Planning for Autonomous Robots

Part 3: *Planning Under Uncertainty*

Professor Nick Hawes
GOALS Research Group
Oxford Robotics Institute, Department of Engineering Science, University of Oxford

Pembroke College

Overview

- Classical Planning to Replanning
- Planning with Non-Deterministic Models
- Planning with Probabilistic Models
  - Value Iteration
  - Beyond Value Iteration

Classical Planning relies on:

Classical Planning relies on:

a **factored representation** — one that represents the world by a collection of variables, called a **state**

Classical Planning relies on:

a **factored representation** — one that represents the world by a collection of variables, called a **state**

an **action representation** — that lifts reasoning to a subset of first-order logic, compactly expressing a range of **successor states**

Classical Planning

fully observable

deterministic

static environments

Classical Planning is insufficient…

Classical Planning is insufficient…

It has nothing to say about **execution**

Classical Planning is insufficient…

It has nothing to say about **execution**

The outcome of an action is not always what the **model** describes

Classical Planning is insufficient…

It has nothing to say about **execution**

The outcome of an action is not always what the **model** describes

- Unintended outcomes
- Exogenous events
- Inherent uncertainty

Classical Planning is insufficient…

It has nothing to say about **execution**

The outcome of an action is not always what the **model** describes

- Unintended outcomes
- Exogenous events
- Inherent uncertainty

The domain **model** does not adequately capture the required behaviour

Classical Planning is insufficient…

It has nothing to say about **execution**

The outcome of an action is not always what the **model** describes

- Unintended outcomes
- Exogenous events
- Inherent uncertainty

The domain **model** does not adequately capture the required behaviour

- Time and durative actions
- Other numeric state contents
- Inherent uncertainty
- Open world
- Inaccurate abstractions

Classical Planning is insufficient…

It has nothing to say about **execution**

The outcome of an action is not always what the **model** describes

- Unintended outcomes
- Exogenous events
- Inherent uncertainty

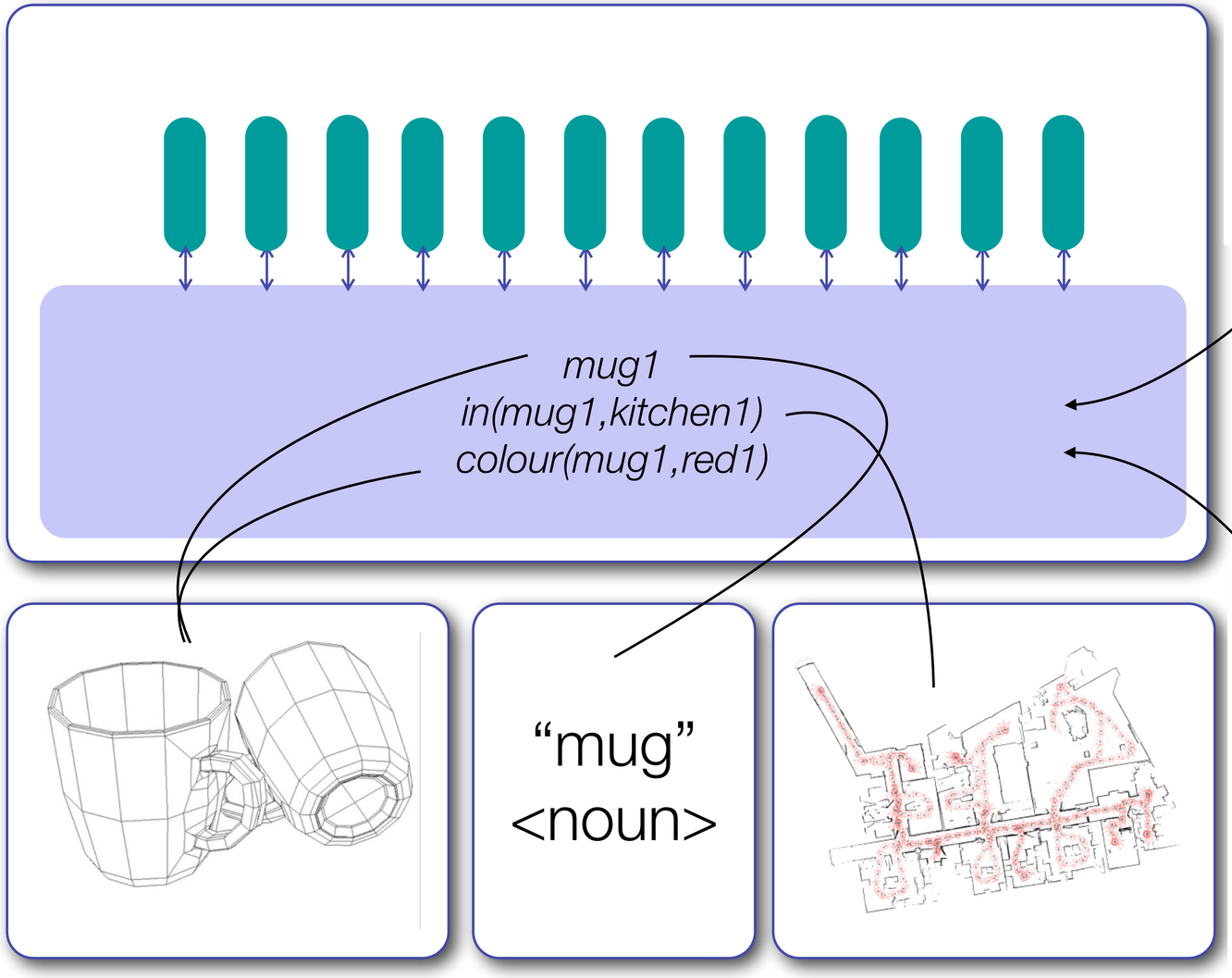
Classical Planning is insufficient…

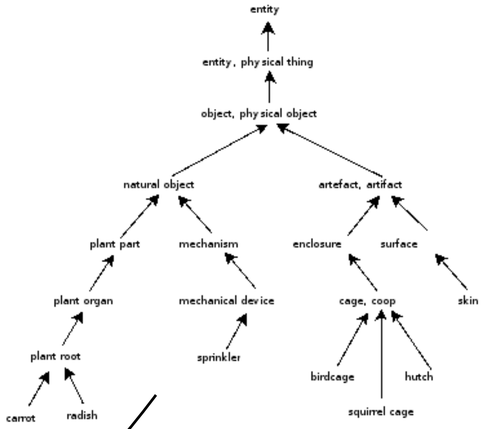
It has nothing to say about **execution**

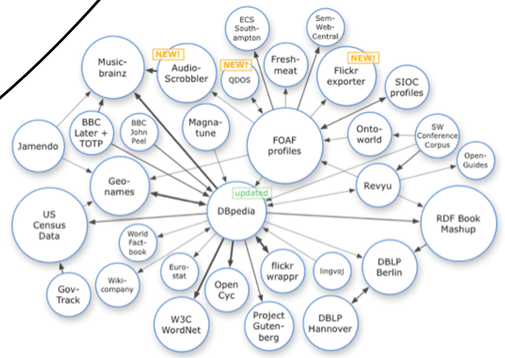
The outcome of an action is not always what the **model** describes

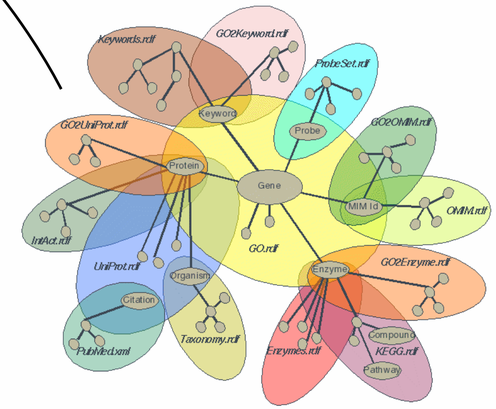
- Unintended outcomes
- Exogenous events
- Inherent uncertainty

Interleaving planning and execution


mug1
in(mug1,kitchen1)
colour(mug1,red1)
“mug”
<noun>

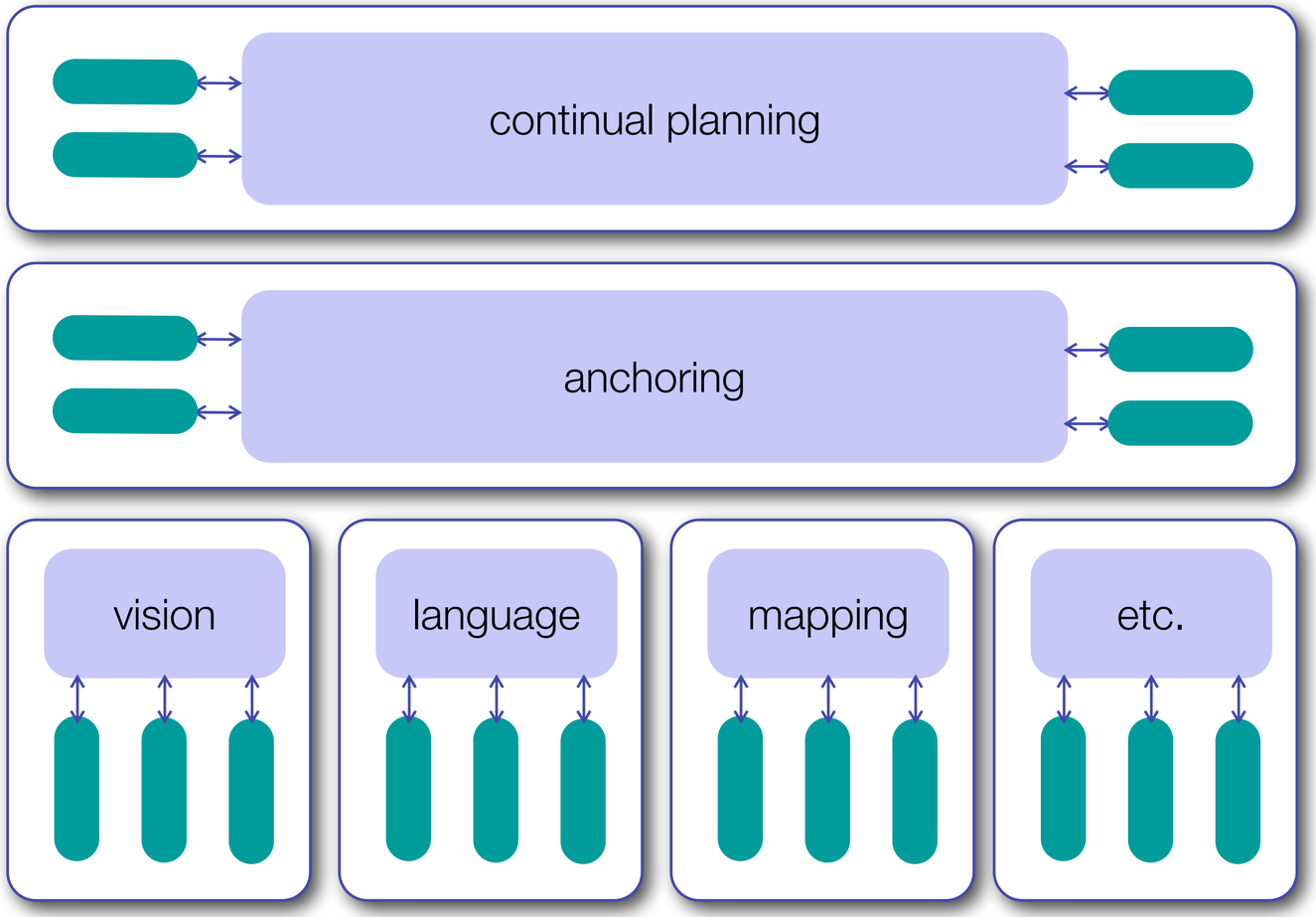

entity
entity, physical thing
object, physical object
natural object
artefact, artifact
plant part
mechanism
enclosure
surface
plant organ
mechanical device
cage, coop
skin
plant root
sprinkler
birdcage
hutch
carrot
radish
squirrel cage


Music-brainz
Audio-Scrobbler
QDOS
Fresh-meat
Flickr exporter
SIOC profiles
BBC Later + TOTP
BBC John Peel
Magna-tune
FOAF profiles
Onto-world
SW Conference Corpus
Open-Guides
Jamendo
Geo-names
DBpedia
Revyu
US Census Data
RDF Book Mashup
World Fact-book
Euro-stat
flickr wrappr
lingvoj
DBLP Berlin
Gov-Track
Wiki-company
Open Cyc
W3C WordNet
Project Guten-berg
DBLP Hannover


Keywords.rdf
GO2Keyword.rdf
ProbeSet.rdf
GO2UniProt.rdf
Keyword
Probe
GO2OMIM.rdf
Protein
Gene
MIM Id
OMIM.rdf
IntAct.rdf
GO.rdf
UniProt.rdf
Organism
Enzyme
GO2Enzyme.rdf
Citation
Taxonomy.rdf
Enzymes.rdf
Compound
KEGG.rdf
PubMed.xml
Pathway

N. Hawes, A. Sloman, J. Wyatt, M. Zillich, H. Jacobsson, G.-J. Kruijff, M. Brenner, G. Berginc, and D. Skocaj. *Towards an integrated robot with multiple cognitive functions*. In AAAI'07.

M. Brenner, N. Hawes, and J Kelleher. *Mediating Between Qualitative and Quantitative Representations for Task-Orientated Human-Robot Interaction*. In IJCAI'07.



Interleaving planning and execution

fully observable

deterministic

static environments

Interleaving planning and execution

fully observable

deterministic

static environments

But?

Interleaving planning and execution

fully observable

deterministic

static environments

But?

outcomes are not considered in advance

Planning with non-deterministic models

The outcome of an action is not always what the **model** describes

- Unintended outcomes
- Exogenous events
- Inherent uncertainty

solution: Extend the basic action definition to specify **multiple outcomes**

Planning with non-deterministic models

Actions define **multiple outcomes**

Planning with non-deterministic models

Actions define **multiple outcomes**

This creates some new problems!

Planning with non-deterministic models

Actions define **multiple outcomes**

This creates some new problems!

What does a **solution** to planning problem look like?

Planning with non-deterministic models

Actions define **multiple outcomes**

This creates some new problems!

What does a **solution** to planning problem look like?

What are **valid solutions** to a planning problem?

From plans to policies

A single action can result in one of many possible states

From plans to policies

A single action can result in one of many possible states

A **plan** is a sequence of actions, and doesn't take into account this non-determinism

From plans to policies

A single action can result in one of many possible states

A **plan** is a sequence of actions, and doesn't take into account this non-determinism

We need a structure which allows the actor to **look up the next action given the resulting state**

From plans to policies

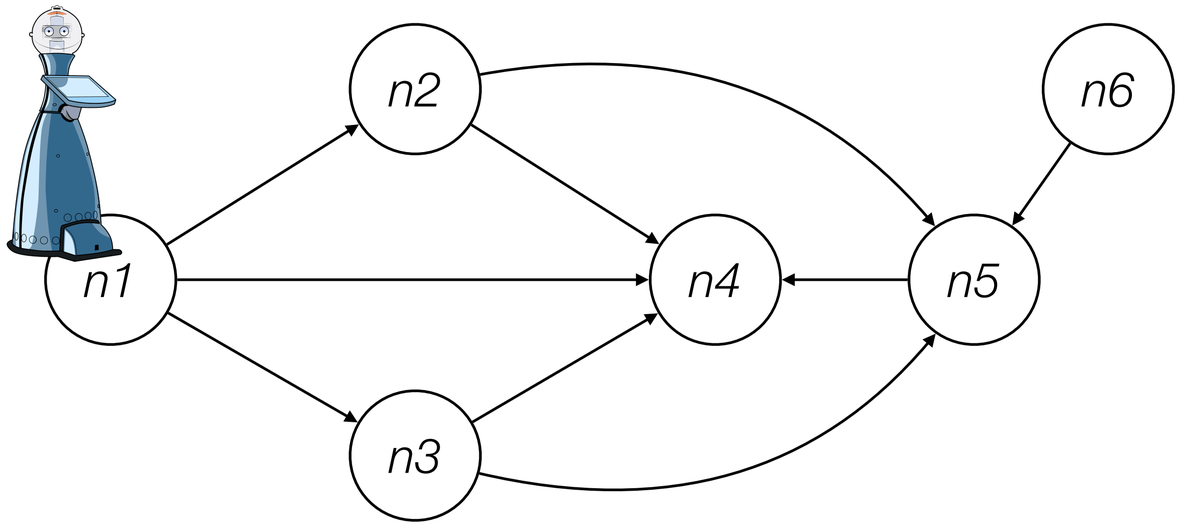
A single action can result in one of many possible states

A **plan** is a sequence of actions, and doesn't take into account this non-determinism

We need a structure which allows the actor to **look up the next action given the resulting state**

Most general structure: a **policy**

Navigation example



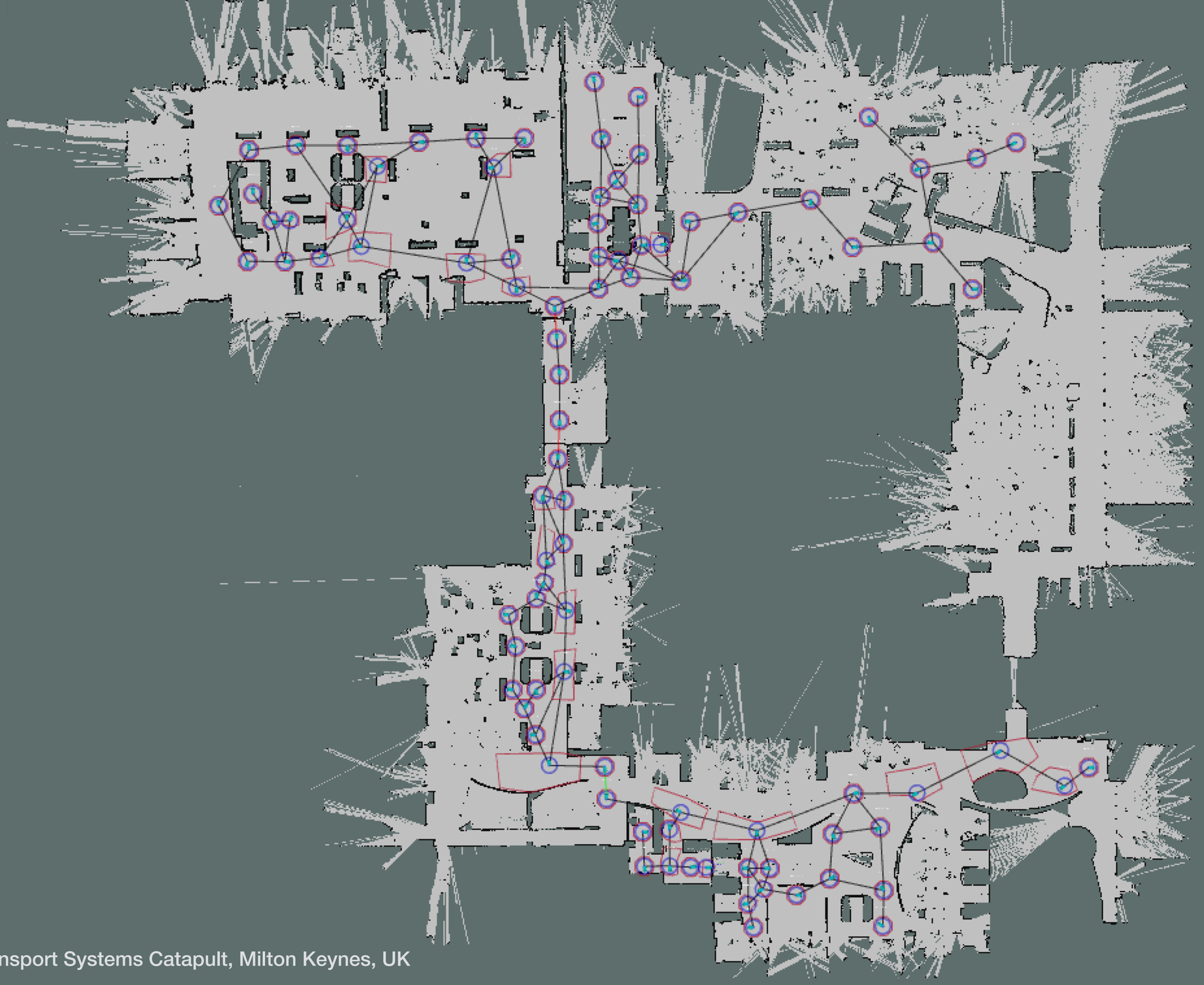
Transport Systems Catapult, Milton Keynes, UK

120 days of *autonomous* behaviour

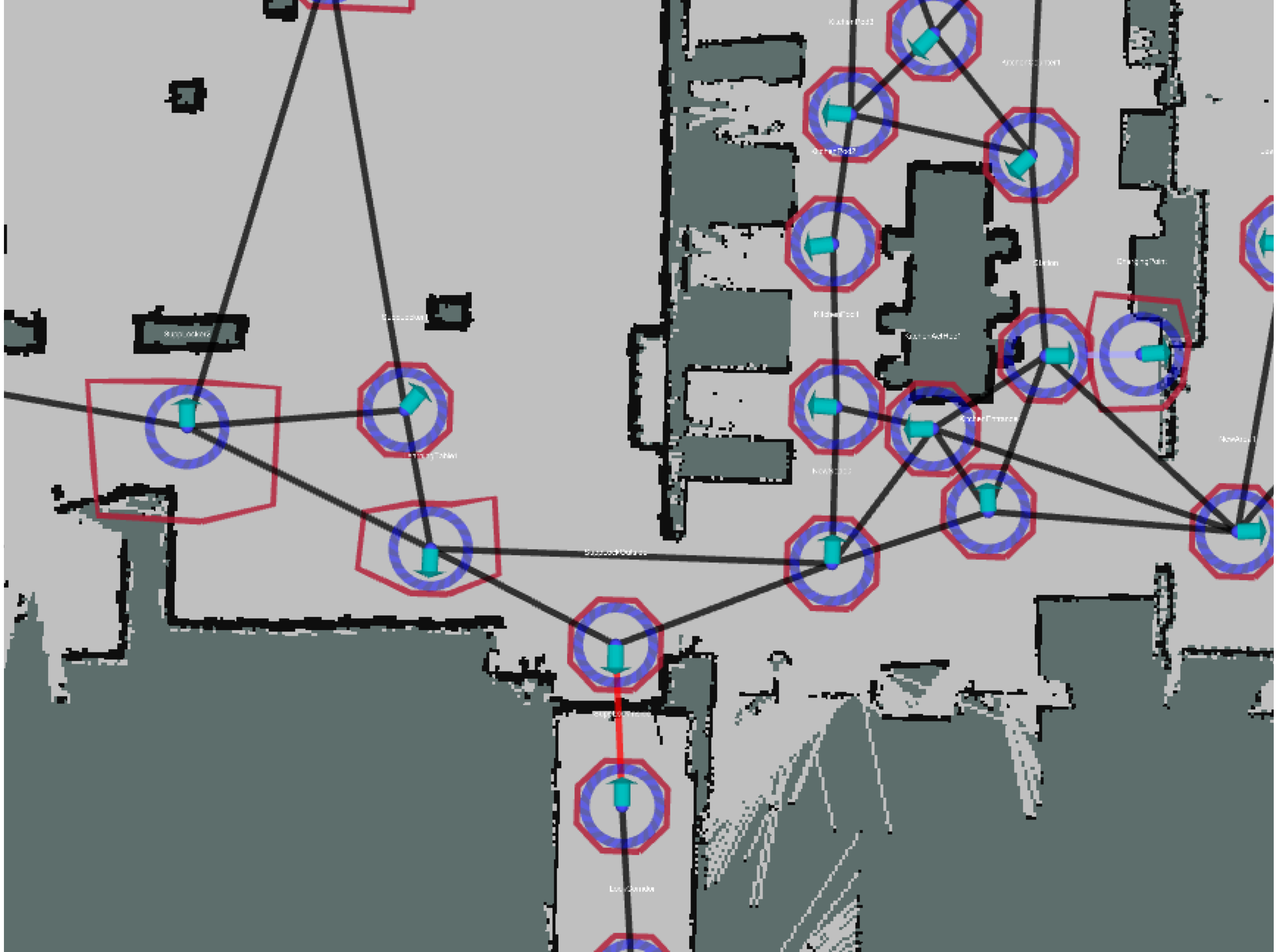
Transport Systems Catapult, Milton Keynes, UK

120 days of *autonomous* behaviour

Transport Systems Catapult, Milton Keynes, UK



Transport Systems Catapult, Milton Keynes, UK

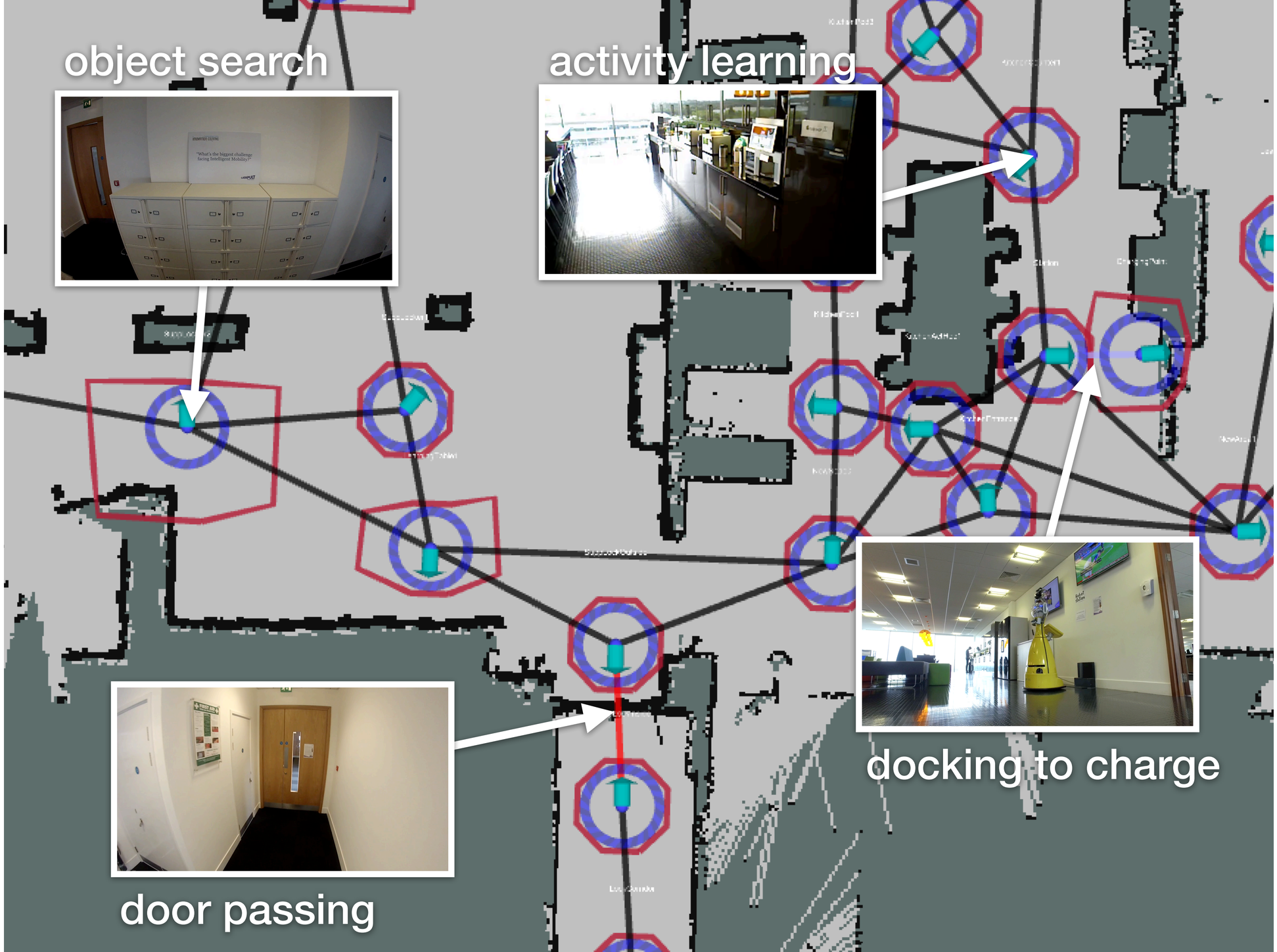


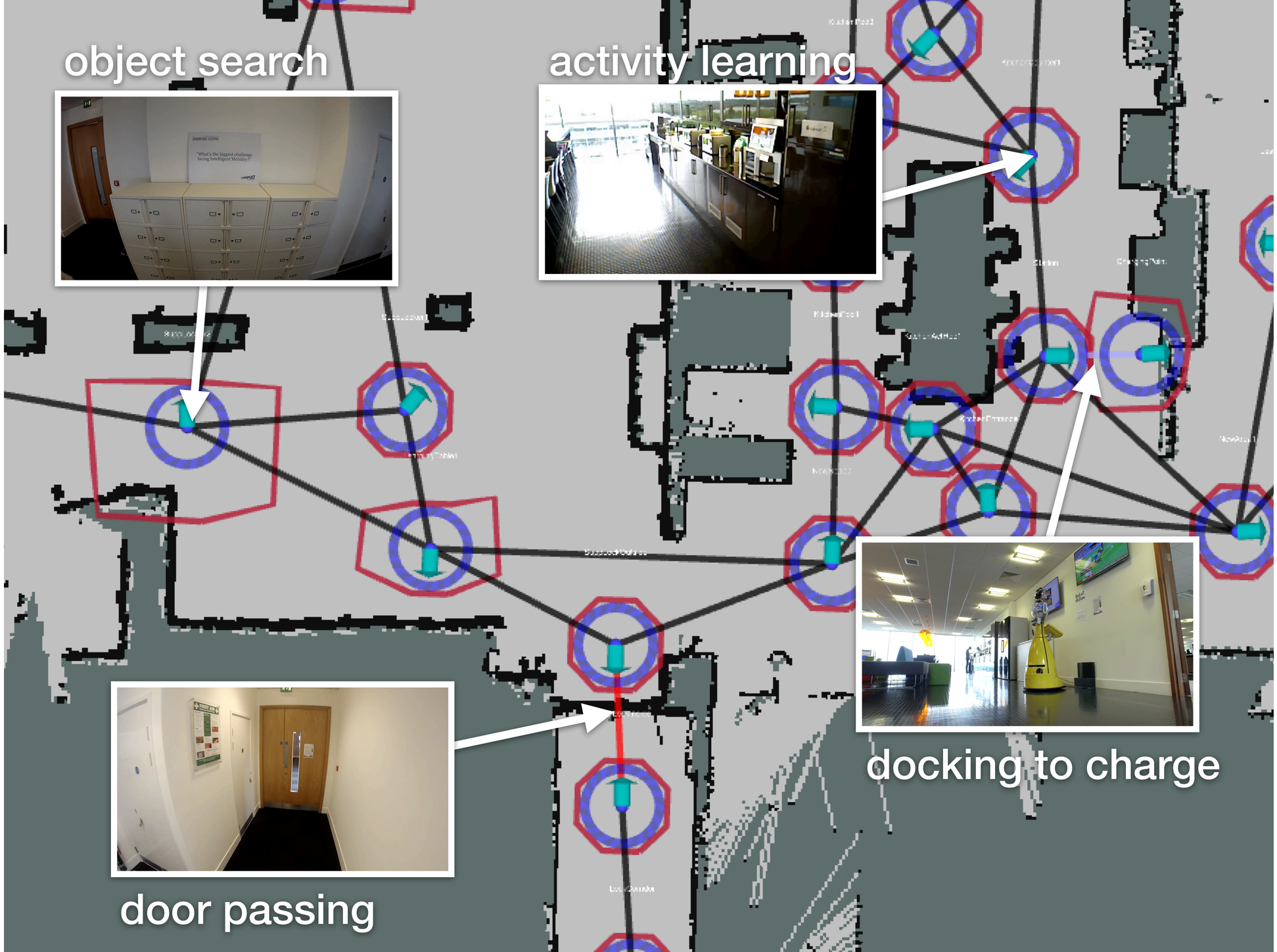
door passing

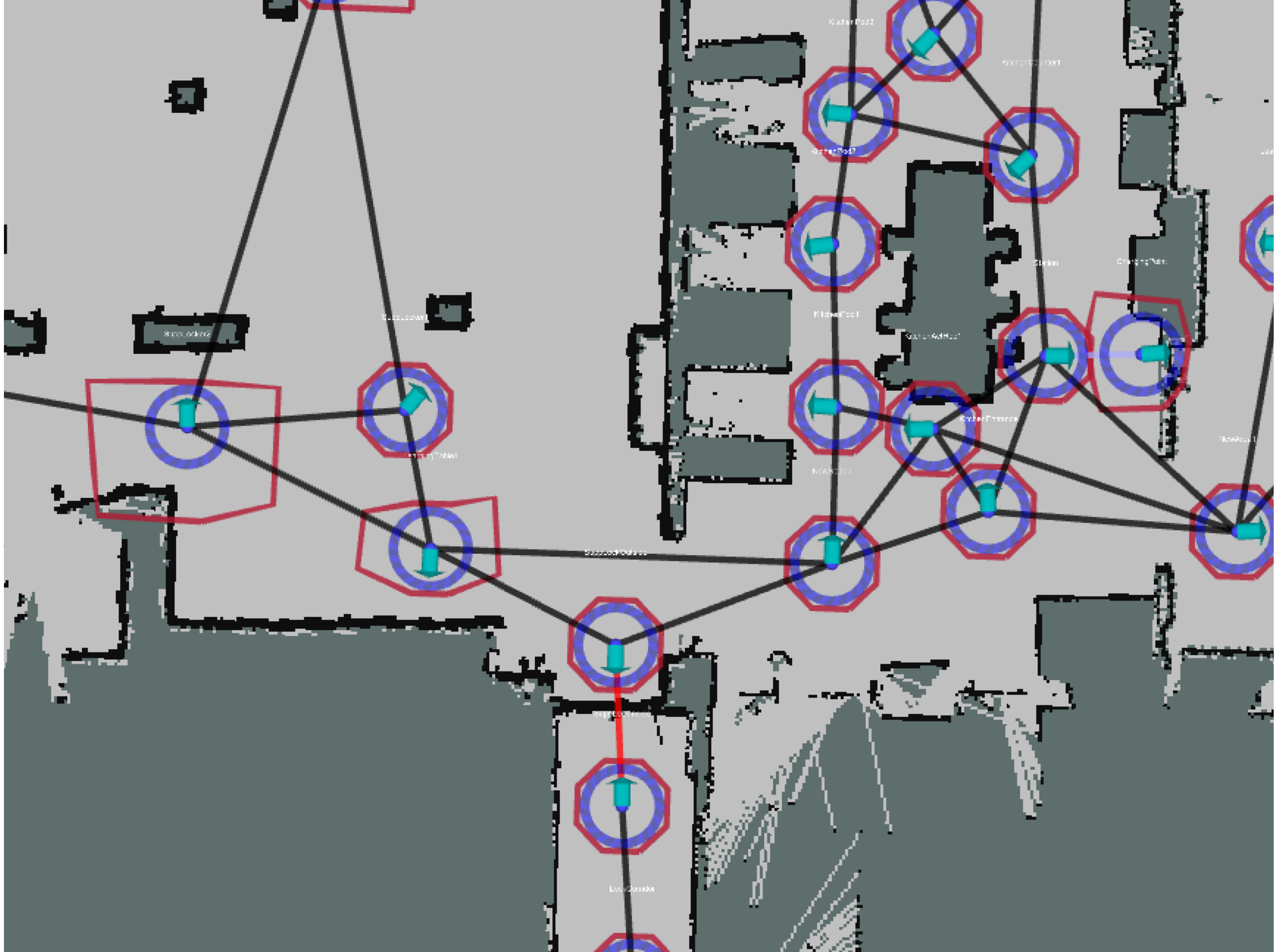
docking to charge

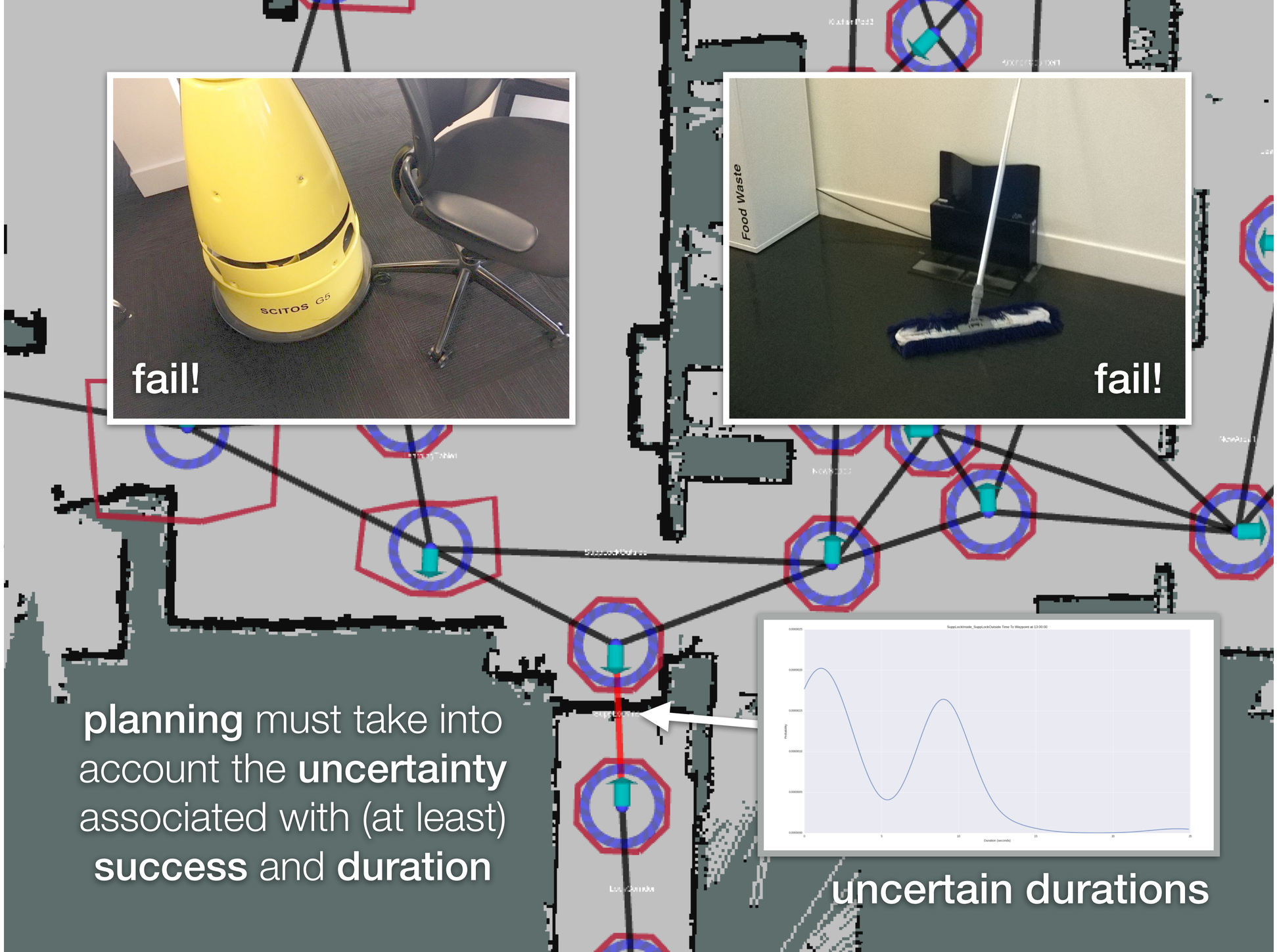
door passing

docking to charge


object search
activity learning
docking to charge
door passing

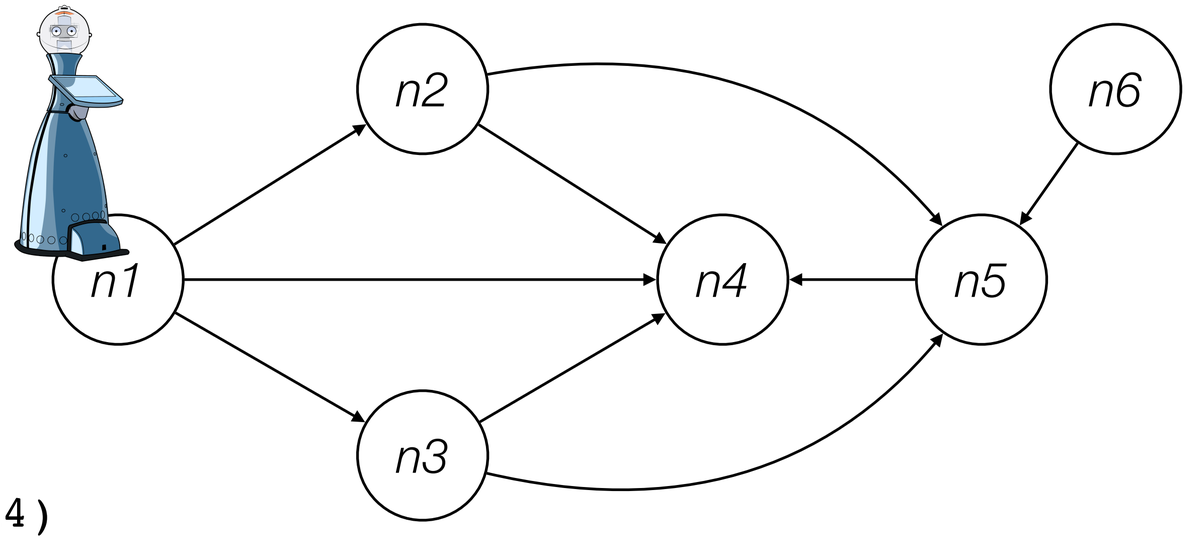

object search
activity learning
docking to charge
door passing




SCITOS G5
fail!
Food Waste
fail!
planning must take into account the uncertainty associated with (at least) success and duration
uncertain durations

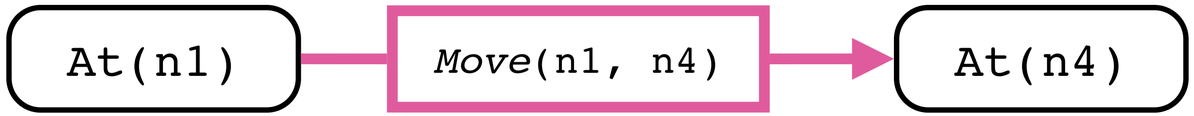
Reliable navigation example

Moving deterministically results in the target node



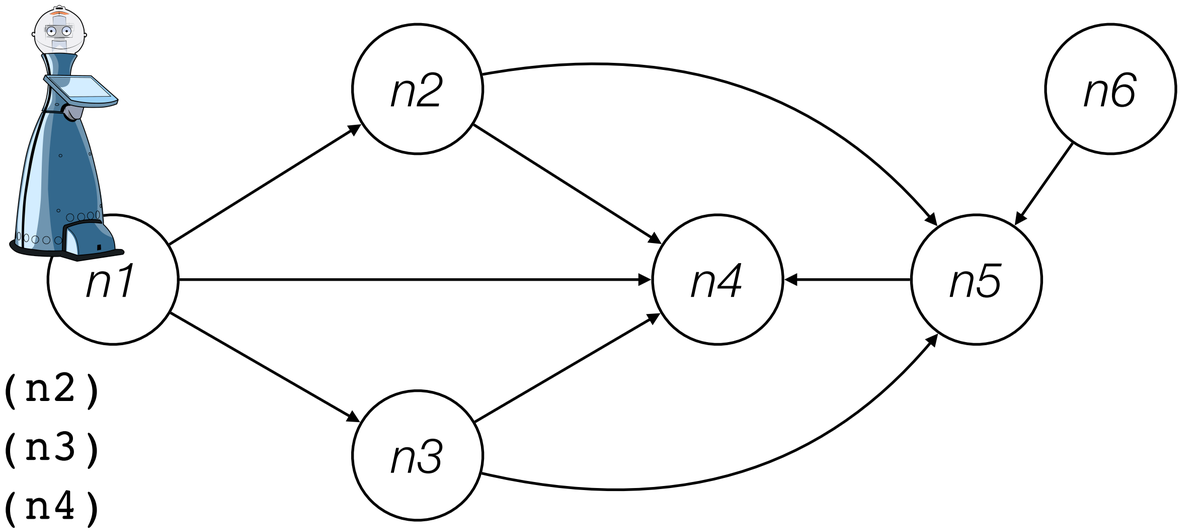
deterministic

```
Action( Move(n1, n4),
  PRECOND: At(n1)
  EFFECT: ¬At(n1) ∧ At(n4)

Init( At(n1))

Goal( At(n4))
```



Unreliable navigation example

Moving non-deterministically results in a connected node

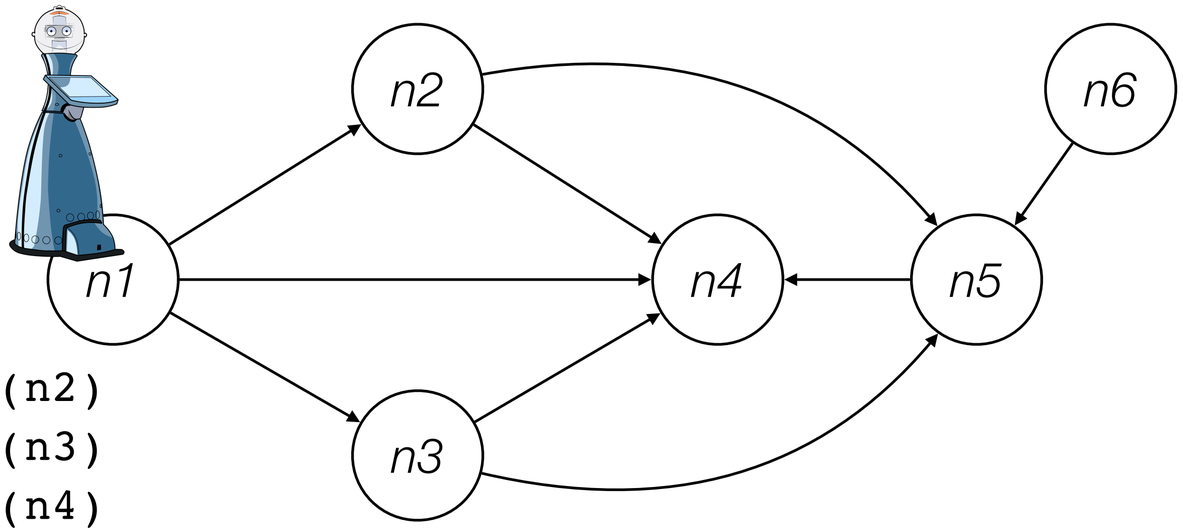


non-deterministic

```
Action( Move(n1, n4),
  PRECOND: At(n1)
  EFFECT 1: ¬At(n1) ∧ At(n2)
  EFFECT 2: ¬At(n1) ∧ At(n3)
  EFFECT 3: ¬At(n1) ∧ At(n4)

Init( At(n1))

Goal( At(n4))
```

Unreliable navigation example

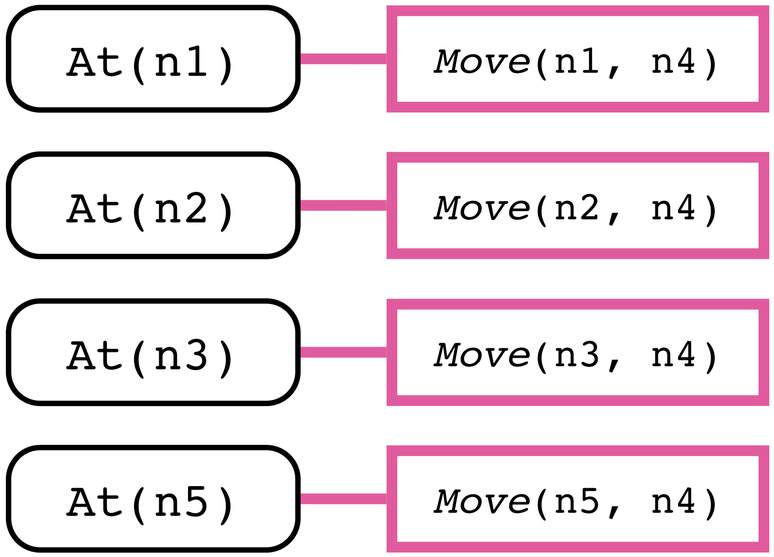
Moving non-deterministically results in a connected node



non-deterministic

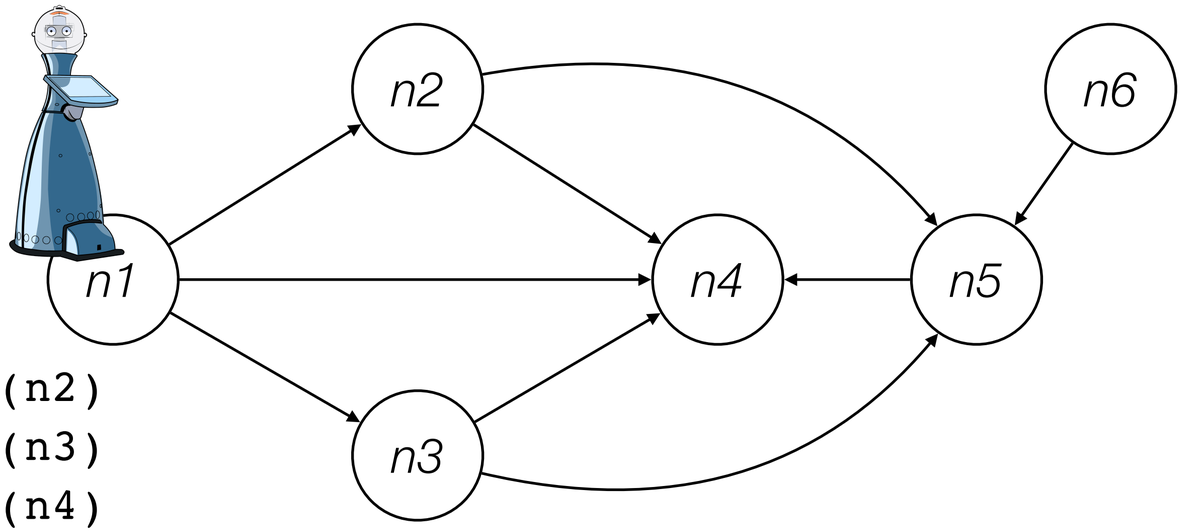
```
Action( Move(n1, n4),
  PRECOND: At(n1)
  EFFECT 1: ¬At(n1) ∧ At(n2)
  EFFECT 2: ¬At(n1) ∧ At(n3)
  EFFECT 3: ¬At(n1) ∧ At(n4)
```

```
Init( At(n1))
Goal( At(n4))
```



Unreliable navigation example

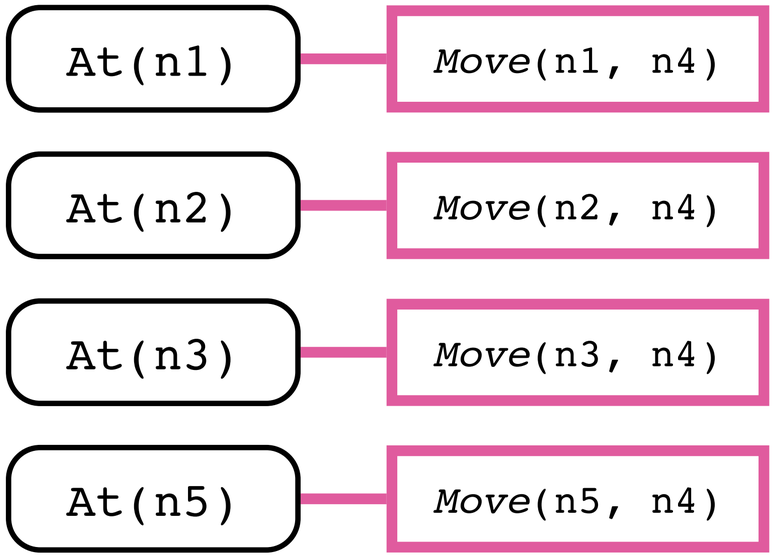
Moving non-deterministically results in a connected node



non-deterministic

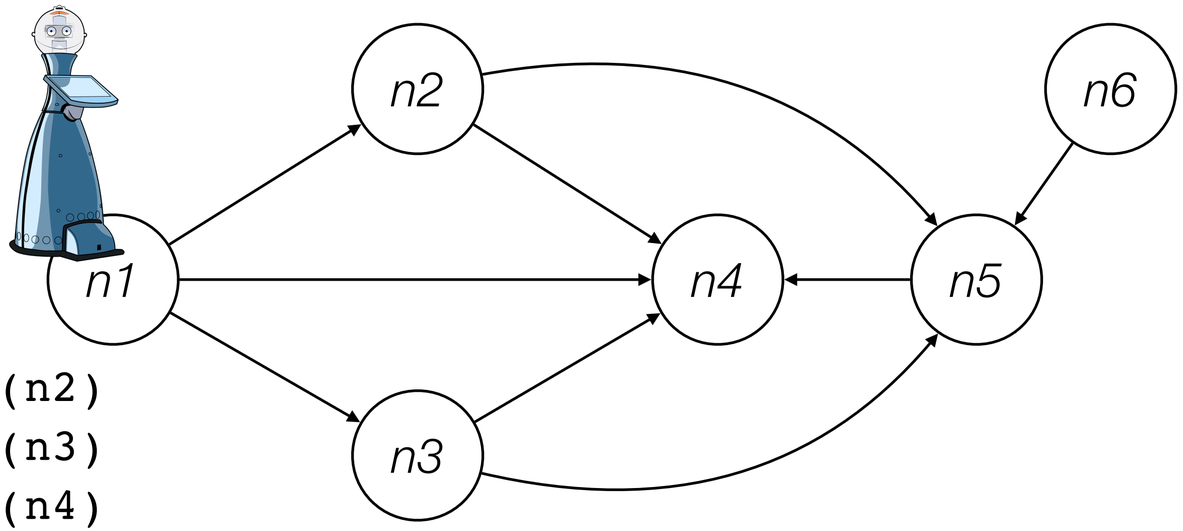
```
Action( Move(n1, n4),
  PRECOND: At(n1)
  EFFECT 1: ¬At(n1) ∧ At(n2)
  EFFECT 2: ¬At(n1) ∧ At(n3)
  EFFECT 3: ¬At(n1) ∧ At(n4)

Init( At(n1))
Goal( At(n4))
```

Why no action for n6?



