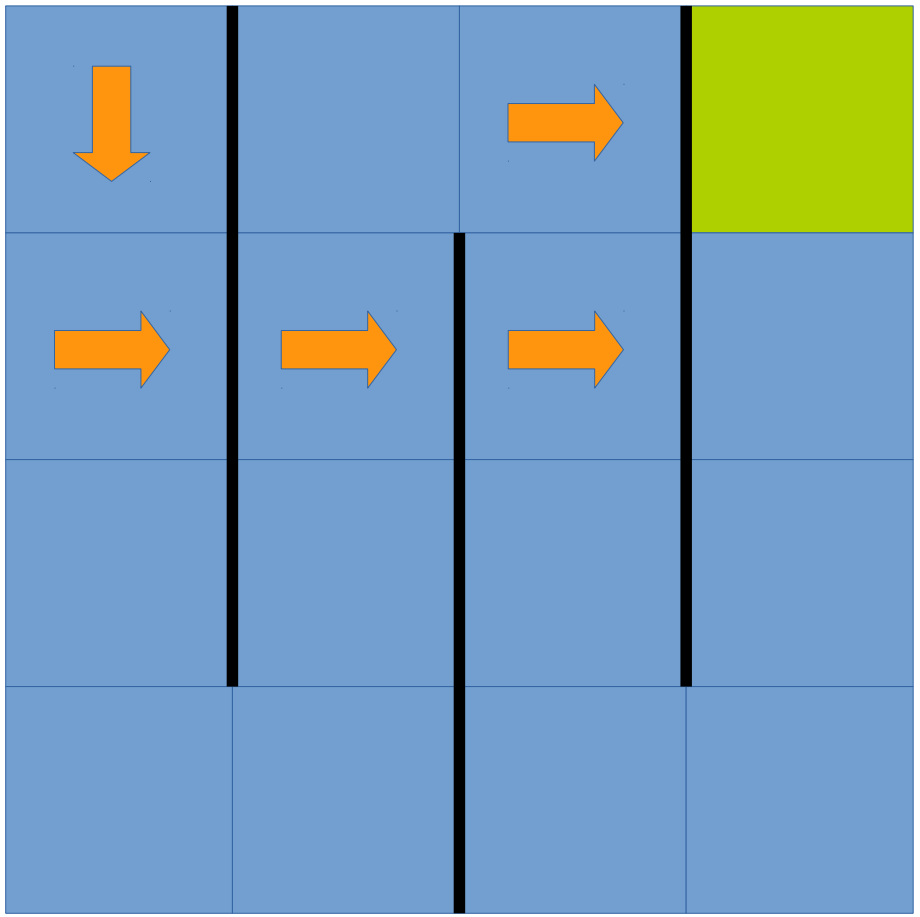


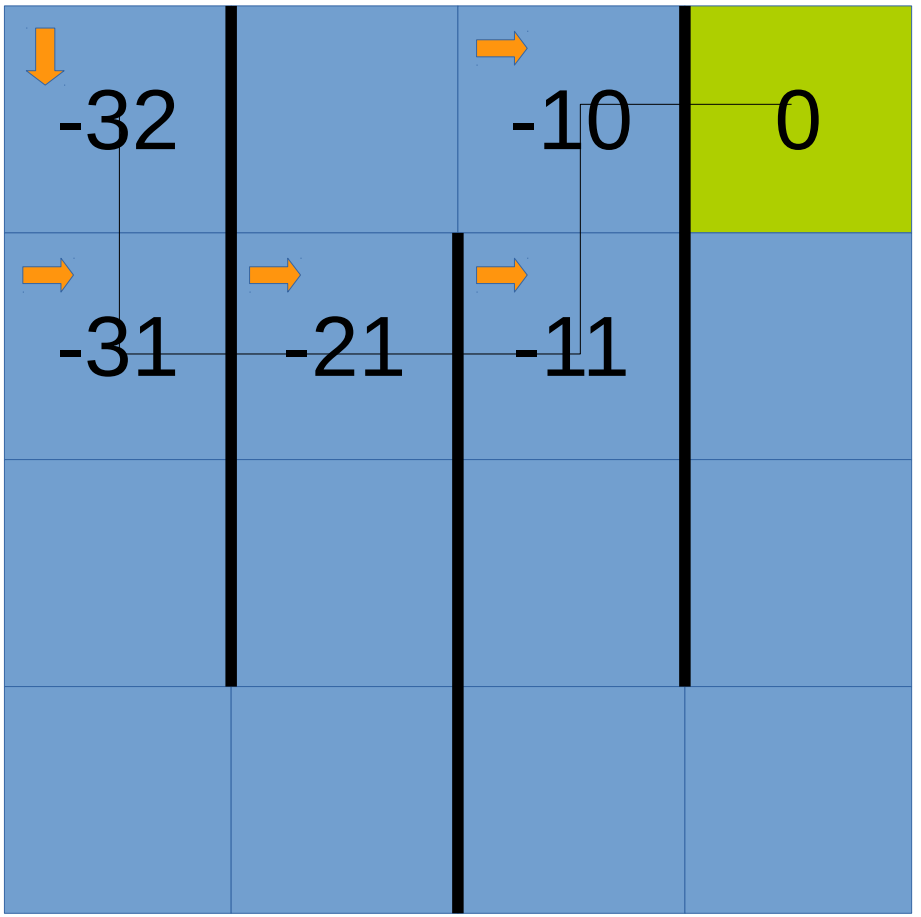




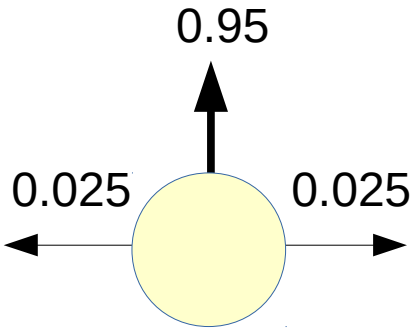
Monte Carlo Methods

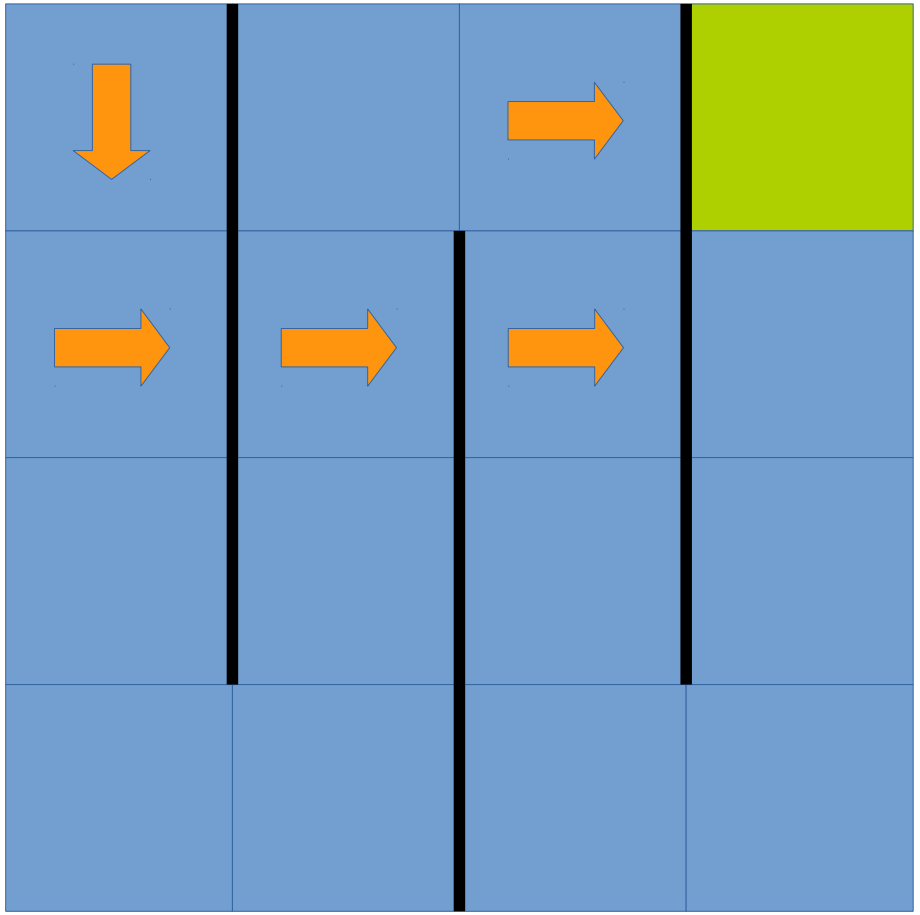


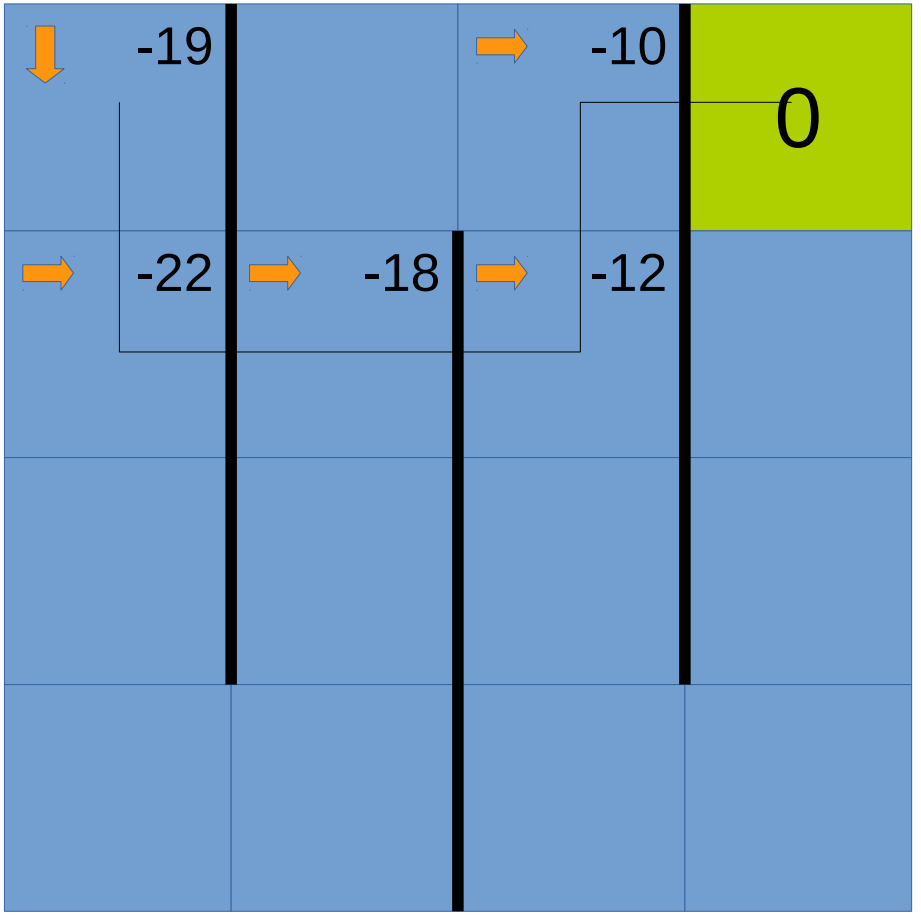




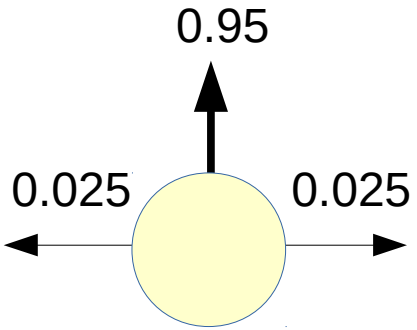
Temporal Difference

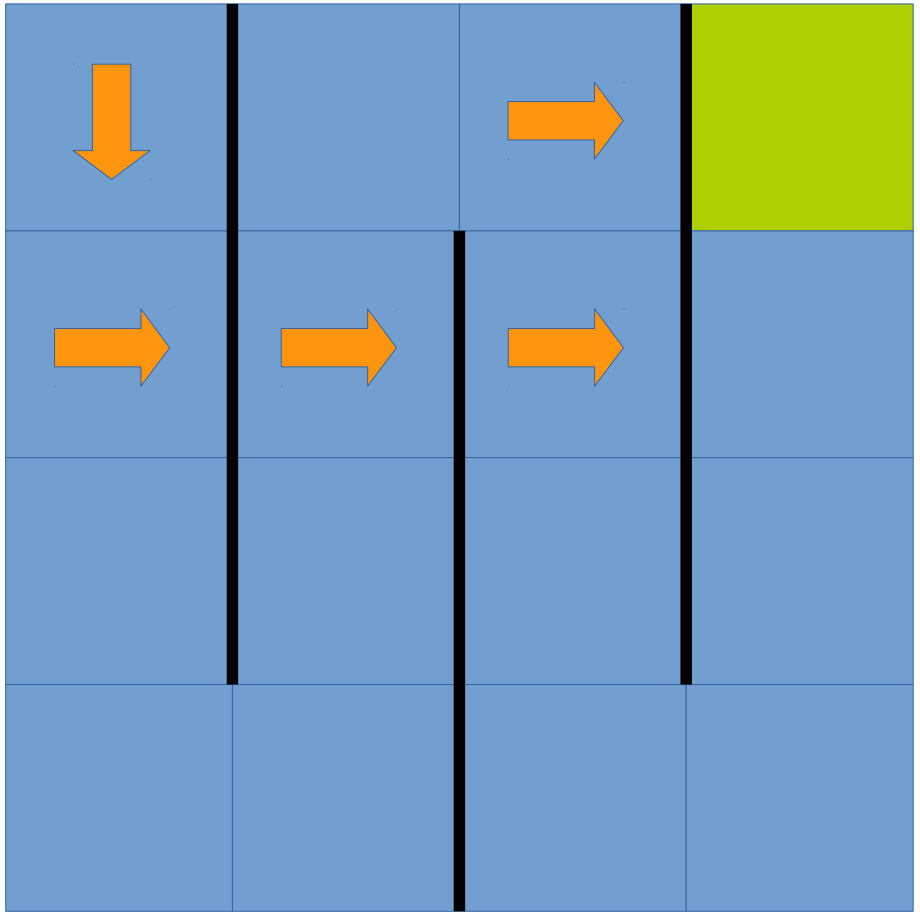