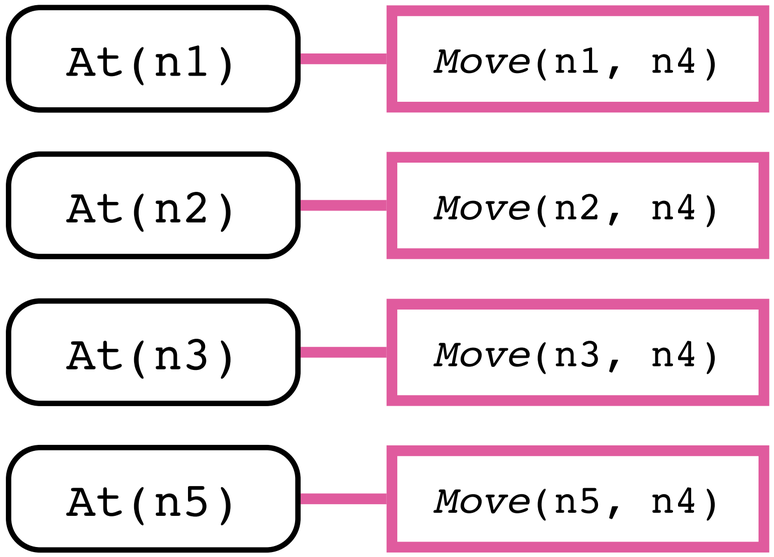
Unreliable navigation example

Moving non-deterministically results in a connected node



non-deterministic

```
Action( Move(n1, n4),
  PRECOND: At(n1)
  EFFECT 1: ¬At(n1) ∧ At(n2)
  EFFECT 2: ¬At(n1) ∧ At(n3)
  EFFECT 3: ¬At(n1) ∧ At(n4)

Init( At(n1))
Goal( At(n4))
```



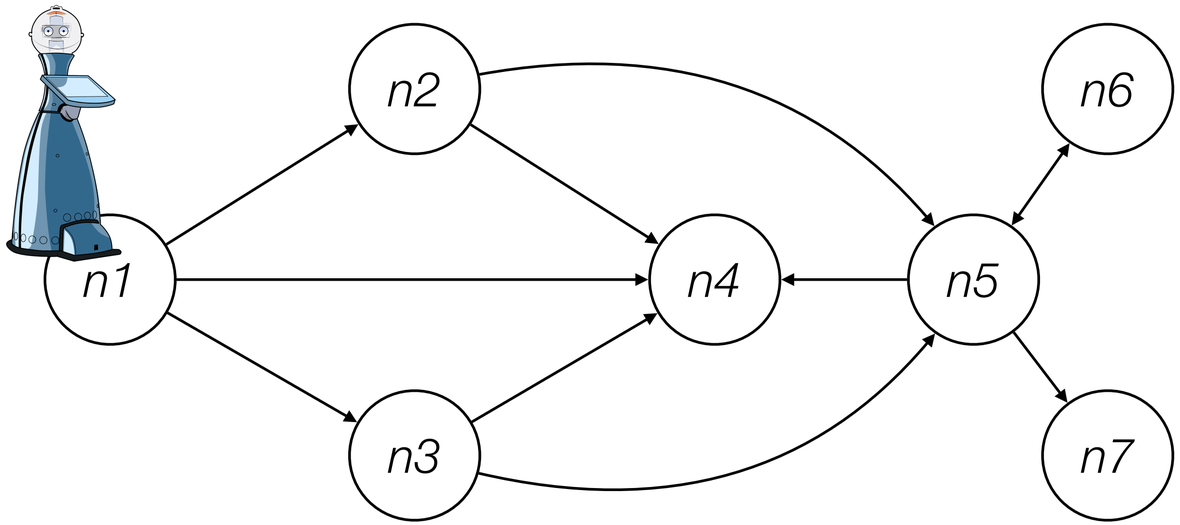
Why no action for n6?

What is the optimal action choice in this model?

Unreliable navigation example

Moving non-deterministically results in a connected node

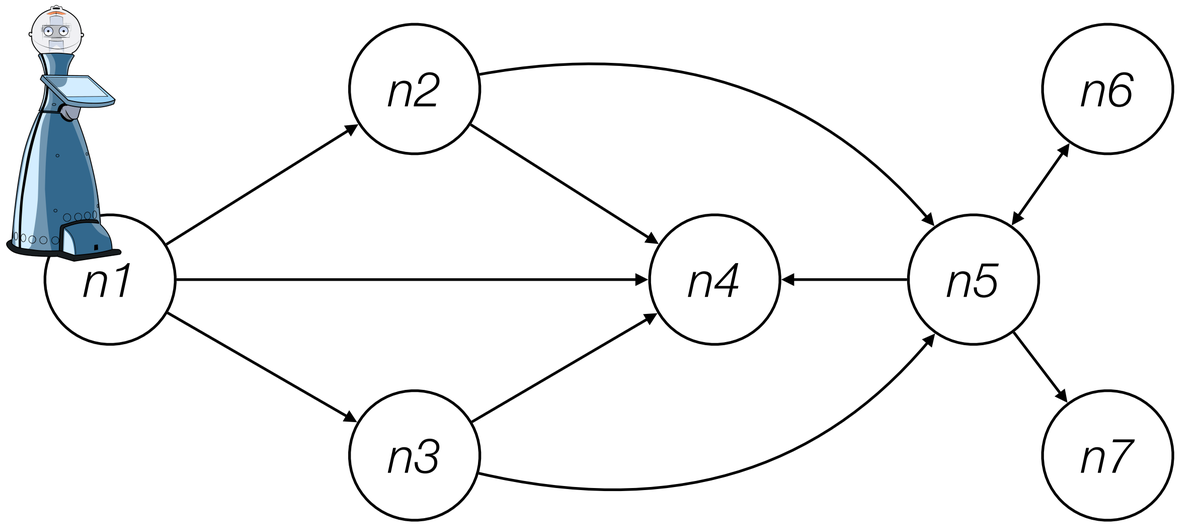
updated model



Unreliable navigation example

Moving non-deterministically results in a connected node

updated model

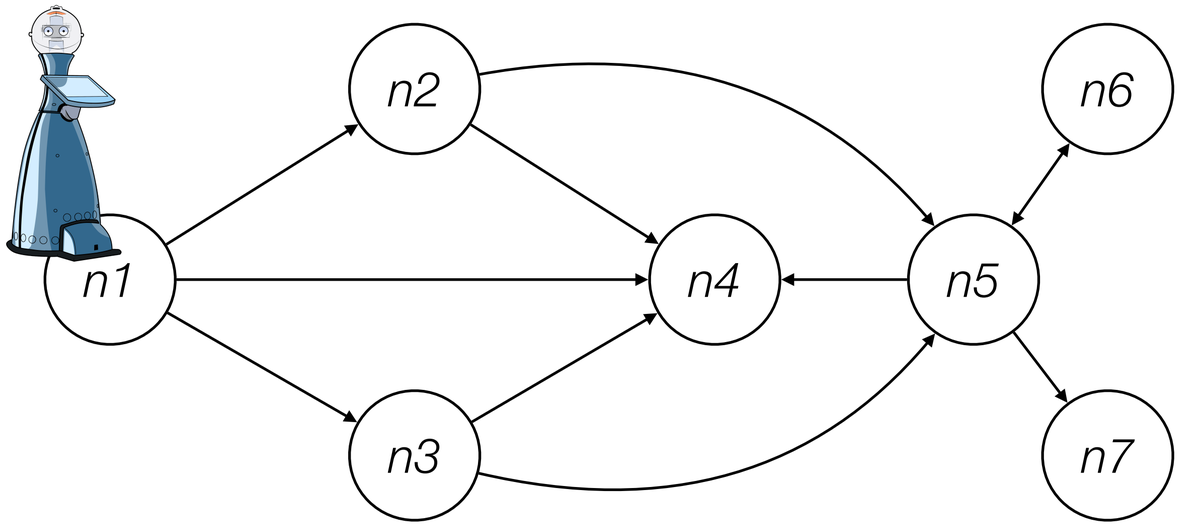


There is a (potentially infinite) **cycle** between *n5* and *n6*

Unreliable navigation example

Moving non-deterministically results in a connected node

updated model



There is a (potentially infinite) **cycle** between *n5* and *n6*

There is no solution from *n7*. It's a **dead end**.

Planning with non-deterministic models

The outcome of an action is not always what the model describes

- Unintended outcomes
- Exogenous events
- Inherent uncertainty

In practice few robot systems use purely non-deterministic models

Planning with non-deterministic models

The outcome of an action is not always what the model describes

- Unintended outcomes
- Exogenous events
- **Inherent uncertainty**

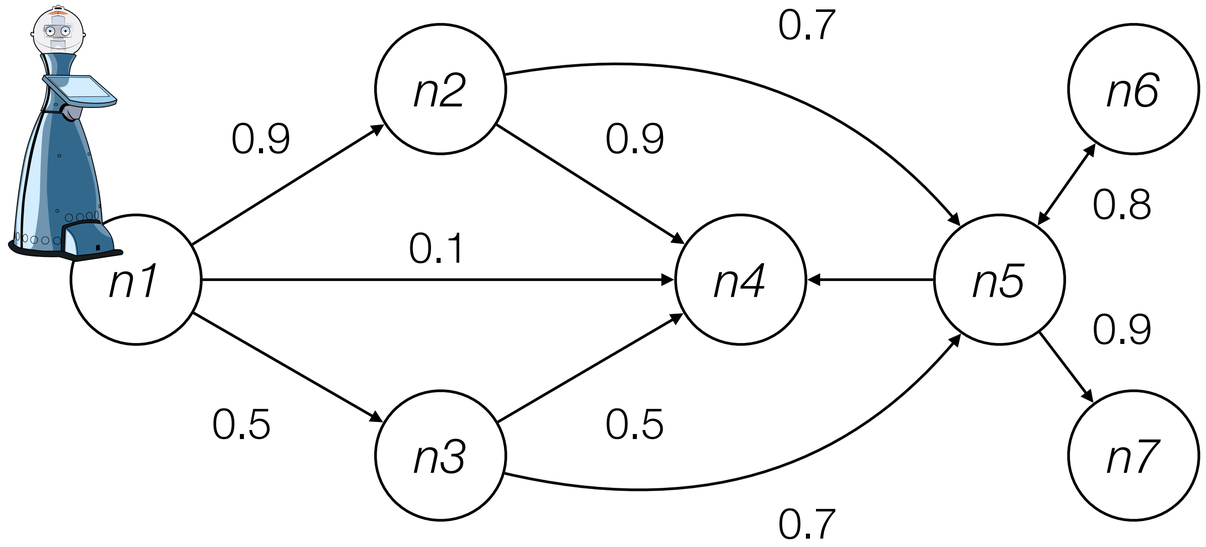
In practice few practical robot systems
use purely non-deterministic models

**If we can quantify the uncertainty,
then we can create better plans**

Unreliable navigation example

Target node is achieved with given probability, other nodes are reached with uniform probability remainder

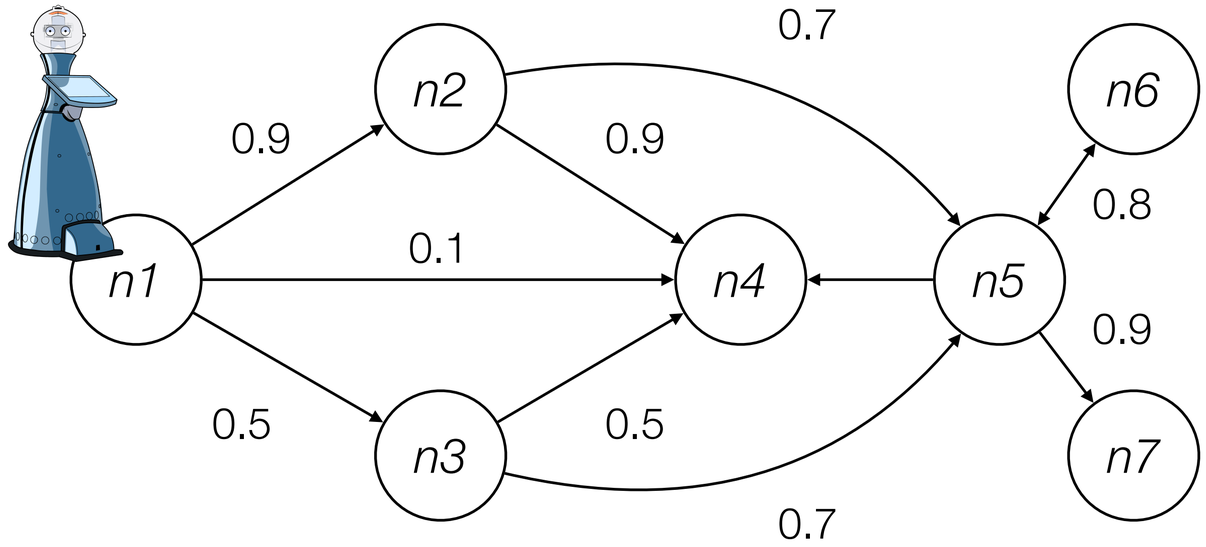
probabilistic



Unreliable navigation example

Target node is achieved with given probability, other nodes are reached with uniform probability remainder

probabilistic



In the non-deterministic case, there was no best action to take

What about the probabilistic case?

Markov Decision Process (MDP)

The most general version

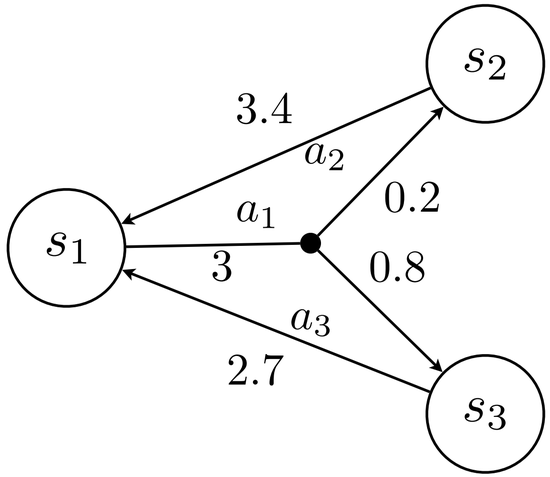
$$\mathcal{M} = \langle S, \bar{s}, A, T \rangle$$

states

initial state

actions

transition probs



- S is a set of states - typically attributions of values to state variables/features
- $\bar{s} \in S$ is the initial state
- A is a set of actions
- $T : S \times A \rightarrow Dist(S)$ is the probabilistic transition function

Policies

Policies

- A policy is a mapping of finite paths of the MDP to (distributions over) actions

$$\pi : FPath_{\mathcal{M},\bar{s}} \rightarrow Dist(A)$$

Policies

- A policy is a mapping of finite paths of the MDP to (distributions over) actions

$$\pi : FPath_{\mathcal{M},\bar{s}} \rightarrow Dist(A)$$

- Different sub-classes:
  - Deterministic - action choice is Dirac delta

$$\pi : FPath_{\mathcal{M},\bar{s}} \rightarrow A$$

Policies

- A policy is a mapping of finite paths of the MDP to (distributions over) actions

$$\pi : FPath_{\mathcal{M},\bar{s}} \to Dist(A)$$

- Different sub-classes:
  - Deterministic - action choice is Dirac delta

$$\pi : FPath_{\mathcal{M},\bar{s}} \to A$$

  - Stationary - action choice only depends on current state

$$\pi : S \to Dist(A)$$

Policies

- A policy is a mapping of finite paths of the MDP to (distributions over) actions

$$\pi : FPath_{\mathcal{M},\bar{s}} \to Dist(A)$$

- Different sub-classes:
  - Deterministic - action choice is Dirac delta

$$\pi : FPath_{\mathcal{M},\bar{s}} \to A$$

  - Stationary - action choice only depends on current state

$$\pi : S \to Dist(A)$$

  - Finite memory - action choice only depends on current state and finite set of “modes”

$$\pi : S \times \{1, ..., m\} \to Dist(A)$$

Policies

Policies

- A policy is a mapping of finite paths of the MDP to (distributions over) actions

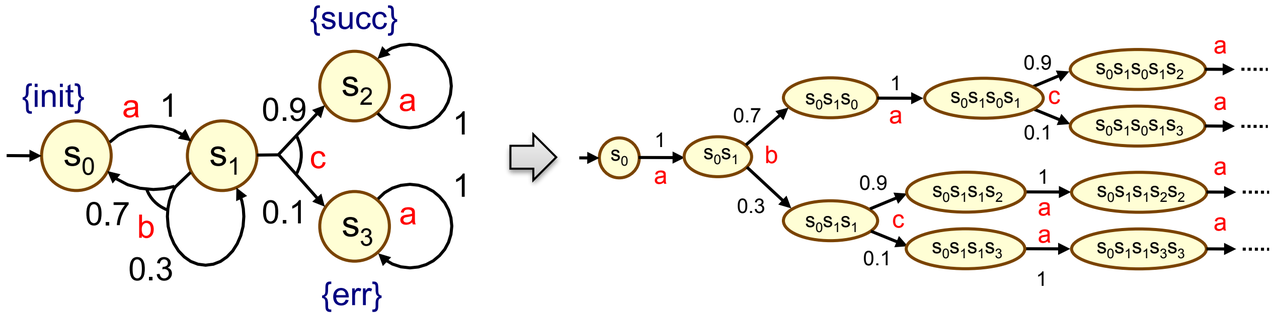
$$\pi : FPath_{\mathcal{M},\bar{s}} \rightarrow Dist(A)$$

Policies

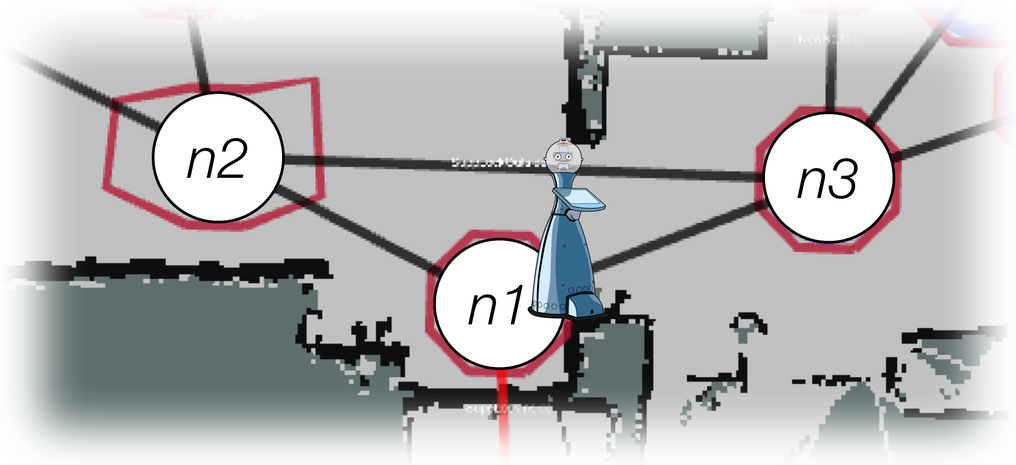
- A policy is a mapping of finite paths of the MDP to (distributions over) actions

$$\pi : FPath_{\mathcal{M},\bar{s}} \rightarrow Dist(A)$$

- Induces (infinite-state) Markov chain (and probability space)
  - *Pick action b the first time you visit s_1, then c afterwards*



Topological Map

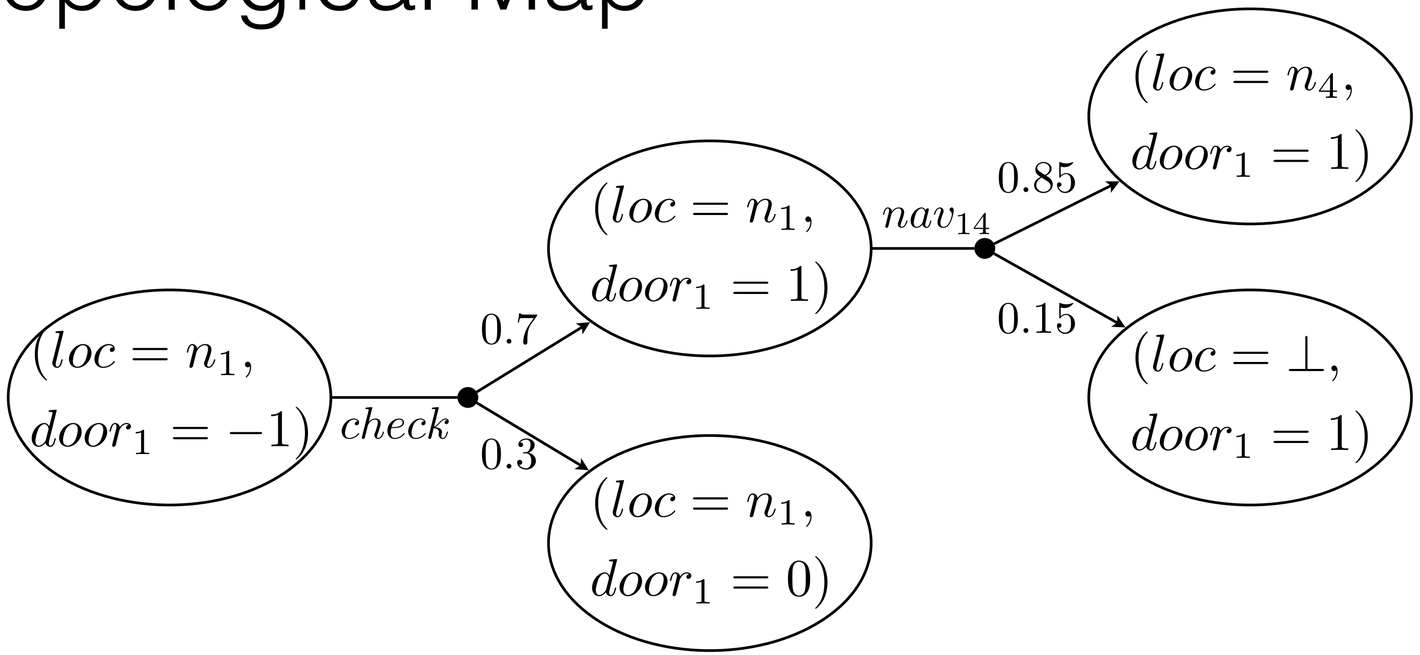


$$\bar{s} = (loc = n_1, door = -1)$$

$$check = \quad pre = \langle (loc = n_1), (door_1 = -1) \rangle,$$
$$eff = 0.7 : (door_1 = 1) + 0.3 : (door_1 = 0) \rangle$$

$$nav_e = \quad pre = \langle (loc = n_1), (door_1 = 1) \rangle,$$
$$eff = 0.85 : (loc = n_4) + 0.15 : (loc = \perp) \rangle$$

Topological Map



$$\bar{s} = (loc = n_1, door = -1)$$

$$\begin{aligned} check = \quad & pre = \langle (loc = n_1), (door_1 = -1) \rangle, \\ & eff = 0.7 : (door_1 = 1) + 0.3 : (door_1 = 0) \rangle \end{aligned}$$

$$\begin{aligned} nav_e = \quad & pre = \langle (loc = n_1), (door_1 = 1) \rangle, \\ & eff = 0.85 : (loc = n_4) + 0.15 : (loc = \perp) \rangle \end{aligned}$$

Common Goal Specifications

$$\mathcal{M} = \langle S, \bar{s}, A, T \rangle$$

- Given reward structure, $r : S \times A \to \mathbb{R}$and discount factor $0 < \gamma < 1$ maximise *expected infinite-horizon discounted cumulative reward*

$$V^*(\bar{s}) = E^{\max}_{\mathcal{M},\bar{s}} \left(\sum_{i=0}^{\infty} \gamma^i r(s_i, a_i) \right) = \sup_{\pi} E^{\pi}_{\mathcal{M},\bar{s}} \left(\sum_{i=0}^{\infty} \gamma^i r(s_i, a_i) \right)$$

- Stationary deterministic policies suffice

$$\pi : S \to A$$

Common Goal Specifications

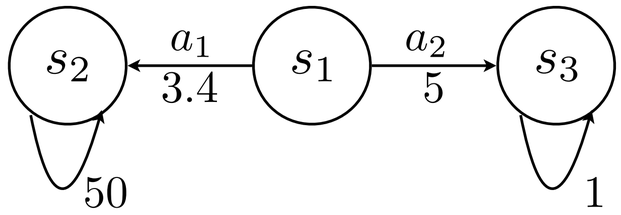
$$\mathcal{M} = \langle S, \bar{s}, A, T \rangle$$

- Given reward structure, $r : S \times A \to \mathbb{R}$ and horizon H maximise *finite-horizon cumulative reward*

$$V^*(\bar{s}) = E^{\max}_{\mathcal{M},\bar{s}} \left(\sum_{i=0}^{H} r(s_i, a_i) \right) = \sup_{\pi} E^{\pi}_{\mathcal{M},\bar{s}} \left(\sum_{i=0}^{H} r(s_i, a_i) \right)$$

- Finite-memory deterministic policies suffice

$$\pi : S \times \{0, ..., H-1\} \to A$$



Common Goal Specifications

$$\mathcal{M} = \langle S, \bar{s}, A, T \rangle$$

- Given a set of goal states $G \subseteq S$ maximise *probability of reaching a goal state*

$$V^*(\bar{s}) = Pr^{\max}_{\mathcal{M},\bar{s}}(reach_G) = \sup_{\pi} Pr^{\pi}_{\mathcal{M},\bar{s}}(reach_G)$$

- Stationary deterministic policies suffice

$$\pi : S \to A$$

Common Goal Specifications

$$\mathcal{M} = \langle S, \bar{s}, A, T \rangle$$

- Given an LTL formula φ maximise probability of generating an infinite path that satisfies it

$$V^*(\bar{s}) = Pr^{\max}_{\mathcal{M},\bar{s}}(\varphi) = \sup_{\pi} Pr^{\pi}_{\mathcal{M},\bar{s}}(\varphi)$$

- Finite-memory deterministic policies suffice

$$\pi : S \times \{0, ..., |Q_\varphi|\} \to A$$

Common Goal Specifications

$$\mathcal{M} = \langle S, \bar{s}, A, T \rangle$$

- Given cost structure, $c : S \times A \to \mathbb{R}_{\geq 0}$ and set of goal states $G \subseteq S$ minimise *expected cumulative cost to reach the goal*

$$V^*(\bar{s}) = E^{\min}_{\mathcal{M},\bar{s}} \left(\sum_{i=0}^{n_G} c(s_i, a_i) \right) = \inf_{\pi} E^{\pi}_{\mathcal{M},\bar{s}} \left(\sum_{i=0}^{n_G} c(s_i, a_i) \right)$$

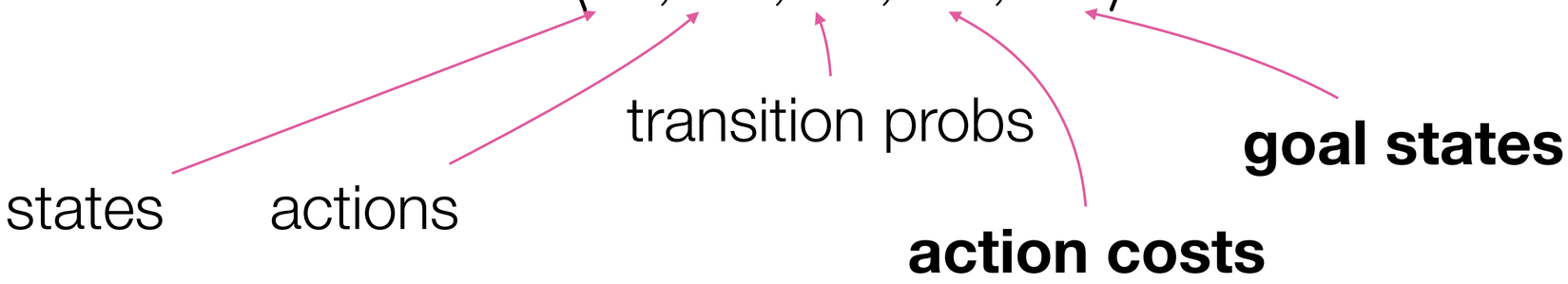
- Extra assumptions:
  - Probability of reaching *G is one*
  - *No zero cost loops*
- Stationary deterministic policies suffice

$$\pi : S \to A$$

Markov Decision Process (MDP)

Stochastic Shortest Path (SSP) MDP

$$\mathcal{M} = \langle S, A, T, C, G \rangle$$



Markov Decision Process (MDP)

Stochastic Shortest Path (SSP) MDP

$$\mathcal{M} = \langle S, A, T, C, G \rangle$$

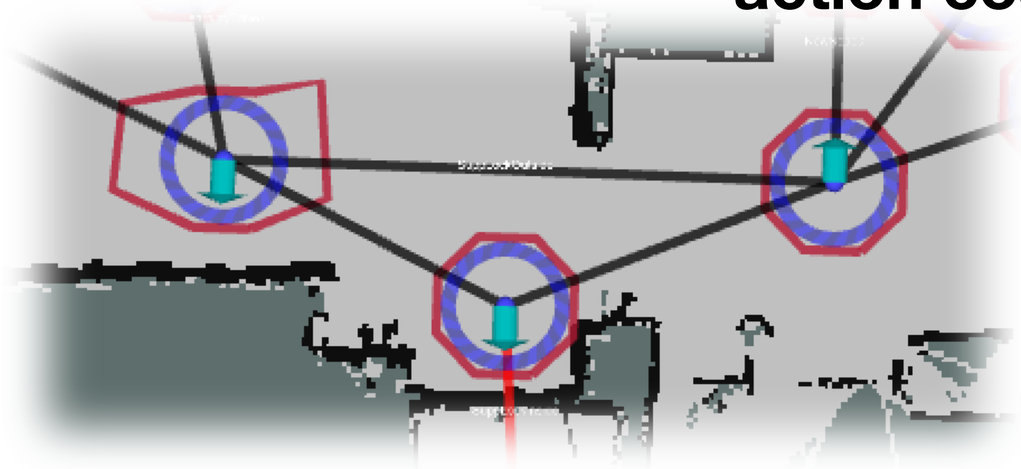
states

actions

transition probs

action costs

goal states



Markov Decision Process (MDP)

Stochastic Shortest Path (SSP) MDP

$$\mathcal{M} = \langle S, A, T, C, G \rangle$$

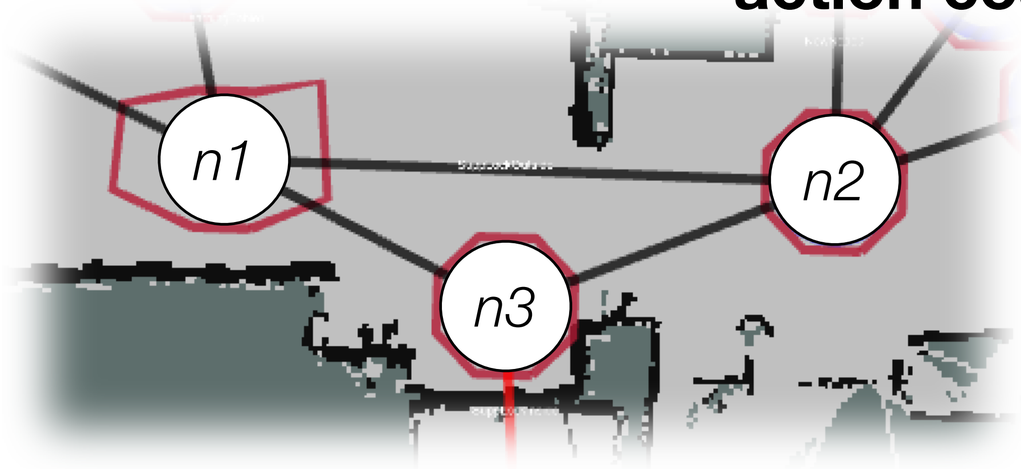
states

actions

transition probs

action costs

goal states

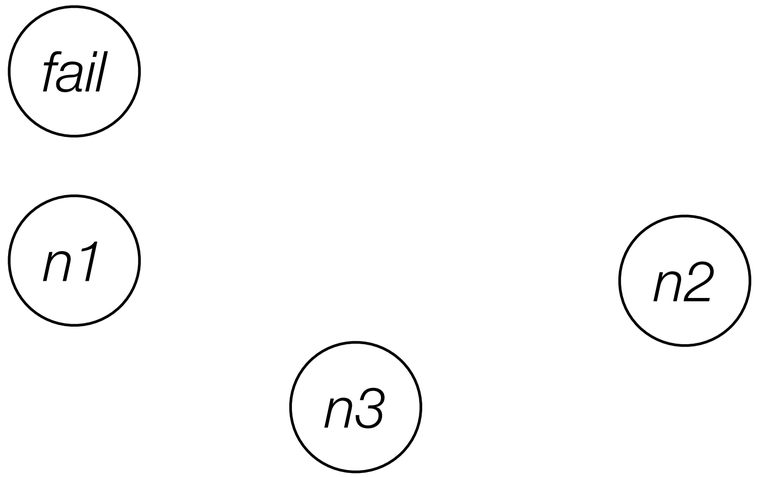


Markov Decision Process (MDP)

Stochastic Shortest Path (SSP) MDP

$$\mathcal{M} = \langle S, A, T, C, G \rangle$$

$n1, n2, n3, fail \in S$

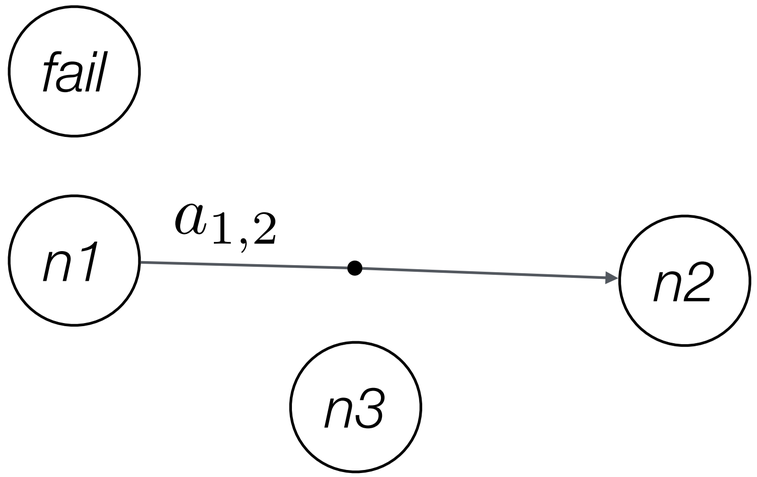


Markov Decision Process (MDP)

Stochastic Shortest Path (SSP) MDP

$$\mathcal{M} = \langle S, A, T, C, G \rangle$$

$n1, n2, n3, fail \in S$ $\quad$ $a_{1,2} \in A$

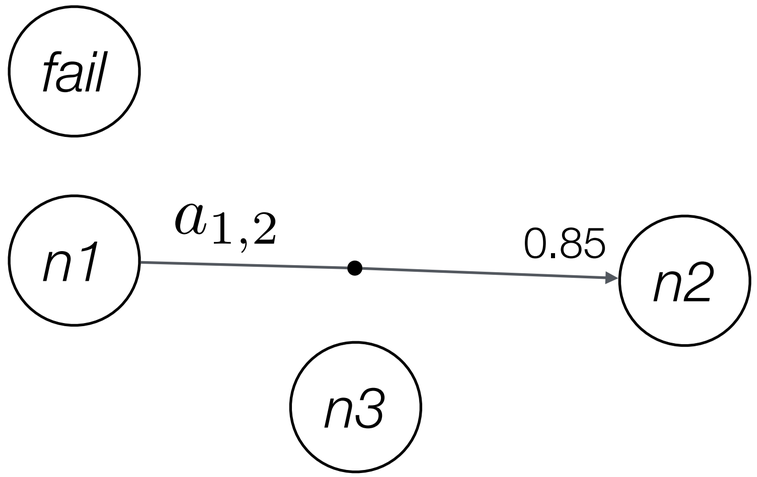


Markov Decision Process (MDP)

Stochastic Shortest Path (SSP) MDP

$$\mathcal{M} = \langle S, A, T, C, G \rangle$$

$n1, n2, n3, fail \in S$ $\quad$ $a_{1,2} \in A$

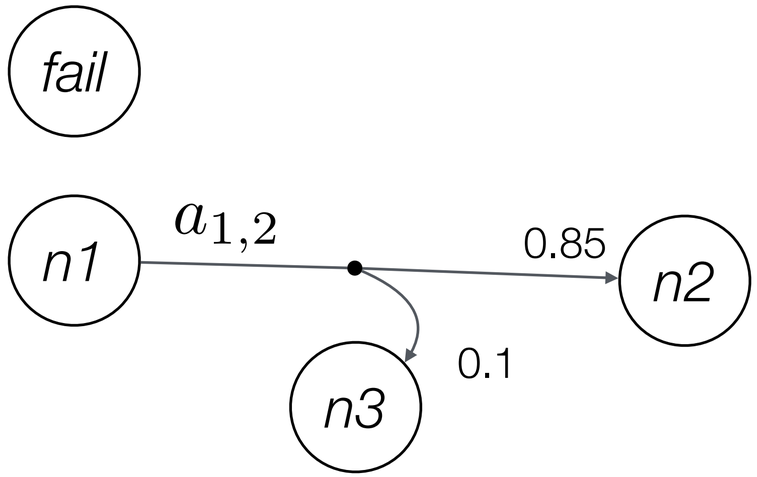


Markov Decision Process (MDP)

Stochastic Shortest Path (SSP) MDP

$$\mathcal{M} = \langle S, A, T, C, G \rangle$$

$n1, n2, n3, fail \in S$ $\quad$ $a_{1,2} \in A$

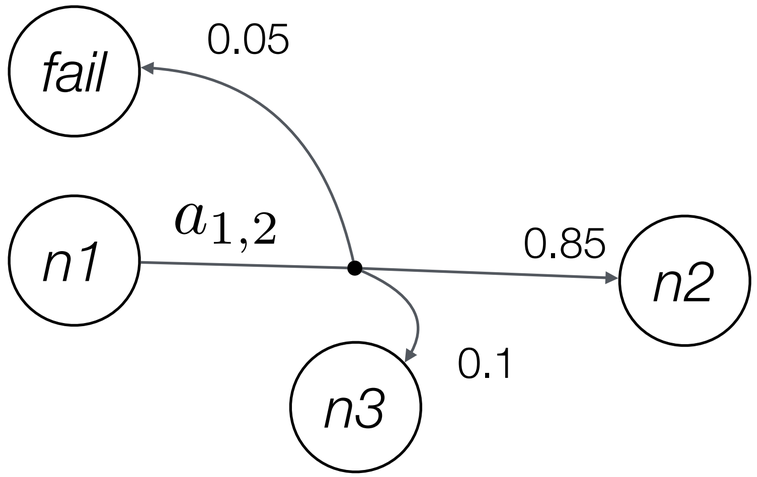


Markov Decision Process (MDP)

Stochastic Shortest Path (SSP) MDP

$$\mathcal{M} = \langle S, A, T, C, G \rangle$$

$n1, n2, n3, fail \in S$ $\quad$ $a_{1,2} \in A$

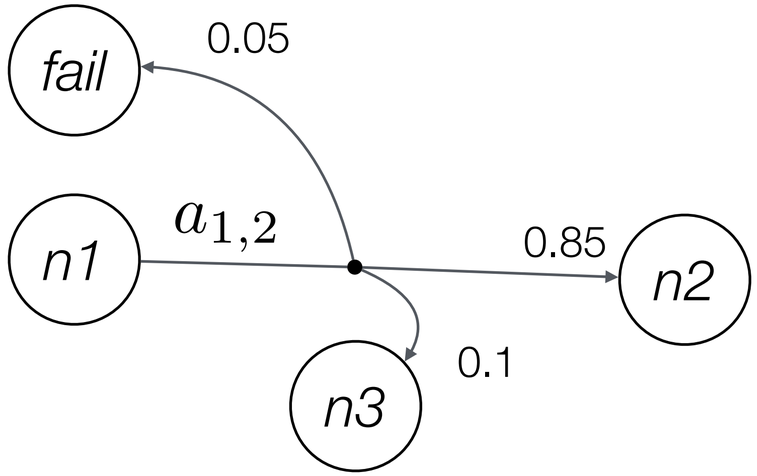


Markov Decision Process (MDP)

Stochastic Shortest Path (SSP) MDP

$$\mathcal{M} = \langle S, A, T, C, G \rangle$$

$n1, n2, n3, fail \in S$ $\quad$ $a_{1,2} \in A$



(seconds?)

$$T(n1, a_{1,2}, n2) = 0.85$$
$$T(n1, a_{1,2}, n3) = 0.1$$
$$T(n1, a_{1,2}, fail) = 0.05$$

$$C(n1, a_{1,2}, n2) = 5$$
$$C(n1, a_{1,2}, n3) = 3$$
$$C(n1, a_{1,2}, fail) = 12$$

Markov Decision Process (MDP)

Stochastic Shortest Path (SSP) MDP

$$\mathcal{M} = \langle S, A, T, C, G \rangle$$

$n1, n2, n3, fail \in S$ $\quad$ $a_{1,2} \in A$

Note that states are usually **sets of variables/values**, e.g.

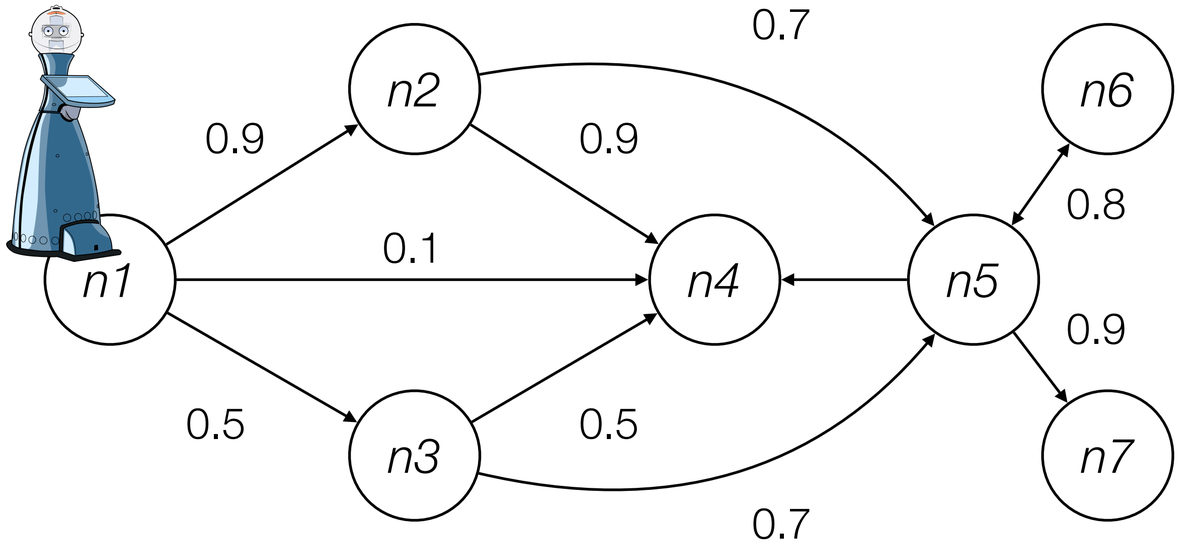
s0 = (*n1*=1, *n2*=0, *n3*=0, *fail*=0, *battery*=high)

Actions may only be applicable/enabled given certain values of state variables (similar to classical planning)

SSP navigation example

Target node is achieved with given probability, other nodes are reached with uniform probability remainder. Cost is 1 for all edges.

$$G = \{n4\}$$

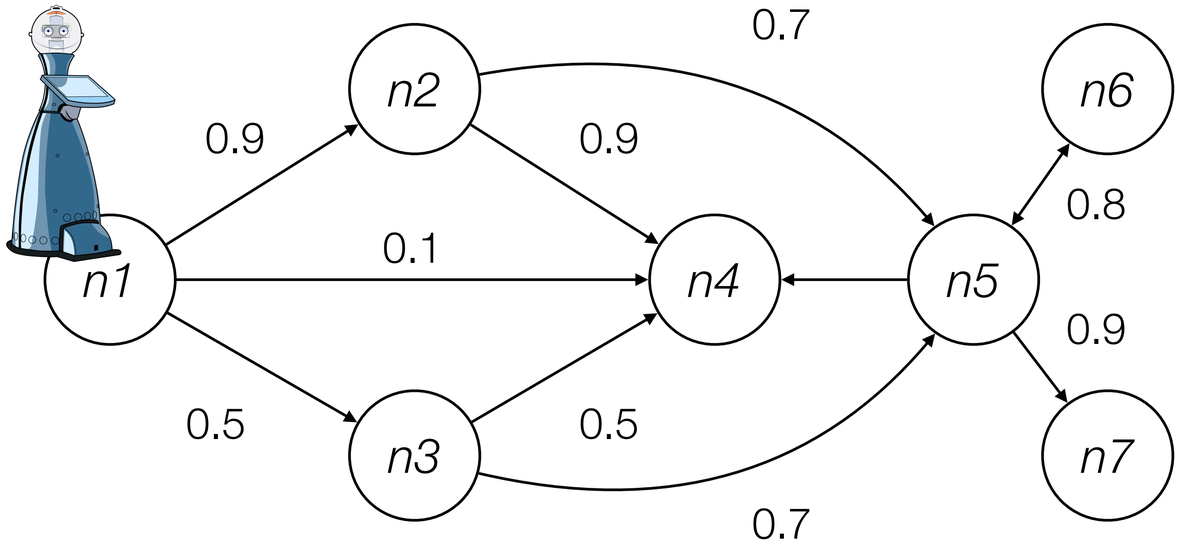


SSP navigation example

Target node is achieved with given probability, other nodes are reached with uniform probability remainder. Cost is 1 for all edges.

$$\mathrm{G} = \{\mathrm{n4}\}$$

Solution will be an optimal policy π^* which **minimises *expected cost* to reach a goal** state



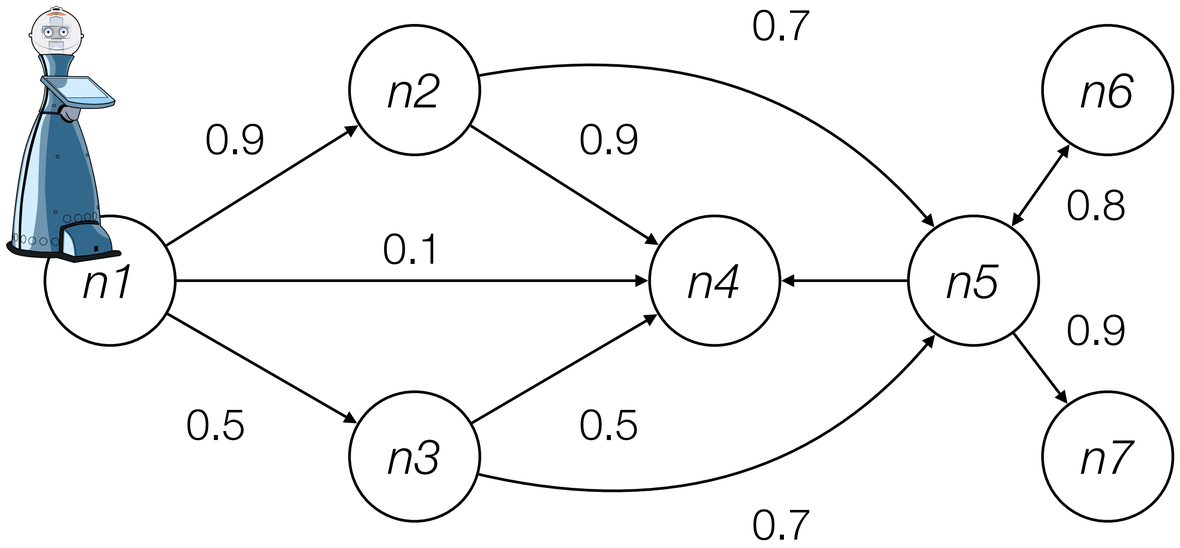
SSP navigation example

Target node is achieved with given probability, other nodes are reached with uniform probability remainder. Cost is 1 for all edges.

$$\mathrm{G} = \{\mathrm{n4}\}$$

Solution will be an optimal policy π^* which **minimises *expected cost* to reach a goal** state

Most MDP solution methods require some assumptions about the model



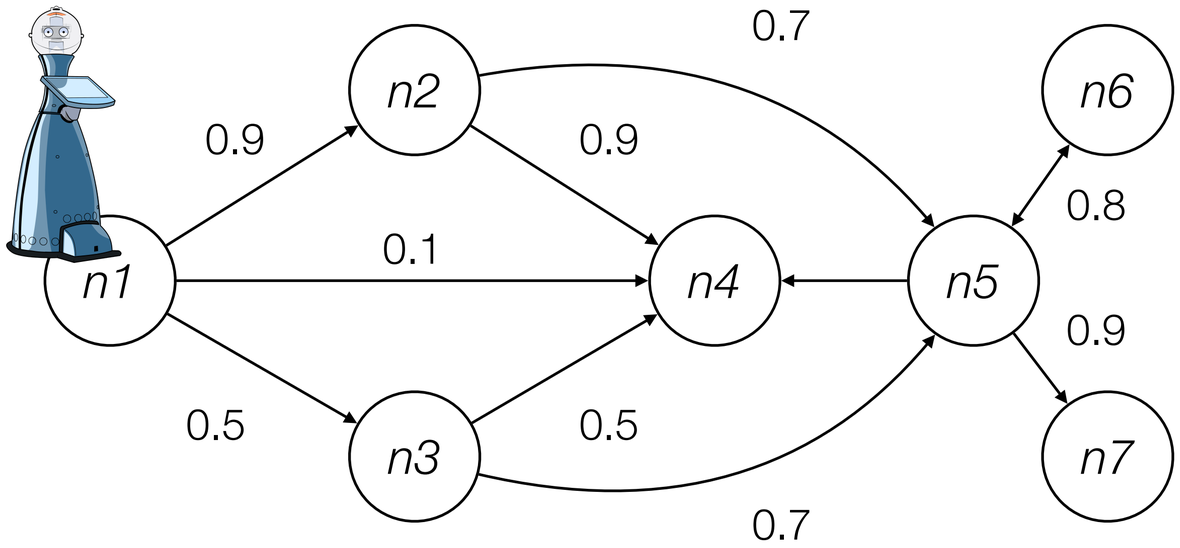
SSP navigation example

Target node is achieved with given probability, other nodes are reached with uniform probability remainder. Cost is 1 for all edges.

$$\mathrm{G} = \{\mathrm{n4}\}$$

Solution will be an optimal policy π^* which **minimises *expected cost* to reach a goal** state

Most MDP solution methods require some assumptions about the model



Goal can be reached with **probability 1**

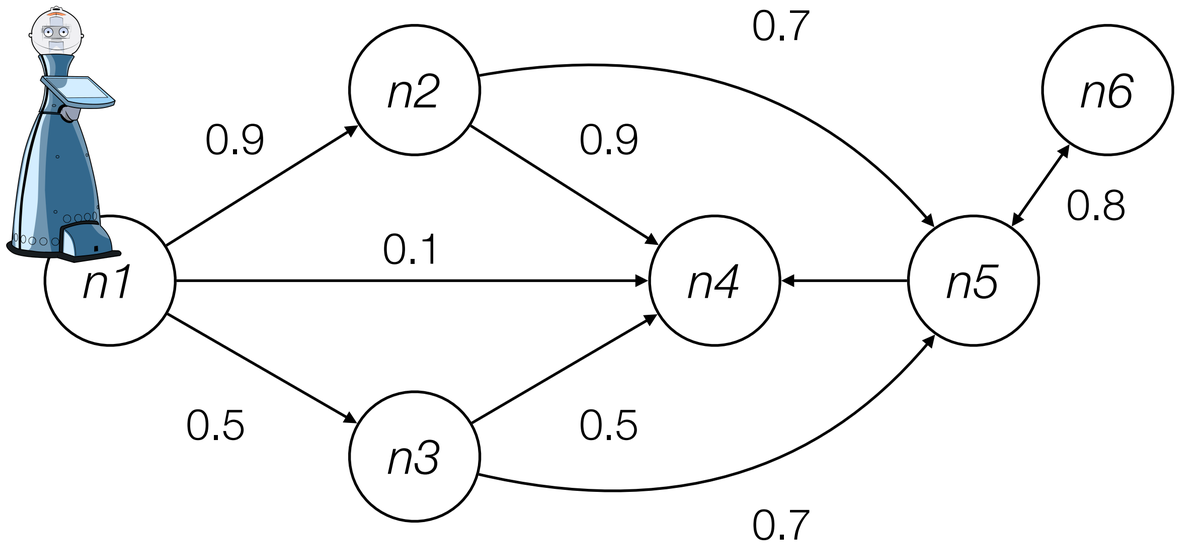
SSP navigation example

Target node is achieved with given probability, other nodes are reached with uniform probability remainder. Cost is 1 for all edges.

$$\mathrm{G} = \{\mathrm{n4}\}$$

Solution will be an optimal policy π^* which **minimises *expected cost* to reach a goal** state

Most MDP solution methods require some assumptions about the model



Goal can be reached with **probability 1**

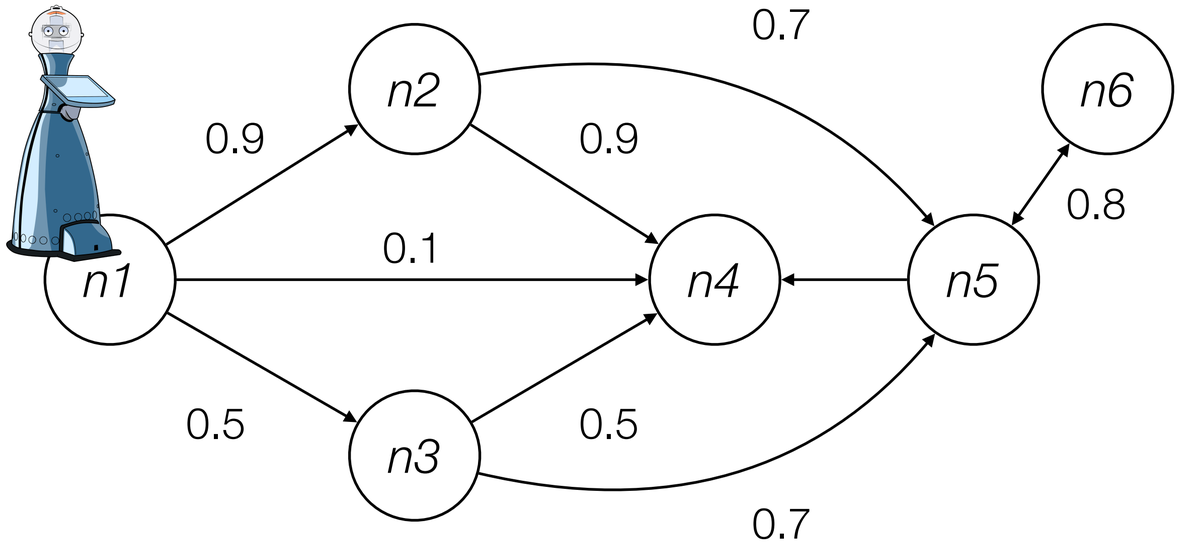
SSP navigation example

Target node is achieved with given probability, other nodes are reached with uniform probability remainder. Cost is 1 for all edges.

$$\mathrm{G} = \{\mathrm{n4}\}$$

Solution will be an optimal policy π^* which **minimises *expected cost* to reach a goal** state

Most MDP solution methods require some assumptions about the model



Goal can be reached with **probability 1**

No **0-cost cycles**.

Value Iteration

Value Iteration

The **optimal policy** provides the **action a** to take in **state s** which minimises the cost to reach a goal state.

Value Iteration

The **optimal policy** provides the **action a** to take in **state s** which minimises the cost to reach a goal state.

Optimal **Q value** (cost-to-go) is the expected cost to first execute **action a** in **state s**, and **then follow an optimal policy** thereafter.

sum over outcome states s' from action a

$$Q^*(s,a) = \sum T(s,a,s')[C(s,a,s') + V^*(s')]$$

transition prob

action cost

value of state

Value Iteration

The **optimal policy** provides the **action a** to take in **state s** which minimises the cost to reach a goal state.

Optimal **Q value** (cost-to-go) is the expected cost to first execute **action a** in **state s**, and **then follow an optimal policy** thereafter.

sum over outcome states s' from action a

$$Q^*(s,a) = \sum T(s,a,s')[C(s,a,s') + V^*(s')]$$

transition prob — action cost — value of state

Where the **optimal value function** is defined as

$$V^*(s) = 0 \ (\text{if } s \in G)$$
$$= \min Q^*(s,a) \ (\text{if } s \notin G)$$

Value Iteration

Bellman Equations

The **optimal policy** provides the **action a** to take in **state s** which minimises the cost to reach a goal state.

Optimal **Q value** (cost-to-go) is the expected cost to first execute **action a** in **state s**, and **then follow an optimal policy** thereafter.

sum over outcome states s' from action a

$$Q^*(s,a) = \sum T(s,a,s')[C(s,a,s') + V^*(s')]$$

transition prob — action cost — value of state

Where the **optimal value function** is defined as

$$V^*(s) = 0 \ (\text{if } s \in G)$$
$$= \min Q^*(s,a) \ (\text{if } s \notin G)$$

Value Iteration

Bellman Backup

Use dynamic programming to successively approximate V* with V_n

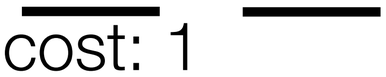
$$V_n(s) \leftarrow \min_{a \in A} \sum_{s' \in S} T(s, a, s')[C(s, a, s') + V_{n-1}(s')]$$

1 initialize V_0 arbitrarily for each state
2 $n \leftarrow 0$
3 **repeat**
4 $\quad n \leftarrow n + 1$
5 $\quad$ **foreach** $s \in \mathcal{S}$ **do**
6 $\quad\quad$ compute $V_n(s)$ using Bellman backup at s
7 $\quad\quad$ compute $\text{residual}_n(s) = |V_n(s) - V_{n-1}(s)|$
8 $\quad$ **end**
9 **until** $\max_{s \in \mathcal{S}} \text{residual}_n(s) < \epsilon$;
10 return greedy policy: $\pi^{V_n}(s) = \text{argmin}_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s') \left[\mathcal{C}(s, a, s') + V_n(s')\right]$

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

Value Iteration Example

C(s, a):



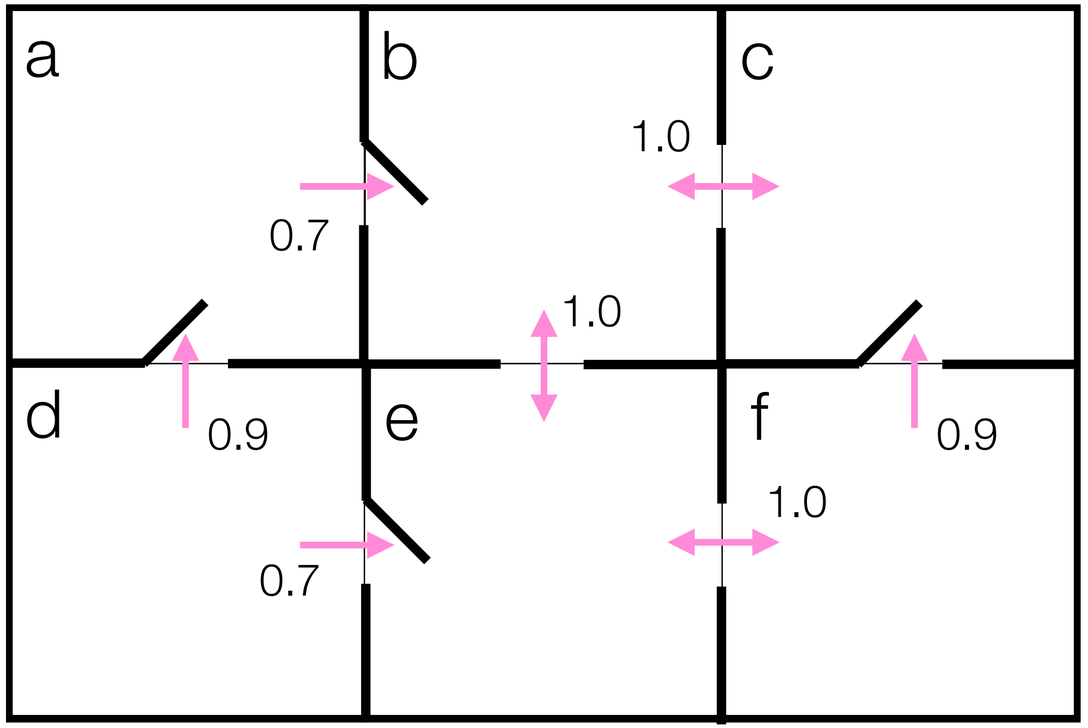
cost: 1

cost: 2

T(s, a, s’):

P(s’ | s, a) = 1

P(s’ | s, a) = d
P(s | s, a) = 1 - d



Value Iteration Example

C(s, a):

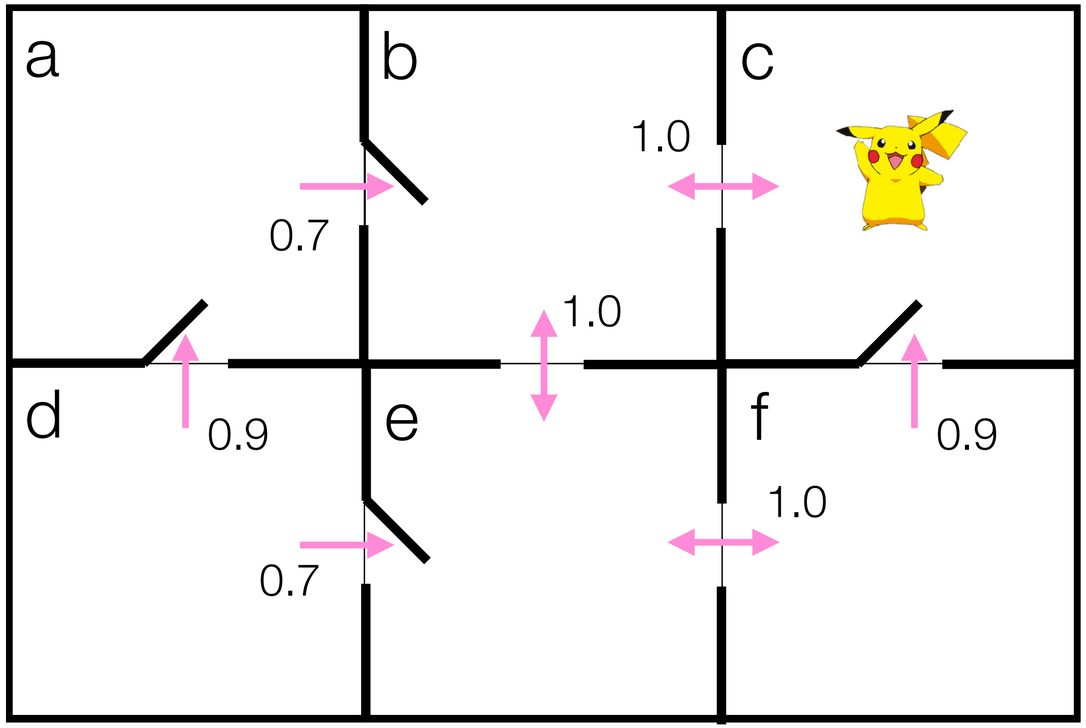
cost: 1

cost: 2

T(s, a, s'):

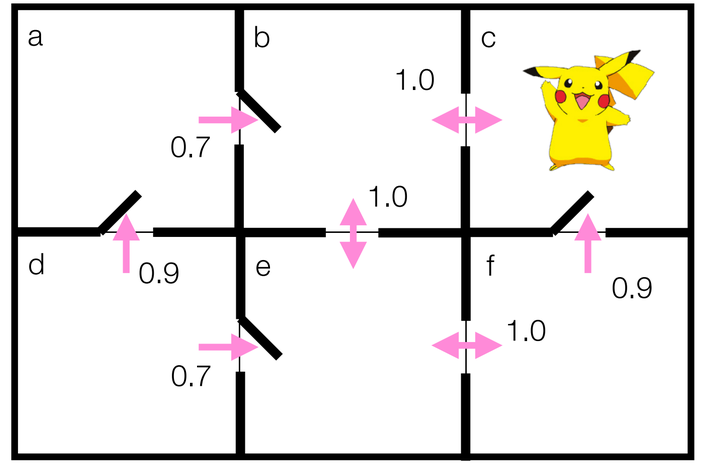
P(s' | s, a) = 1

P(s' | s, a) = d
P(s | s, a) = 1 - d

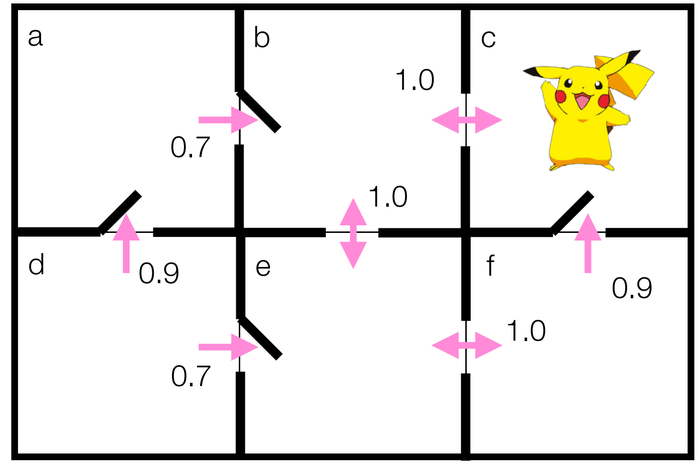


Value Iteration Example

	0
V(a)	100.000
Q(a, b)	100.000
V(b)	100.000
Q(b, c)	100.000
Q(b, e)	100.000
V(c)	0.000
Q(c, b)	100.000
V(d)	100.000
Q(d, a)	100.000
Q(d, e)	100.000
V(e)	100.000
Q(e, b)	100.000
Q(e, f)	100.000
V(f)	100.000
Q(f, c)	100.000
Q(f, e)	100.000



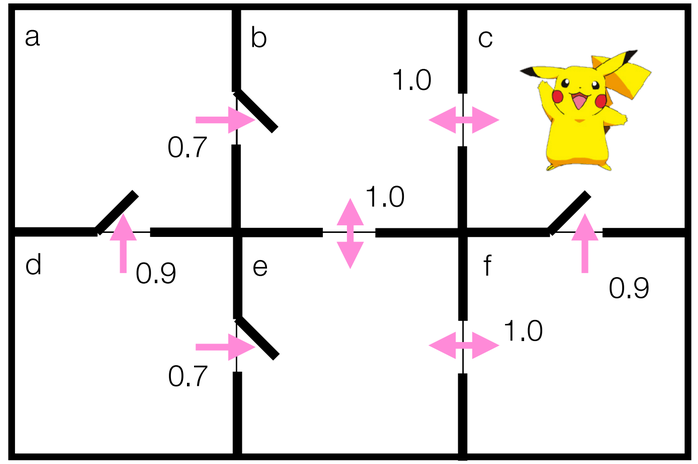
Value Iteration Example



$$\mathbf{Q_1(f, c)} = C(f, c) + (0.9 * \mathbf{V_0(c)}) + (0.1 * \mathbf{V_0(f)})$$

Value Iteration Example

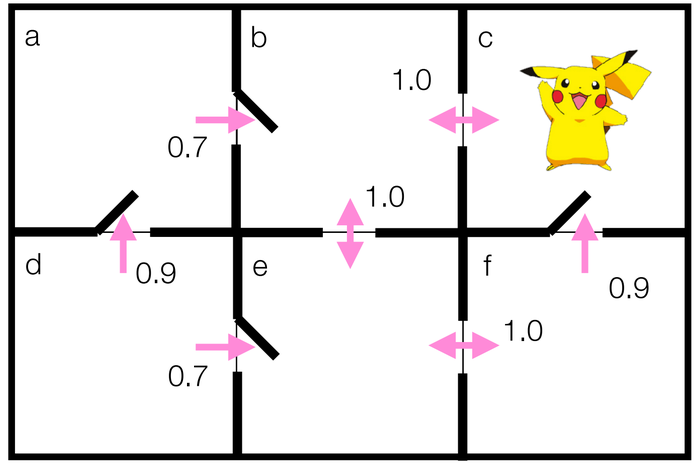
	0
V(a)	100.000
Q(a, b)	100.000
V(b)	100.000
Q(b, c)	100.000
Q(b, e)	100.000
V(c)	0.000
Q(c, b)	100.000
V(d)	100.000
Q(d, a)	100.000
Q(d, e)	100.000
V(e)	100.000
Q(e, b)	100.000
Q(e, f)	100.000
V(f)	100.000
Q(f, c)	100.000
Q(f, e)	100.000



$$\mathbf{Q_1(f, c)} = C(f, c) + (0.9 * \mathbf{V_0(c)}) + (0.1 * \mathbf{V_0(f)})$$

Value Iteration Example

	0
V(a)	100.000
Q(a, b)	100.000
V(b)	100.000
Q(b, c)	100.000
Q(b, e)	100.000
V(c)	0.000
Q(c, b)	100.000
V(d)	100.000
Q(d, a)	100.000
Q(d, e)	100.000
V(e)	100.000
Q(e, b)	100.000
Q(e, f)	100.000
V(f)	100.000
Q(f, c)	100.000
Q(f, e)	100.000

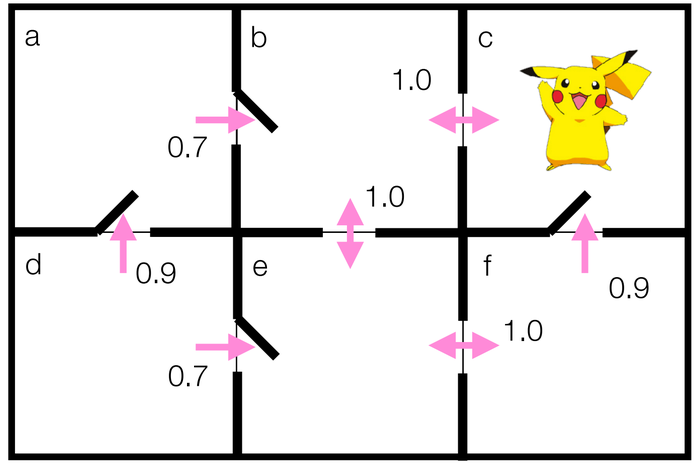


"cost to go"

$$\mathbf{Q_1(f, c)} = C(f, c) + (0.9 * \mathbf{V_0(c)}) + (0.1 * \mathbf{V_0(f)})$$

Value Iteration Example

	0
V(a)	100.000
Q(a, b)	100.000
V(b)	100.000
Q(b, c)	100.000
Q(b, e)	100.000
V(c)	0.000
Q(c, b)	100.000
V(d)	100.000
Q(d, a)	100.000
Q(d, e)	100.000
V(e)	100.000
Q(e, b)	100.000
Q(e, f)	100.000
V(f)	100.000
Q(f, c)	100.000
Q(f, e)	100.000



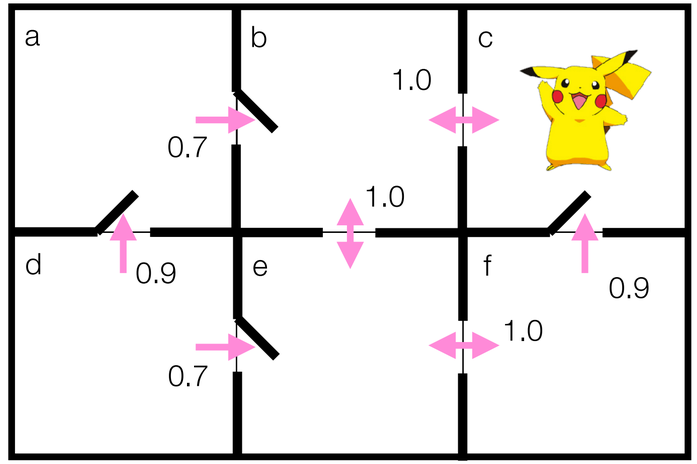
"cost to go"

$$\mathbf{Q_1(f, c)} = C(f, c) + (0.9 * \mathbf{V_0(c)}) + (0.1 * \mathbf{V_0(f)})$$

iteration

Value Iteration Example

	0
V(a)	100.000
Q(a, b)	100.000
V(b)	100.000
Q(b, c)	100.000
Q(b, e)	100.000
V(c)	0.000
Q(c, b)	100.000
V(d)	100.000
Q(d, a)	100.000
Q(d, e)	100.000
V(e)	100.000
Q(e, b)	100.000
Q(e, f)	100.000
V(f)	100.000
Q(f, c)	100.000
Q(f, e)	100.000



action cost

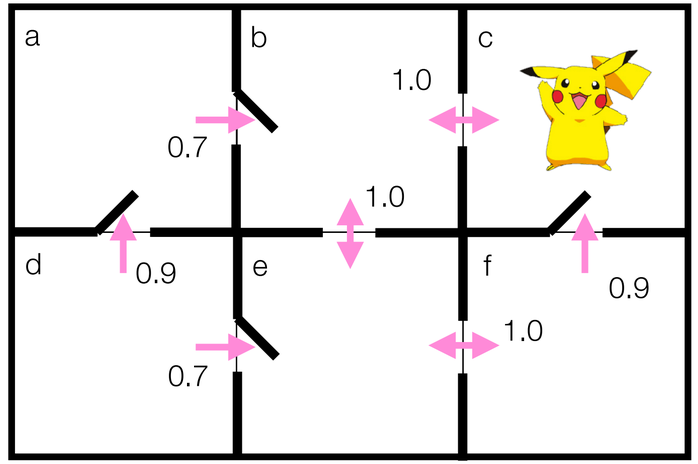
"cost to go"

$$\mathbf{Q_1(f, c)} = C(f, c) + (0.9 * \mathbf{V_0(c)}) + (0.1 * \mathbf{V_0(f)})$$

iteration

Value Iteration Example

	0
V(a)	100.000
Q(a, b)	100.000
V(b)	100.000
Q(b, c)	100.000
Q(b, e)	100.000
V(c)	0.000
Q(c, b)	100.000
V(d)	100.000
Q(d, a)	100.000
Q(d, e)	100.000
V(e)	100.000
Q(e, b)	100.000
Q(e, f)	100.000
V(f)	100.000
Q(f, c)	100.000
Q(f, e)	100.000



action cost

outcome prob

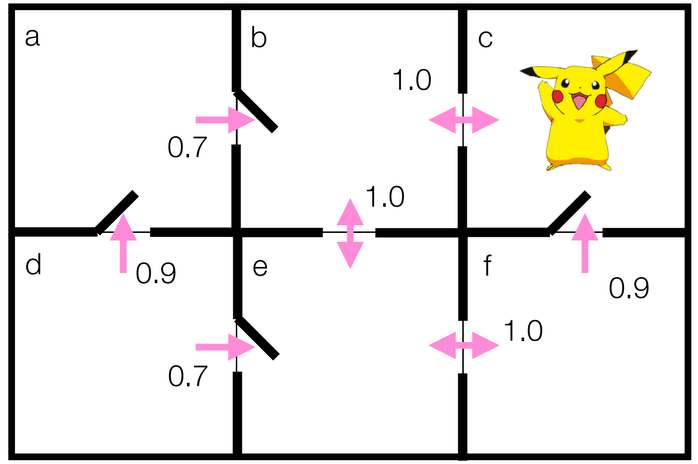
"cost to go"

$$\mathbf{Q_1(f, c)} = C(f, c) + (0.9 * \mathbf{V_0(c)}) + (0.1 * \mathbf{V_0(f)})$$

iteration

Value Iteration Example

	0
V(a)	100.000
Q(a, b)	100.000
V(b)	100.000
Q(b, c)	100.000
Q(b, e)	100.000
V(c)	0.000
Q(c, b)	100.000
V(d)	100.000
Q(d, a)	100.000
Q(d, e)	100.000
V(e)	100.000
Q(e, b)	100.000
Q(e, f)	100.000
V(f)	100.000
Q(f, c)	100.000
Q(f, e)	100.000



action cost

outcome prob

"cost to go"

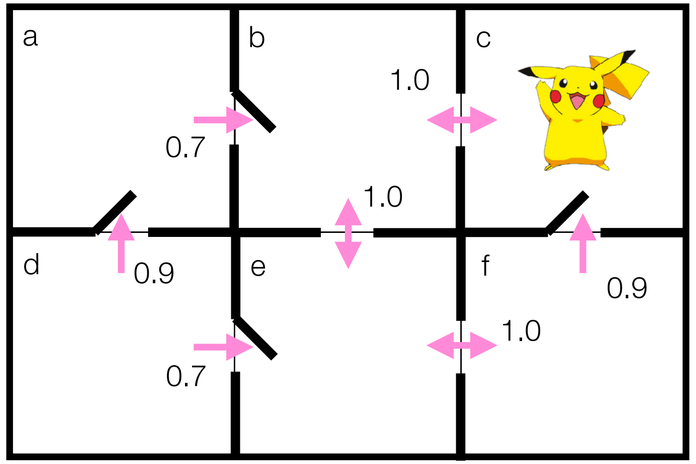
outcome value

$$\mathbf{Q_1(f, c)} = C(f, c) + (0.9 * \mathbf{V_0(c)}) + (0.1 * \mathbf{V_0(f)})$$

iteration

Value Iteration Example

	0
V(a)	100.000
Q(a, b)	100.000
V(b)	100.000
Q(b, c)	100.000
Q(b, e)	100.000
V(c)	0.000
Q(c, b)	100.000
V(d)	100.000
Q(d, a)	100.000
Q(d, e)	100.000
V(e)	100.000
Q(e, b)	100.000
Q(e, f)	100.000
V(f)	100.000
Q(f, c)	100.000
Q(f, e)	100.000



action cost

outcome prob

"cost to go"

outcome value

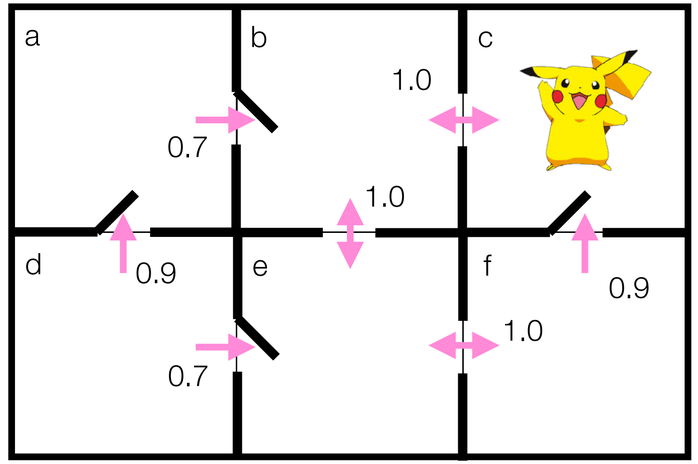
$$\mathbf{Q_1(f, c)} = C(f, c) + (0.9 * \mathbf{V_0(c)}) + (0.1 * \mathbf{V_0(f)})$$

iteration

$$= 2 + (0.9 * 0) + (0.1 * 100) = 12.000$$

Value Iteration Example

	0
V(a)	100.000
Q(a, b)	100.000
V(b)	100.000
Q(b, c)	100.000
Q(b, e)	100.000
V(c)	0.000
Q(c, b)	100.000
V(d)	100.000
Q(d, a)	100.000
Q(d, e)	100.000
V(e)	100.000
Q(e, b)	100.000
Q(e, f)	100.000
V(f)	100.000
Q(f, c)	100.000
Q(f, e)	100.000



action cost

outcome prob

"cost to go"

outcome value

$$\mathbf{Q_1(f, c)} = C(f, c) + (0.9 * \mathbf{V_0(c)}) + (0.1 * \mathbf{V_0(f)})$$

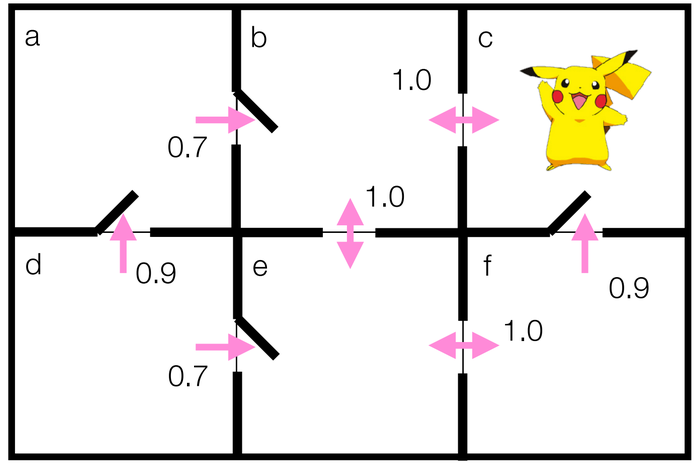
iteration

$$= 2 + (0.9 * 0) + (0.1 * 100) = 12.000$$

$$\mathbf{Q_1(f, e)} = C(f, e) + (1.0 * \mathbf{V_0(e)}) = 101.00$$

Value Iteration Example

	0
V(a)	100.000
Q(a, b)	100.000
V(b)	100.000
Q(b, c)	100.000
Q(b, e)	100.000
V(c)	0.000
Q(c, b)	100.000
V(d)	100.000
Q(d, a)	100.000
Q(d, e)	100.000
V(e)	100.000
Q(e, b)	100.000
Q(e, f)	100.000
V(f)	100.000
Q(f, c)	100.000
Q(f, e)	100.000



action cost

outcome prob

"cost to go"

outcome value

$$\mathbf{Q_1(f, c)} = C(f, c) + (0.9 * \mathbf{V_0(c)}) + (0.1 * \mathbf{V_0(f)})$$

iteration

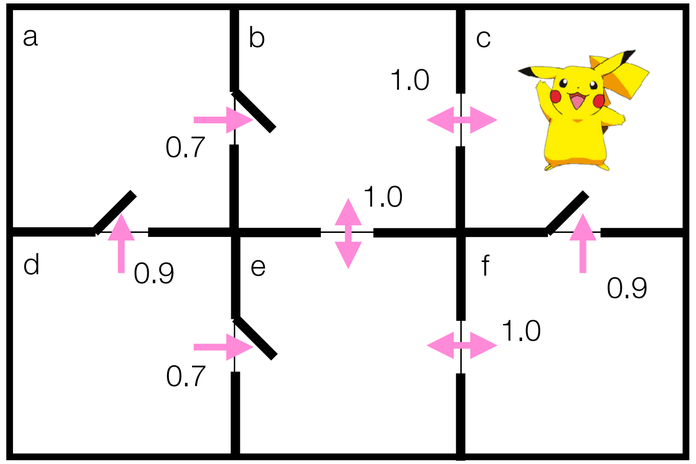
$$= 2 + (0.9 * 0) + (0.1 * 100) = 12.000$$

$$\mathbf{Q_1(f, e)} = C(f, e) + (1.0 * \mathbf{V_0(e)}) = 101.00$$

$$\mathbf{V_1(f)} = \textit{min}\ (\mathbf{Q_1(f, c)}, \mathbf{Q_1(f, e)}) = 12.000$$

Value Iteration Example

	0	1
V(a)	100.000	102.000
Q(a, b)	100.000	102.000
V(b)	100.000	1.000
Q(b, c)	100.000	1.000
Q(b, e)	100.000	101.000
V(c)	0.000	0.000
Q(c, b)	100.000	101.000
V(d)	100.000	102.000
Q(d, a)	100.000	102.000
Q(d, e)	100.000	102.000
V(e)	100.000	101.000
Q(e, b)	100.000	101.000
Q(e, f)	100.000	101.000
V(f)	100.000	12.000
Q(f, c)	100.000	12.000
Q(f, e)	100.000	101.000



action cost

outcome prob

"cost to go"

outcome value

$$\mathbf{Q_1(f, c)} = C(f, c) + (0.9 * \mathbf{V_0(c)}) + (0.1 * \mathbf{V_0(f)})$$

iteration

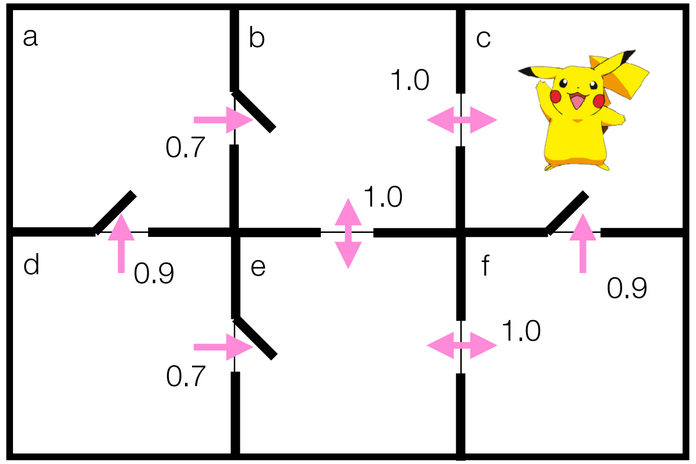
$$= 2 + (0.9 * 0) + (0.1 * 100) = 12.000$$

$$\mathbf{Q_1(f, e)} = C(f, e) + (1.0 * \mathbf{V_0(e)}) = 101.00$$

$$\mathbf{V_1(f)} = \textit{min}\ (\mathbf{Q_1(f, c)}, \mathbf{Q_1(f, e)}) = 12.000$$

Value Iteration Example

	0	1
V(a)	100.000	102.000
Q(a, b)	100.000	102.000
V(b)	100.000	1.000
Q(b, c)	100.000	1.000
Q(b, e)	100.000	101.000
V(c)	0.000	0.000
Q(c, b)	100.000	101.000
V(d)	100.000	102.000
Q(d, a)	100.000	102.000
Q(d, e)	100.000	102.000
V(e)	100.000	101.000
Q(e, b)	100.000	101.000
Q(e, f)	100.000	101.000
V(f)	100.000	12.000
Q(f, c)	100.000	12.000
Q(f, e)	100.000	101.000



action cost

outcome prob

"cost to go"

outcome value

$$\mathbf{Q_1(f, c)} = C(f, c) + (0.9 * \mathbf{V_0(c)}) + (0.1 * \mathbf{V_0(f)})$$

iteration

$$= 2 + (0.9 * 0) + (0.1 * 100) = 12.000$$

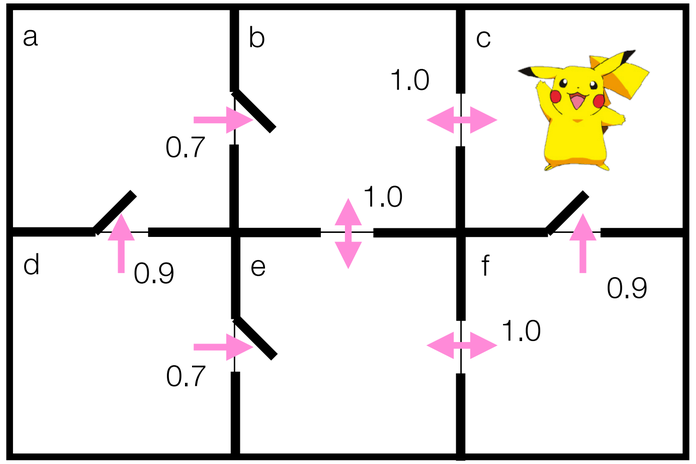
$$\mathbf{Q_1(f, e)} = C(f, e) + (1.0 * \mathbf{V_0(e)}) = 101.00$$

$$\mathbf{V_1(f)} = \textit{min}\ (\mathbf{Q_1(f, c)}, \mathbf{Q_1(f, e)}) = 12.000$$

$$\mathbf{Res_1(f)} = \mathbf{V_1(f)} - \mathbf{V_0(f)} = 100 - 12 = 88$$

Value Iteration Example

	0	1
V(a)	100.000	102.000
Q(a, b)	100.000	102.000
V(b)	100.000	1.000
Q(b, c)	100.000	1.000
Q(b, e)	100.000	101.000
V(c)	0.000	0.000
Q(c, b)	100.000	101.000
V(d)	100.000	102.000
Q(d, a)	100.000	102.000
Q(d, e)	100.000	102.000
V(e)	100.000	101.000
Q(e, b)	100.000	101.000
Q(e, f)	100.000	101.000
V(f)	100.000	12.000
Q(f, c)	100.000	12.000
Q(f, e)	100.000	101.000
Res(a)		2.000
Res(b)		99.000
Res(c)		0.000
Res(d)		2.000
Res(e)		1.000
Res(f)		88.000
MaxRes		99.000



action cost

outcome prob

"cost to go"

outcome value

$$\mathbf{Q_1(f, c)} = C(f, c) + (0.9 * \mathbf{V_0(c)}) + (0.1 * \mathbf{V_0(f)})$$

iteration

$$= 2 + (0.9 * 0) + (0.1 * 100) = 12.000$$

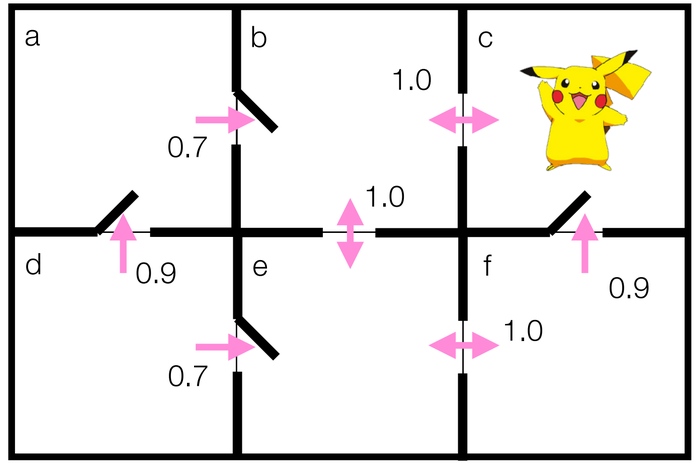
$$\mathbf{Q_1(f, e)} = C(f, e) + (1.0 * \mathbf{V_0(e)}) = 101.00$$

$$\mathbf{V_1(f)} = \textit{min}\ (\mathbf{Q_1(f, c)}, \mathbf{Q_1(f, e)}) = 12.000$$