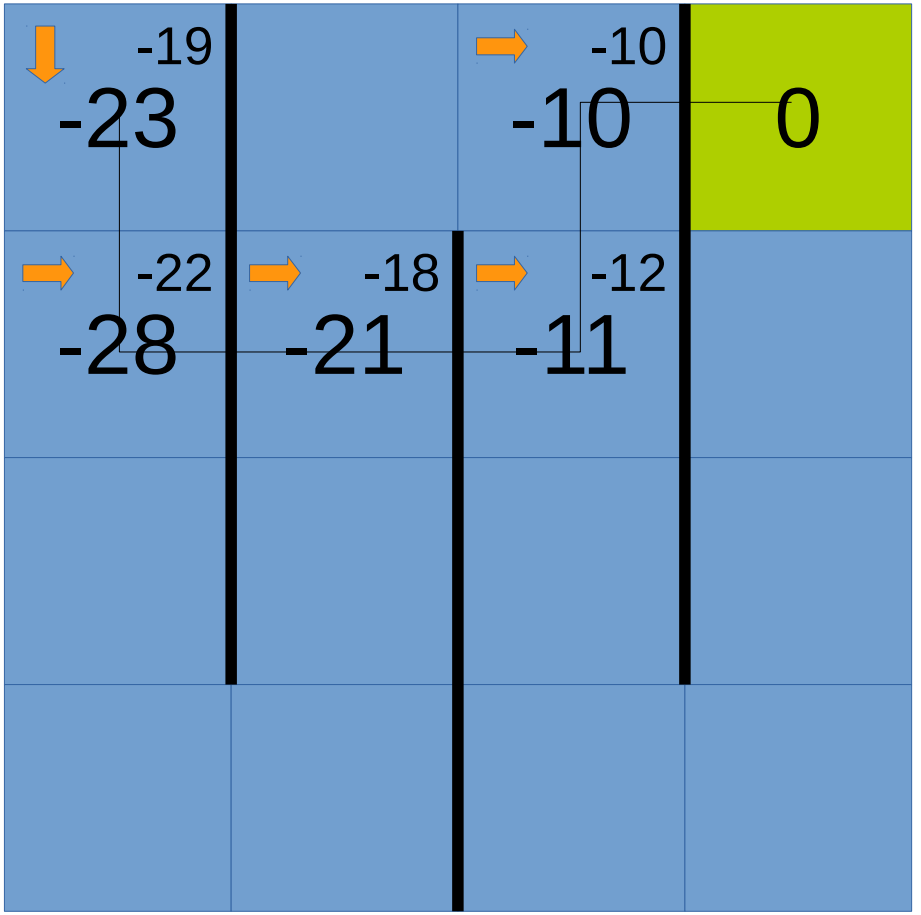




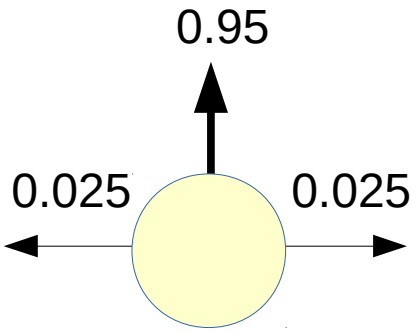
Temporal Difference







Temporal Difference



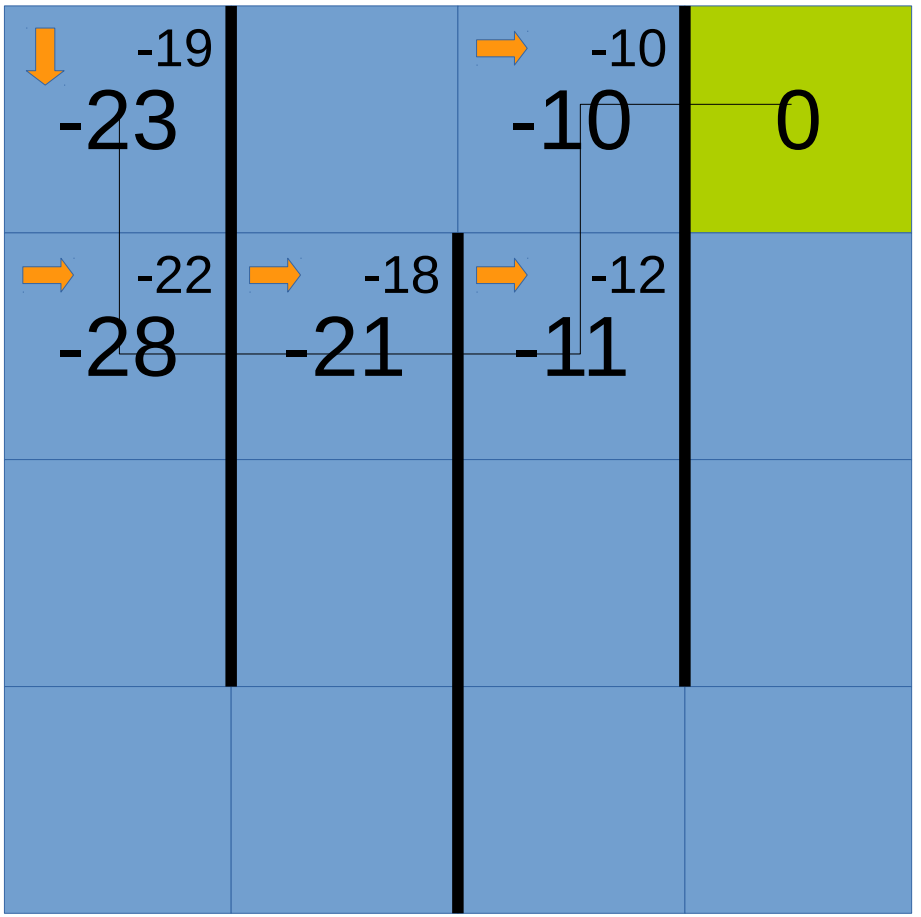
23 = 1 + 22

28 = 10 + 18

21 = 10 + 11

11 = 1 + 10

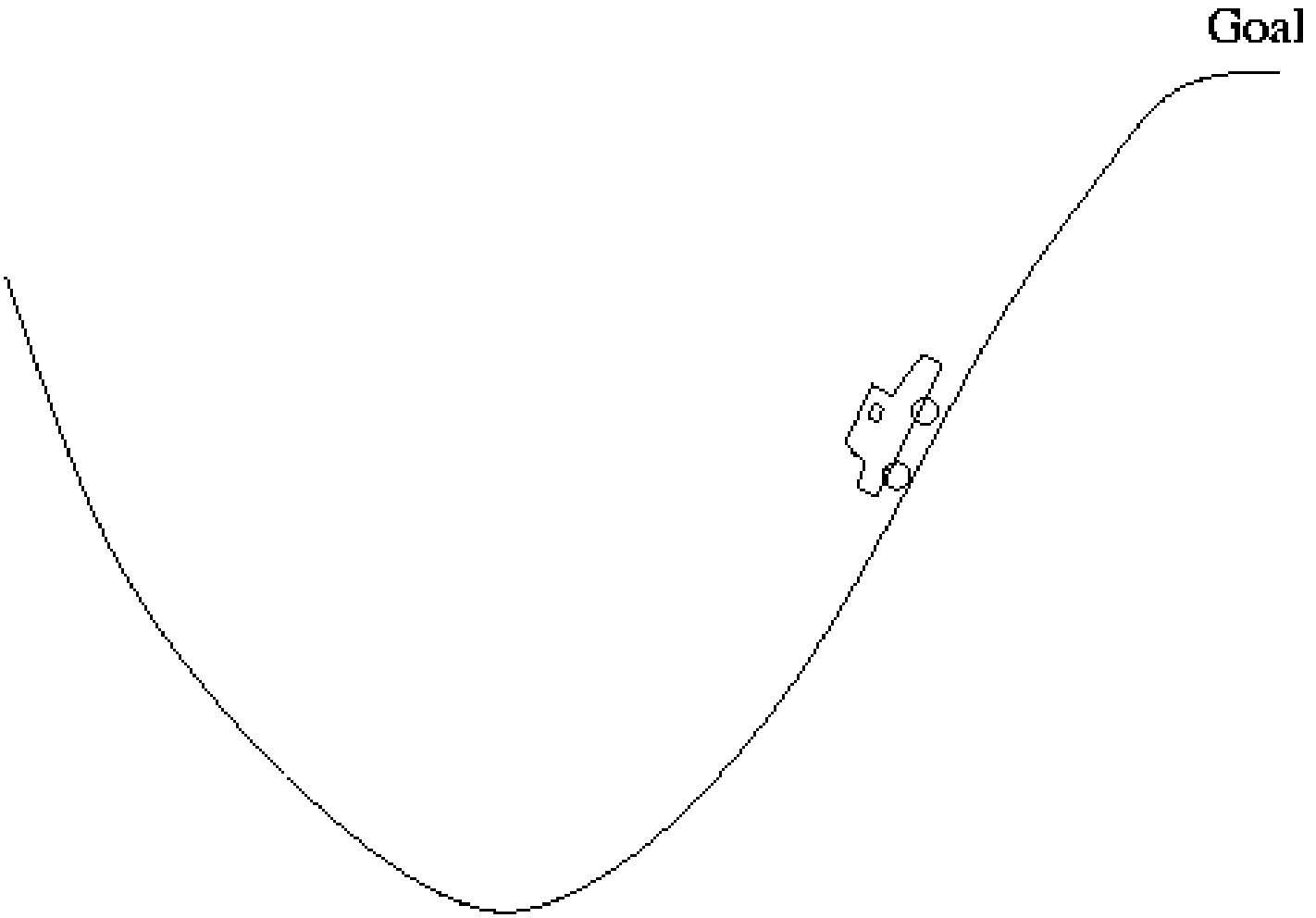
10 = 10 + 0



Function Approximation