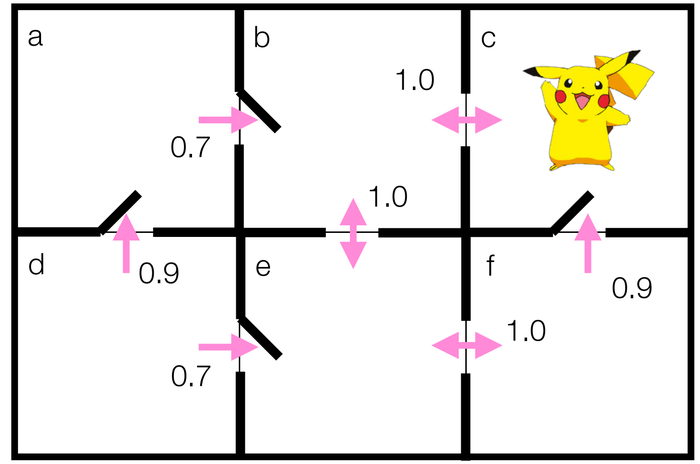
$$\mathbf{Res_1(f)} = \mathbf{V_1(f)} - \mathbf{V_0(f)} = 100 - 12 = 88$$

Value Iteration Example

	0	1
V(a)	100.000	102.000
Q(a, b)	100.000	102.000
V(b)	100.000	1.000
Q(b, c)	100.000	1.000
Q(b, e)	100.000	101.000
V(c)	0.000	0.000
Q(c, b)	100.000	101.000
V(d)	100.000	102.000
Q(d, a)	100.000	102.000
Q(d, e)	100.000	102.000
V(e)	100.000	101.000
Q(e, b)	100.000	101.000
Q(e, f)	100.000	101.000
V(f)	100.000	12.000
Q(f, c)	100.000	12.000
Q(f, e)	100.000	101.000
Res(a)		2.000
Res(b)		99.000
Res(c)		0.000
Res(d)		2.000
Res(e)		1.000
Res(f)		88.000
MaxRes		99.000

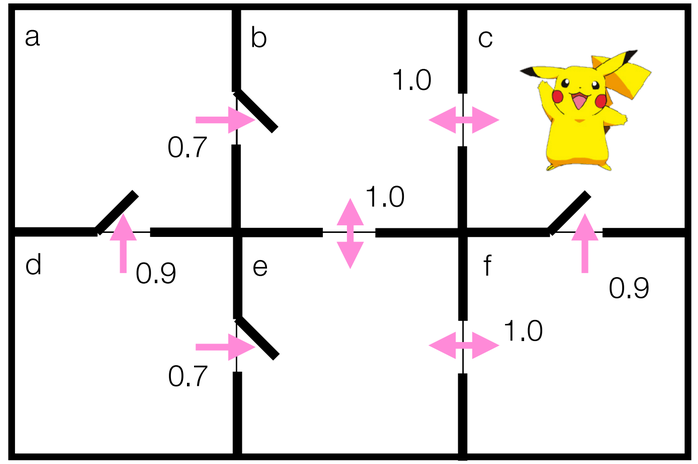


Value Iteration Example



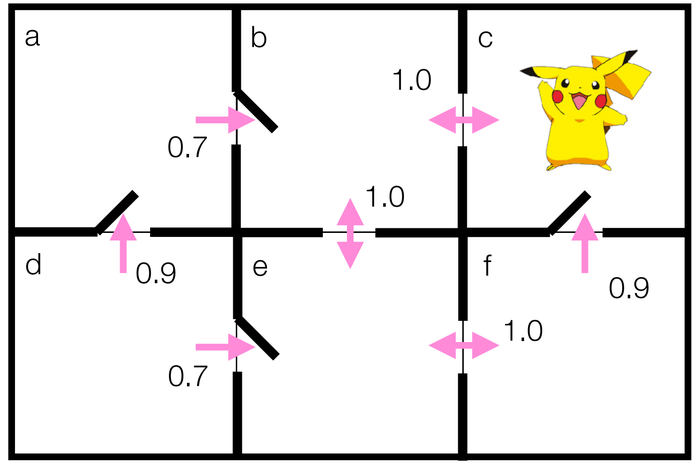
Value Iteration Example

	0	1
V(a)	100.000	102.000
Q(a, b)	100.000	102.000
V(b)	100.000	1.000
Q(b, c)	100.000	1.000
Q(b, e)	100.000	101.000
V(c)	0.000	0.000
Q(c, b)	100.000	101.000
V(d)	100.000	102.000
Q(d, a)	100.000	102.000
Q(d, e)	100.000	102.000
V(e)	100.000	101.000
Q(e, b)	100.000	101.000
Q(e, f)	100.000	101.000
V(f)	100.000	12.000
Q(f, c)	100.000	12.000
Q(f, e)	100.000	101.000
Res(a)		2.000
Res(b)		99.000
Res(c)		0.000
Res(d)		2.000
Res(e)		1.000
Res(f)		88.000
MaxRes		99.000



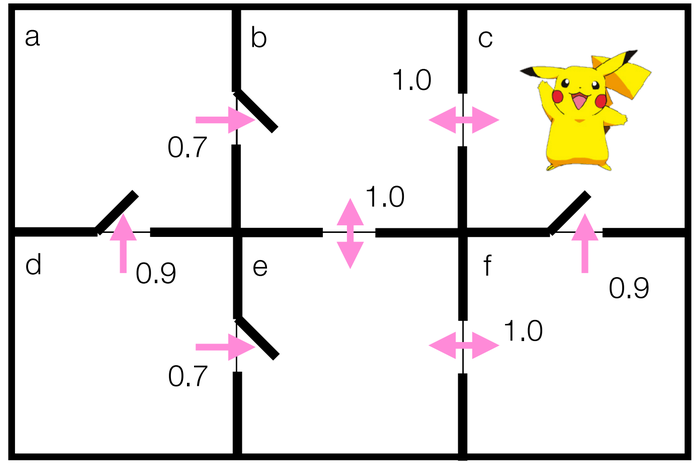
Value Iteration Example

	0	1	2
V(a)	100.000	102.000	33.300
Q(a, b)	100.000	102.000	33.300
V(b)	100.000	1.000	1.000
Q(b, c)	100.000	1.000	1.000
Q(b, e)	100.000	101.000	102.000
V(c)	0.000	0.000	0.000
Q(c, b)	100.000	101.000	2.000
V(d)	100.000	102.000	103.300
Q(d, a)	100.000	102.000	104.000
Q(d, e)	100.000	102.000	103.300
V(e)	100.000	101.000	2.000
Q(e, b)	100.000	101.000	2.000
Q(e, f)	100.000	101.000	13.000
V(f)	100.000	12.000	3.200
Q(f, c)	100.000	12.000	3.200
Q(f, e)	100.000	101.000	102.000
Res(a)		2.000	68.700
Res(b)		99.000	0.000
Res(c)		0.000	0.000
Res(d)		2.000	1.300
Res(e)		1.000	99.000
Res(f)		88.000	8.800
MaxRes		99.000	99.000



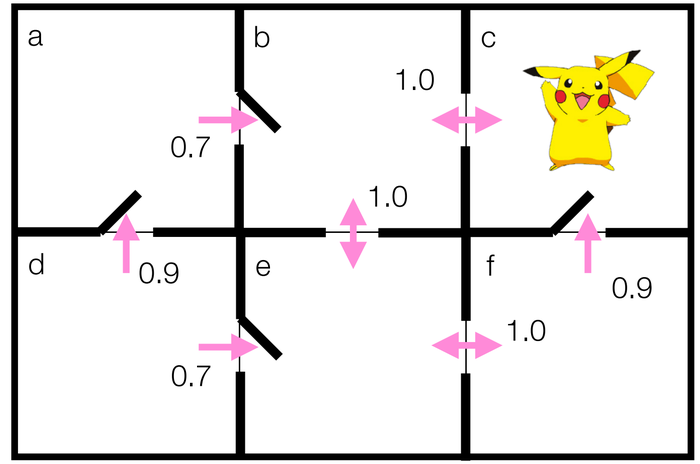
Value Iteration Example

	0	1	2	3
V(a)	100.000	102.000	33.300	12.690
Q(a, b)	100.000	102.000	33.300	12.690
V(b)	100.000	1.000	1.000	1.000
Q(b, c)	100.000	1.000	1.000	1.000
Q(b, e)	100.000	101.000	102.000	3.000
V(c)	0.000	0.000	0.000	0.000
Q(c, b)	100.000	101.000	2.000	2.000
V(d)	100.000	102.000	103.300	34.390
Q(d, a)	100.000	102.000	104.000	42.300
Q(d, e)	100.000	102.000	103.300	34.390
V(e)	100.000	101.000	2.000	2.000
Q(e, b)	100.000	101.000	2.000	2.000
Q(e, f)	100.000	101.000	13.000	4.200
V(f)	100.000	12.000	3.200	2.320
Q(f, c)	100.000	12.000	3.200	2.320
Q(f, e)	100.000	101.000	102.000	3.000
Res(a)		2.000	68.700	20.610
Res(b)		99.000	0.000	0.000
Res(c)		0.000	0.000	0.000
Res(d)		2.000	1.300	68.910
Res(e)		1.000	99.000	0.000
Res(f)		88.000	8.800	0.880
MaxRes		99.000	99.000	68.910



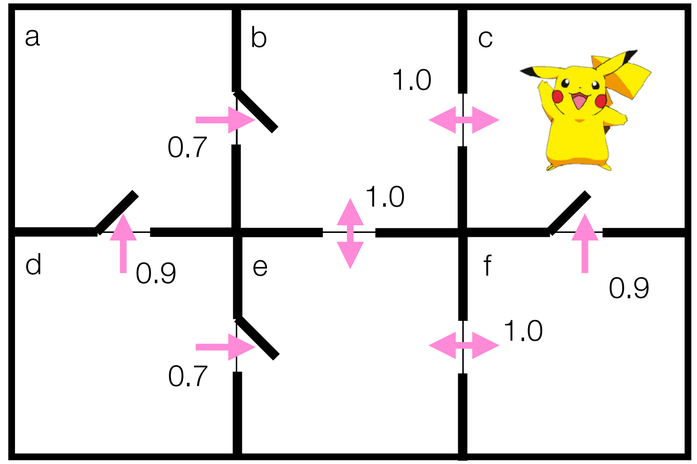
Value Iteration Example

	0	1	2	3	4
V(a)	100.000	102.000	33.300	12.690	6.507
Q(a, b)	100.000	102.000	33.300	12.690	6.507
V(b)	100.000	1.000	1.000	1.000	1.000
Q(b, c)	100.000	1.000	1.000	1.000	1.000
Q(b, e)	100.000	101.000	102.000	3.000	3.000
V(c)	0.000	0.000	0.000	0.000	0.000
Q(c, b)	100.000	101.000	2.000	2.000	2.000
V(d)	100.000	102.000	103.300	34.390	13.717
Q(d, a)	100.000	102.000	104.000	42.300	16.860
Q(d, e)	100.000	102.000	103.300	34.390	13.717
V(e)	100.000	101.000	2.000	2.000	2.000
Q(e, b)	100.000	101.000	2.000	2.000	2.000
Q(e, f)	100.000	101.000	13.000	4.200	3.320
V(f)	100.000	12.000	3.200	2.320	2.232
Q(f, c)	100.000	12.000	3.200	2.320	2.232
Q(f, e)	100.000	101.000	102.000	3.000	3.000
Res(a)		2.000	68.700	20.610	6.183
Res(b)		99.000	0.000	0.000	0.000
Res(c)		0.000	0.000	0.000	0.000
Res(d)		2.000	1.300	68.910	20.673
Res(e)		1.000	99.000	0.000	0.000
Res(f)		88.000	8.800	0.880	0.088
MaxRes		99.000	99.000	68.910	20.673



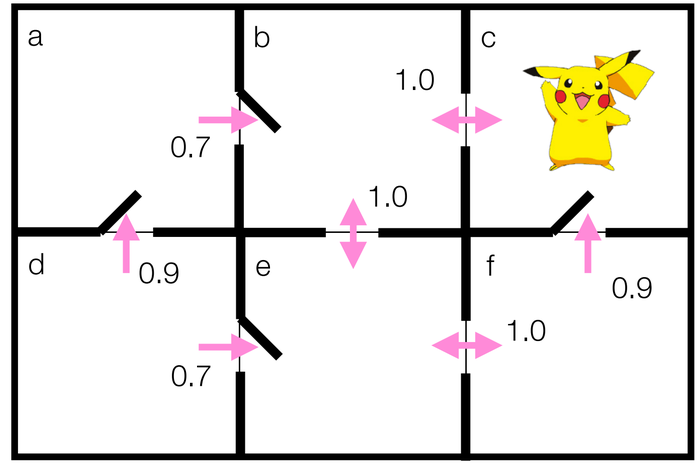
Value Iteration Example

	0	1	2	3	4	5
V(a)	100.000	102.000	33.300	12.690	6.507	4.652
Q(a, b)	100.000	102.000	33.300	12.690	6.507	4.652
V(b)	100.000	1.000	1.000	1.000	1.000	1.000
Q(b, c)	100.000	1.000	1.000	1.000	1.000	1.000
Q(b, e)	100.000	101.000	102.000	3.000	3.000	3.000
V(c)	0.000	0.000	0.000	0.000	0.000	0.000
Q(c, b)	100.000	101.000	2.000	2.000	2.000	2.000
V(d)	100.000	102.000	103.300	34.390	13.717	7.515
Q(d, a)	100.000	102.000	104.000	42.300	16.860	9.228
Q(d, e)	100.000	102.000	103.300	34.390	13.717	7.515
V(e)	100.000	101.000	2.000	2.000	2.000	2.000
Q(e, b)	100.000	101.000	2.000	2.000	2.000	2.000
Q(e, f)	100.000	101.000	13.000	4.200	3.320	3.232
V(f)	100.000	12.000	3.200	2.320	2.232	2.223
Q(f, c)	100.000	12.000	3.200	2.320	2.232	2.223
Q(f, e)	100.000	101.000	102.000	3.000	3.000	3.000
Res(a)		2.000	68.700	20.610	6.183	1.855
Res(b)		99.000	0.000	0.000	0.000	0.000
Res(c)		0.000	0.000	0.000	0.000	0.000
Res(d)		2.000	1.300	68.910	20.673	6.202
Res(e)		1.000	99.000	0.000	0.000	0.000
Res(f)		88.000	8.800	0.880	0.088	0.009
MaxRes		99.000	99.000	68.910	20.673	6.202



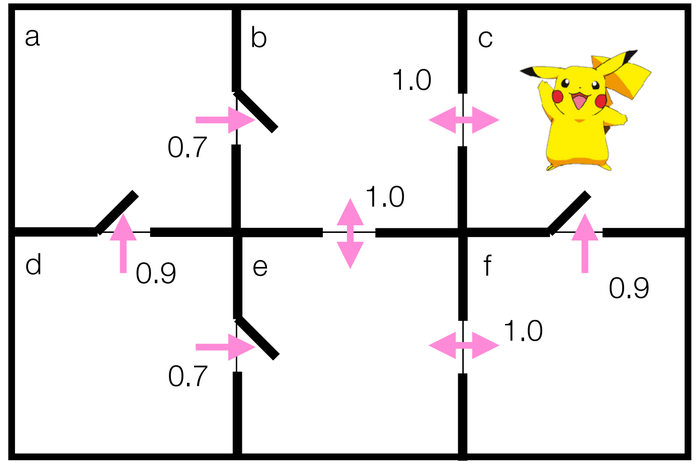
Value Iteration Example

	0	1	2	3	4	5	6
V(a)	100.000	102.000	33.300	12.690	6.507	4.652	4.096
Q(a, b)	100.000	102.000	33.300	12.690	6.507	4.652	4.096
V(b)	100.000	1.000	1.000	1.000	1.000	1.000	1.000
Q(b, c)	100.000	1.000	1.000	1.000	1.000	1.000	1.000
Q(b, e)	100.000	101.000	102.000	3.000	3.000	3.000	3.000
V(c)	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Q(c, b)	100.000	101.000	2.000	2.000	2.000	2.000	2.000
V(d)	100.000	102.000	103.300	34.390	13.717	7.515	5.655
Q(d, a)	100.000	102.000	104.000	42.300	16.860	9.228	6.938
Q(d, e)	100.000	102.000	103.300	34.390	13.717	7.515	5.655
V(e)	100.000	101.000	2.000	2.000	2.000	2.000	2.000
Q(e, b)	100.000	101.000	2.000	2.000	2.000	2.000	2.000
Q(e, f)	100.000	101.000	13.000	4.200	3.320	3.232	3.223
V(f)	100.000	12.000	3.200	2.320	2.232	2.223	2.222
Q(f, c)	100.000	12.000	3.200	2.320	2.232	2.223	2.222
Q(f, e)	100.000	101.000	102.000	3.000	3.000	3.000	3.000
Res(a)		2.000	68.700	20.610	6.183	1.855	0.556
Res(b)		99.000	0.000	0.000	0.000	0.000	0.000
Res(c)		0.000	0.000	0.000	0.000	0.000	0.000
Res(d)		2.000	1.300	68.910	20.673	6.202	1.861
Res(e)		1.000	99.000	0.000	0.000	0.000	0.000
Res(f)		88.000	8.800	0.880	0.088	0.009	0.001
MaxRes		99.000	99.000	68.910	20.673	6.202	1.861



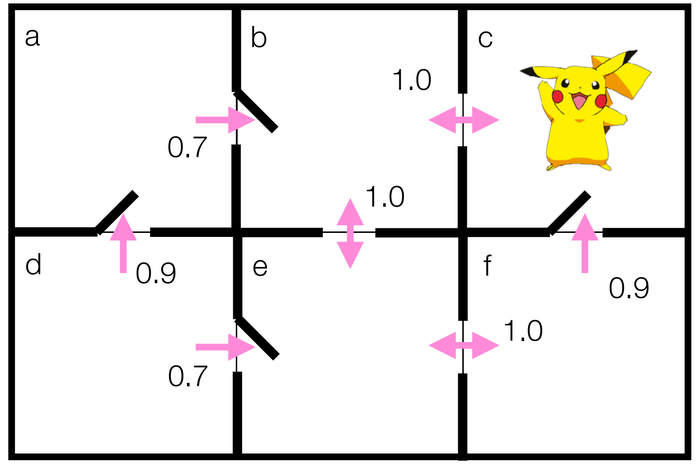
Value Iteration Example

	0	1	2	3	4	5	6	7
V(a)	100.000	102.000	33.300	12.690	6.507	4.652	4.096	3.929
Q(a, b)	100.000	102.000	33.300	12.690	6.507	4.652	4.096	3.929
V(b)	100.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Q(b, c)	100.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Q(b, e)	100.000	101.000	102.000	3.000	3.000	3.000	3.000	3.000
V(c)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Q(c, b)	100.000	101.000	2.000	2.000	2.000	2.000	2.000	2.000
V(d)	100.000	102.000	103.300	34.390	13.717	7.515	5.655	5.096
Q(d, a)	100.000	102.000	104.000	42.300	16.860	9.228	6.938	6.252
Q(d, e)	100.000	102.000	103.300	34.390	13.717	7.515	5.655	5.096
V(e)	100.000	101.000	2.000	2.000	2.000	2.000	2.000	2.000
Q(e, b)	100.000	101.000	2.000	2.000	2.000	2.000	2.000	2.000
Q(e, f)	100.000	101.000	13.000	4.200	3.320	3.232	3.223	3.222
V(f)	100.000	12.000	3.200	2.320	2.232	2.223	2.222	2.222
Q(f, c)	100.000	12.000	3.200	2.320	2.232	2.223	2.222	2.222
Q(f, e)	100.000	101.000	102.000	3.000	3.000	3.000	3.000	3.000
Res(a)		2.000	68.700	20.610	6.183	1.855	0.556	0.167
Res(b)		99.000	0.000	0.000	0.000	0.000	0.000	0.000
Res(c)		0.000	0.000	0.000	0.000	0.000	0.000	0.000
Res(d)		2.000	1.300	68.910	20.673	6.202	1.861	0.558
Res(e)		1.000	99.000	0.000	0.000	0.000	0.000	0.000
Res(f)		88.000	8.800	0.880	0.088	0.009	0.001	0.000
MaxRes		99.000	99.000	68.910	20.673	6.202	1.861	0.558



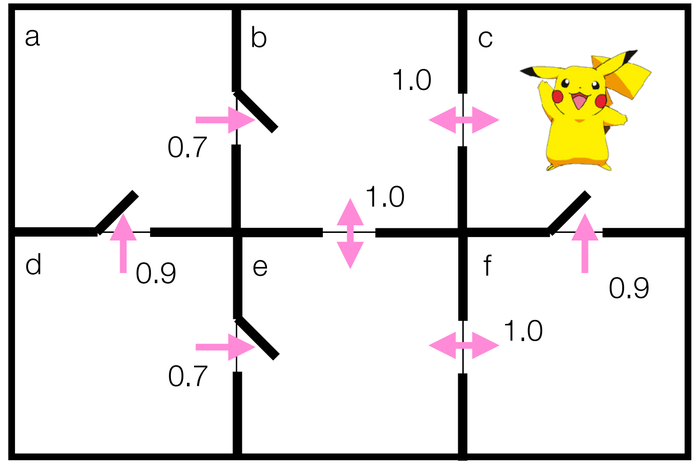
Value Iteration Example

	0	1	2	3	4	5	6	7	8
V(a)	100.000	102.000	33.300	12.690	6.507	4.652	4.096	3.929	3.879
Q(a, b)	100.000	102.000	33.300	12.690	6.507	4.652	4.096	3.929	3.879
V(b)	100.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Q(b, c)	100.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Q(b, e)	100.000	101.000	102.000	3.000	3.000	3.000	3.000	3.000	3.000
V(c)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Q(c, b)	100.000	101.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000
V(d)	100.000	102.000	103.300	34.390	13.717	7.515	5.655	5.096	4.929
Q(d, a)	100.000	102.000	104.000	42.300	16.860	9.228	6.938	6.252	6.045
Q(d, e)	100.000	102.000	103.300	34.390	13.717	7.515	5.655	5.096	4.929
V(e)	100.000	101.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000
Q(e, b)	100.000	101.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000
Q(e, f)	100.000	101.000	13.000	4.200	3.320	3.232	3.223	3.222	3.222
V(f)	100.000	12.000	3.200	2.320	2.232	2.223	2.222	2.222	2.222
Q(f, c)	100.000	12.000	3.200	2.320	2.232	2.223	2.222	2.222	2.222
Q(f, e)	100.000	101.000	102.000	3.000	3.000	3.000	3.000	3.000	3.000
Res(a)		2.000	68.700	20.610	6.183	1.855	0.556	0.167	0.050
Res(b)		99.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Res(c)		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Res(d)		2.000	1.300	68.910	20.673	6.202	1.861	0.558	0.167
Res(e)		1.000	99.000	0.000	0.000	0.000	0.000	0.000	0.000
Res(f)		88.000	8.800	0.880	0.088	0.009	0.001	0.000	0.000
MaxRes		99.000	99.000	68.910	20.673	6.202	1.861	0.558	0.167



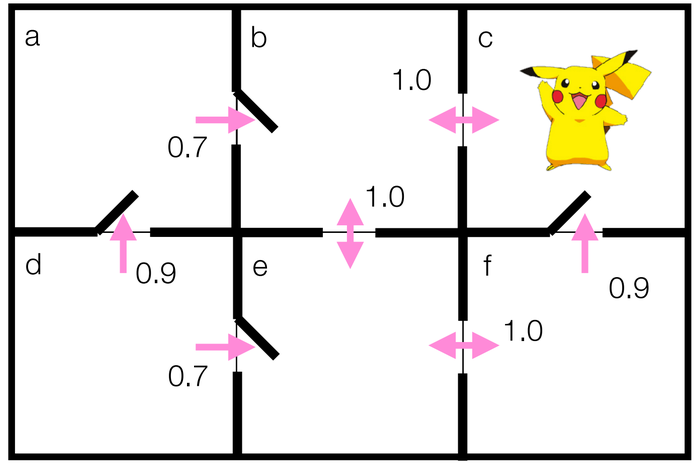
Value Iteration Example

	0	1	2	3	4	5	6	7	8	9
V(a)	100.000	102.000	33.300	12.690	6.507	4.652	4.096	3.929	3.879	3.864
Q(a, b)	100.000	102.000	33.300	12.690	6.507	4.652	4.096	3.929	3.879	3.864
V(b)	100.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Q(b, c)	100.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Q(b, e)	100.000	101.000	102.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000
V(c)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Q(c, b)	100.000	101.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000
V(d)	100.000	102.000	103.300	34.390	13.717	7.515	5.655	5.096	4.929	4.879
Q(d, a)	100.000	102.000	104.000	42.300	16.860	9.228	6.938	6.252	6.045	5.984
Q(d, e)	100.000	102.000	103.300	34.390	13.717	7.515	5.655	5.096	4.929	4.879
V(e)	100.000	101.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000
Q(e, b)	100.000	101.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000
Q(e, f)	100.000	101.000	13.000	4.200	3.320	3.232	3.223	3.222	3.222	3.222
V(f)	100.000	12.000	3.200	2.320	2.232	2.223	2.222	2.222	2.222	2.222
Q(f, c)	100.000	12.000	3.200	2.320	2.232	2.223	2.222	2.222	2.222	2.222
Q(f, e)	100.000	101.000	102.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000
Res(a)		2.000	68.700	20.610	6.183	1.855	0.556	0.167	0.050	0.015
Res(b)		99.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Res(c)		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Res(d)		2.000	1.300	68.910	20.673	6.202	1.861	0.558	0.167	0.050
Res(e)		1.000	99.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Res(f)		88.000	8.800	0.880	0.088	0.009	0.001	0.000	0.000	0.000
MaxRes		99.000	99.000	68.910	20.673	6.202	1.861	0.558	0.167	0.050



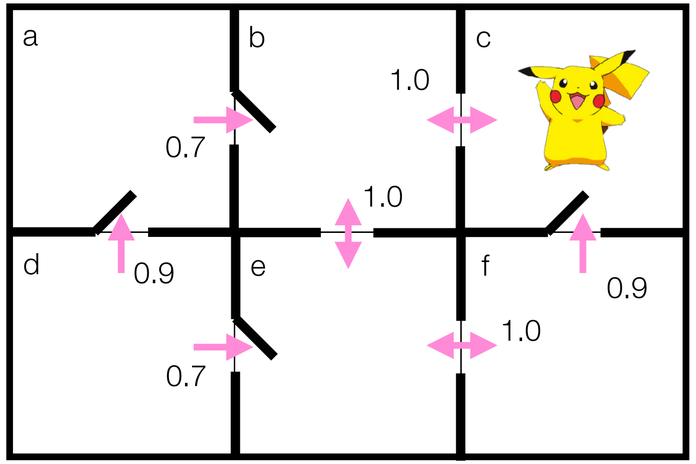
Value Iteration Example

	0	1	2	3	4	5	6	7	8	9	10
V(a)	100.000	102.000	33.300	12.690	6.507	4.652	4.096	3.929	3.879	3.864	3.859
Q(a, b)	100.000	102.000	33.300	12.690	6.507	4.652	4.096	3.929	3.879	3.864	3.859
V(b)	100.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Q(b, c)	100.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Q(b, e)	100.000	101.000	102.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000
V(c)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Q(c, b)	100.000	101.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000
V(d)	100.000	102.000	103.300	34.390	13.717	7.515	5.655	5.096	4.929	4.879	4.864
Q(d, a)	100.000	102.000	104.000	42.300	16.860	9.228	6.938	6.252	6.045	5.984	5.965
Q(d, e)	100.000	102.000	103.300	34.390	13.717	7.515	5.655	5.096	4.929	4.879	4.864
V(e)	100.000	101.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000
Q(e, b)	100.000	101.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000
Q(e, f)	100.000	101.000	13.000	4.200	3.320	3.232	3.223	3.222	3.222	3.222	3.222
V(f)	100.000	12.000	3.200	2.320	2.232	2.223	2.222	2.222	2.222	2.222	2.222
Q(f, c)	100.000	12.000	3.200	2.320	2.232	2.223	2.222	2.222	2.222	2.222	2.222
Q(f, e)	100.000	101.000	102.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000
Res(a)		2.000	68.700	20.610	6.183	1.855	0.556	0.167	0.050	0.015	0.005
Res(b)		99.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Res(c)		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Res(d)		2.000	1.300	68.910	20.673	6.202	1.861	0.558	0.167	0.050	0.015
Res(e)		1.000	99.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Res(f)		88.000	8.800	0.880	0.088	0.009	0.001	0.000	0.000	0.000	0.000
MaxRes		99.000	99.000	68.910	20.673	6.202	1.861	0.558	0.167	0.050	0.015



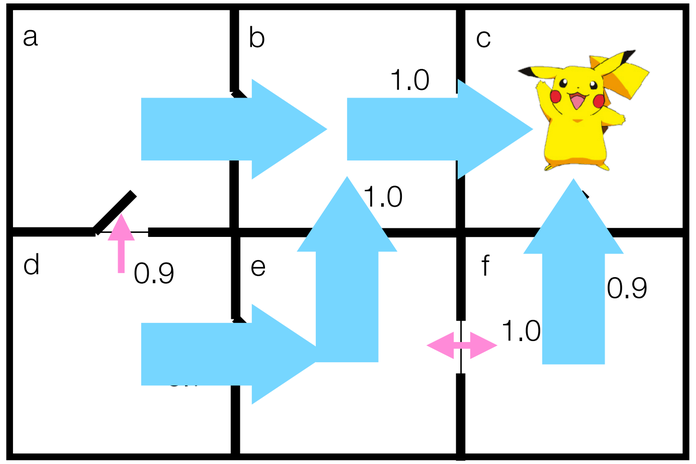
Value Iteration Example

	0	1	2	3	4	5	6	7	8	9	10	11
V(a)	100.000	102.000	33.300	12.690	6.507	4.652	4.096	3.929	3.879	3.864	3.859	3.858
Q(a, b)	100.000	102.000	33.300	12.690	6.507	4.652	4.096	3.929	3.879	3.864	3.859	3.858
V(b)	100.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Q(b, c)	100.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Q(b, e)	100.000	101.000	102.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000
V(c)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Q(c, b)	100.000	101.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000
V(d)	100.000	102.000	103.300	34.390	13.717	7.515	5.655	5.096	4.929	4.879	4.864	4.859
Q(d, a)	100.000	102.000	104.000	42.300	16.860	9.228	6.938	6.252	6.045	5.984	5.965	5.960
Q(d, e)	100.000	102.000	103.300	34.390	13.717	7.515	5.655	5.096	4.929	4.879	4.864	4.859
V(e)	100.000	101.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000
Q(e, b)	100.000	101.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000
Q(e, f)	100.000	101.000	13.000	4.200	3.320	3.232	3.223	3.222	3.222	3.222	3.222	3.222
V(f)	100.000	12.000	3.200	2.320	2.232	2.223	2.222	2.222	2.222	2.222	2.222	2.222
Q(f, c)	100.000	12.000	3.200	2.320	2.232	2.223	2.222	2.222	2.222	2.222	2.222	2.222
Q(f, e)	100.000	101.000	102.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000
Res(a)		2.000	68.700	20.610	6.183	1.855	0.556	0.167	0.050	0.015	0.005	0.001
Res(b)		99.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Res(c)		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Res(d)		2.000	1.300	68.910	20.673	6.202	1.861	0.558	0.167	0.050	0.015	0.005
Res(e)		1.000	99.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Res(f)		88.000	8.800	0.880	0.088	0.009	0.001	0.000	0.000	0.000	0.000	0.000
MaxRes		99.000	99.000	68.910	20.673	6.202	1.861	0.558	0.167	0.050	0.015	0.005



Value Iteration Example

	0	1	2	3	4	5	6	7	8	9	10	11
V(a)	100.000	102.000	33.300	12.690	6.507	4.652	4.096	3.929	3.879	3.864	3.859	3.858
Q(a, b)	100.000	102.000	33.300	12.690	6.507	4.652	4.096	3.929	3.879	3.864	3.859	3.858
V(b)	100.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Q(b, c)	100.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Q(b, e)	100.000	101.000	102.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000
V(c)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Q(c, b)	100.000	101.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000
V(d)	100.000	102.000	103.300	34.390	13.717	7.515	5.655	5.096	4.929	4.879	4.864	4.859
Q(d, a)	100.000	102.000	104.000	42.300	16.860	9.228	6.938	6.252	6.045	5.984	5.965	5.960
Q(d, e)	100.000	102.000	103.300	34.390	13.717	7.515	5.655	5.096	4.929	4.879	4.864	4.859
V(e)	100.000	101.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000
Q(e, b)	100.000	101.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000
Q(e, f)	100.000	101.000	13.000	4.200	3.320	3.232	3.223	3.222	3.222	3.222	3.222	3.222
V(f)	100.000	12.000	3.200	2.320	2.232	2.223	2.222	2.222	2.222	2.222	2.222	2.222
Q(f, c)	100.000	12.000	3.200	2.320	2.232	2.223	2.222	2.222	2.222	2.222	2.222	2.222
Q(f, e)	100.000	101.000	102.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000
Res(a)		2.000	68.700	20.610	6.183	1.855	0.556	0.167	0.050	0.015	0.005	0.001
Res(b)		99.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Res(c)		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Res(d)		2.000	1.300	68.910	20.673	6.202	1.861	0.558	0.167	0.050	0.015	0.005
Res(e)		1.000	99.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Res(f)		88.000	8.800	0.880	0.088	0.009	0.001	0.000	0.000	0.000	0.000	0.000
MaxRes		99.000	99.000	68.910	20.673	6.202	1.861	0.558	0.167	0.050	0.015	0.005



Create the ***greedy*** **policy** by choosing the actions which minimise the **Q** values

The policy is greedy with respect to **Q** and **V**, but **V** is exactly optimal, so the policy is **optimal**

Value Iteration

Value iteration is optimal (in the limit) and finds solutions from all states (not just initial), so it exhaustively solves the MDP.

Complexity of a single iteration of VI: $\mathcal{O}(|S|^2|A|)$

For large problems VI becomes intractable. It suffers particularly on problems with **large branching factors** (due to having to enumerate state-action pairs). For example, multi-robot planning problems.

Beyond Value Iteration

For large problems VI becomes intractable. It suffers particularly on problems with **large branching factors** (due to having to enumerate state-action pairs). For example, multi-robot planning problems.

VI is a good baseline approach for many MDPs, but for large SSPs it ignores two useful bits of information:

Beyond Value Iteration

For large problems VI becomes intractable. It suffers particularly on problems with **large branching factors** (due to having to enumerate state-action pairs). For example, multi-robot planning problems.

VI is a good baseline approach for many MDPs, but for large SSPs it ignores two useful bits of information:

- we typically have an **initial state** s_0

Beyond Value Iteration

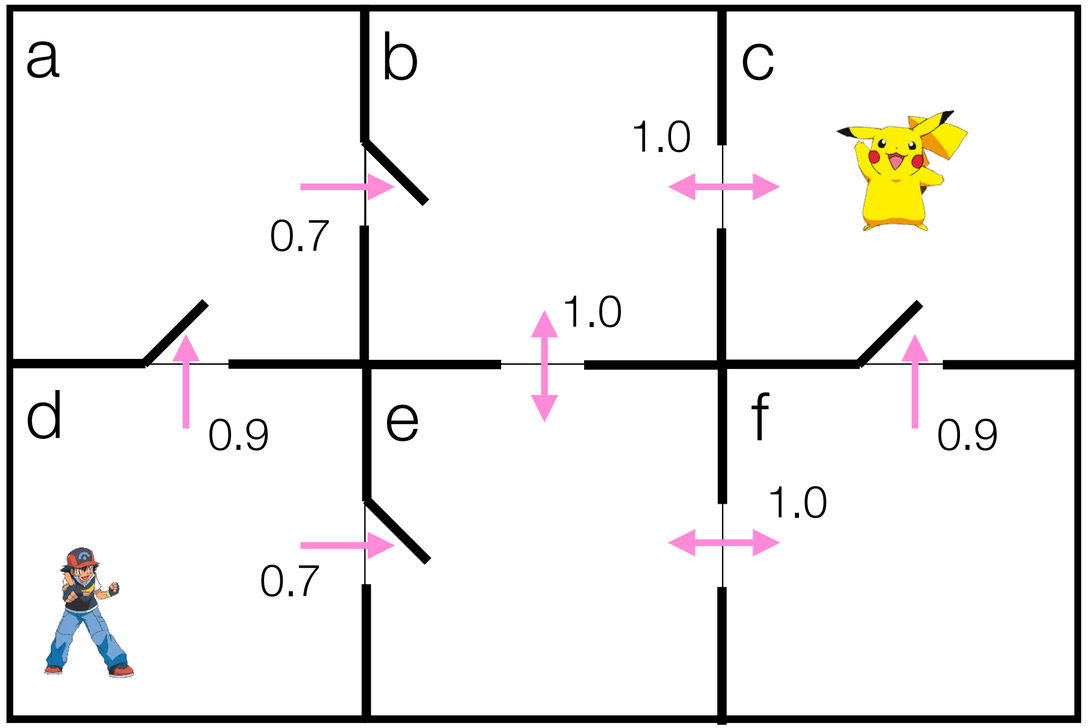
For large problems VI becomes intractable. It suffers particularly on problems with **large branching factors** (due to having to enumerate state-action pairs). For example, multi-robot planning problems.

VI is a good baseline approach for many MDPs, but for large SSPs it ignores two useful bits of information:

- we typically have an **initial state** s_0
- we can use a **heuristic** to *prioritise* the states we run Bellman backups for, and to *initialise* the value function in an informed manner

MDP Heuristic - All Outcomes Determinisation

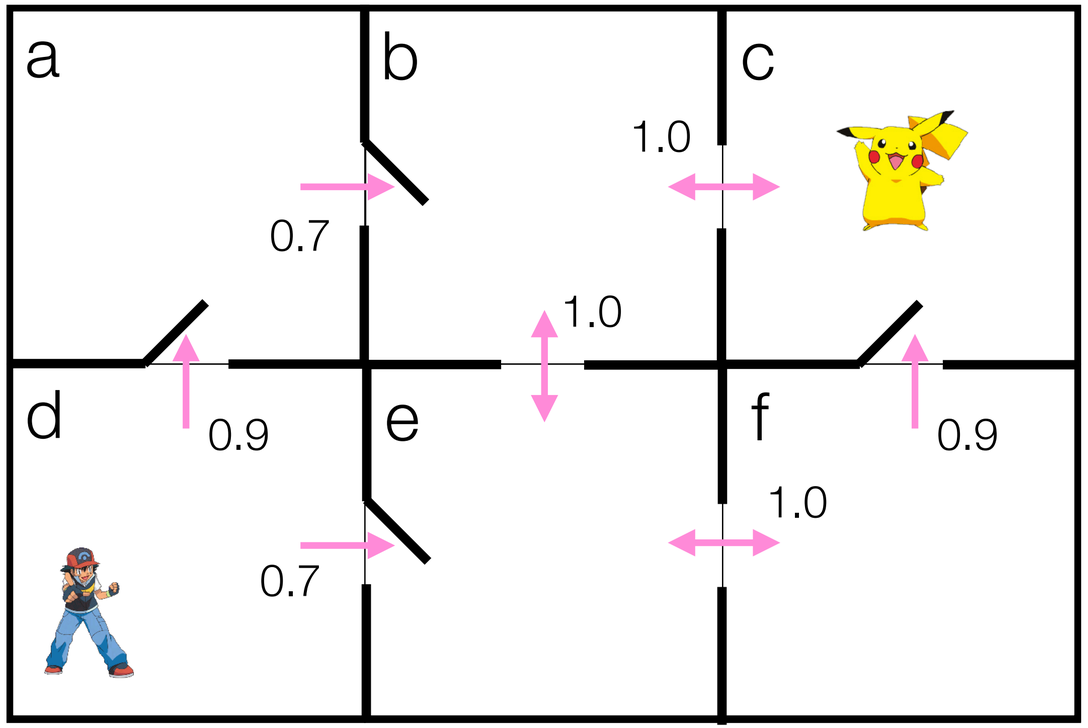
Create a ***relaxed*** version of the problem



MDP Heuristic - All Outcomes Determinisation

Create a ***relaxed*** version of the problem

We will use
All-Outcomes Determinisation



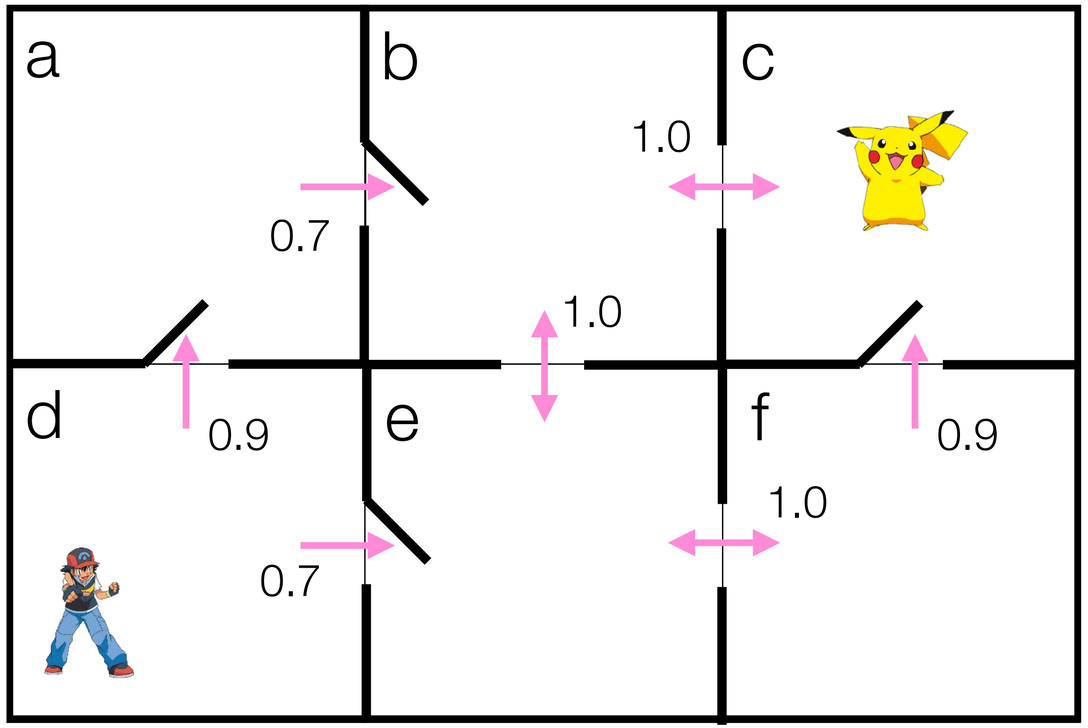
MDP Heuristic - All Outcomes Determinisation

Create a ***relaxed*** version of the problem

We will use
All-Outcomes Determinisation

Original:

a_b: 0.7 -> b, 0.3 -> a, cost 2



MDP Heuristic - All Outcomes Determinisation

Create a ***relaxed*** version of the problem

We will use
All-Outcomes Determinisation

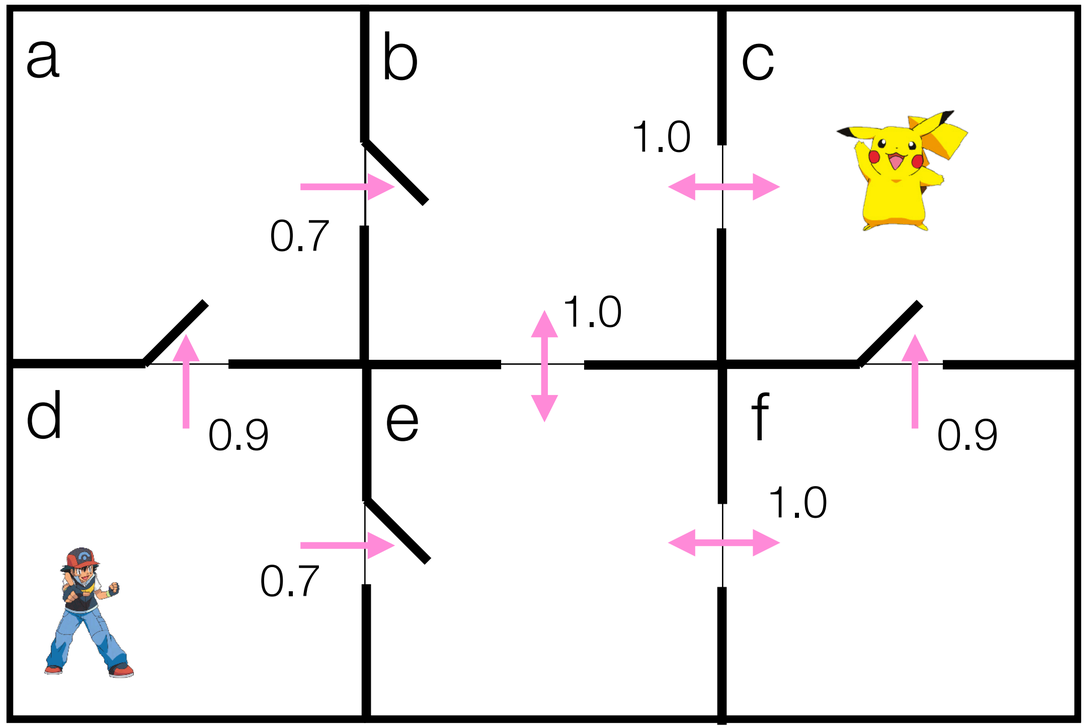
Original:

a_b: 0.7 -> b, 0.3 -> a, cost 2

Relaxed:

a_b_b: 1.0 -> b, cost 2

a_b_a: 1.0 -> a, cost 2



MDP Heuristic - All Outcomes Determinisation

Create a ***relaxed*** version of the problem

We will use
All-Outcomes Determinisation

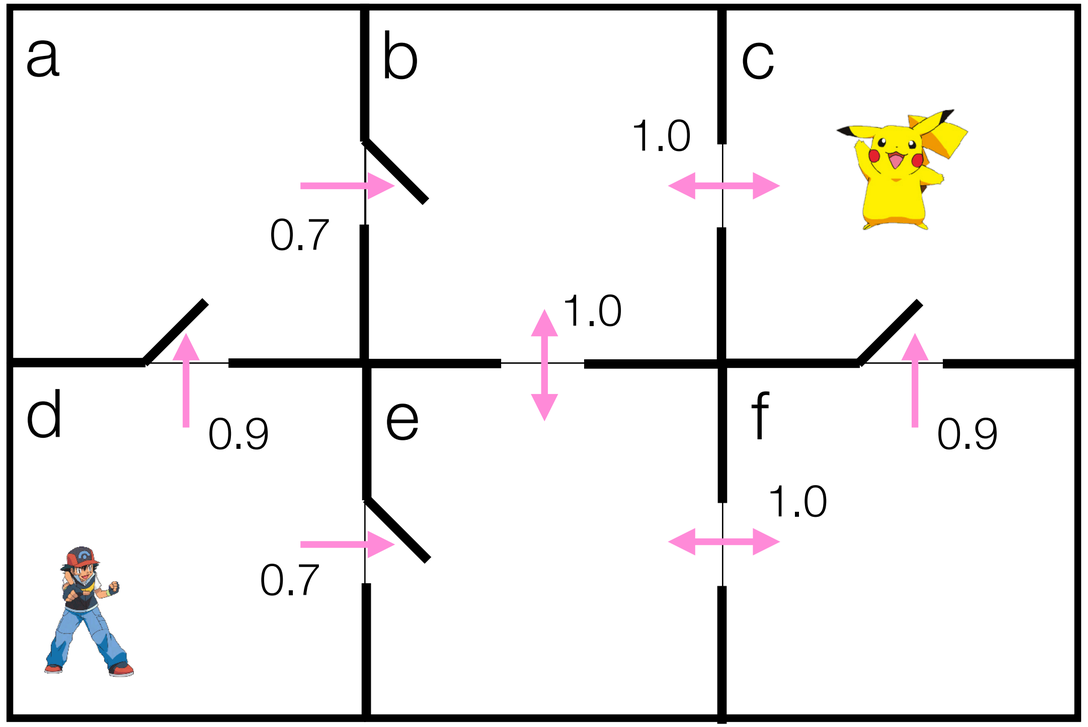
Original:

a_b: 0.7 -> b, 0.3 -> a, cost 2

Relaxed:

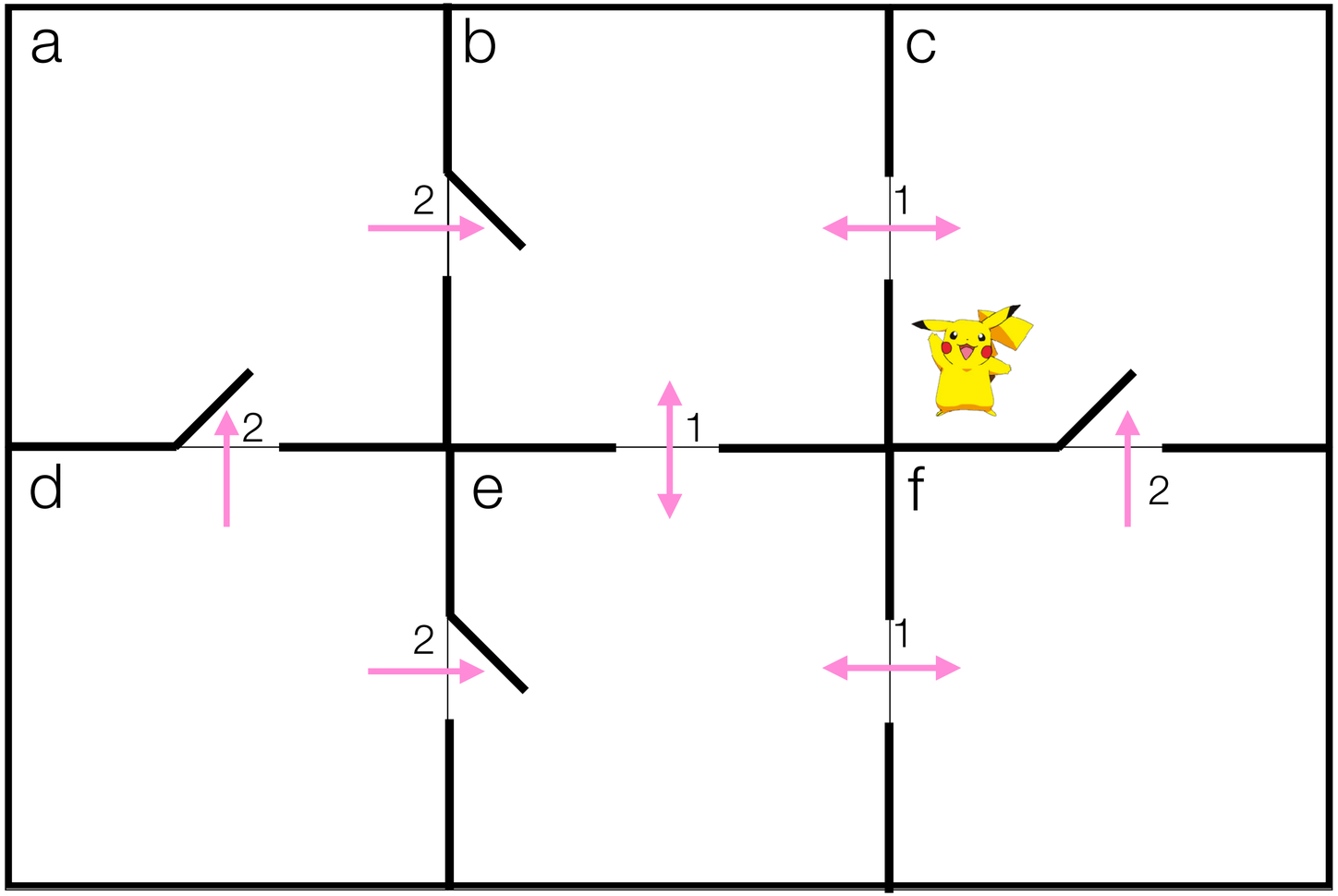
a_b_b: 1.0 -> b, cost 2

a_b_a: 1.0 -> a, cost 2



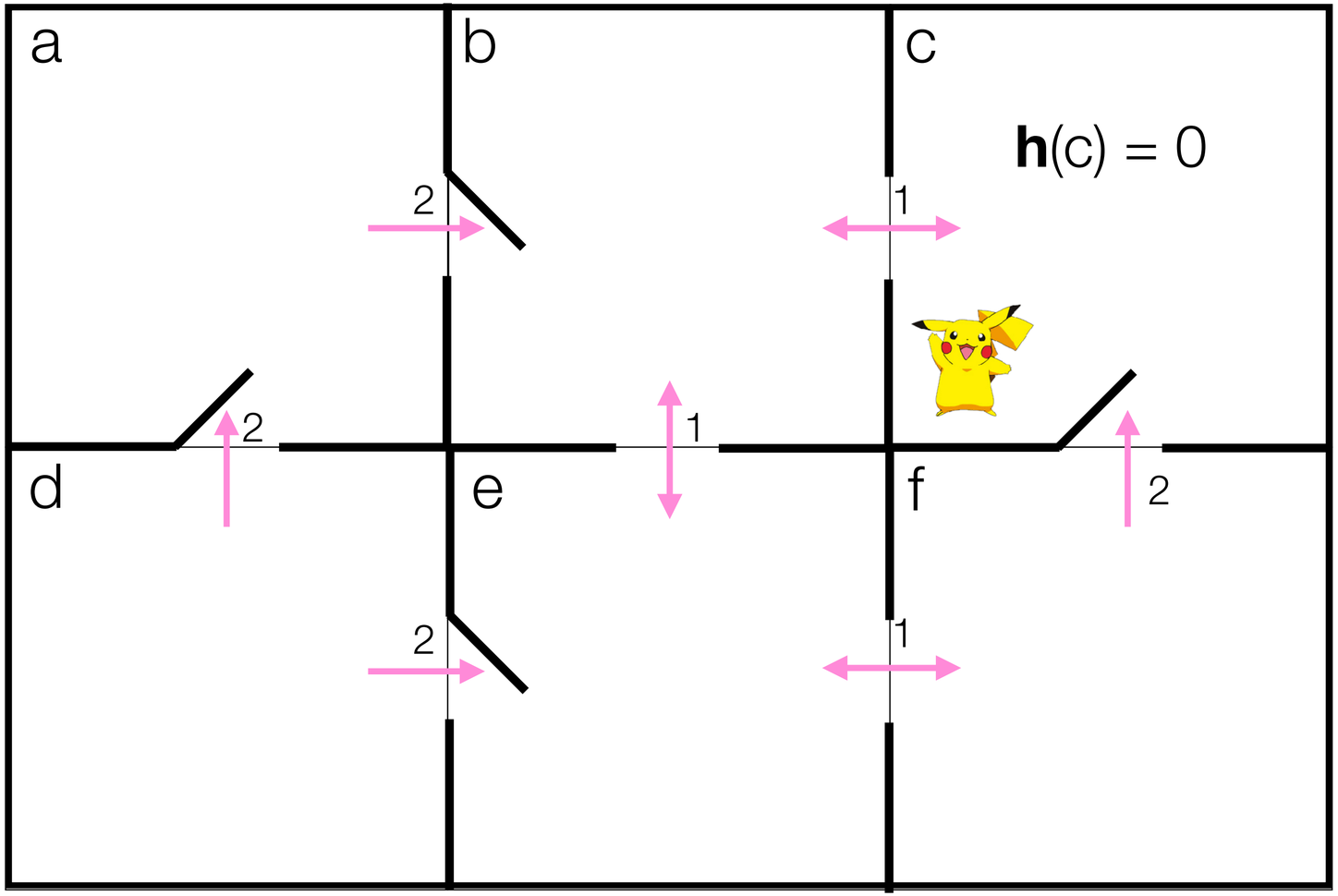
This relaxes the SSP into a **deterministic shortest path problem**,
which we we **solve to provide heuristic** for the SSP MDP

MDP Heuristic - All Outcomes Determinisation



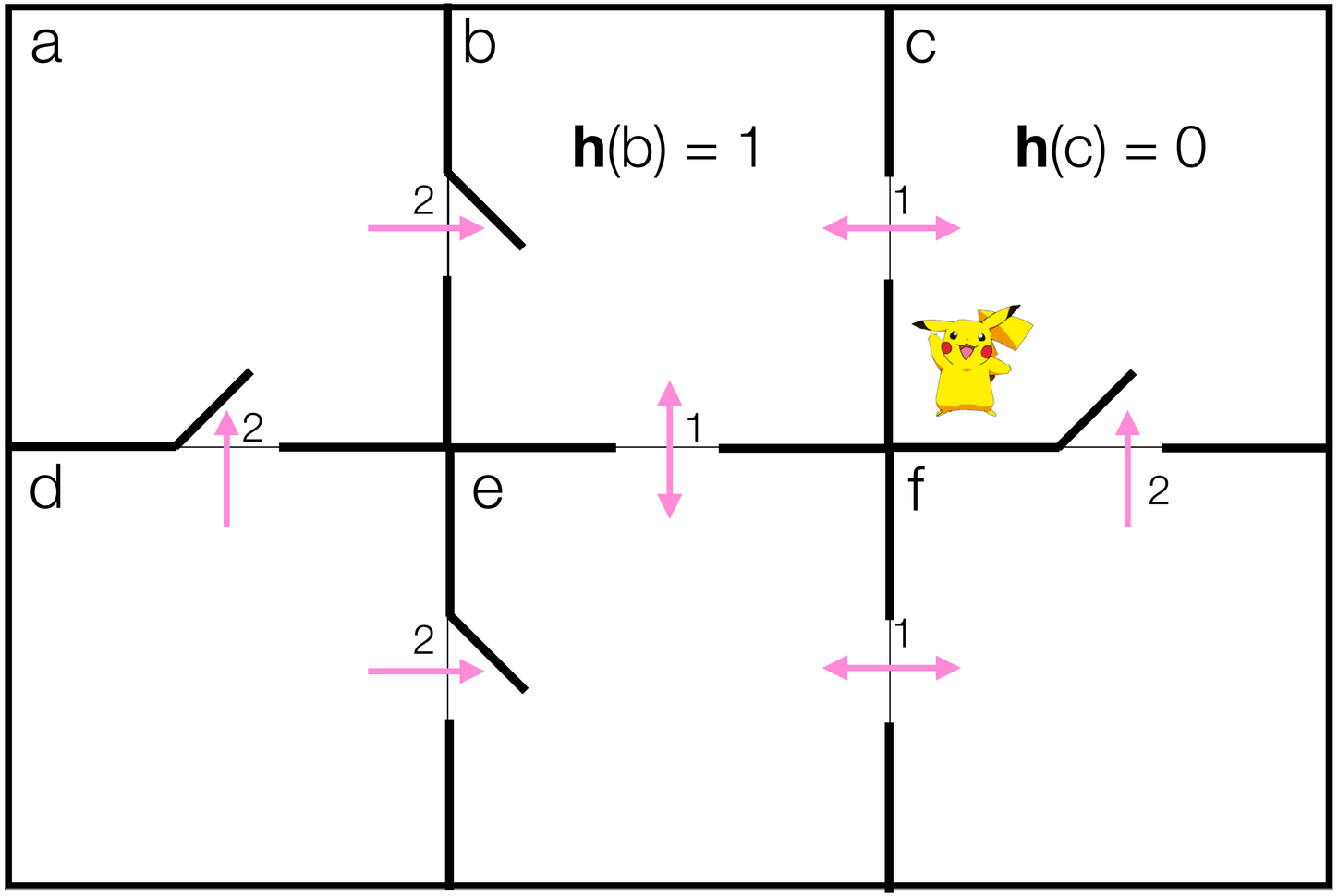
This relaxes the SSP into a **deterministic shortest path problem**, which we we **solve to provide heuristic** for the SSP MDP

MDP Heuristic - All Outcomes Determinisation



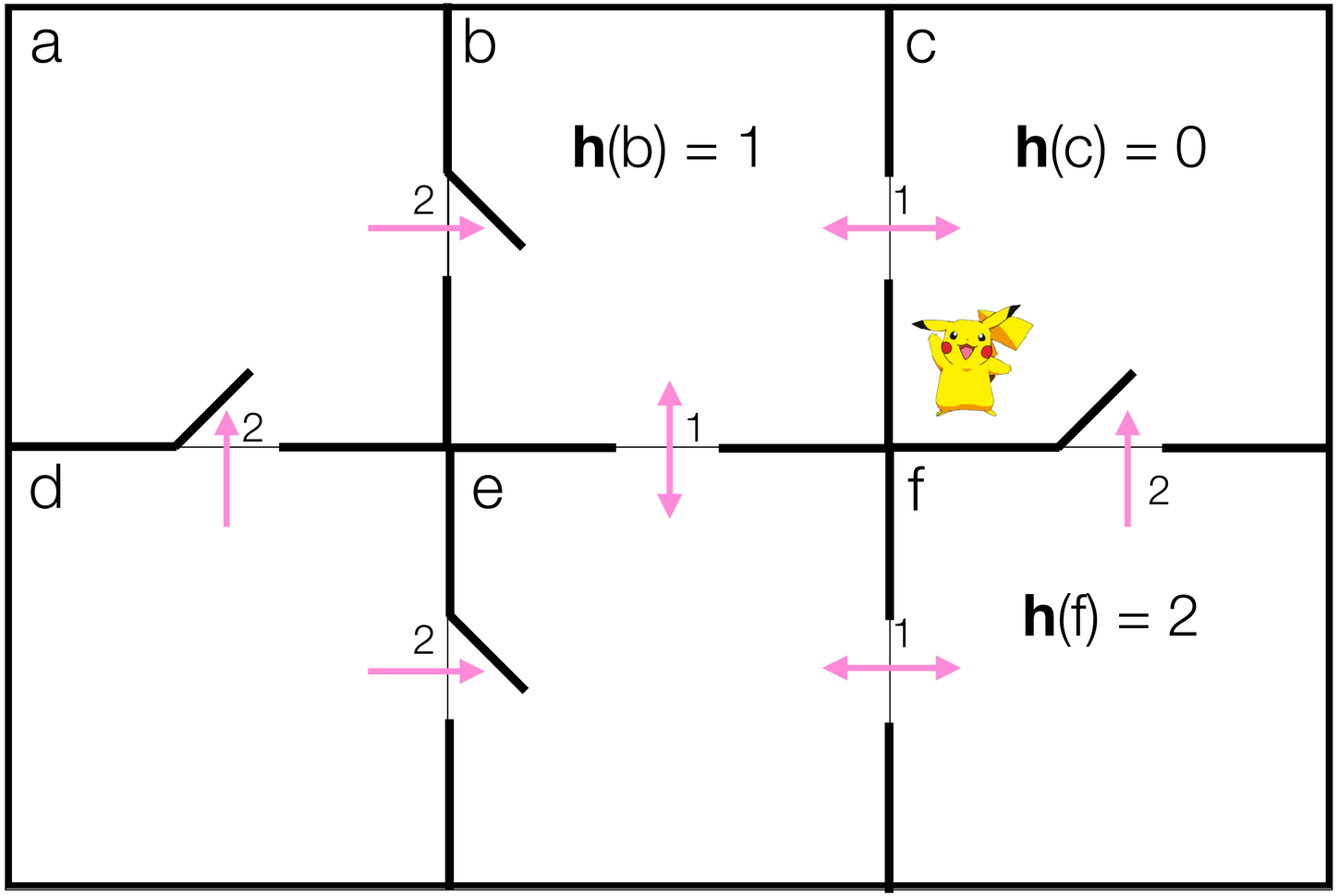
This relaxes the SSP into a **deterministic shortest path problem**, which we we **solve to provide heuristic** for the SSP MDP

MDP Heuristic - All Outcomes Determinisation



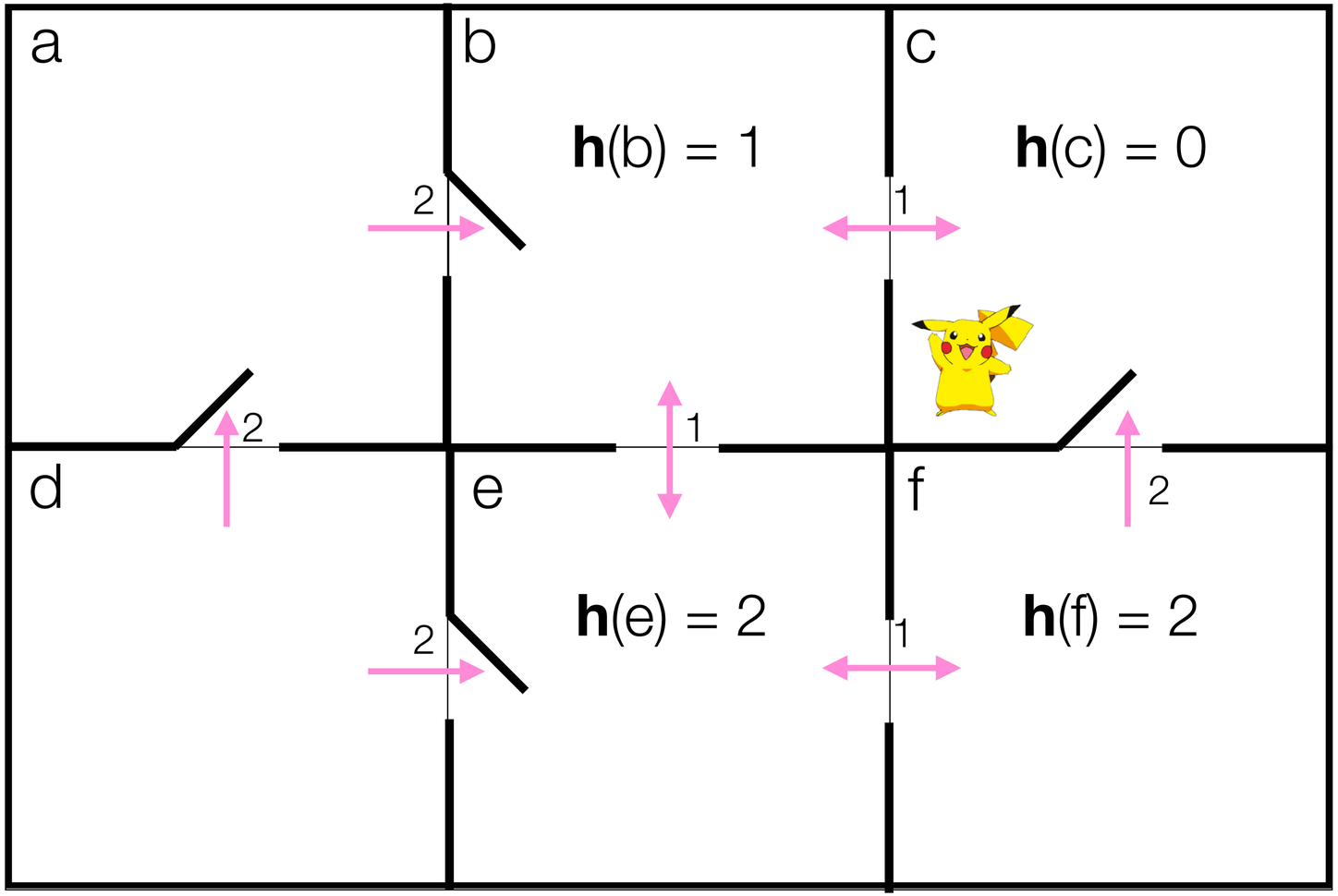
This relaxes the SSP into a **deterministic shortest path problem**, which we we **solve to provide heuristic** for the SSP MDP

MDP Heuristic - All Outcomes Determinisation



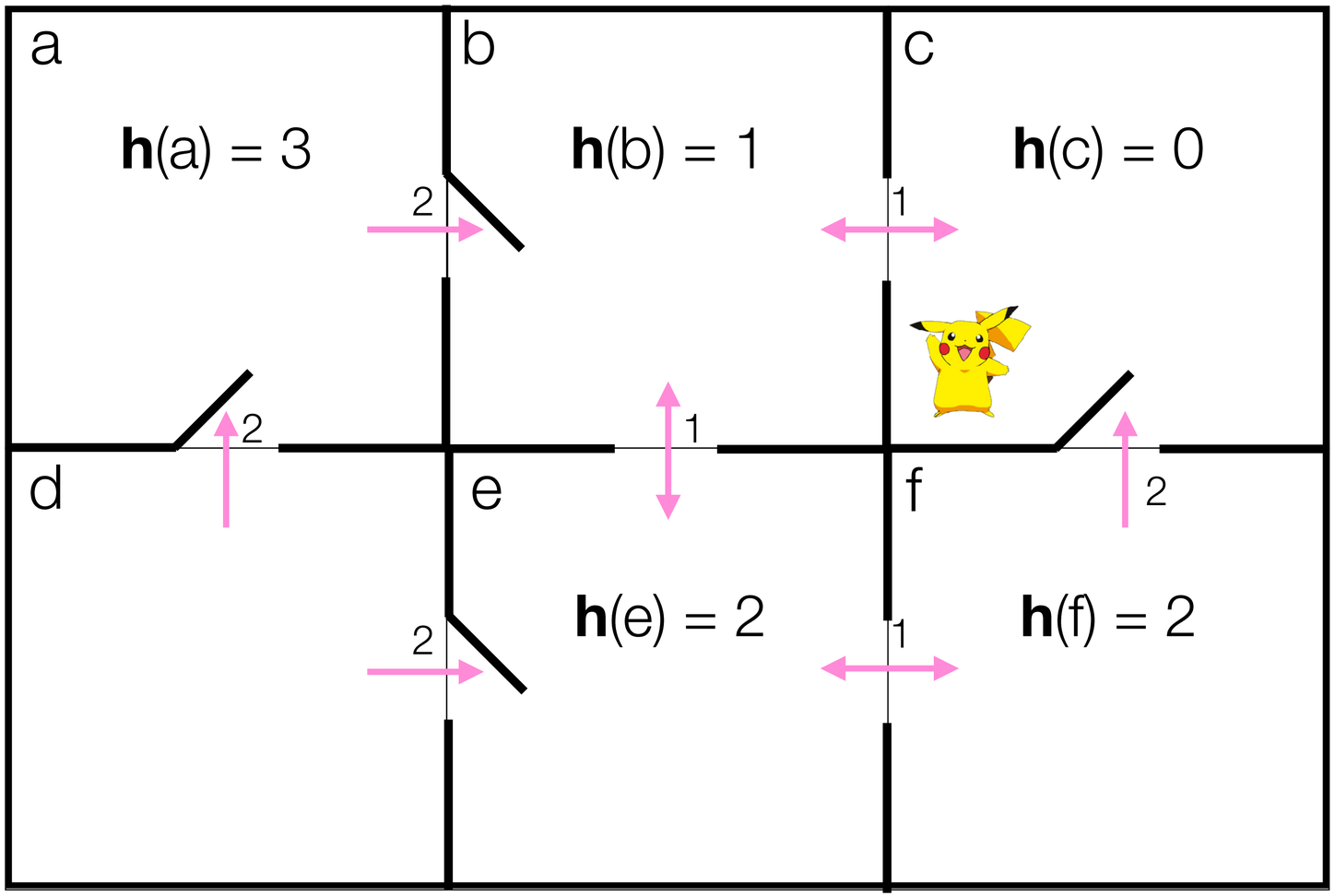
This relaxes the SSP into a **deterministic shortest path problem**, which we we **solve to provide heuristic** for the SSP MDP

MDP Heuristic - All Outcomes Determinisation



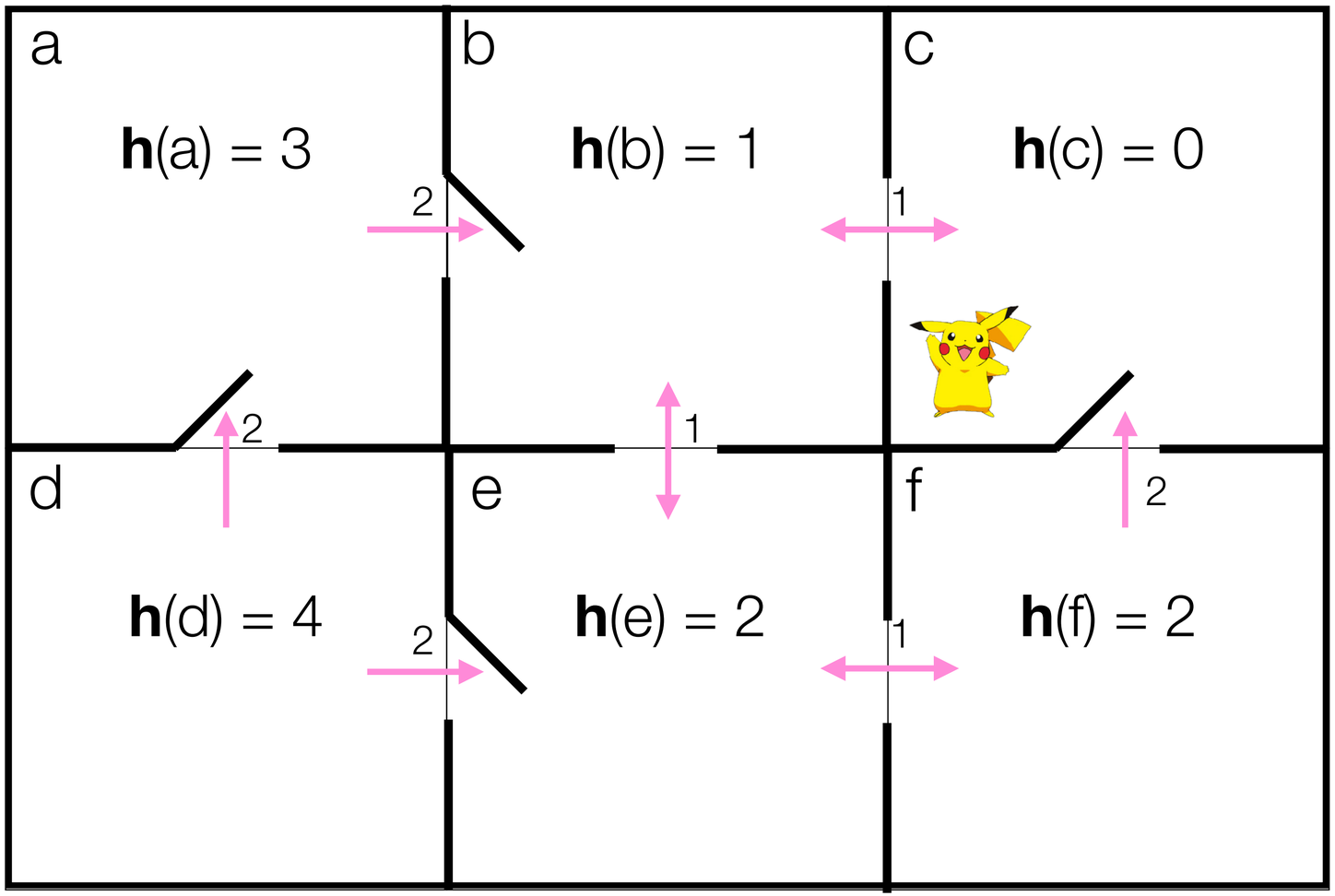
This relaxes the SSP into a **deterministic shortest path problem**, which we we **solve to provide heuristic** for the SSP MDP

MDP Heuristic - All Outcomes Determinisation



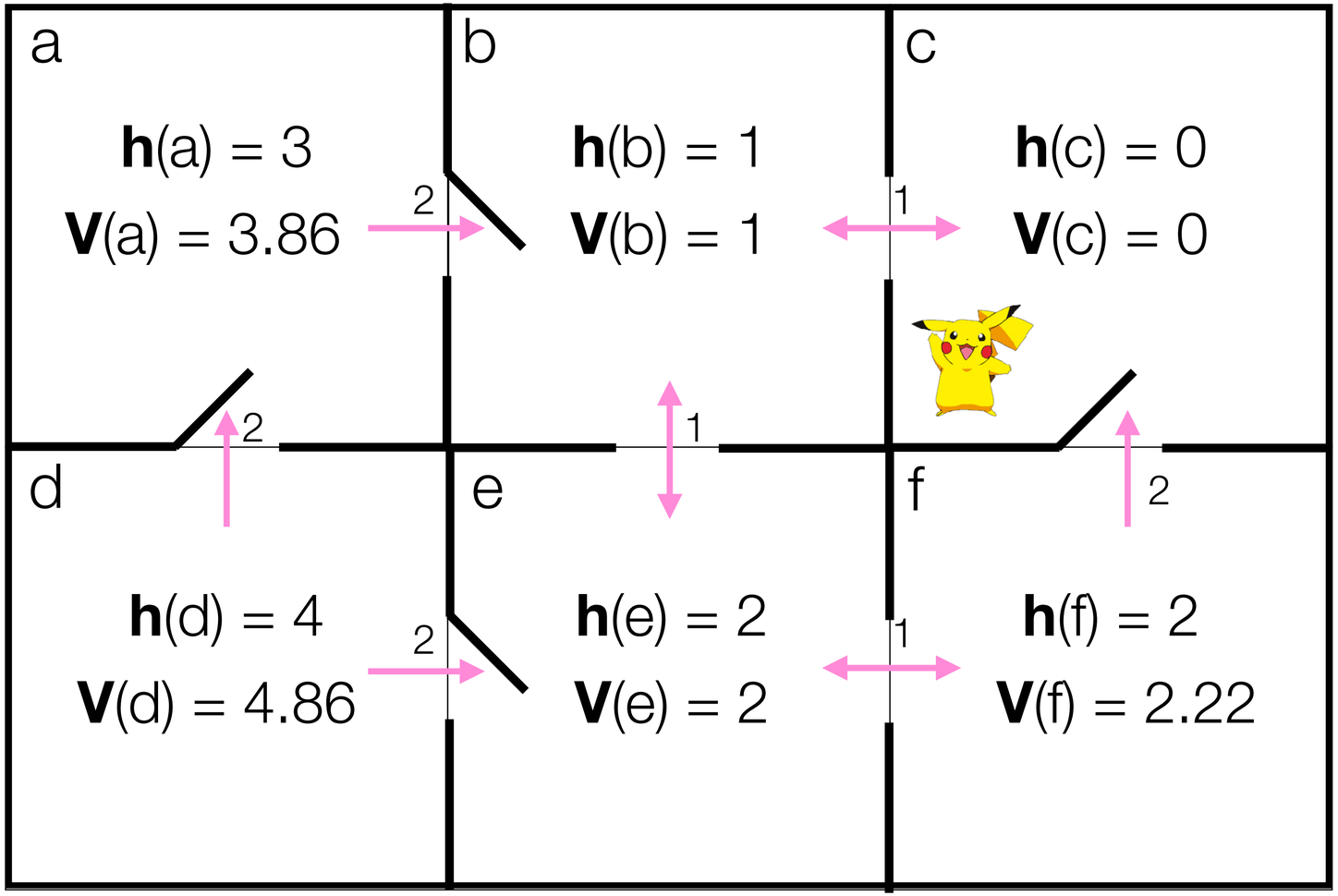
This relaxes the SSP into a **deterministic shortest path problem**, which we we **solve to provide heuristic** for the SSP MDP

MDP Heuristic - All Outcomes Determinisation



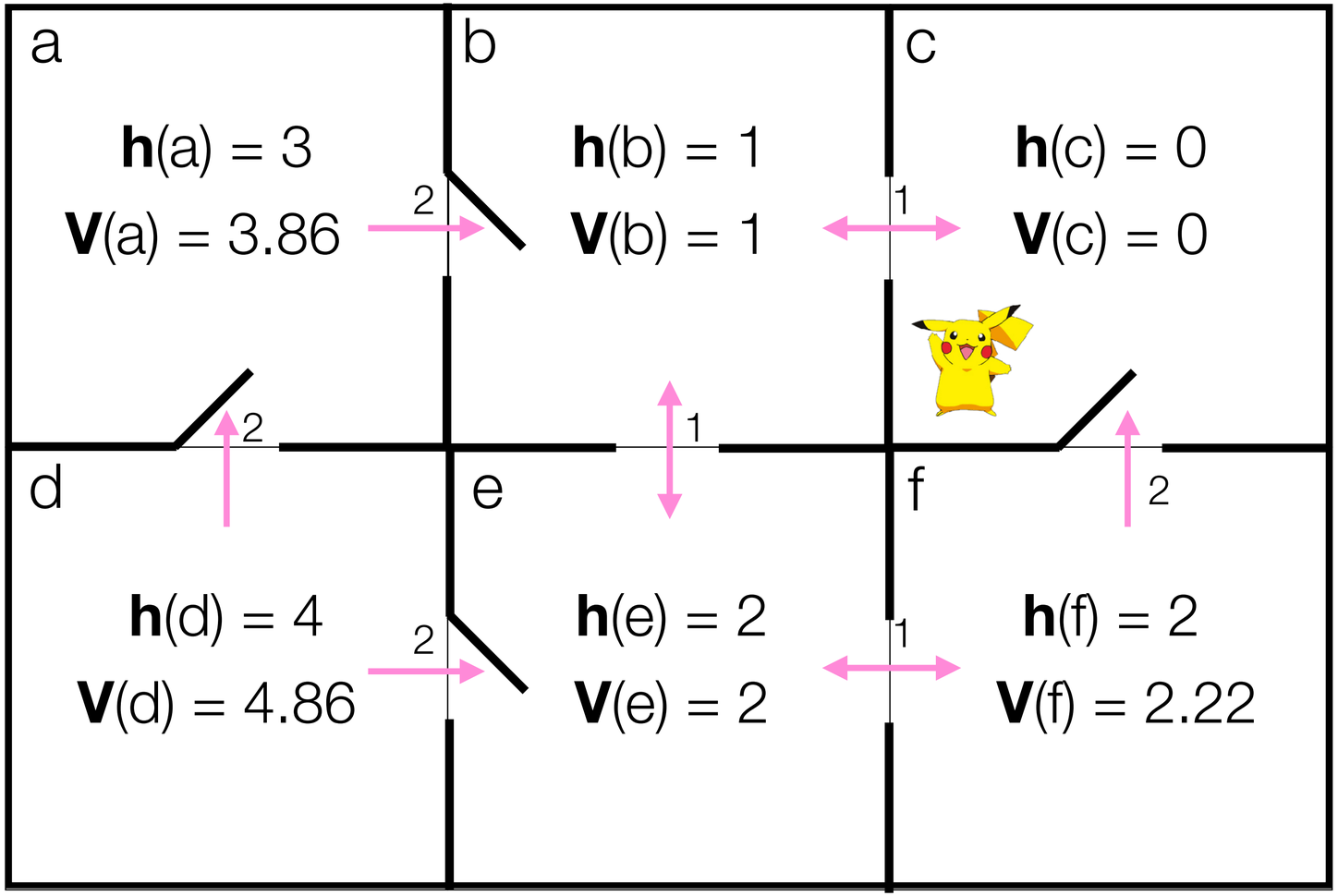
This relaxes the SSP into a **deterministic shortest path problem**, which we we **solve to provide heuristic** for the SSP MDP

MDP Heuristic - All Outcomes Determinisation



This relaxes the SSP into a **deterministic shortest path problem**, which we we **solve to provide heuristic** for the SSP MDP

MDP Heuristic - All Outcomes Determinisation



h(e) is a lower bound for **V**(e), and an *admissible* heuristic

This relaxes the SSP into a **deterministic shortest path problem**, which we we **solve to provide heuristic** for the SSP MDP

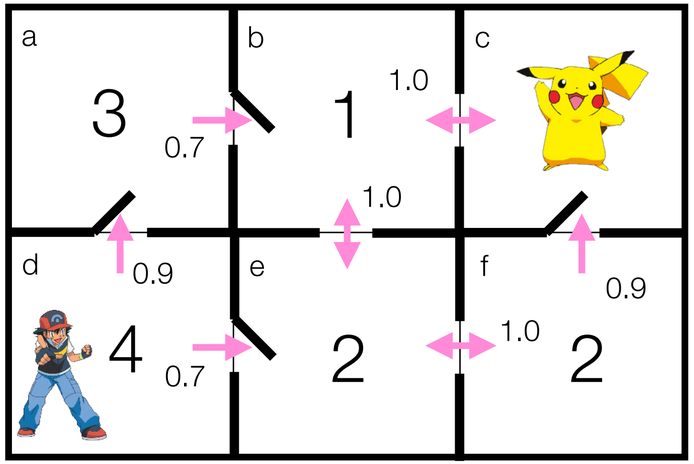
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

heuristic



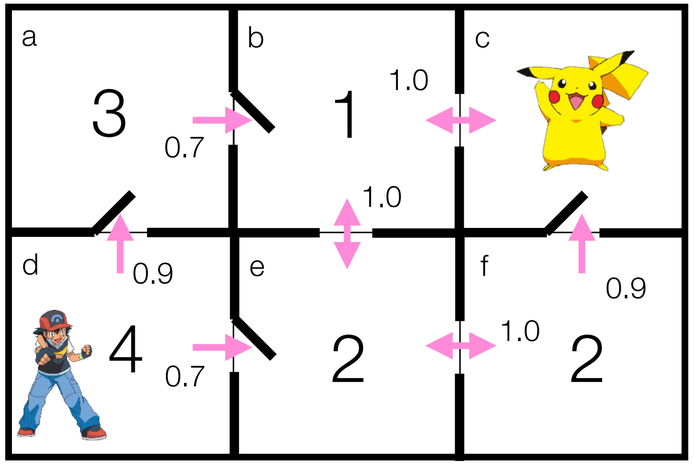
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

Trial 1

heuristic



	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

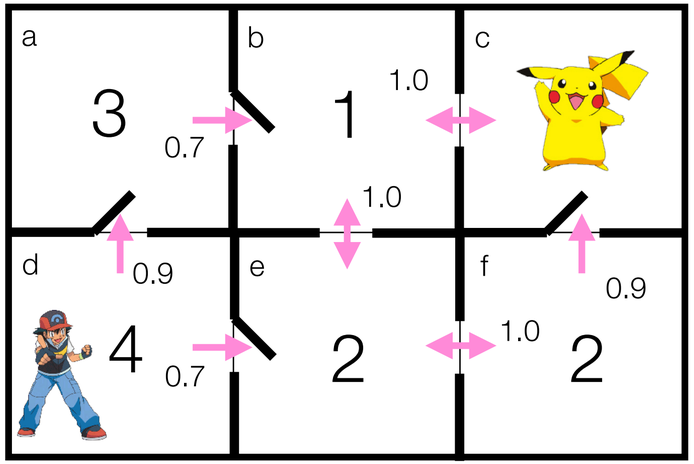
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

Trial 1

heuristic



	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

	Init
V(a)	3.0
Q(a, b)	
V(b)	1.0
Q(b, c)	
Q(b, e)	
V(c)	0.0
Q(c, b)	
V(d)	**4.0**
Q(d, a)	
Q(d, e)	
V(e)	2.0
Q(e, b)	
Q(e, f)	
V(f)	2.0
Q(f, c)	
Q(f, e)	

Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

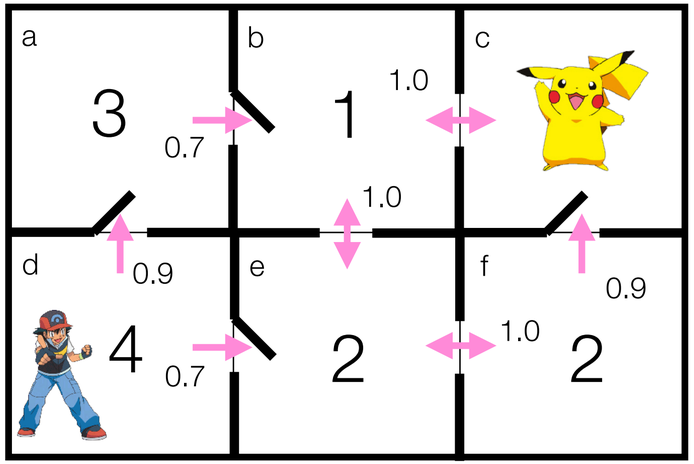
Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

Trial 1

d
V(d) = 4

heuristic



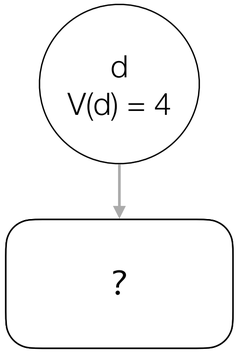
	Init
V(a)	3.0
Q(a, b)	
V(b)	1.0
Q(b, c)	
Q(b, e)	
V(c)	0.0
Q(c, b)	
V(d)	**4.0**
Q(d, a)	
Q(d, e)	
V(e)	2.0
Q(e, b)	
Q(e, f)	
V(f)	2.0
Q(f, c)	
Q(f, e)	