- Most problems have large state space.
- We can generally design an approximation for the state space.
- Choosing the correct approximation has a large influence on system performance.

Mountain Car Problem



Mountain Car Problem

- Car cannot make it to top.
- Can can swing back and forth to gain momentum.
- We know x and $\dot{x}$.
- x and $\dot{x}$ give an infinite state space.
- Random – may get to top in 1000 steps.
- Optimal – may get to top in 102 steps.

Function Approximation

- We can partition state space in 200 x 200 grid.
- Coarse coding – different ways of partitioning state space.
- We can approximate $V = w^T f$
- E.g. $f = (x \quad \dot{x} \quad \textit{height} \quad \dot{x}^2)^T$
- We can estimate w to solve problem.

Problems with Reinforcement Learning

Policy sometimes gets worse:

- Safe Reinforcement Learning (Phil Thomas) guarantees an improved policy over the current policy.

Very specific to training task:

- Learning Parameterized Skills
  Bruno Castro da Silva PhD Thesis

Checkers

- Arthur Samuel (IBM) 1959

TD-Gammon

- Neural networks and temporal difference.
- Current programs play better than human experts.
- Expert work in input selection.

Deep Learning: Atari

- Inputs: score and pixels.
- Deep learning used to discover features.
- Some games played at super-human level.
- Some games played at mediocre level.