Algorithm 4.3: RTDP

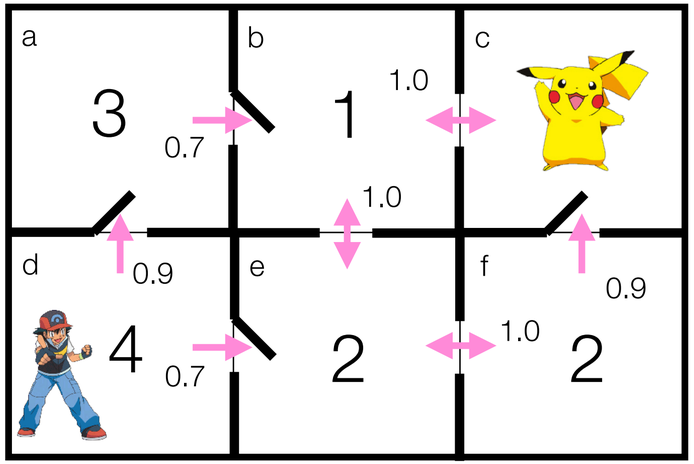
```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

Trial 1



heuristic



	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

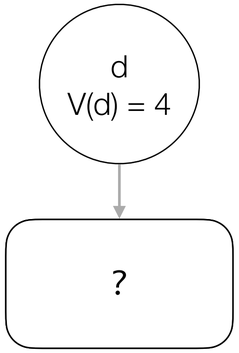
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

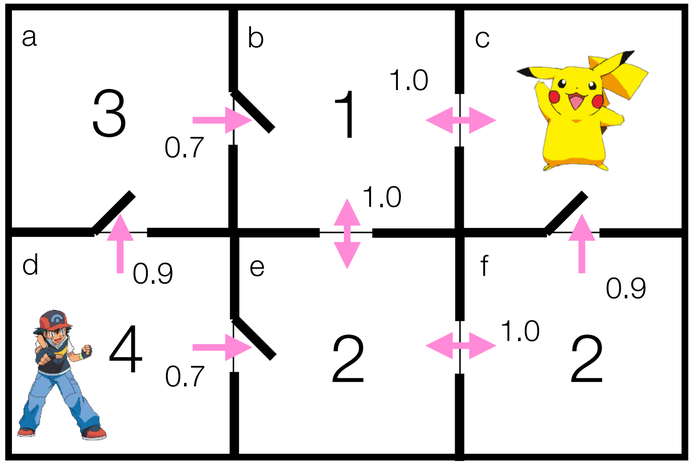
Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

Trial 1



heuristic



	Init
V(a)	3.0
Q(a, b)	
V(b)	1.0
Q(b, c)	
Q(b, e)	
V(c)	0.0
Q(c, b)	
V(d)	**4.0**
Q(d, a)	
Q(d, e)	
V(e)	2.0
Q(e, b)	
Q(e, f)	
V(f)	2.0
Q(f, c)	
Q(f, e)	

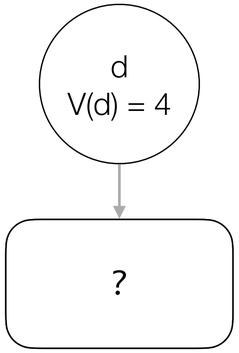
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

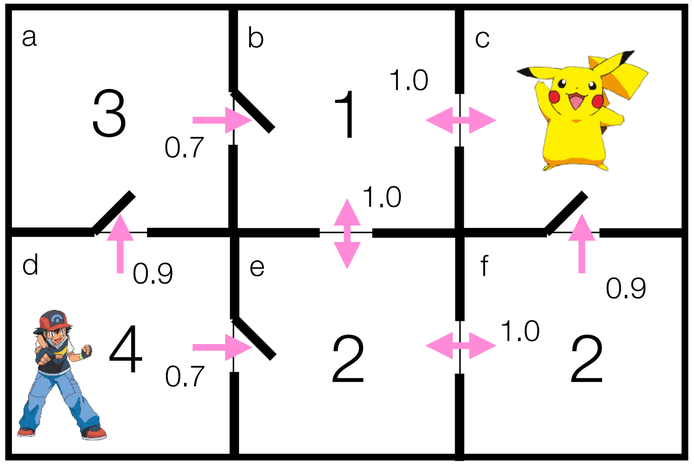
	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

Trial 1



What successor do we generate?

heuristic



	Init
V(a)	3.0
Q(a, b)	
V(b)	1.0
Q(b, c)	
Q(b, e)	
V(c)	0.0
Q(c, b)	
V(d)	**4.0**
Q(d, a)	
Q(d, e)	
V(e)	2.0
Q(e, b)	
Q(e, f)	
V(f)	2.0
Q(f, c)	
Q(f, e)	

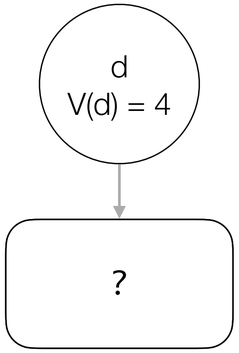
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

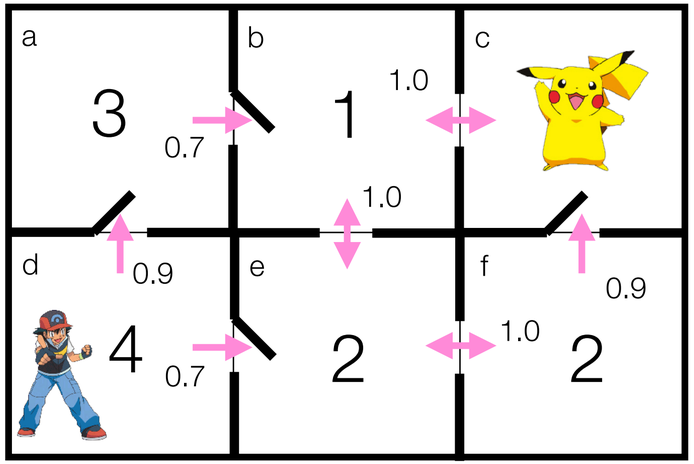
Trial 1



What successor do we generate?

Follow the **greedy policy** given current $\mathbf{Q_l}$ estimate and **sample** and outcome state

heuristic



	Init
V(a)	3.0
Q(a, b)	
V(b)	1.0
Q(b, c)	
Q(b, e)	
V(c)	0.0
Q(c, b)	
V(d)	**4.0**
Q(d, a)	
Q(d, e)	
V(e)	2.0
Q(e, b)	
Q(e, f)	
V(f)	2.0
Q(f, c)	
Q(f, e)	

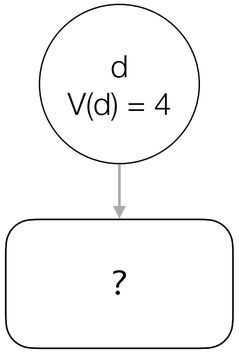
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

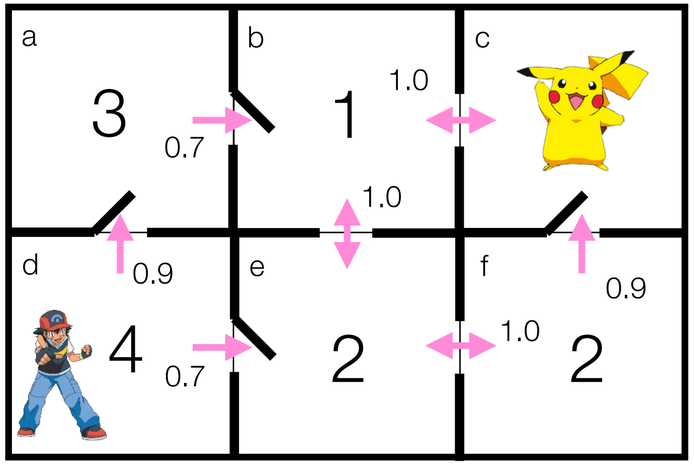
Trial 1



What successor do we generate?

Follow the **greedy policy** given current $\mathbf{Q_l}$ estimate and **sample** and outcome state

heuristic



	Init	Q?
V(a)	3.0	3.0
Q(a, b)		
V(b)	1.0	1.0
Q(b, c)		
Q(b, e)		
V(c)	0.0	0.0
Q(c, b)		
V(d)	**4.0**	4.6
Q(d, a)		5.1
Q(d, e)		**4.6**
V(e)	2.0	2.0
Q(e, b)		
Q(e, f)		
V(f)	2.0	2.0
Q(f, c)		
Q(f, e)		

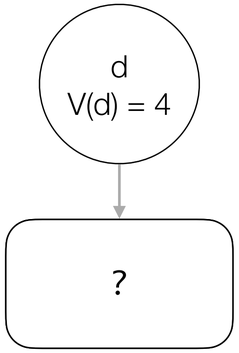
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

Trial 1

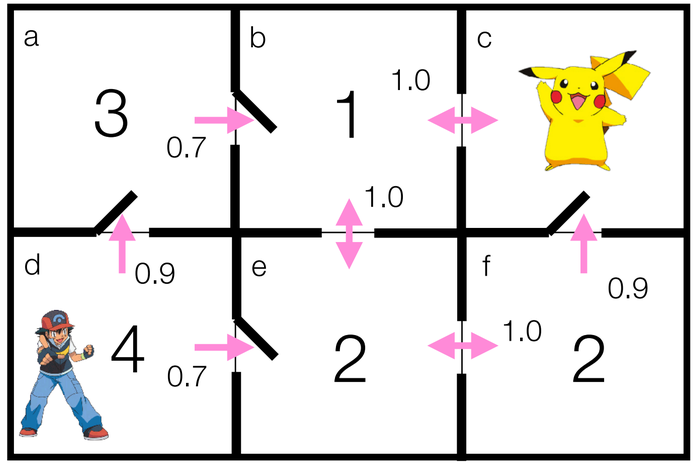


Q(d, a) = 5.1
Q(d, e) = 4.6

What successor do we generate?

Follow the **greedy policy** given current $\mathbf{Q_l}$ estimate and **sample** and outcome state

heuristic



	Init	Q?
V(a)	3.0	3.0
Q(a, b)		
V(b)	1.0	1.0
Q(b, c)		
Q(b, e)		
V(c)	0.0	0.0
Q(c, b)		
V(d)	**4.0**	4.6
Q(d, a)		5.1
Q(d, e)		**4.6**
V(e)	2.0	2.0
Q(e, b)		
Q(e, f)		
V(f)	2.0	2.0
Q(f, c)		
Q(f, e)		

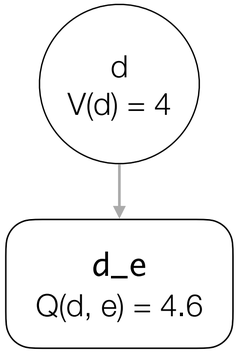
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

Trial 1

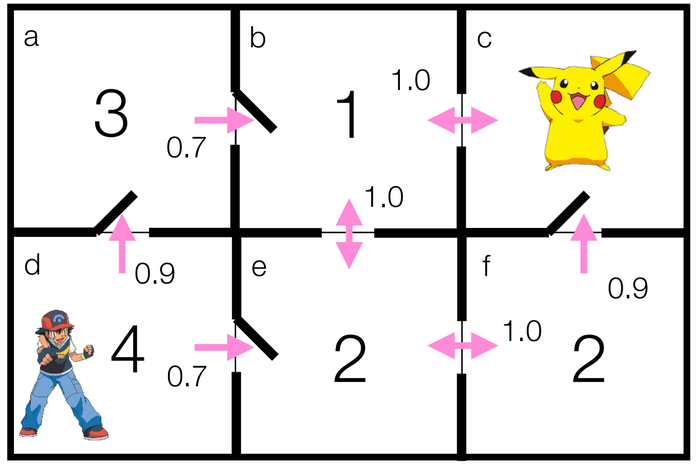


Q(d, a) = 5.1
Q(d, e) = 4.6

What successor do we generate?

Follow the **greedy policy** given current $\mathbf{Q_l}$ estimate and **sample** and outcome state

heuristic



	Init	Q?
V(a)	3.0	3.0
Q(a, b)		
V(b)	1.0	1.0
Q(b, c)		
Q(b, e)		
V(c)	0.0	0.0
Q(c, b)		
V(d)	**4.0**	4.6
Q(d, a)		5.1
Q(d, e)		**4.6**
V(e)	2.0	2.0
Q(e, b)		
Q(e, f)		
V(f)	2.0	2.0
Q(f, c)		
Q(f, e)		

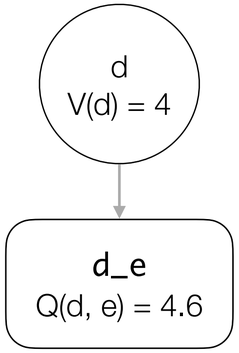
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

Trial 1



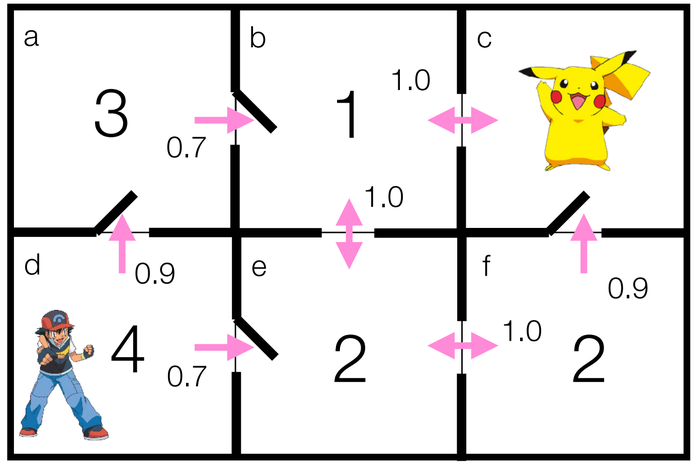
Q(d, a) = 5.1
Q(d, e) = 4.6

What successor do we generate?

Follow the **greedy policy** given current $\mathbf{Q_l}$ estimate and **sample** and outcome state

Now we we can (*Bellman*) **backup** the value of the expanded state

heuristic



	Init	Q?
V(a)	3.0	3.0
Q(a, b)		
V(b)	1.0	1.0
Q(b, c)		
Q(b, e)		
V(c)	0.0	0.0
Q(c, b)		
V(d)	**4.0**	4.6
Q(d, a)		5.1
Q(d, e)		**4.6**
V(e)	2.0	2.0
Q(e, b)		
Q(e, f)		
V(f)	2.0	2.0
Q(f, c)		
Q(f, e)		

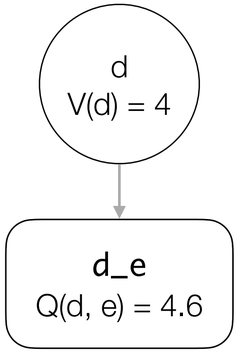
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

Trial 1



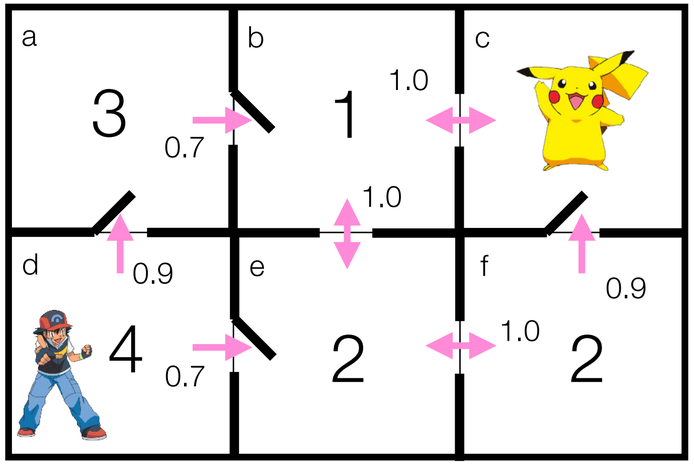
Q(d, a) = 5.1
Q(d, e) = 4.6

What successor do we generate?

Follow the **greedy policy** given current $\mathbf{Q_l}$ estimate and **sample** and outcome state

Now we we can (*Bellman*) **backup** the value of the expanded state

heuristic



	Init	Q?	Backup
V(a)	3.0	3.0	3.0
Q(a, b)			
V(b)	1.0	1.0	1.0
Q(b, c)			
Q(b, e)			
V(c)	0.0	0.0	0.0
Q(c, b)			
V(d)	**4.0**	4.6	**4.6**
Q(d, a)		5.1	
Q(d, e)		**4.6**	
V(e)	2.0	2.0	2.0
Q(e, b)			
Q(e, f)			
V(f)	2.0	2.0	2.0
Q(f, c)			
Q(f, e)			

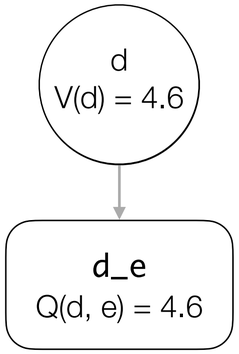
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

Trial 1



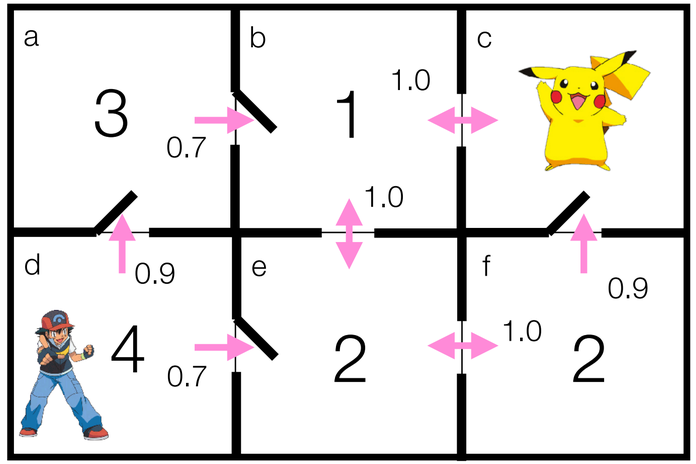
Q(d, a) = 5.1
Q(d, e) = 4.6

What successor do we generate?

Follow the **greedy policy** given current $\mathbf{Q_l}$ estimate and **sample** and outcome state

Now we we can (*Bellman*) **backup** the value of the expanded state

heuristic



	Init	Q?	Backup
V(a)	3.0	3.0	3.0
Q(a, b)			
V(b)	1.0	1.0	1.0
Q(b, c)			
Q(b, e)			
V(c)	0.0	0.0	0.0
Q(c, b)			
V(d)	**4.0**	4.6	**4.6**
Q(d, a)		5.1	
Q(d, e)		**4.6**	
V(e)	2.0	2.0	2.0
Q(e, b)			
Q(e, f)			
V(f)	2.0	2.0	2.0
Q(f, c)			
Q(f, e)			

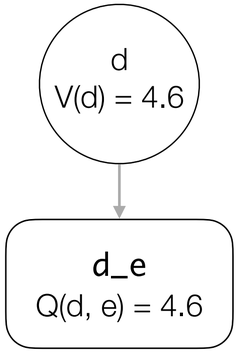
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

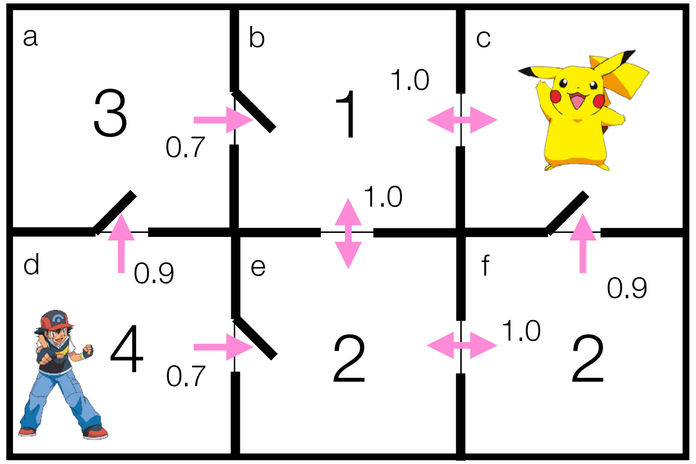
Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

Trial 1



The action has two **possible outcomes**, which one is the successor state?



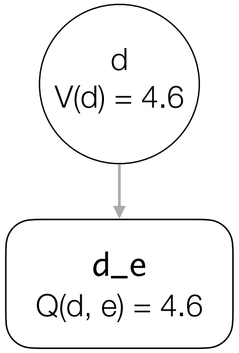
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

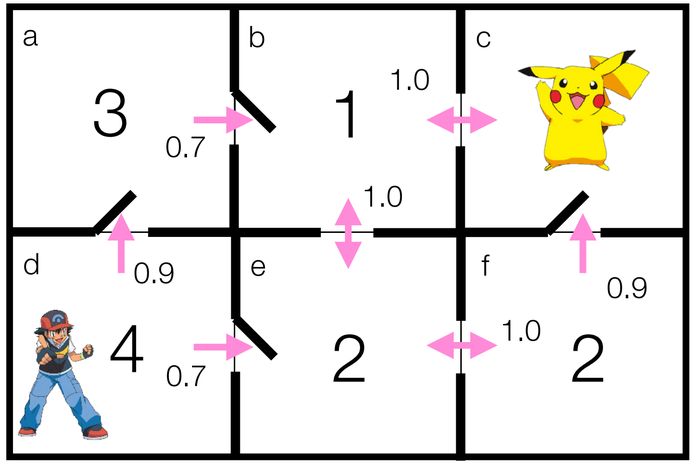
Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

Trial 1



The action has two **possible outcomes**, which one is the successor state?



	Init	Q?	Backup
V(a)	3.0	3.0	3.0
Q(a, b)			
V(b)	1.0	1.0	1.0
Q(b, c)			
Q(b, e)			
V(c)	0.0	0.0	0.0
Q(c, b)			
V(d)	**4.0**	4.6	**4.6**
Q(d, a)		5.1	
Q(d, e)		**4.6**	
V(e)	2.0	2.0	**2.0**
Q(e, b)			
Q(e, f)			
V(f)	2.0	2.0	2.0
Q(f, c)			
Q(f, e)			

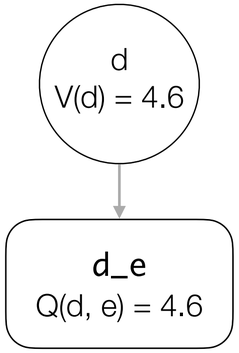
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

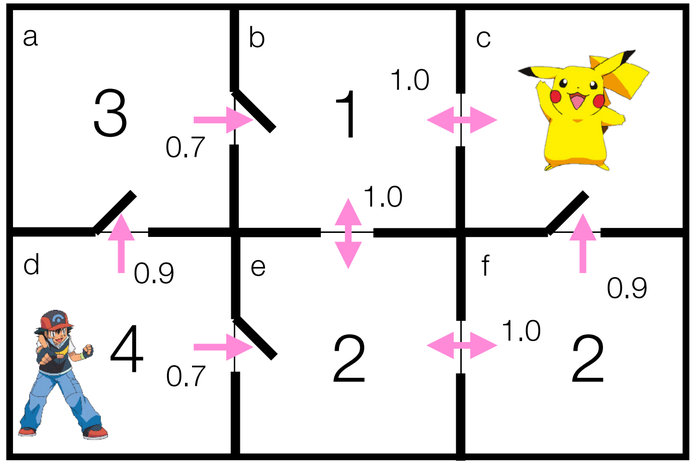
	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

Trial 1



The action has two **possible outcomes**, which one is the successor state?

We ***sample*** an outcome following the transition distribution



	Init	Q?	Backup
V(a)	3.0	3.0	3.0
Q(a, b)			
V(b)	1.0	1.0	1.0
Q(b, c)			
Q(b, e)			
V(c)	0.0	0.0	0.0
Q(c, b)			
V(d)	**4.0**	4.6	**4.6**
Q(d, a)		5.1	
Q(d, e)		**4.6**	
V(e)	2.0	2.0	**2.0**
Q(e, b)			
Q(e, f)			
V(f)	2.0	2.0	2.0
Q(f, c)			
Q(f, e)			

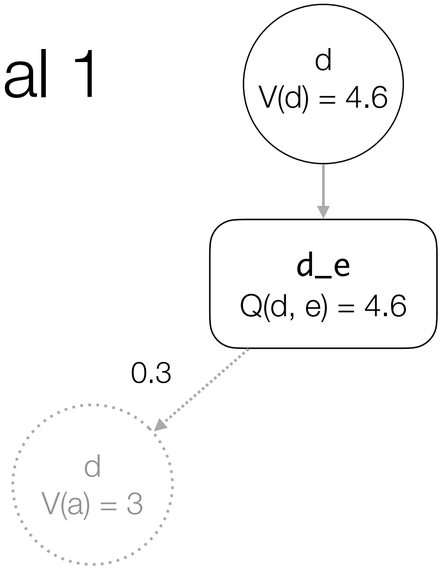
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

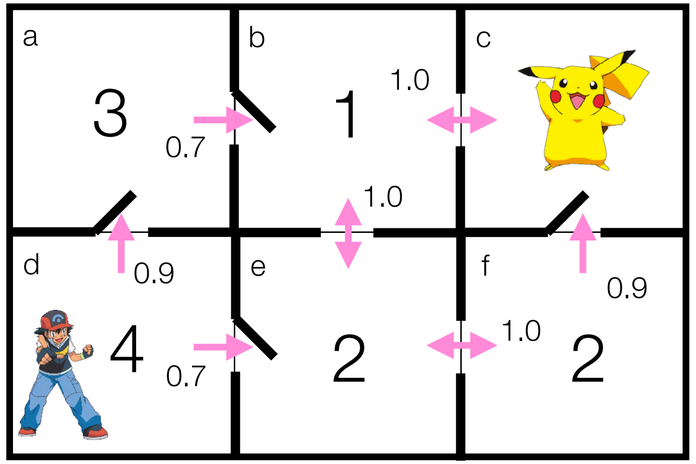
	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

Trial 1



The action has two **possible outcomes**, which one is the successor state?

We ***sample*** an outcome following the transition distribution



	Init	Q?	Backup
V(a)	3.0	3.0	3.0
Q(a, b)			
V(b)	1.0	1.0	1.0
Q(b, c)			
Q(b, e)			
V(c)	0.0	0.0	0.0
Q(c, b)			
V(d)	**4.0**	4.6	**4.6**
Q(d, a)		5.1	
Q(d, e)		**4.6**	
V(e)	2.0	2.0	**2.0**
Q(e, b)			
Q(e, f)			
V(f)	2.0	2.0	2.0
Q(f, c)			
Q(f, e)			

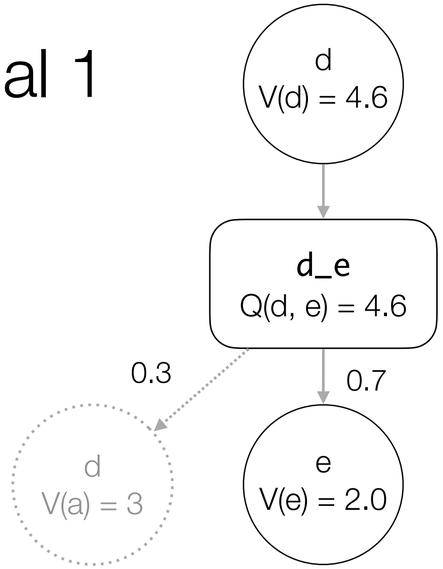
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

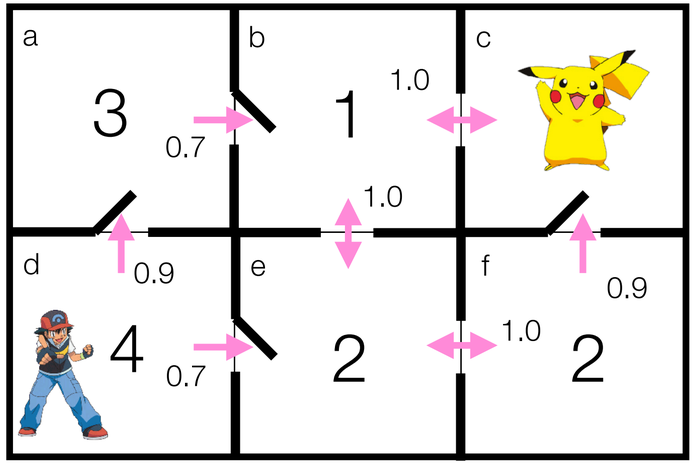
	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

Trial 1



The action has two **possible outcomes**, which one is the successor state?

We ***sample*** an outcome following the transition distribution



	Init	Q?	Backup
V(a)	3.0	3.0	3.0
Q(a, b)			
V(b)	1.0	1.0	1.0
Q(b, c)			
Q(b, e)			
V(c)	0.0	0.0	0.0
Q(c, b)			
V(d)	**4.0**	4.6	**4.6**
Q(d, a)		5.1	
Q(d, e)		**4.6**	
V(e)	2.0	2.0	**2.0**
Q(e, b)			
Q(e, f)			
V(f)	2.0	2.0	2.0
Q(f, c)			
Q(f, e)			

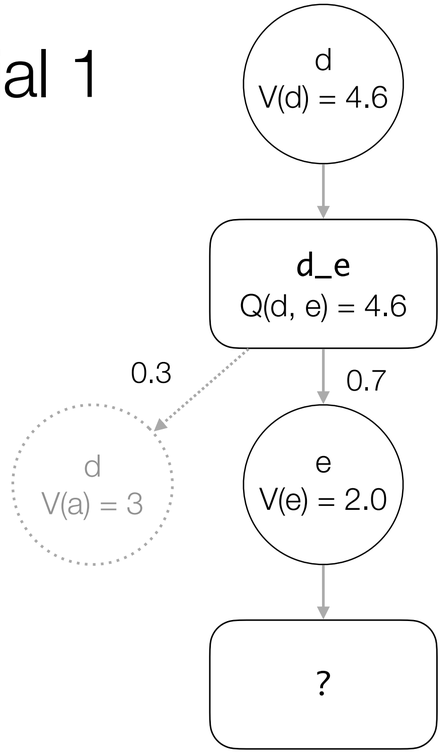
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

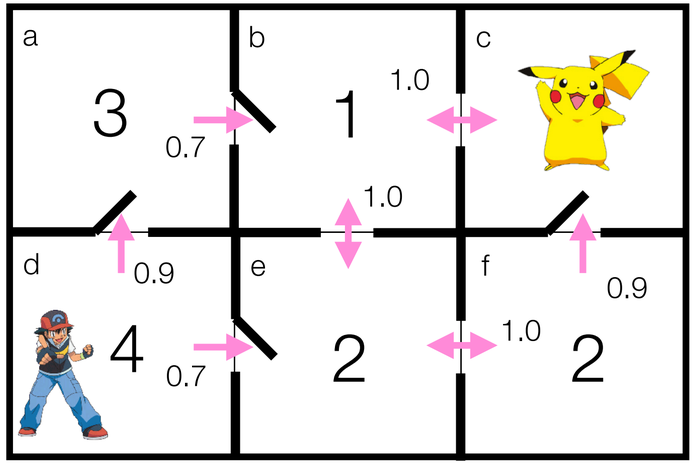
	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

Trial 1



The action has two **possible outcomes**, which one is the successor state?

We ***sample*** an outcome following the transition distribution



	Init	Q?	Backup
V(a)	3.0	3.0	3.0
Q(a, b)			
V(b)	1.0	1.0	1.0
Q(b, c)			
Q(b, e)			
V(c)	0.0	0.0	0.0
Q(c, b)			
V(d)	**4.0**	4.6	**4.6**
Q(d, a)		5.1	
Q(d, e)		**4.6**	
V(e)	2.0	2.0	**2.0**
Q(e, b)			
Q(e, f)			
V(f)	2.0	2.0	2.0
Q(f, c)			
Q(f, e)			

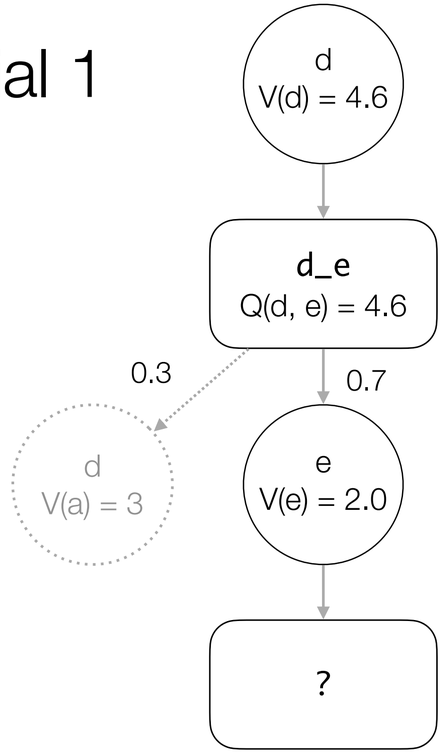
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

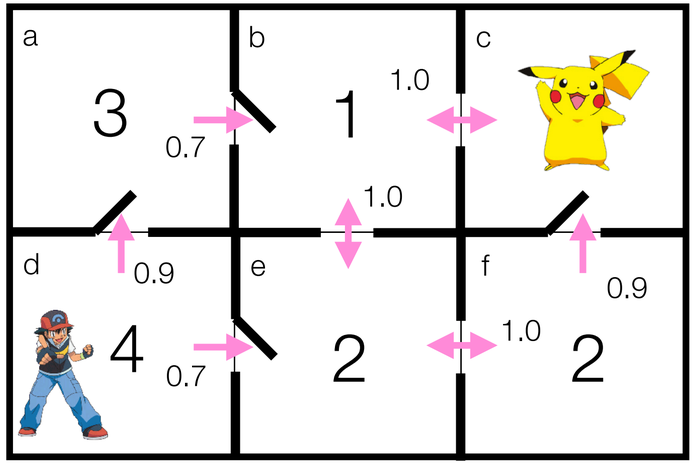
	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

Trial 1



The action has two **possible outcomes**, which one is the successor state?

We ***sample*** an outcome following the transition distribution



	Init	Q?	Backup	Q?
V(a)	3.0	3.0	3.0	3.0
Q(a, b)				
V(b)	1.0	1.0	1.0	1.0
Q(b, c)				
Q(b, e)				
V(c)	0.0	0.0	0.0	0.0
Q(c, b)				
V(d)	**4.0**	4.6	**4.6**	4.6
Q(d, a)		5.1		
Q(d, e)		**4.6**		
V(e)	2.0	2.0	**2.0**	2.0
Q(e, b)				**2.0**
Q(e, f)				3.0
V(f)	2.0	2.0	2.0	2.0
Q(f, c)				
Q(f, e)				

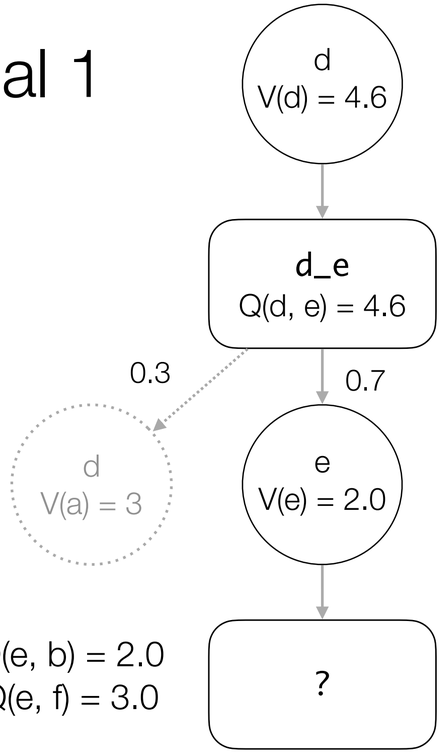
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

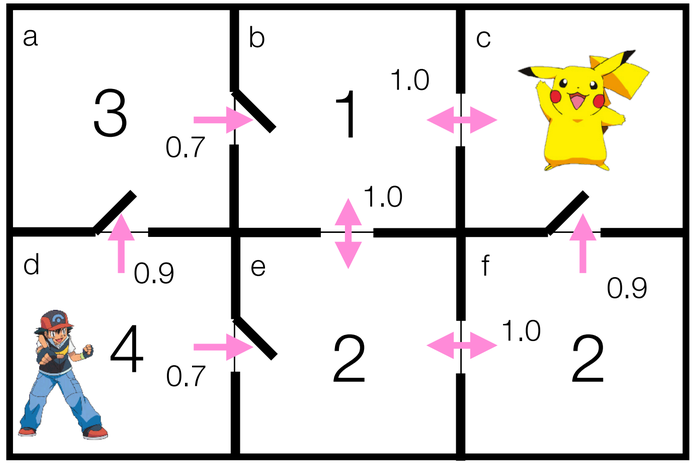
Trial 1



Q(e, b) = 2.0
Q(e, f) = 3.0

The action has two **possible outcomes**, which one is the successor state?

We ***sample*** an outcome following the transition distribution



	Init	Q?	Backup	Q?
V(a)	3.0	3.0	3.0	3.0
Q(a, b)				
V(b)	1.0	1.0	1.0	1.0
Q(b, c)				
Q(b, e)				
V(c)	0.0	0.0	0.0	0.0
Q(c, b)				
V(d)	**4.0**	4.6	**4.6**	4.6
Q(d, a)		5.1		
Q(d, e)		**4.6**		
V(e)	2.0	2.0	**2.0**	2.0
Q(e, b)				**2.0**
Q(e, f)				3.0
V(f)	2.0	2.0	2.0	2.0
Q(f, c)				
Q(f, e)				

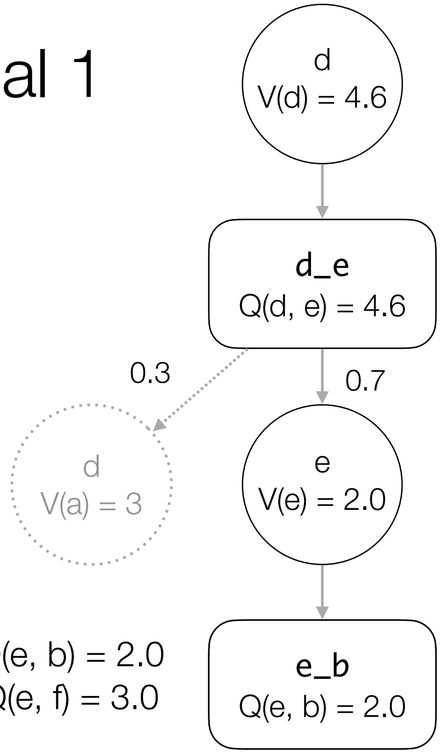
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

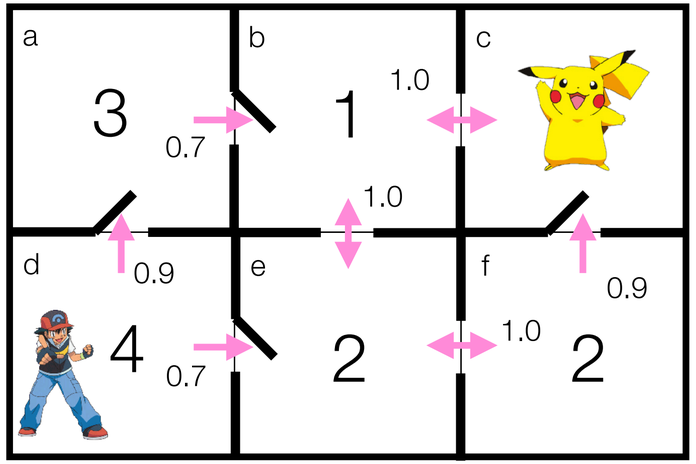
Trial 1



Q(e, b) = 2.0
Q(e, f) = 3.0

The action has two **possible outcomes**, which one is the successor state?

We ***sample*** an outcome following the transition distribution



	Init	Q?	Backup	Q?
V(a)	3.0	3.0	3.0	3.0
Q(a, b)				
V(b)	1.0	1.0	1.0	1.0
Q(b, c)				
Q(b, e)				
V(c)	0.0	0.0	0.0	0.0
Q(c, b)				
V(d)	**4.0**	4.6	**4.6**	4.6
Q(d, a)		5.1		
Q(d, e)		**4.6**		
V(e)	2.0	2.0	**2.0**	2.0
Q(e, b)				**2.0**
Q(e, f)				3.0
V(f)	2.0	2.0	2.0	2.0
Q(f, c)				
Q(f, e)				

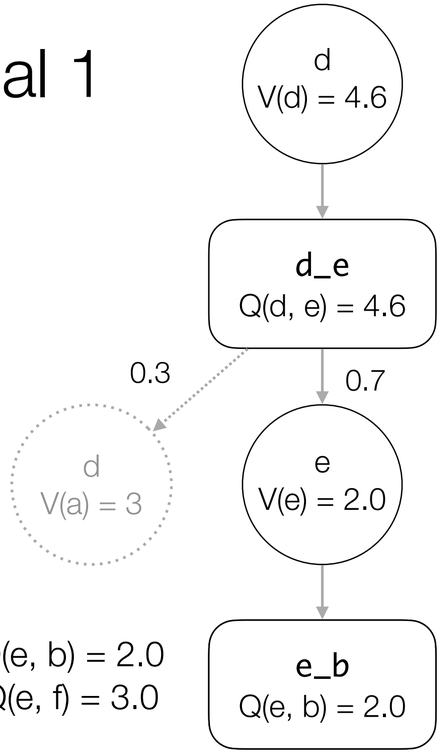
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

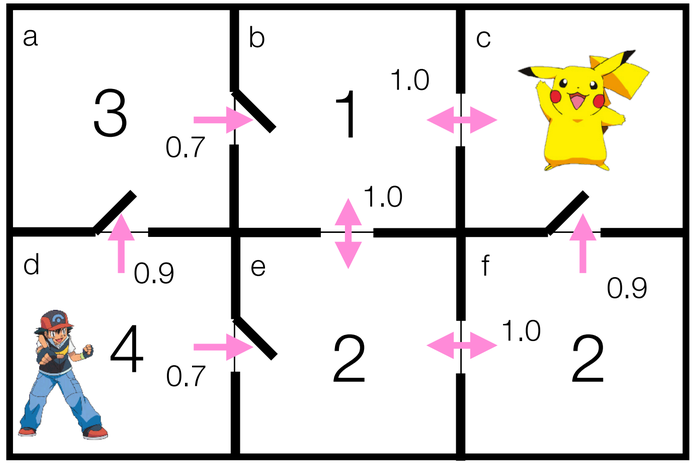
Trial 1



Q(e, b) = 2.0
Q(e, f) = 3.0

The action has two **possible outcomes**, which one is the successor state?

We ***sample*** an outcome following the transition distribution



	Init	Q?	Backup	Q?	Backup
V(a)	3.0	3.0	3.0	3.0	3.0
Q(a, b)					
V(b)	1.0	1.0	1.0	1.0	1.0
Q(b, c)					
Q(b, e)					
V(c)	0.0	0.0	0.0	0.0	0.0
Q(c, b)					
V(d)	**4.0**	4.6	**4.6**	4.6	4.6
Q(d, a)		5.1			
Q(d, e)		**4.6**			
V(e)	2.0	2.0	**2.0**	2.0	**2.0**
Q(e, b)				**2.0**	
Q(e, f)				3.0	
V(f)	2.0	2.0	2.0	2.0	2.0
Q(f, c)					
Q(f, e)					

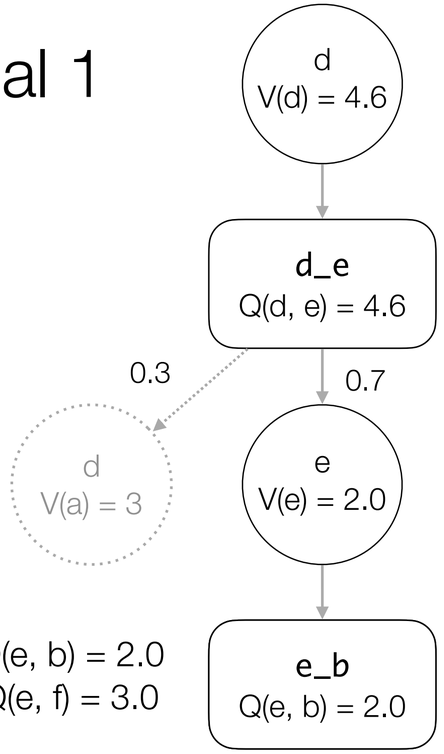
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

Trial 1

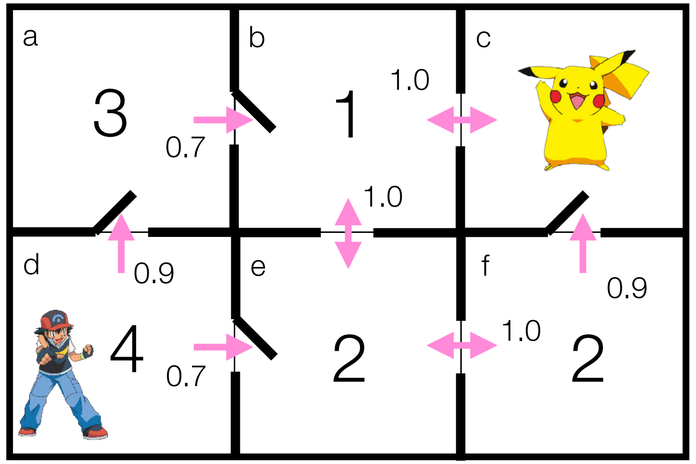


Q(e, b) = 2.0
Q(e, f) = 3.0

The action has two **possible outcomes**, which one is the successor state?

We ***sample*** an outcome following the transition distribution

Them repeat this until we reach the **goal** (state c)



	Init	Q?	Backup	Q?	Backup
V(a)	3.0	3.0	3.0	3.0	3.0
Q(a, b)					
V(b)	1.0	1.0	1.0	1.0	1.0
Q(b, c)					
Q(b, e)					
V(c)	0.0	0.0	0.0	0.0	0.0
Q(c, b)					
V(d)	**4.0**	4.6	**4.6**	4.6	4.6
Q(d, a)		5.1			
Q(d, e)		**4.6**			
V(e)	2.0	2.0	**2.0**	2.0	**2.0**
Q(e, b)				**2.0**	
Q(e, f)				3.0	
V(f)	2.0	2.0	2.0	2.0	2.0
Q(f, c)					
Q(f, e)					

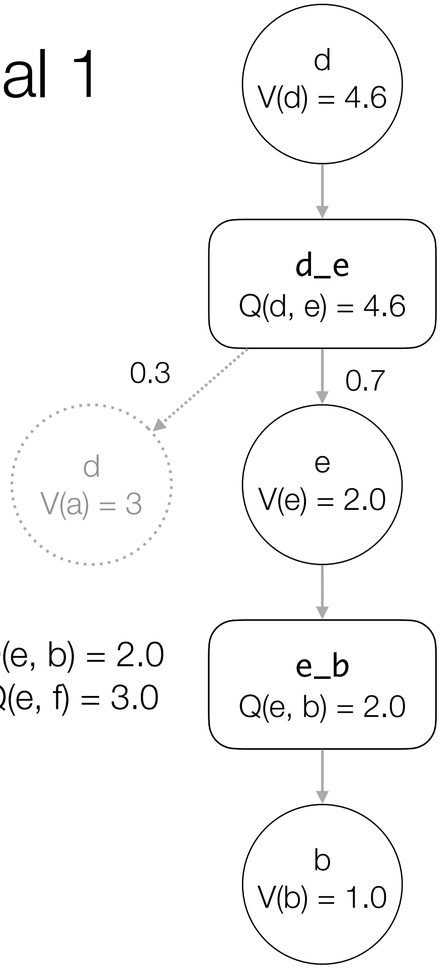
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

Trial 1

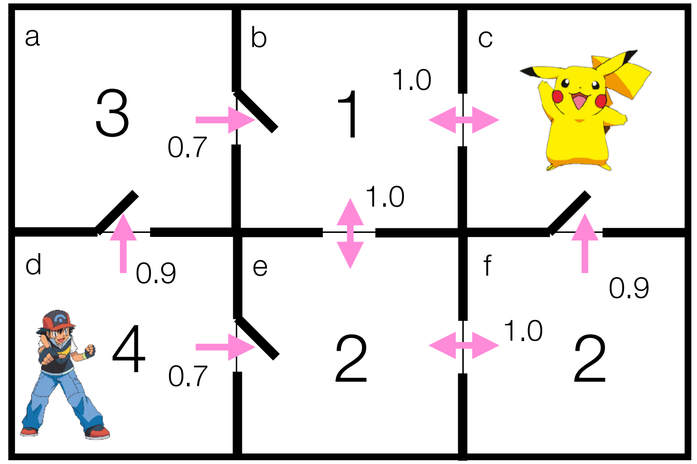


Q(e, b) = 2.0
Q(e, f) = 3.0

The action has two **possible outcomes**, which one is the successor state?

We ***sample*** an outcome following the transition distribution

Them repeat this until we reach the **goal** (state c)



	Init	Q?	Backup	Q?	Backup
V(a)	3.0	3.0	3.0	3.0	3.0
Q(a, b)					
V(b)	1.0	1.0	1.0	1.0	1.0
Q(b, c)					
Q(b, e)					
V(c)	0.0	0.0	0.0	0.0	0.0
Q(c, b)					
V(d)	**4.0**	4.6	**4.6**	4.6	4.6
Q(d, a)		5.1			
Q(d, e)		**4.6**			
V(e)	2.0	2.0	**2.0**	2.0	**2.0**
Q(e, b)				**2.0**	
Q(e, f)				3.0	
V(f)	2.0	2.0	2.0	2.0	2.0
Q(f, c)					
Q(f, e)					

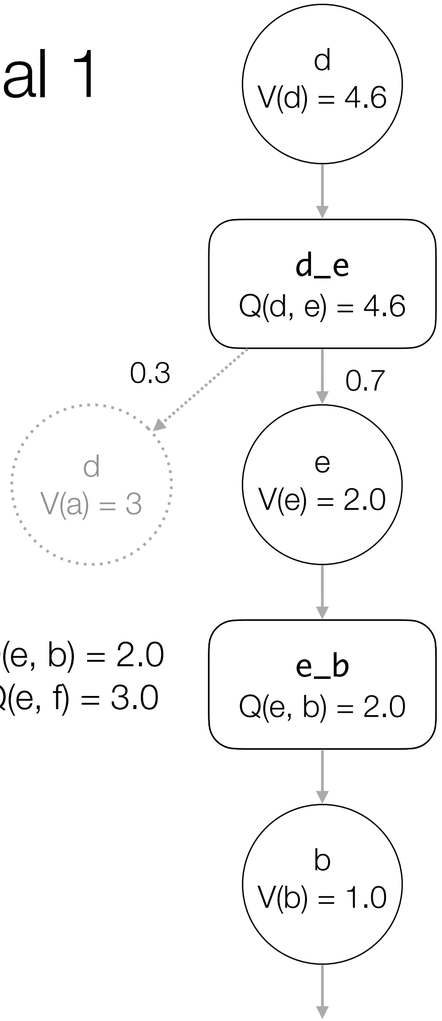
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

Trial 1

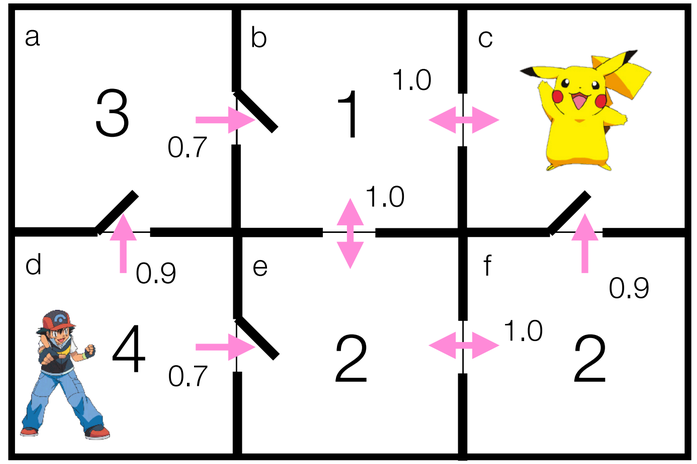


Q(e, b) = 2.0
Q(e, f) = 3.0

The action has two **possible outcomes**, which one is the successor state?

We ***sample*** an outcome following the transition distribution

Them repeat this until we reach the **goal** (state c)



	Init	Q?	Backup	Q?	Backup
V(a)	3.0	3.0	3.0	3.0	3.0
Q(a, b)					
V(b)	1.0	1.0	1.0	1.0	1.0
Q(b, c)					
Q(b, e)					
V(c)	0.0	0.0	0.0	0.0	0.0
Q(c, b)					
V(d)	**4.0**	4.6	**4.6**	4.6	4.6
Q(d, a)		5.1			
Q(d, e)		**4.6**			
V(e)	2.0	2.0	**2.0**	2.0	**2.0**
Q(e, b)				**2.0**	
Q(e, f)				3.0	
V(f)	2.0	2.0	2.0	2.0	2.0
Q(f, c)					
Q(f, e)					

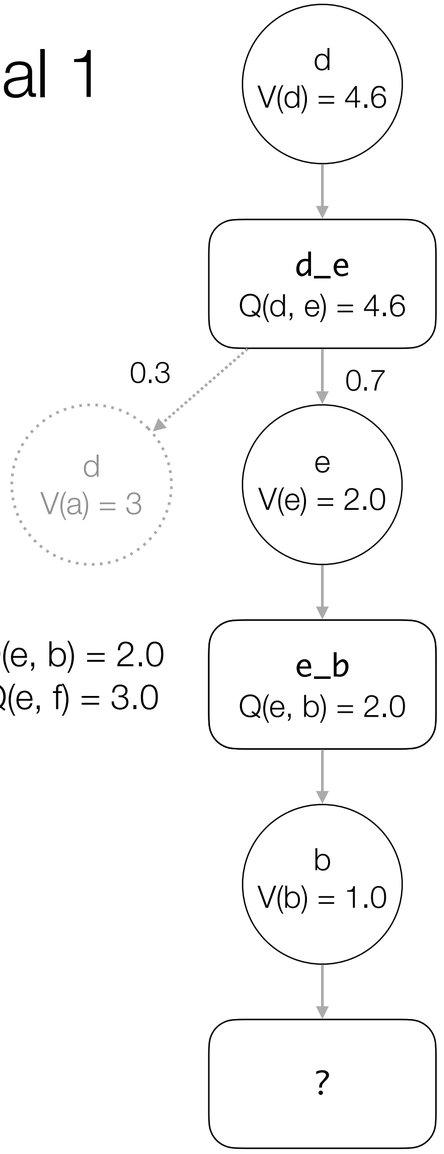
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

Trial 1

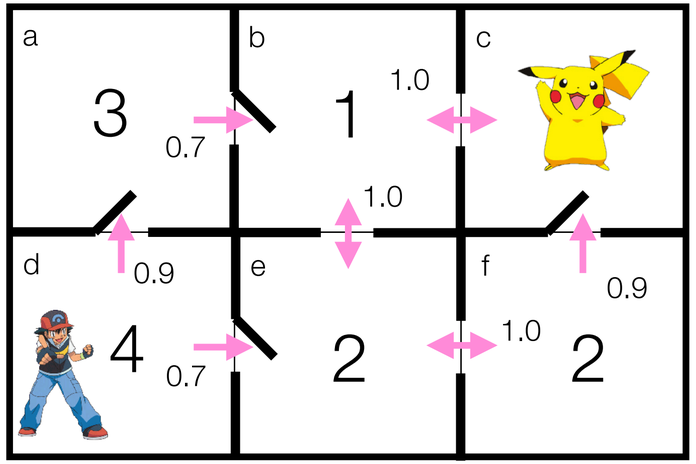


Q(e, b) = 2.0
Q(e, f) = 3.0

The action has two **possible outcomes**, which one is the successor state?

We ***sample*** an outcome following the transition distribution

Them repeat this until we reach the **goal** (state c)



	Init	Q?	Backup	Q?	Backup
V(a)	3.0	3.0	3.0	3.0	3.0
Q(a, b)					
V(b)	1.0	1.0	1.0	1.0	1.0
Q(b, c)					
Q(b, e)					
V(c)	0.0	0.0	0.0	0.0	0.0
Q(c, b)					
V(d)	**4.0**	4.6	**4.6**	4.6	4.6
Q(d, a)		5.1			
Q(d, e)		**4.6**			
V(e)	2.0	2.0	**2.0**	2.0	**2.0**
Q(e, b)				**2.0**	
Q(e, f)				3.0	
V(f)	2.0	2.0	2.0	2.0	2.0
Q(f, c)					
Q(f, e)					

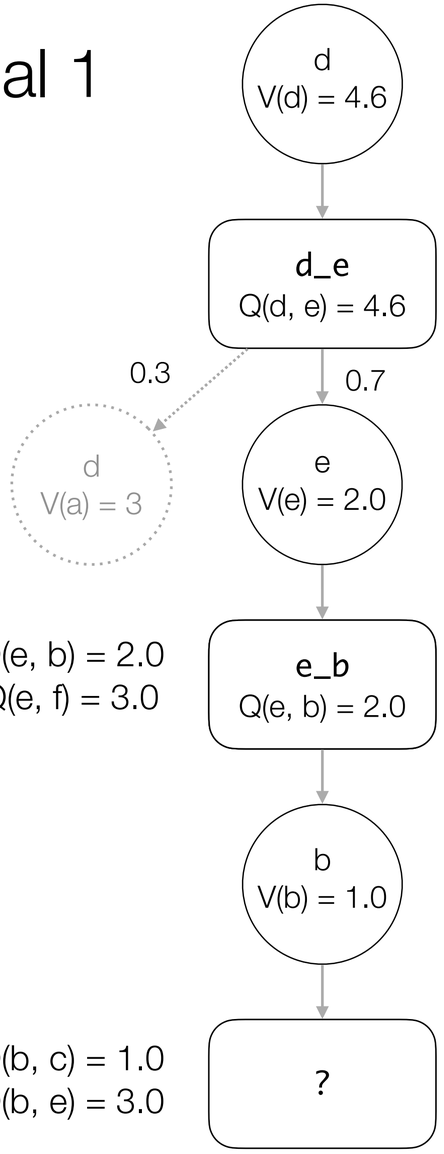
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

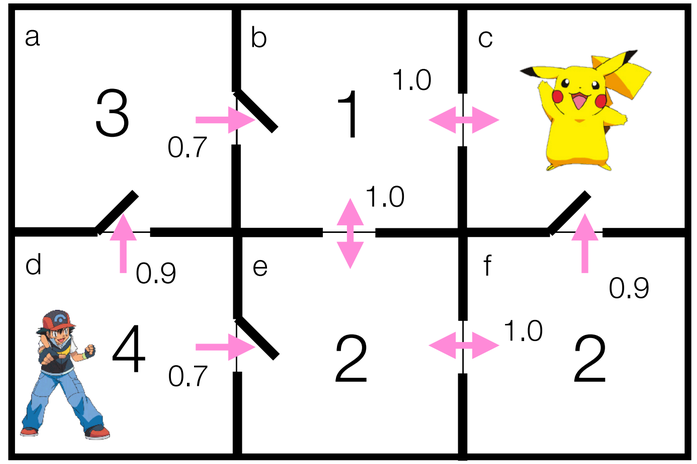
Trial 1



The action has two **possible outcomes**, which one is the successor state?

We ***sample*** an outcome following the transition distribution

Them repeat this until we reach the **goal** (state c)



	Init	Q?	Backup	Q?	Backup
V(a)	3.0	3.0	3.0	3.0	3.0
Q(a, b)					
V(b)	1.0	1.0	1.0	1.0	1.0
Q(b, c)					
Q(b, e)					
V(c)	0.0	0.0	0.0	0.0	0.0
Q(c, b)					
V(d)	**4.0**	4.6	**4.6**	4.6	4.6
Q(d, a)		5.1			
Q(d, e)		**4.6**			
V(e)	2.0	2.0	**2.0**	2.0	**2.0**
Q(e, b)				**2.0**	
Q(e, f)				3.0	
V(f)	2.0	2.0	2.0	2.0	2.0
Q(f, c)					
Q(f, e)					

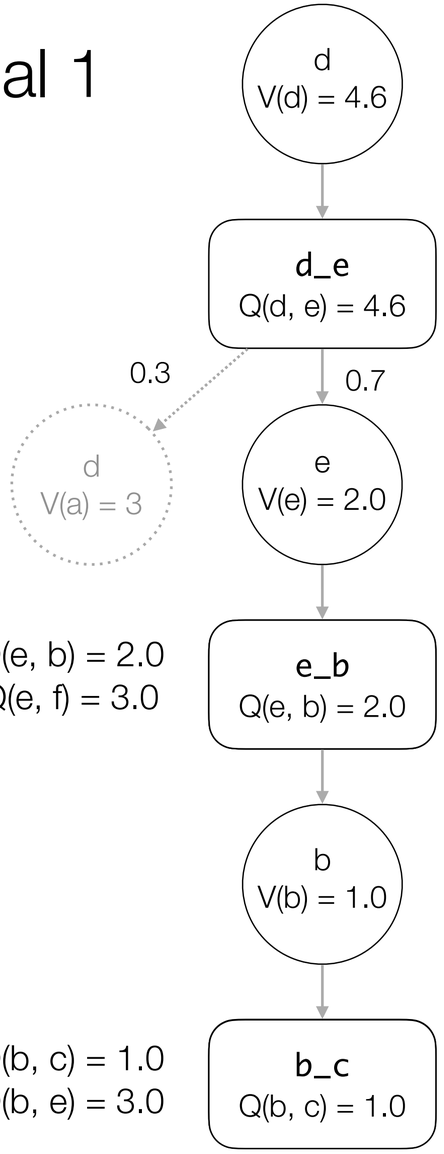
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

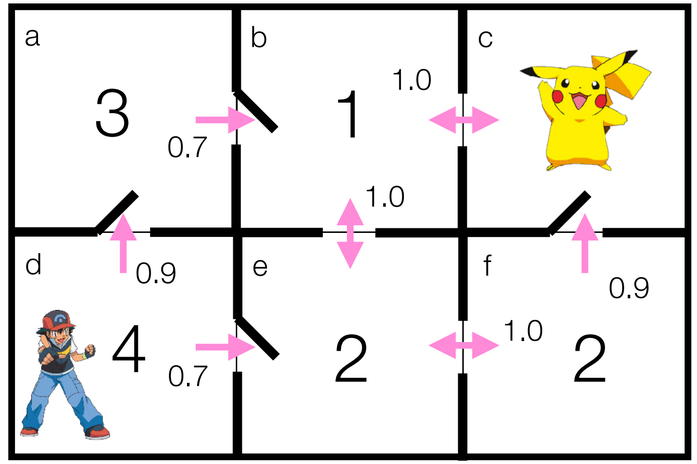
Trial 1



The action has two **possible outcomes**, which one is the successor state?

We ***sample*** an outcome following the transition distribution

Them repeat this until we reach the **goal** (state c)



	Init	Q?	Backup	Q?	Backup
V(a)	3.0	3.0	3.0	3.0	3.0
Q(a, b)					
V(b)	1.0	1.0	1.0	1.0	1.0
Q(b, c)					
Q(b, e)					
V(c)	0.0	0.0	0.0	0.0	0.0
Q(c, b)					
V(d)	**4.0**	4.6	**4.6**	4.6	4.6
Q(d, a)		5.1			
Q(d, e)		**4.6**			
V(e)	2.0	2.0	**2.0**	2.0	**2.0**
Q(e, b)				**2.0**	
Q(e, f)				3.0	
V(f)	2.0	2.0	2.0	2.0	2.0
Q(f, c)					
Q(f, e)					

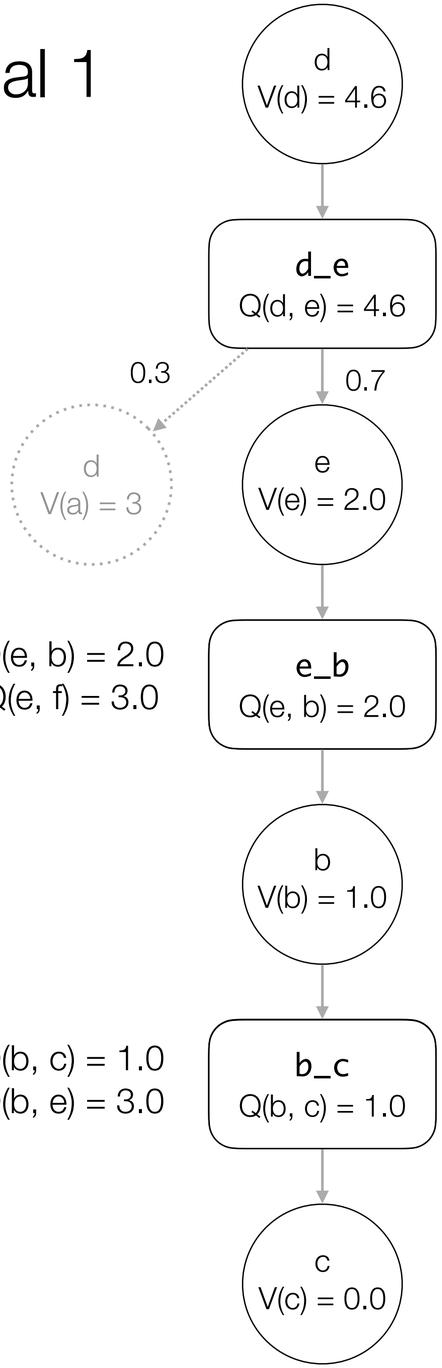
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

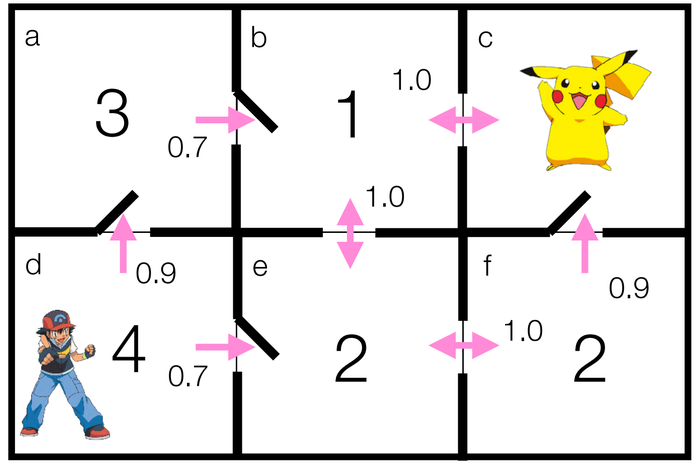
Trial 1



The action has two **possible outcomes**, which one is the successor state?

We ***sample*** an outcome following the transition distribution

Them repeat this until we reach the **goal** (state c)



	Init	Q?	Backup	Q?	Backup
V(a)	3.0	3.0	3.0	3.0	3.0
Q(a, b)					
V(b)	1.0	1.0	1.0	1.0	1.0
Q(b, c)					
Q(b, e)					
V(c)	0.0	0.0	0.0	0.0	0.0
Q(c, b)					
V(d)	**4.0**	4.6	**4.6**	4.6	4.6
Q(d, a)		5.1			
Q(d, e)		**4.6**			
V(e)	2.0	2.0	**2.0**	2.0	**2.0**
Q(e, b)				**2.0**	
Q(e, f)				3.0	
V(f)	2.0	2.0	2.0	2.0	2.0
Q(f, c)					
Q(f, e)					

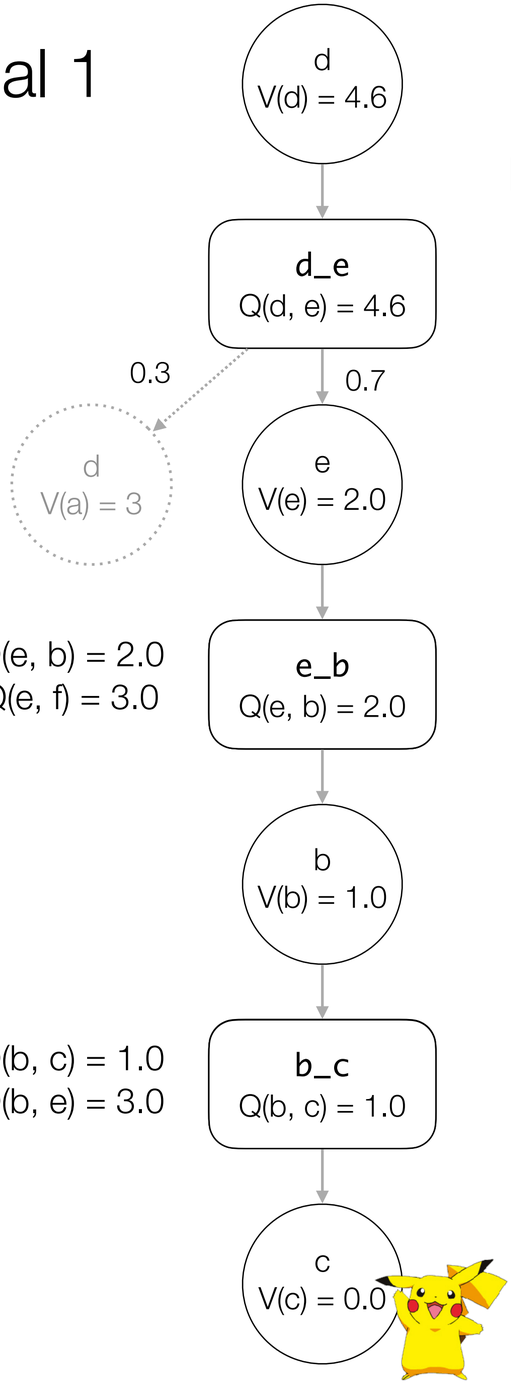
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact
V(a)	3.858
V(b)	1.000
V(c)	0.000
V(d)	4.859
V(e)	2.000
V(f)	2.222

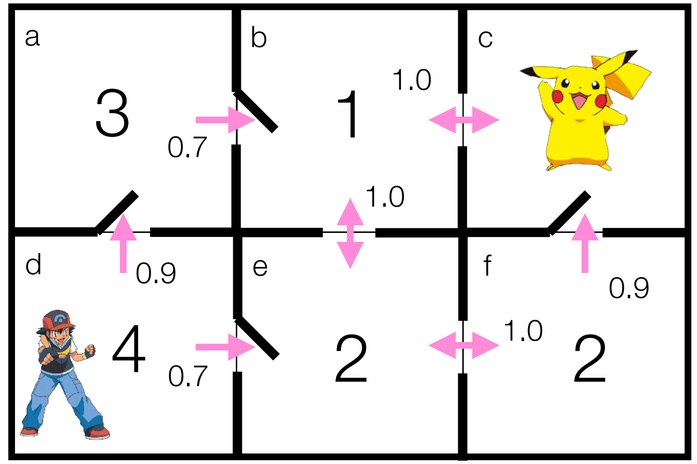
Trial 1



The action has two **possible outcomes**, which one is the successor state?

We ***sample*** an outcome following the transition distribution

Them repeat this until we reach the **goal** (state c)



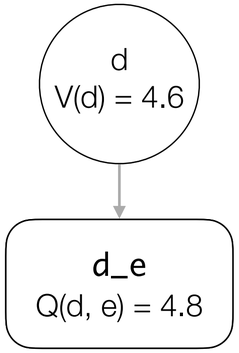
	Init	Q?	Backup	Q?	Backup
V(a)	3.0	3.0	3.0	3.0	3.0
Q(a, b)					
V(b)	1.0	1.0	1.0	1.0	1.0
Q(b, c)					
Q(b, e)					
V(c)	0.0	0.0	0.0	0.0	0.0
Q(c, b)					
V(d)	**4.0**	4.6	**4.6**	4.6	4.6
Q(d, a)		5.1			
Q(d, e)		**4.6**			
V(e)	2.0	2.0	**2.0**	2.0	**2.0**
Q(e, b)				**2.0**	
Q(e, f)				3.0	
V(f)	2.0	2.0	2.0	2.0	2.0
Q(f, c)					
Q(f, e)					

Algorithm 4.3: RTDP

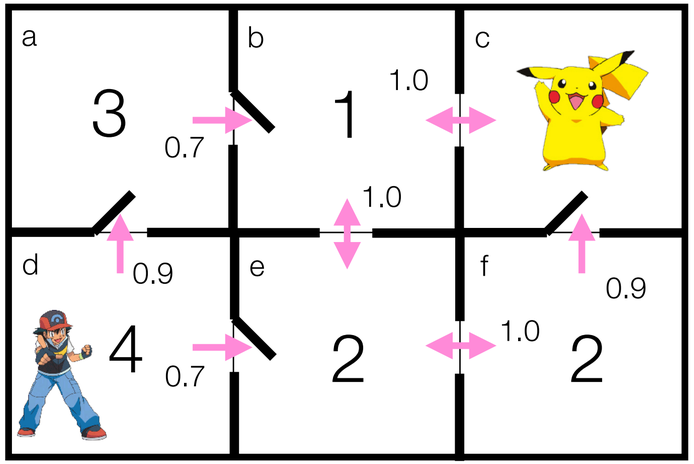
```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

Trial 2



heuristic



	Exact	Trial 1
V(a)	3.858	3.000
V(b)	1.000	1.000
V(c)	0.000	0.000
V(d)	4.859	4.600
V(e)	2.000	2.000
V(f)	2.222	2.000

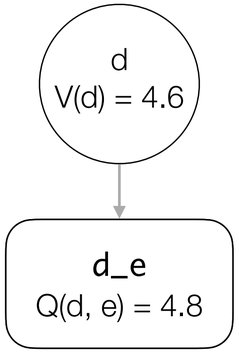
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

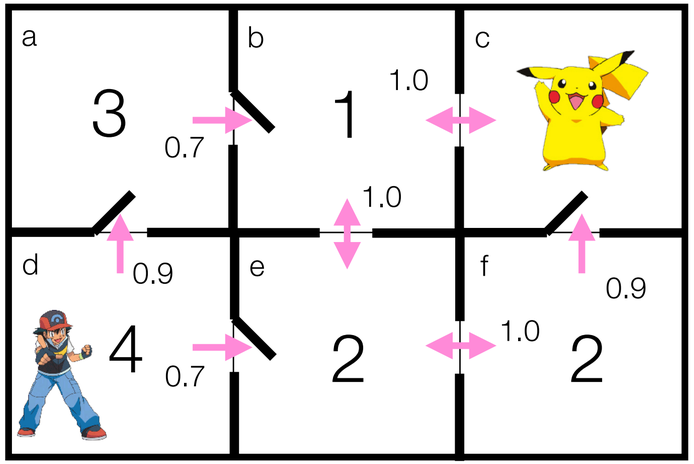
Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact	Trial 1
V(a)	3.858	3.000
V(b)	1.000	1.000
V(c)	0.000	0.000
V(d)	4.859	4.600
V(e)	2.000	2.000
V(f)	2.222	2.000

Trial 2



heuristic



	Init
V(a)	3.0
Q(a, b)	
V(b)	1.0
Q(b, c)	
Q(b, e)	
V(c)	0.0
Q(c, b)	
V(d)	**4.6**
Q(d, a)	
Q(d, e)	
V(e)	2.0
Q(e, b)	
Q(e, f)	
V(f)	2.0
Q(f, c)	
Q(f, e)	

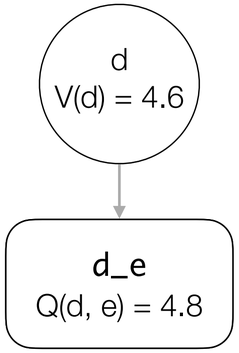
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

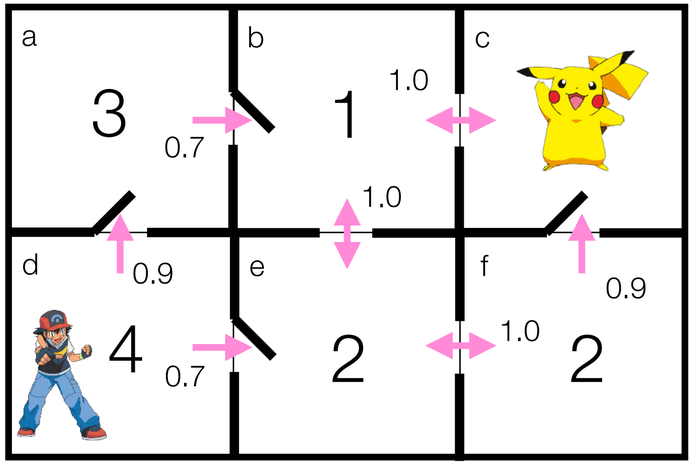
	Exact	Trial 1
V(a)	3.858	3.000
V(b)	1.000	1.000
V(c)	0.000	0.000
V(d)	4.859	4.600
V(e)	2.000	2.000
V(f)	2.222	2.000

Trial 2



Q(d, a) = 5.2
Q(d, e) = 4.8

heuristic



	Init	Q?
V(a)	3.0	3.0
Q(a, b)		
V(b)	1.0	1.0
Q(b, c)		
Q(b, e)		
V(c)	0.0	0.0
Q(c, b)		
V(d)	**4.6**	4.8
Q(d, a)		5.2
Q(d, e)		**4.8**
V(e)	2.0	2.0
Q(e, b)		
Q(e, f)		
V(f)	2.0	2.0
Q(f, c)		
Q(f, e)		

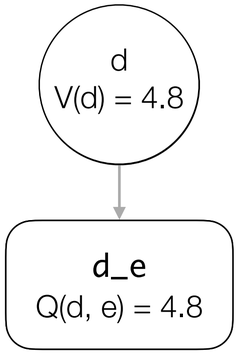
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

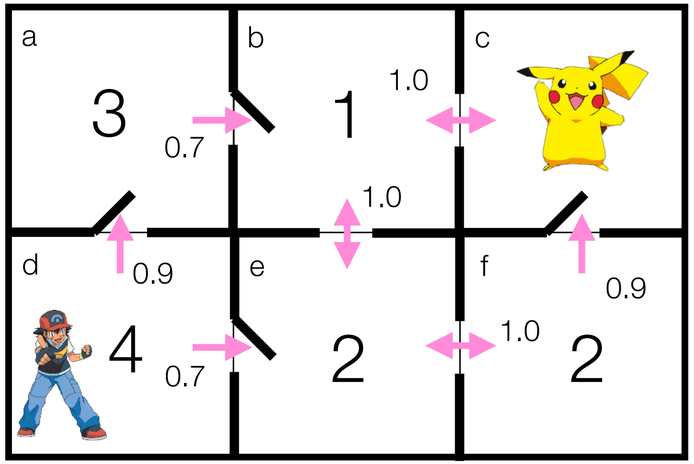
	Exact	Trial 1
V(a)	3.858	3.000
V(b)	1.000	1.000
V(c)	0.000	0.000
V(d)	4.859	4.600
V(e)	2.000	2.000
V(f)	2.222	2.000

Trial 2



Q(d, a) = 5.2
Q(d, e) = 4.8

heuristic



	Init	Q?
V(a)	3.0	3.0
Q(a, b)		
V(b)	1.0	1.0
Q(b, c)		
Q(b, e)		
V(c)	0.0	0.0
Q(c, b)		
V(d)	**4.6**	4.8
Q(d, a)		5.2
Q(d, e)		**4.8**
V(e)	2.0	2.0
Q(e, b)		
Q(e, f)		
V(f)	2.0	2.0
Q(f, c)		
Q(f, e)		

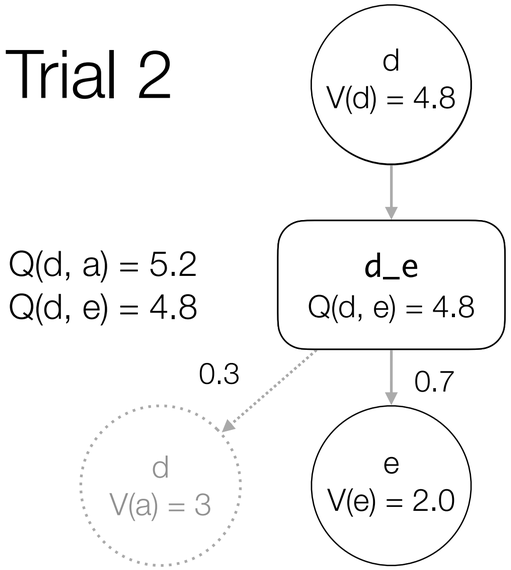
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

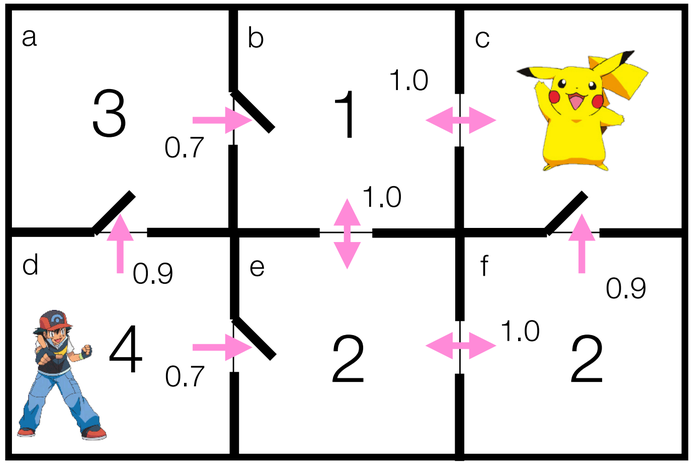
Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact	Trial 1
V(a)	3.858	3.000
V(b)	1.000	1.000
V(c)	0.000	0.000
V(d)	4.859	4.600
V(e)	2.000	2.000
V(f)	2.222	2.000

Trial 2



heuristic



	Init	Q?
V(a)	3.0	3.0
Q(a, b)		
V(b)	1.0	1.0
Q(b, c)		
Q(b, e)		
V(c)	0.0	0.0
Q(c, b)		
V(d)	**4.6**	4.8
Q(d, a)		5.2
Q(d, e)		**4.8**
V(e)	2.0	2.0
Q(e, b)		
Q(e, f)		
V(f)	2.0	2.0
Q(f, c)		
Q(f, e)		

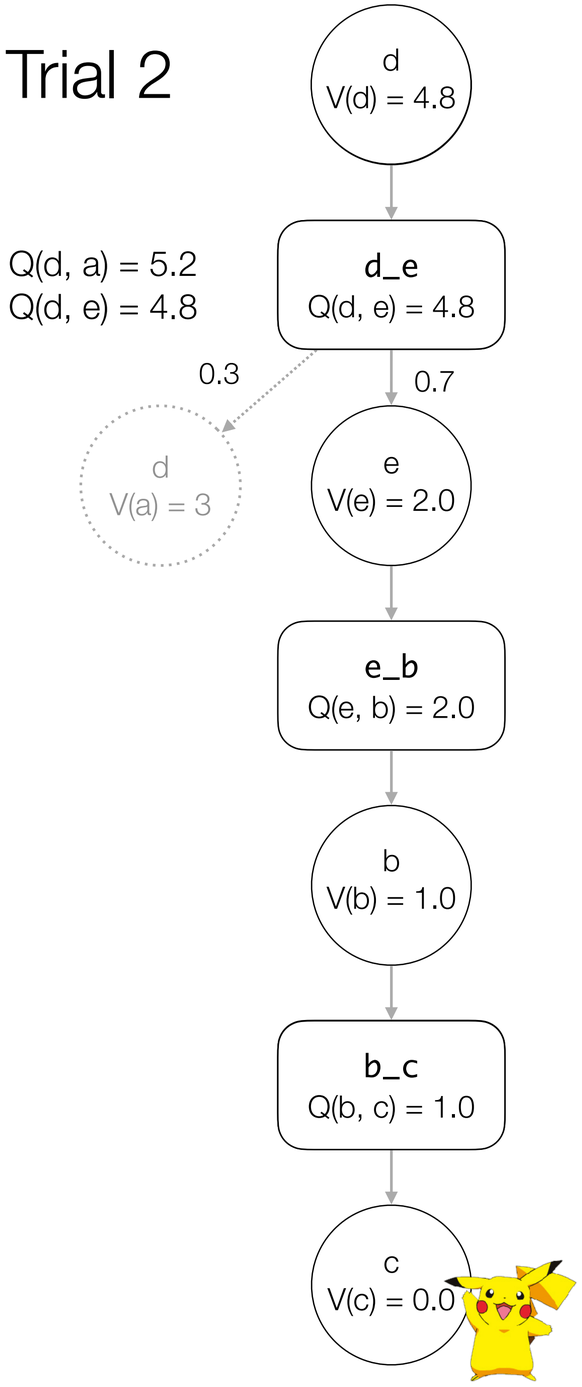
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

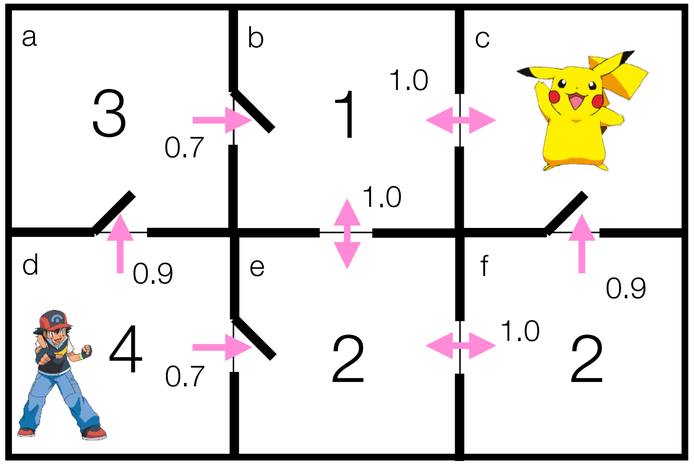
Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact	Trial 1
V(a)	3.858	3.000
V(b)	1.000	1.000
V(c)	0.000	0.000
V(d)	4.859	4.600
V(e)	2.000	2.000
V(f)	2.222	2.000

Trial 2



heuristic



	Init	Q?
V(a)	3.0	3.0
Q(a, b)		
V(b)	1.0	1.0
Q(b, c)		
Q(b, e)		
V(c)	0.0	0.0
Q(c, b)		
V(d)	**4.6**	4.8
Q(d, a)		5.2
Q(d, e)		**4.8**
V(e)	2.0	2.0
Q(e, b)		
Q(e, f)		
V(f)	2.0	2.0
Q(f, c)		
Q(f, e)		

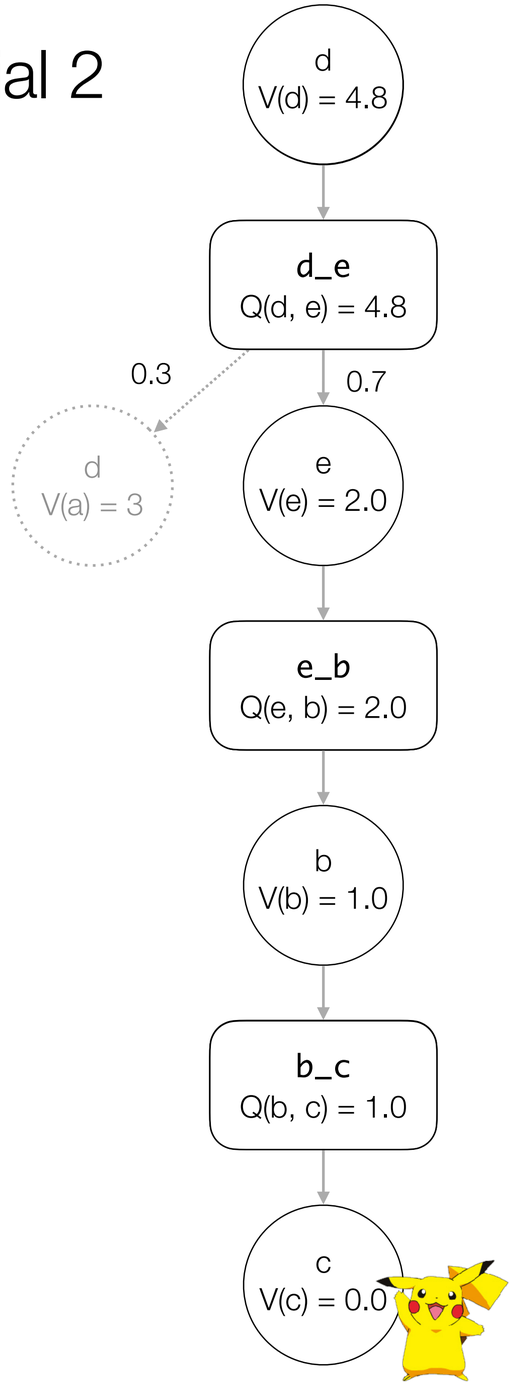
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

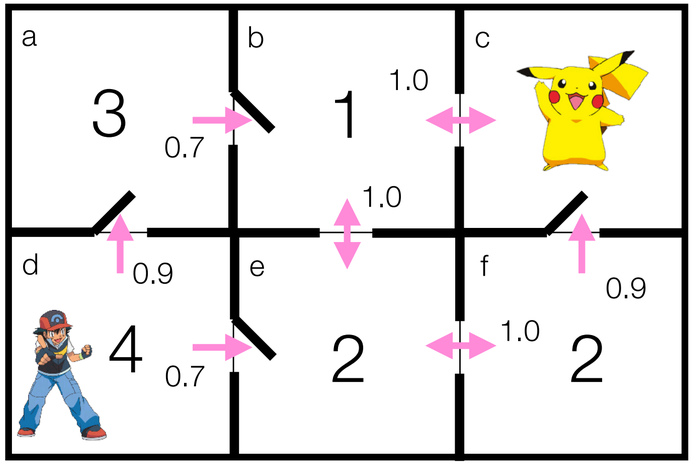
Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

	Exact	Trial 1	Trial 2
V(a)	3.858	3.000	3.000
V(b)	1.000	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**
V(d)	4.859	4.600	4.780
V(e)	2.000	**2.000**	**2.000**
V(f)	2.222	2.000	2.000

Trial 2



heuristic



	Init	Q?
V(a)	3.0	3.0
Q(a, b)		
V(b)	1.0	1.0
Q(b, c)		
Q(b, e)		
V(c)	0.0	0.0
Q(c, b)		
V(d)	**4.6**	4.8
Q(d, a)		5.2
Q(d, e)		**4.8**
V(e)	2.0	2.0
Q(e, b)		
Q(e, f)		
V(f)	2.0	2.0
Q(f, c)		
Q(f, e)		

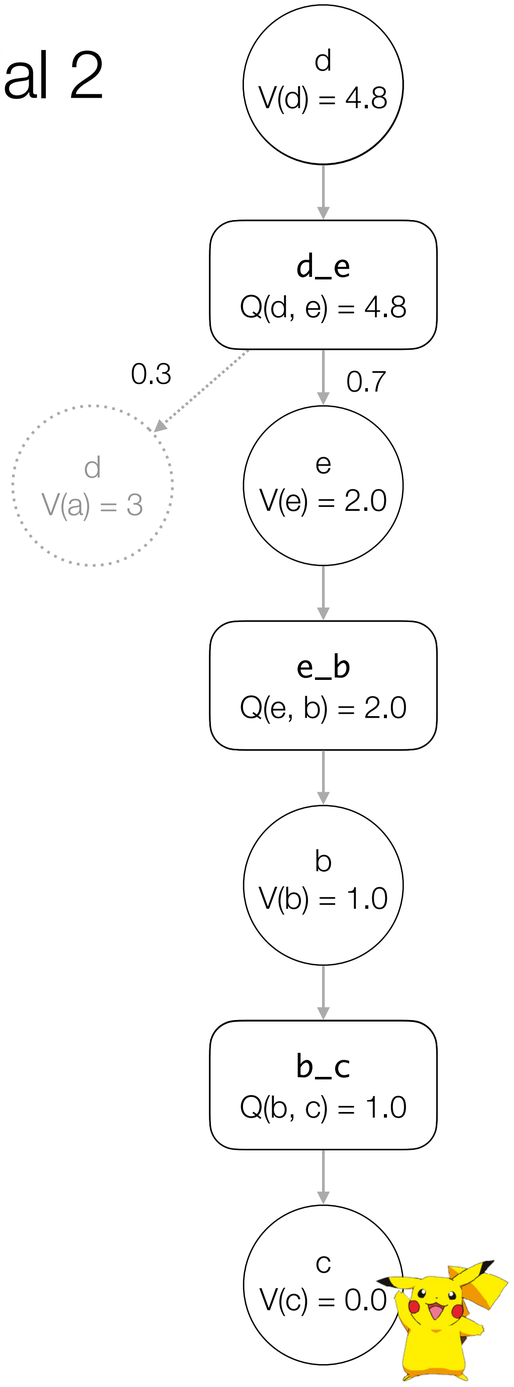
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

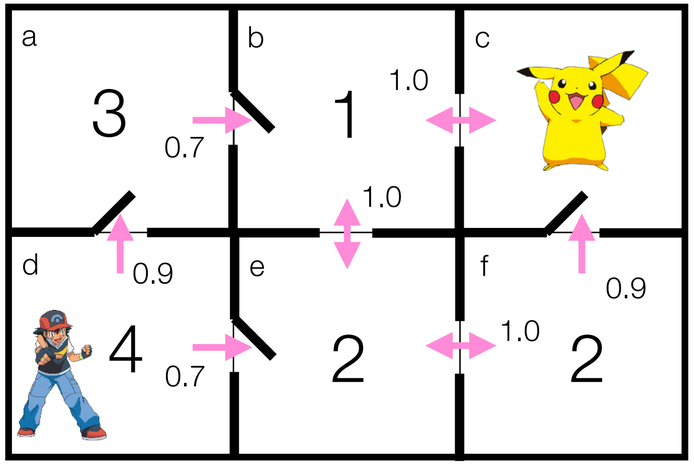
	Exact	Trial 1	Trial 2
V(a)	3.858	3.000	3.000
V(b)	1.000	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**
V(d)	4.859	4.600	4.780
V(e)	2.000	**2.000**	**2.000**
V(f)	2.222	2.000	2.000

Trial 2



Labelled RTDP

heuristic



	Init	Q?
V(a)	3.0	3.0
Q(a, b)		
V(b)	1.0	1.0
Q(b, c)		
Q(b, e)		
V(c)	0.0	0.0
Q(c, b)		
V(d)	**4.6**	4.8
Q(d, a)		5.2
Q(d, e)		**4.8**
V(e)	2.0	2.0
Q(e, b)		
Q(e, f)		
V(f)	2.0	2.0
Q(f, c)		
Q(f, e)		

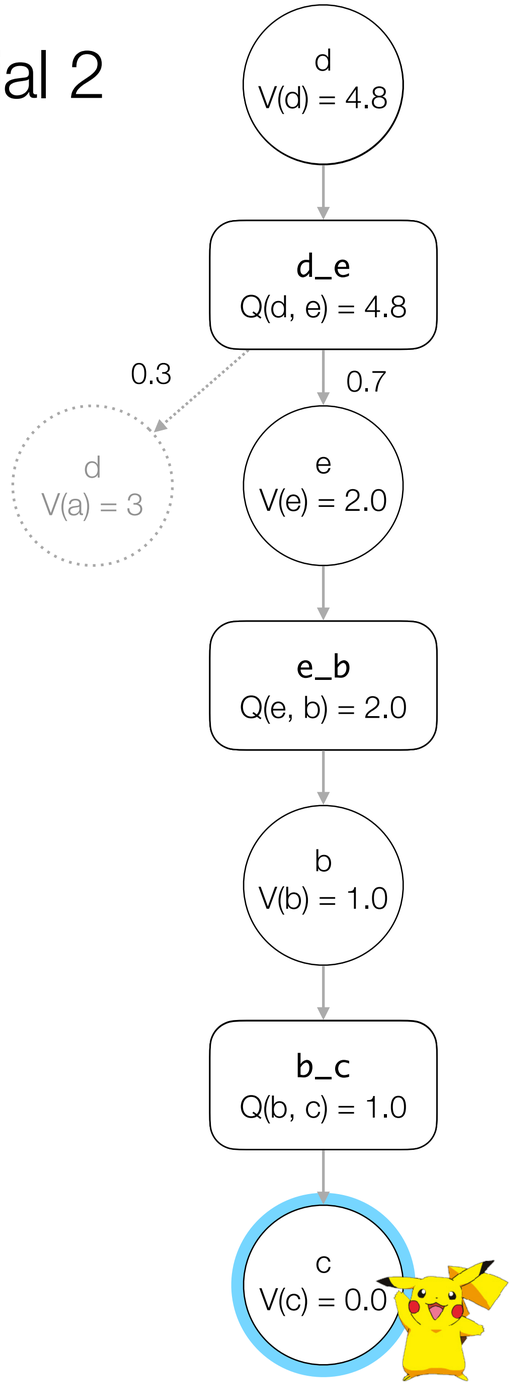
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

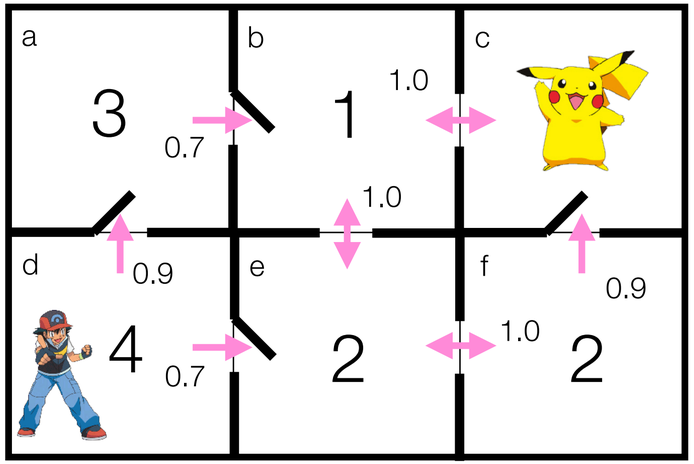
	Exact	Trial 1	Trial 2
V(a)	3.858	3.000	3.000
V(b)	1.000	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**
V(d)	4.859	4.600	4.780
V(e)	2.000	**2.000**	**2.000**
V(f)	2.222	2.000	2.000

Trial 2



Labelled RTDP

heuristic



	Init	Q?
V(a)	3.0	3.0
Q(a, b)		
V(b)	1.0	1.0
Q(b, c)		
Q(b, e)		
V(c)	0.0	0.0
Q(c, b)		
V(d)	**4.6**	4.8
Q(d, a)		5.2
Q(d, e)		**4.8**
V(e)	2.0	2.0
Q(e, b)		
Q(e, f)		
V(f)	2.0	2.0
Q(f, c)		
Q(f, e)		

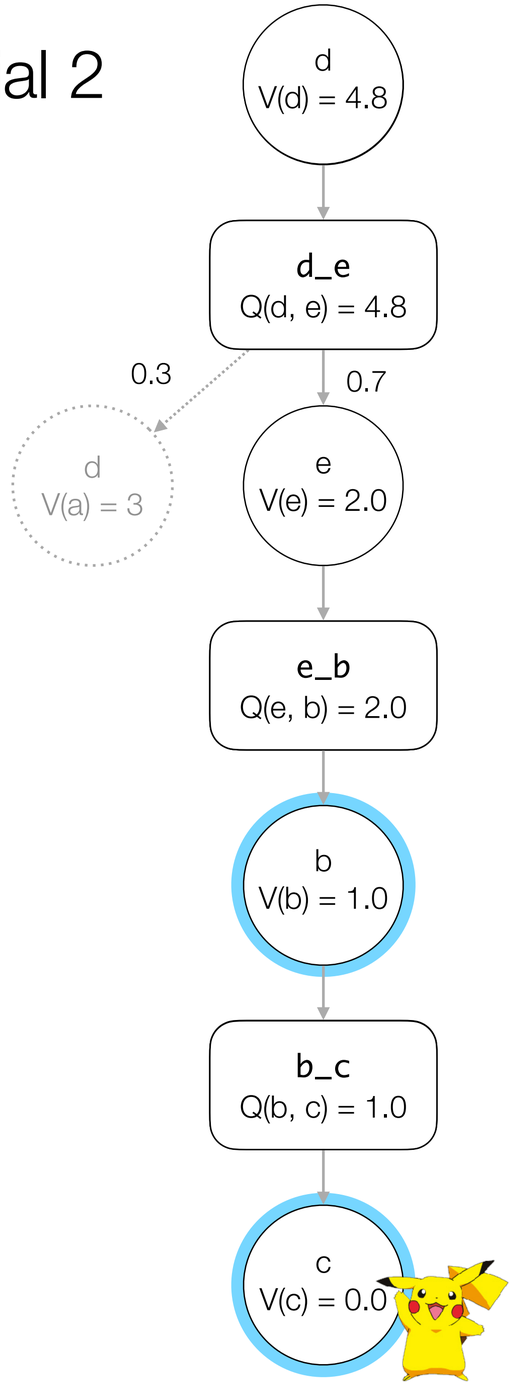
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

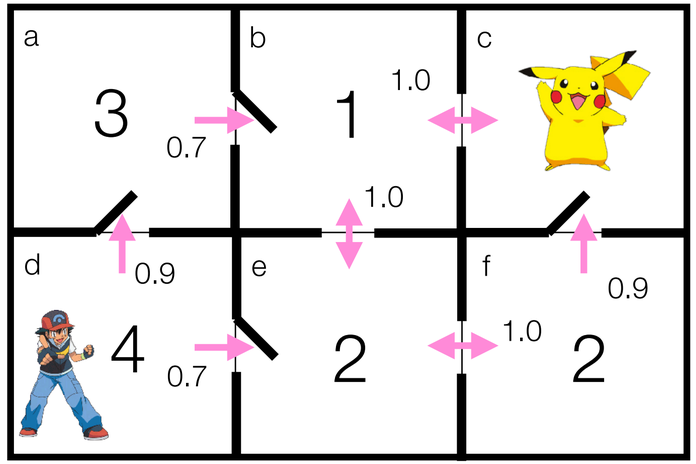
	Exact	Trial 1	Trial 2
V(a)	3.858	3.000	3.000
V(b)	1.000	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**
V(d)	4.859	4.600	4.780
V(e)	2.000	**2.000**	**2.000**
V(f)	2.222	2.000	2.000

Trial 2



Labelled RTDP

heuristic



	Init	Q?
V(a)	3.0	3.0
Q(a, b)		
V(b)	1.0	1.0
Q(b, c)		
Q(b, e)		
V(c)	0.0	0.0
Q(c, b)		
V(d)	**4.6**	4.8
Q(d, a)		5.2
Q(d, e)		**4.8**
V(e)	2.0	2.0
Q(e, b)		
Q(e, f)		
V(f)	2.0	2.0
Q(f, c)		
Q(f, e)		

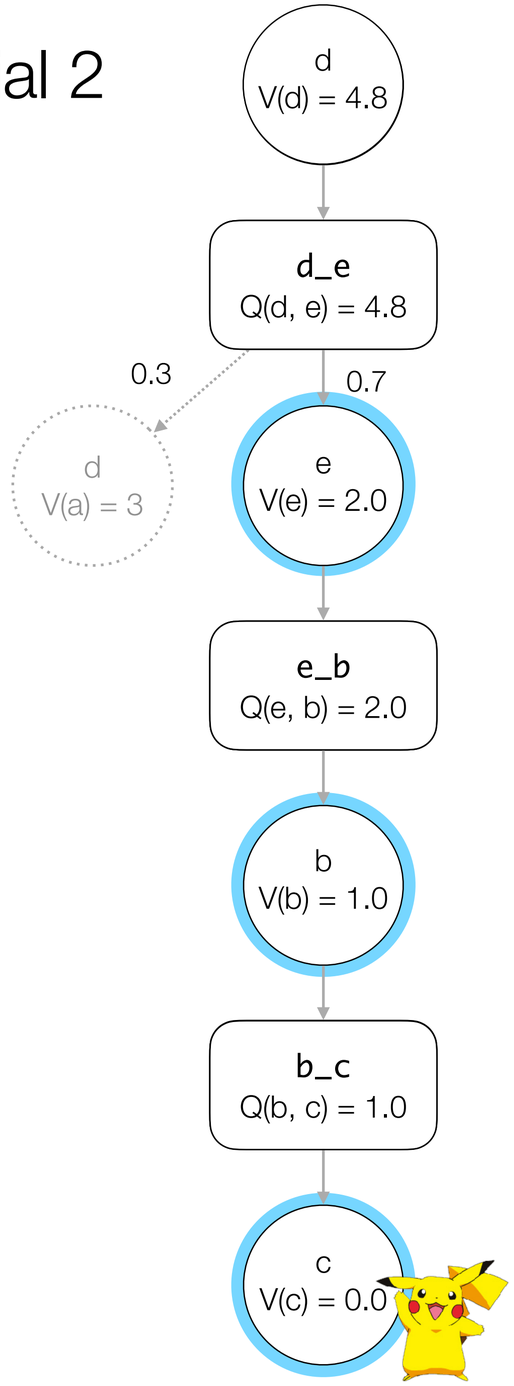
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

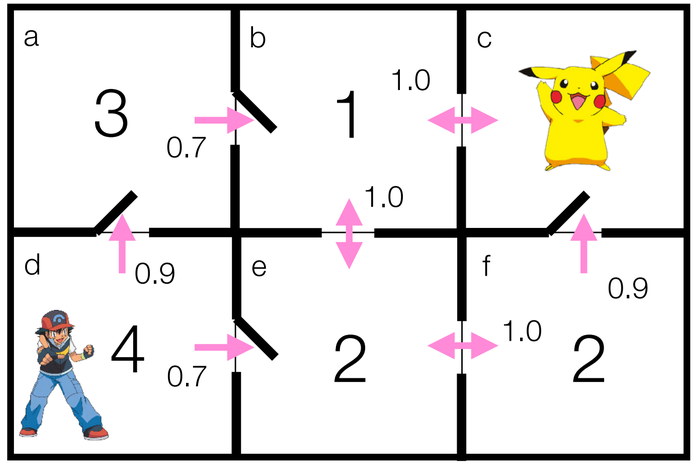
	Exact	Trial 1	Trial 2
V(a)	3.858	3.000	3.000
V(b)	1.000	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**
V(d)	4.859	4.600	4.780
V(e)	2.000	**2.000**	**2.000**
V(f)	2.222	2.000	2.000

Trial 2



Labelled RTDP

heuristic



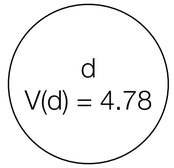
	Init	Q?
V(a)	3.0	3.0
Q(a, b)		
V(b)	1.0	1.0
Q(b, c)		
Q(b, e)		
V(c)	0.0	0.0
Q(c, b)		
V(d)	**4.6**	4.8
Q(d, a)		5.2
Q(d, e)		**4.8**
V(e)	2.0	2.0
Q(e, b)		
Q(e, f)		
V(f)	2.0	2.0
Q(f, c)		
Q(f, e)		

Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

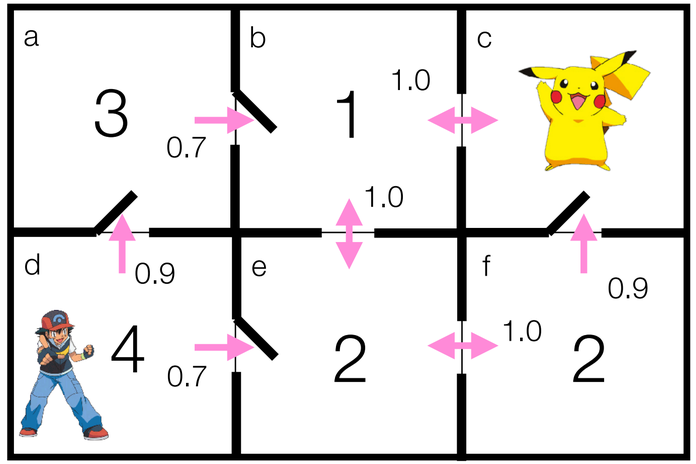
Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

Trial 3



Labelled
RTDP

heuristic

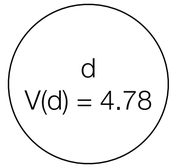


Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

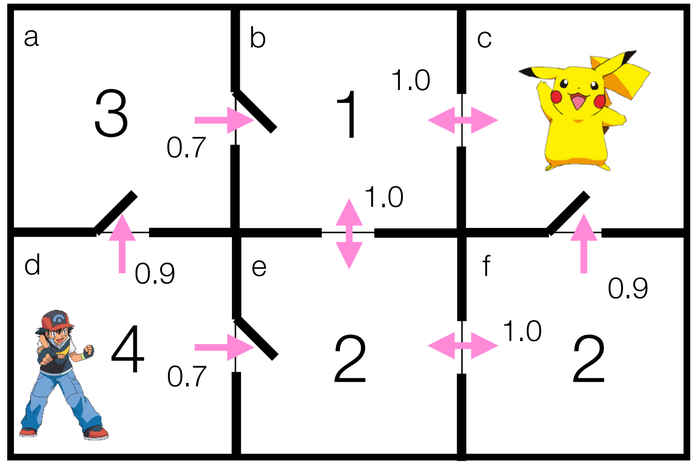
Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

Trial 3



Labelled
RTDP

heuristic



	Exact	Trial 1	Trial 2
V(a)	3.858	3.000	3.000
V(b)	1.000	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**
V(d)	4.859	4.600	4.780
V(e)	2.000	**2.000**	**2.000**
V(f)	2.222	2.000	2.000

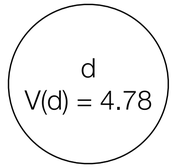
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

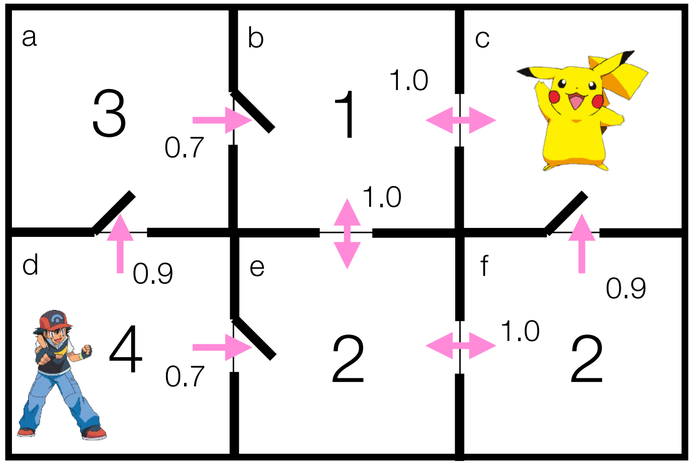
	Exact	Trial 1	Trial 2
V(a)	3.858	3.000	3.000
V(b)	1.000	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**
V(d)	4.859	4.600	4.780
V(e)	2.000	**2.000**	**2.000**
V(f)	2.222	2.000	2.000

Trial 3



Labelled RTDP

heuristic



	Init	Q?	
V(a)	3.00	3.00	3.00
Q(a, b)			
V(b)	1.00	1.00	1.00
Q(b, c)			
Q(b, e)			
V(c)	0.00	0.00	0.00
Q(c, b)			
V(d)	**4.78**	4.83	4.83
Q(d, a)		5.18	
Q(d, e)		**4.83**	
V(e)	2.00	2.00	2.00
Q(e, b)			
Q(e, f)			
V(f)	2.00	2.00	2.00
Q(f, c)			
Q(f, e)			

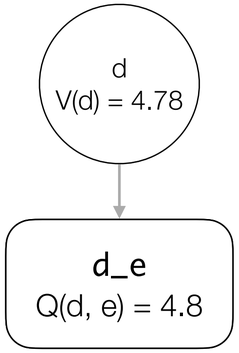
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

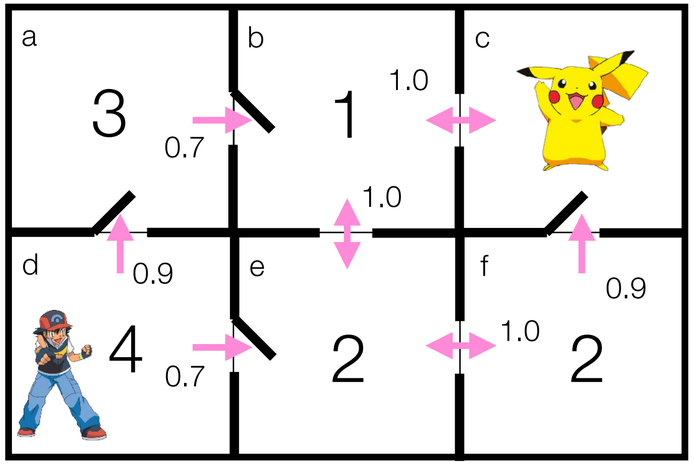
	Exact	Trial 1	Trial 2
V(a)	3.858	3.000	3.000
V(b)	1.000	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**
V(d)	4.859	4.600	4.780
V(e)	2.000	**2.000**	**2.000**
V(f)	2.222	2.000	2.000

Trial 3



Labelled RTDP

heuristic



	Init	Q?	
V(a)	3.00	3.00	3.00
Q(a, b)			
V(b)	1.00	1.00	1.00
Q(b, c)			
Q(b, e)			
V(c)	0.00	0.00	0.00
Q(c, b)			
V(d)	**4.78**	4.83	4.83
Q(d, a)		5.18	
Q(d, e)		**4.83**	
V(e)	2.00	2.00	2.00
Q(e, b)			
Q(e, f)			
V(f)	2.00	2.00	2.00
Q(f, c)			
Q(f, e)			

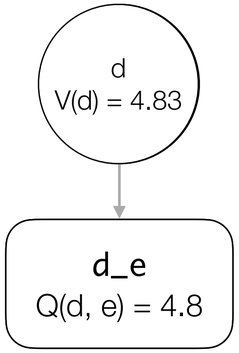
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

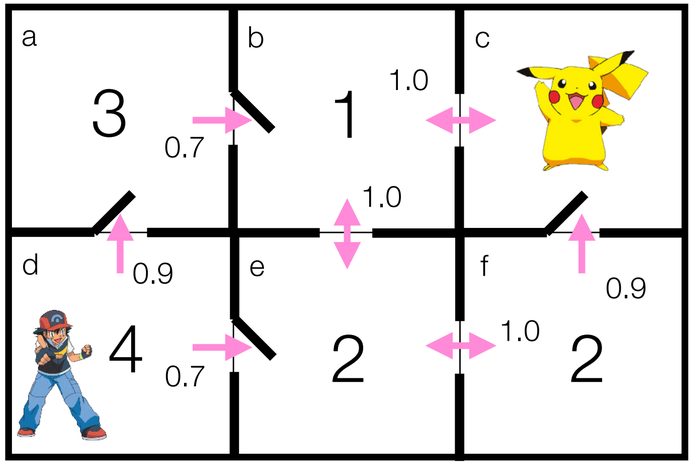
	Exact	Trial 1	Trial 2
V(a)	3.858	3.000	3.000
V(b)	1.000	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**
V(d)	4.859	4.600	4.780
V(e)	2.000	**2.000**	**2.000**
V(f)	2.222	2.000	2.000

Trial 3



Labelled RTDP

heuristic



	Init	Q?	
V(a)	3.00	3.00	3.00
Q(a, b)			
V(b)	1.00	1.00	1.00
Q(b, c)			
Q(b, e)			
V(c)	0.00	0.00	0.00
Q(c, b)			
V(d)	**4.78**	4.83	4.83
Q(d, a)		5.18	
Q(d, e)		**4.83**	
V(e)	2.00	2.00	2.00
Q(e, b)			
Q(e, f)			
V(f)	2.00	2.00	2.00
Q(f, c)			
Q(f, e)			

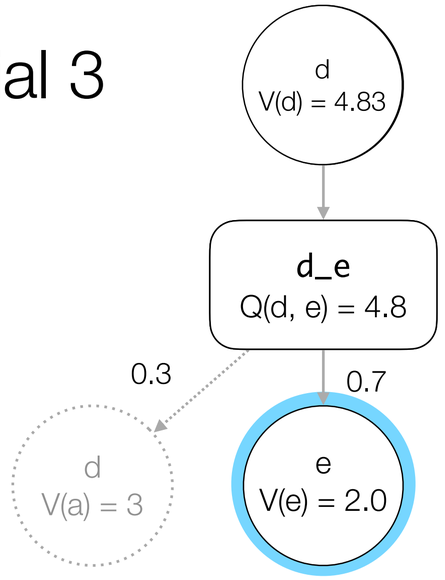
Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

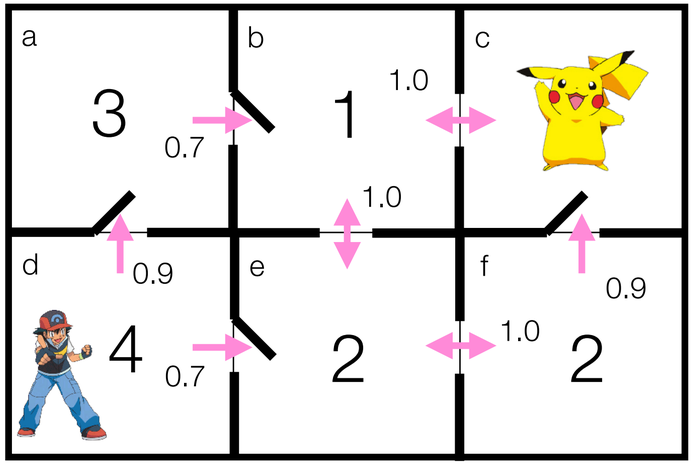
	Exact	Trial 1	Trial 2
V(a)	3.858	3.000	3.000
V(b)	1.000	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**
V(d)	4.859	4.600	4.780
V(e)	2.000	**2.000**	**2.000**
V(f)	2.222	2.000	2.000

Trial 3



Labelled RTDP

heuristic



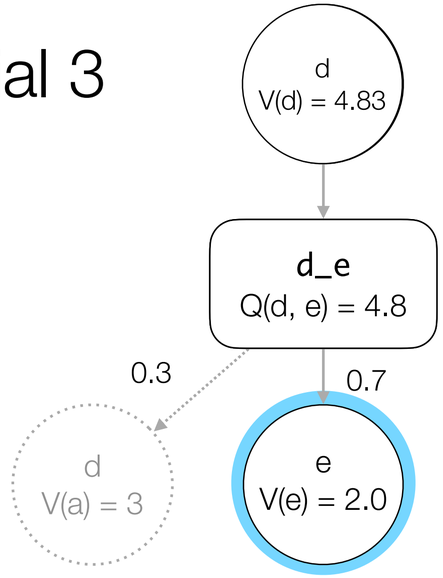
	Init	Q?	
V(a)	3.00	3.00	3.00
Q(a, b)			
V(b)	1.00	1.00	1.00
Q(b, c)			
Q(b, e)			
V(c)	0.00	0.00	0.00
Q(c, b)			
V(d)	**4.78**	4.83	4.83
Q(d, a)		5.18	
Q(d, e)		**4.83**	
V(e)	2.00	2.00	2.00
Q(e, b)			
Q(e, f)			
V(f)	2.00	2.00	2.00
Q(f, c)			
Q(f, e)			

Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

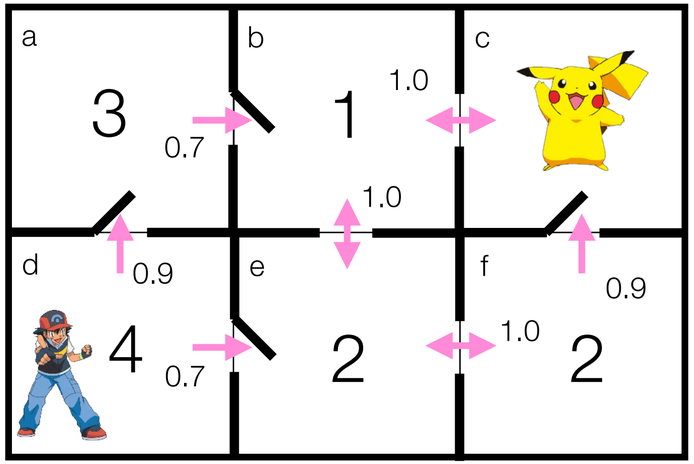
Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

Trial 3



Labelled RTDP

heuristic



	Exact	Trial 1	Trial 2	Trial 3
V(a)	3.858	3.000	3.000	3.000
V(b)	1.000	**1.000**	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**	**0.000**
V(d)	4.859	4.600	4.780	4.834
V(e)	2.000	**2.000**	**2.000**	**2.000**
V(f)	2.222	2.000	2.000	2.000

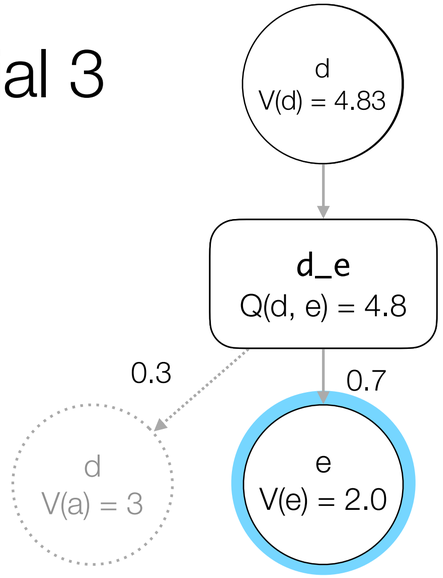
	Init	Q?	
V(a)	3.00	3.00	3.00
Q(a, b)			
V(b)	1.00	1.00	1.00
Q(b, c)			
Q(b, e)			
V(c)	0.00	0.00	0.00
Q(c, b)			
V(d)	**4.78**	4.83	4.83
Q(d, a)		5.18	
Q(d, e)		**4.83**	
V(e)	2.00	2.00	2.00
Q(e, b)			
Q(e, f)			
V(f)	2.00	2.00	2.00
Q(f, c)			
Q(f, e)			

Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

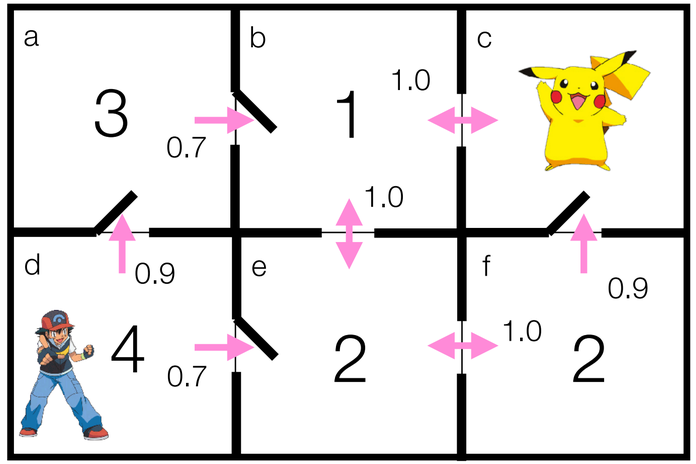
Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

Trial 3



Labelled RTDP

heuristic



	Exact	Trial 1	Trial 2	Trial 3	Trial 4
V(a)	3.858	3.000	3.000	3.000	3.000
V(b)	1.000	**1.000**	**1.000**	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**	**0.000**	**0.000**
V(d)	4.859	4.600	4.780	4.834	4.850
V(e)	2.000	**2.000**	**2.000**	**2.000**	**2.000**
V(f)	2.222	2.000	2.000	2.000	2.000

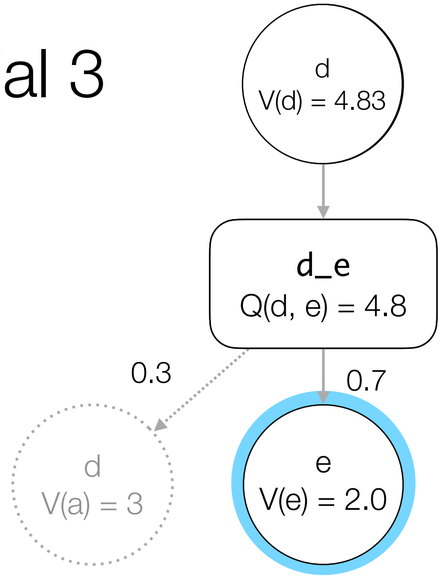
	Init	Q?	
V(a)	3.00	3.00	3.00
Q(a, b)			
V(b)	1.00	1.00	1.00
Q(b, c)			
Q(b, e)			
V(c)	0.00	0.00	0.00
Q(c, b)			
V(d)	**4.78**	4.83	4.83
Q(d, a)		5.18	
Q(d, e)		**4.83**	
V(e)	2.00	2.00	2.00
Q(e, b)			
Q(e, f)			
V(f)	2.00	2.00	2.00
Q(f, c)			
Q(f, e)			

Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

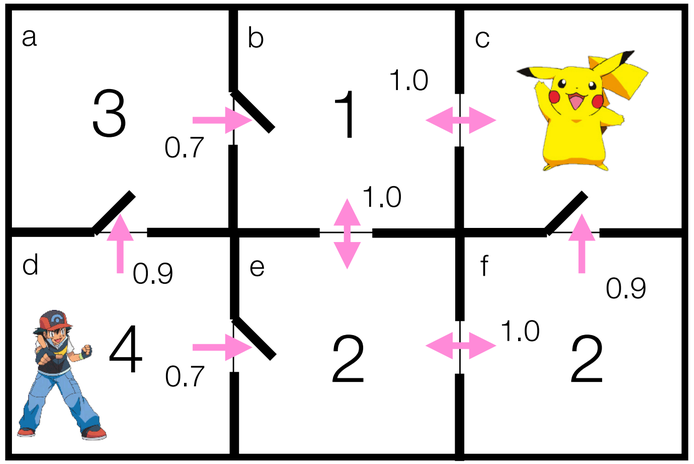
Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

Trial 3



Labelled RTDP

heuristic



	Exact	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
V(a)	3.858	3.000	3.000	3.000	3.000	3.000
V(b)	1.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
V(d)	4.859	4.600	4.780	4.834	4.850	4.855
V(e)	2.000	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**
V(f)	2.222	2.000	2.000	2.000	2.000	2.000

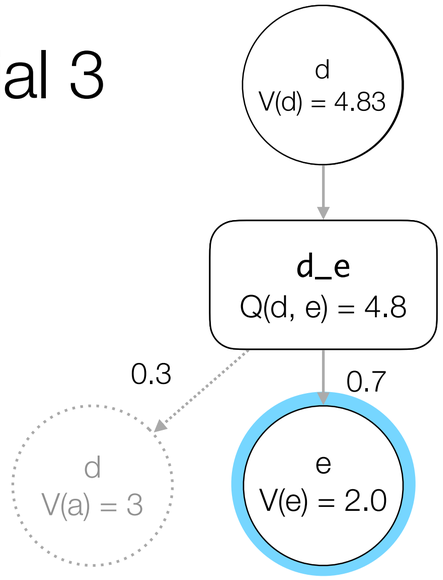
	Init	Q?	
V(a)	3.00	3.00	3.00
Q(a, b)			
V(b)	1.00	1.00	1.00
Q(b, c)			
Q(b, e)			
V(c)	0.00	0.00	0.00
Q(c, b)			
V(d)	**4.78**	4.83	4.83
Q(d, a)		5.18	
Q(d, e)		**4.83**	
V(e)	2.00	2.00	2.00
Q(e, b)			
Q(e, f)			
V(f)	2.00	2.00	2.00
Q(f, c)			
Q(f, e)			

Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

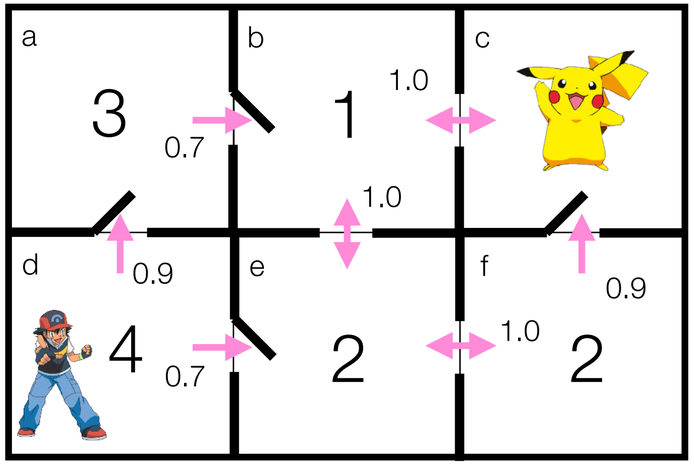
Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

Trial 3



Labelled RTDP

heuristic



	Exact	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6
V(a)	3.858	3.000	3.000	3.000	3.000	3.000	3.000
V(b)	1.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
V(d)	4.859	4.600	4.780	4.834	4.850	4.855	4.857
V(e)	2.000	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**
V(f)	2.222	2.000	2.000	2.000	2.000	2.000	2.000

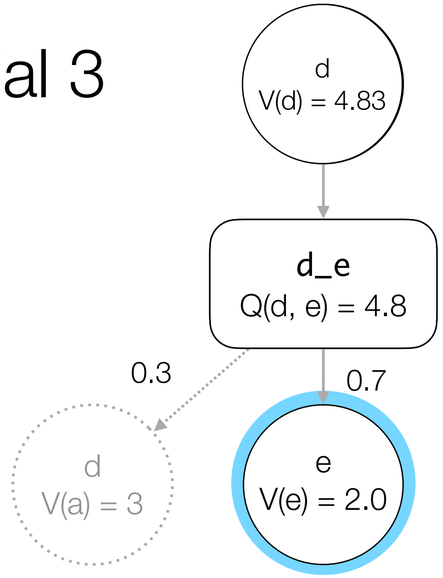
	Init	Q?	
V(a)	3.00	3.00	3.00
Q(a, b)			
V(b)	1.00	1.00	1.00
Q(b, c)			
Q(b, e)			
V(c)	0.00	0.00	0.00
Q(c, b)			
V(d)	**4.78**	4.83	4.83
Q(d, a)		5.18	
Q(d, e)		**4.83**	
V(e)	2.00	2.00	2.00
Q(e, b)			
Q(e, f)			
V(f)	2.00	2.00	2.00
Q(f, c)			
Q(f, e)			

Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

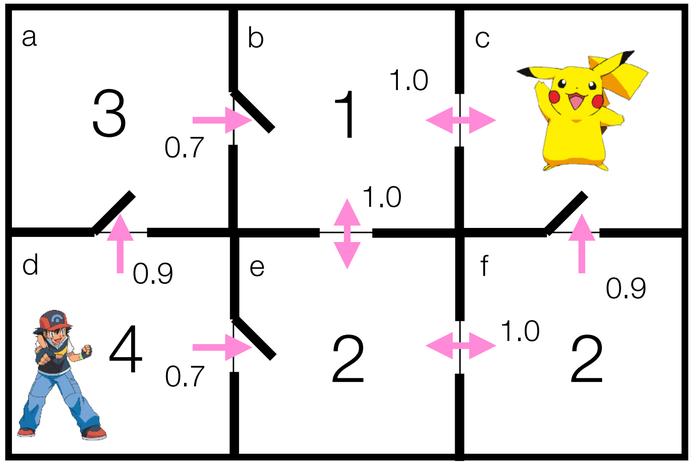
Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

Trial 3



Labelled RTDP

heuristic



	Exact	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7
V(a)	3.858	3.000	3.000	3.000	3.000	3.000	3.000	3.000
V(b)	1.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
V(d)	4.859	4.600	4.780	4.834	4.850	4.855	4.857	**4.857**
V(e)	2.000	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**
V(f)	2.222	2.000	2.000	2.000	2.000	2.000	2.000	2.000

	Init	Q?	
V(a)	3.00	3.00	3.00
Q(a, b)			
V(b)	1.00	1.00	1.00
Q(b, c)			
Q(b, e)			
V(c)	0.00	0.00	0.00
Q(c, b)			
V(d)	**4.78**	4.83	4.83
Q(d, a)		5.18	
Q(d, e)		**4.83**	
V(e)	2.00	2.00	2.00
Q(e, b)			
Q(e, f)			
V(f)	2.00	2.00	2.00
Q(f, c)			
Q(f, e)			

Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3  |   V_l ← h
4  |   while there is time left do
5  |   |   TRIAL(s_0)
6  |   end
7  |   return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13 |   s ← s_0
14 |   while s ∉ G do
15 |   |   a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16 |   |   V_l(s) ← Q^{V_l}(s, a_best)
17 |   |   s ← execute action a_best in s
18 |   end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

Labelled

RTDP can be seen as a version of VI that only:

	Exact	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7
V(a)	3.858	3.000	3.000	3.000	3.000	3.000	3.000	3.000
V(b)	1.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
V(d)	4.859	4.600	4.780	4.834	4.850	4.855	4.857	**4.857**
V(e)	2.000	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**
V(f)	2.222	2.000	2.000	2.000	2.000	2.000	2.000	2.000

Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

Labelled

RTDP can be seen as a version of VI that only:

…expands states likely to be in the optimal policy

	Exact	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7
V(a)	3.858	3.000	3.000	3.000	3.000	3.000	3.000	3.000
V(b)	1.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
V(d)	4.859	4.600	4.780	4.834	4.850	4.855	4.857	**4.857**
V(e)	2.000	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**
V(f)	2.222	2.000	2.000	2.000	2.000	2.000	2.000	2.000

Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

Labelled

RTDP can be seen as a version of VI that only:

…expands states likely to be in the optimal policy

…performs back-ups on those states where necessary

	Exact	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7
V(a)	3.858	3.000	3.000	3.000	3.000	3.000	3.000	3.000
V(b)	1.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
V(d)	4.859	4.600	4.780	4.834	4.850	4.855	4.857	**4.857**
V(e)	2.000	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**
V(f)	2.222	2.000	2.000	2.000	2.000	2.000	2.000	2.000

Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

Labelled

RTDP can be seen as a version of VI that only:

…expands states likely to be in the optimal policy

…performs back-ups on those states where necessary

RDTP converges to a **partial** optimal policy in the limit

	Exact	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7
V(a)	3.858	3.000	3.000	3.000	3.000	3.000	3.000	3.000
V(b)	1.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
V(d)	4.859	4.600	4.780	4.834	4.850	4.855	4.857	**4.857**
V(e)	2.000	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**
V(f)	2.222	2.000	2.000	2.000	2.000	2.000	2.000	2.000

Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

Labelled

RTDP can be seen as a version of VI that only:

…expands states likely to be in the optimal policy

…performs back-ups on those states where necessary

RDTP converges to a **partial** optimal policy in the limit

… but can be stopped early if the agent needs to act

	Exact	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7
V(a)	3.858	3.000	3.000	3.000	3.000	3.000	3.000	3.000
V(b)	1.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
V(d)	4.859	4.600	4.780	4.834	4.850	4.855	4.857	**4.857**
V(e)	2.000	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**
V(f)	2.222	2.000	2.000	2.000	2.000	2.000	2.000	2.000

Algorithm 4.3: RTDP

```
1  RTDP(s_0)
2  begin
3      V_l ← h
4      while there is time left do
5          TRIAL(s_0)
6      end
7      return π*_{s_0}
8  end
9
10
11 TRIAL(s_0)
12 begin
13     s ← s_0
14     while s ∉ G do
15         a_best ← argmin_{a∈A} Q^{V_l}(s, a)
16         V_l(s) ← Q^{V_l}(s, a_best)
17         s ← execute action a_best in s
18     end
19 end
```

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

Labelled

RTDP can be seen as a version of VI that only:

…expands states likely to be in the optimal policy

…performs back-ups on those states where necessary

RDTP converges to a **partial** optimal policy in the limit

… but can be stopped early if the agent needs to act

… although the policy may be arbitrarily bad!

	Exact	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7
V(a)	3.858	3.000	3.000	3.000	3.000	3.000	3.000	3.000
V(b)	1.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
V(c)	0.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
V(d)	4.859	4.600	4.780	4.834	4.850	4.855	4.857	**4.857**
V(e)	2.000	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**	**2.000**
V(f)	2.222	2.000	2.000	2.000	2.000	2.000	2.000	2.000

UCT (Upper Confidence Bounds applied to Trees)

In cases where it is expensive or difficult to **enumerate** states, or we don't have an **explicit transition function,** but can **simulate** the transition between states, we can use **a Monte Carlo (i.e sampling-based)** approach to approximate the value function.

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

UCT (Upper Confidence Bounds applied to Trees)

In cases where it is expensive or difficult to **enumerate** states, or we don't have an **explicit transition function,** but can **simulate** the transition between states, we can use **a Monte Carlo (i.e sampling-based)** approach to approximate the value function.

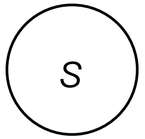
The Q function is **approximated** through random rollouts through the model.

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

UCT (Upper Confidence Bounds applied to Trees)

In cases where it is expensive or difficult to **enumerate** states, or we don't have an **explicit transition function,** but can **simulate** the transition between states, we can use **a Monte Carlo (i.e sampling-based)** approach to approximate the value function.

The Q function is **approximated** through random rollouts through the model.

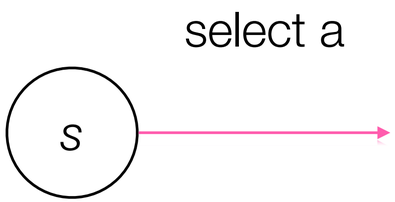


Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

UCT (Upper Confidence Bounds applied to Trees)

In cases where it is expensive or difficult to **enumerate** states, or we don't have an **explicit transition function,** but can **simulate** the transition between states, we can use **a Monte Carlo (i.e sampling-based)** approach to approximate the value function.

The Q function is **approximated** through random rollouts through the model.

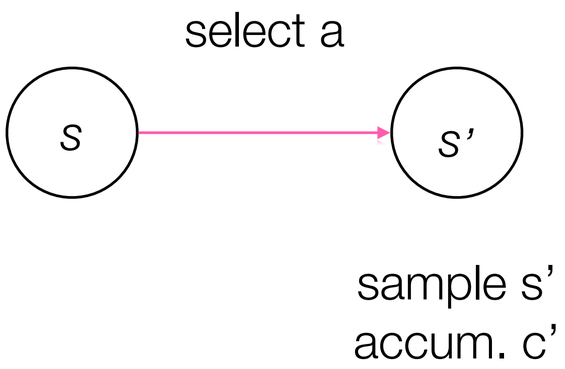


Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

UCT (Upper Confidence Bounds applied to Trees)

In cases where it is expensive or difficult to **enumerate** states, or we don't have an **explicit transition function,** but can **simulate** the transition between states, we can use **a Monte Carlo (i.e sampling-based)** approach to approximate the value function.

The Q function is **approximated** through random rollouts through the model.

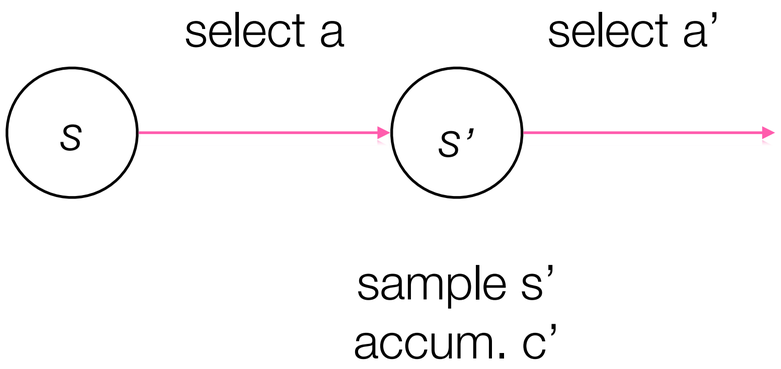


Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

UCT (Upper Confidence Bounds applied to Trees)

In cases where it is expensive or difficult to **enumerate** states, or we don't have an **explicit transition function,** but can **simulate** the transition between states, we can use **a Monte Carlo (i.e sampling-based)** approach to approximate the value function.

The Q function is **approximated** through random rollouts through the model.

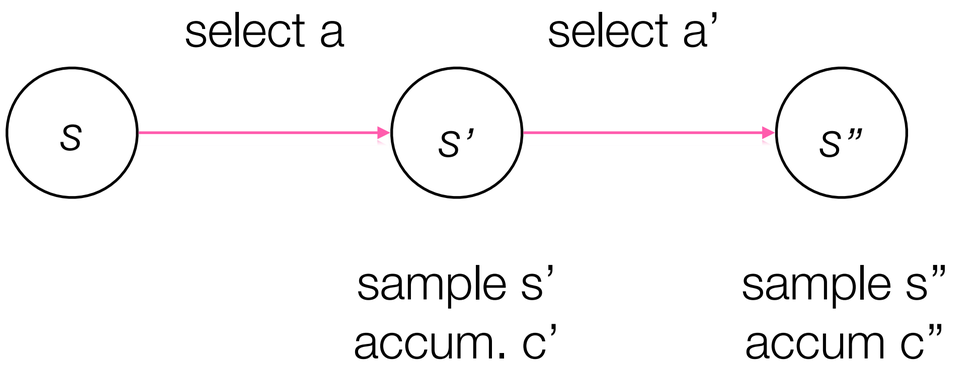


Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

UCT (Upper Confidence Bounds applied to Trees)

In cases where it is expensive or difficult to **enumerate** states, or we don't have an **explicit transition function,** but can **simulate** the transition between states, we can use **a Monte Carlo (i.e sampling-based)** approach to approximate the value function.

The Q function is **approximated** through random rollouts through the model.

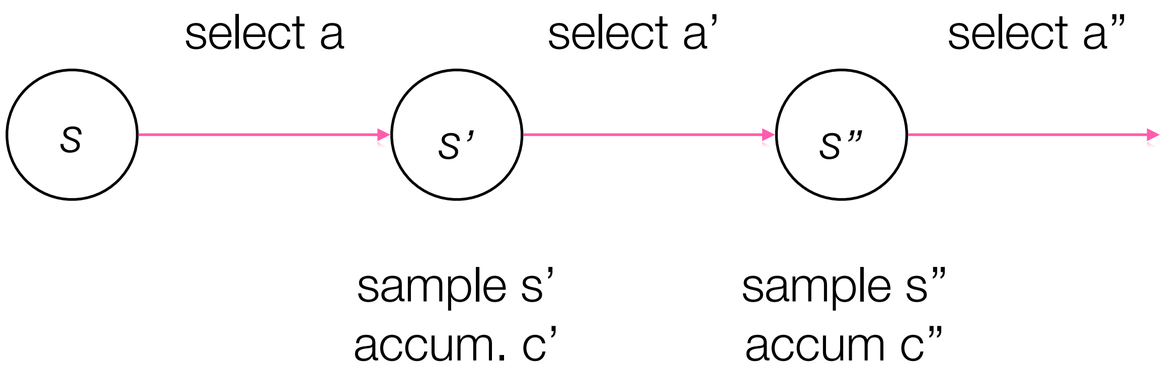


Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

UCT (Upper Confidence Bounds applied to Trees)

In cases where it is expensive or difficult to **enumerate** states, or we don't have an **explicit transition function,** but can **simulate** the transition between states, we can use **a Monte Carlo (i.e sampling-based)** approach to approximate the value function.

The Q function is **approximated** through random rollouts through the model.

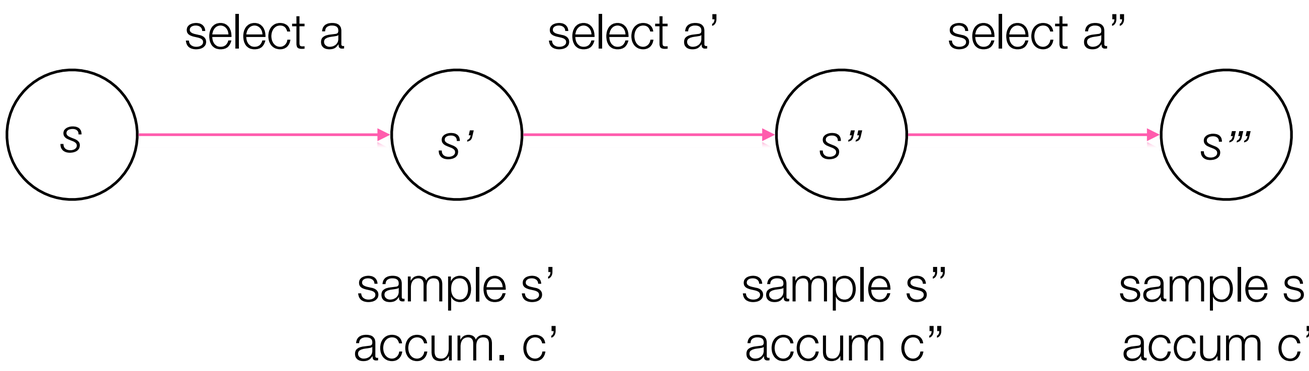


Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

UCT (Upper Confidence Bounds applied to Trees)

In cases where it is expensive or difficult to **enumerate** states, or we don't have an **explicit transition function,** but can **simulate** the transition between states, we can use **a Monte Carlo (i.e sampling-based)** approach to approximate the value function.

The Q function is **approximated** through random rollouts through the model.

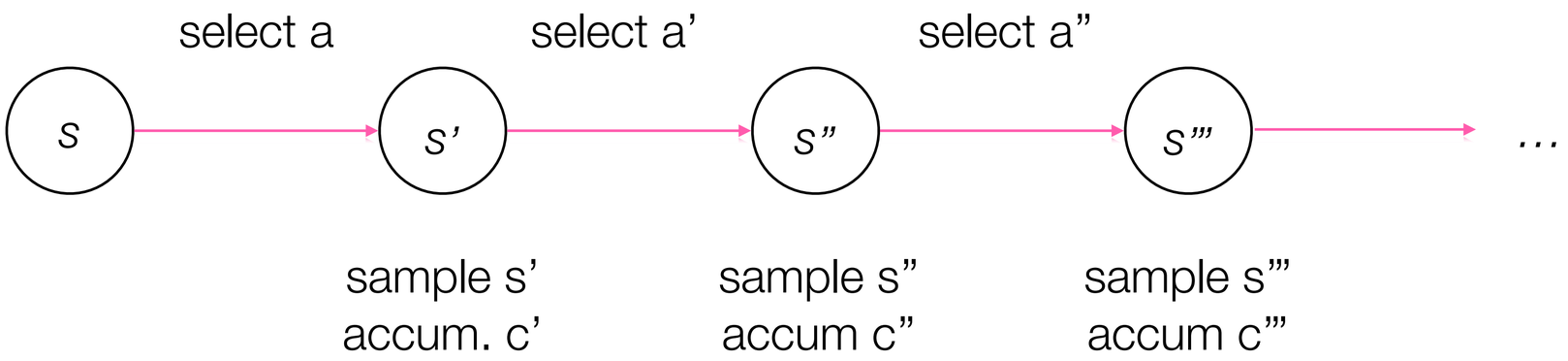


Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

UCT (Upper Confidence Bounds applied to Trees)

In cases where it is expensive or difficult to **enumerate** states, or we don't have an **explicit transition function,** but can **simulate** the transition between states, we can use **a Monte Carlo (i.e sampling-based)** approach to approximate the value function.

The Q function is **approximated** through random rollouts through the model.

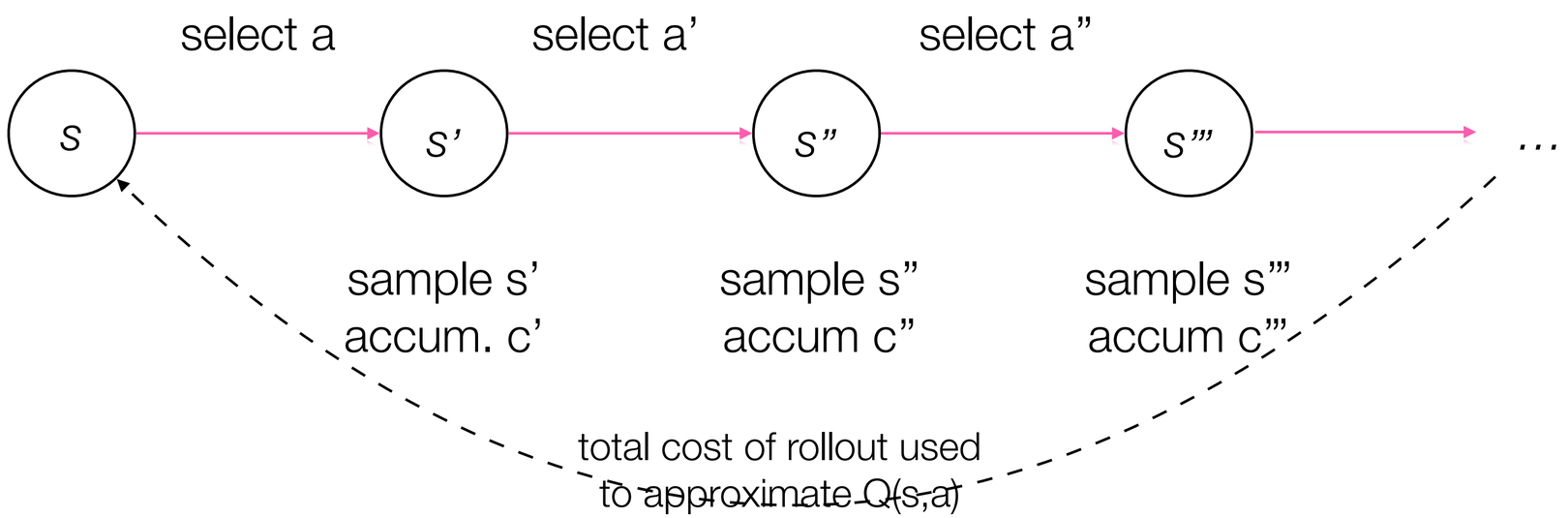


Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

UCT (Upper Confidence Bounds applied to Trees)

In cases where it is expensive or difficult to **enumerate** states, or we don't have an **explicit transition function,** but can **simulate** the transition between states, we can use **a Monte Carlo (i.e sampling-based)** approach to approximate the value function.

The Q function is **approximated** through random rollouts through the model.

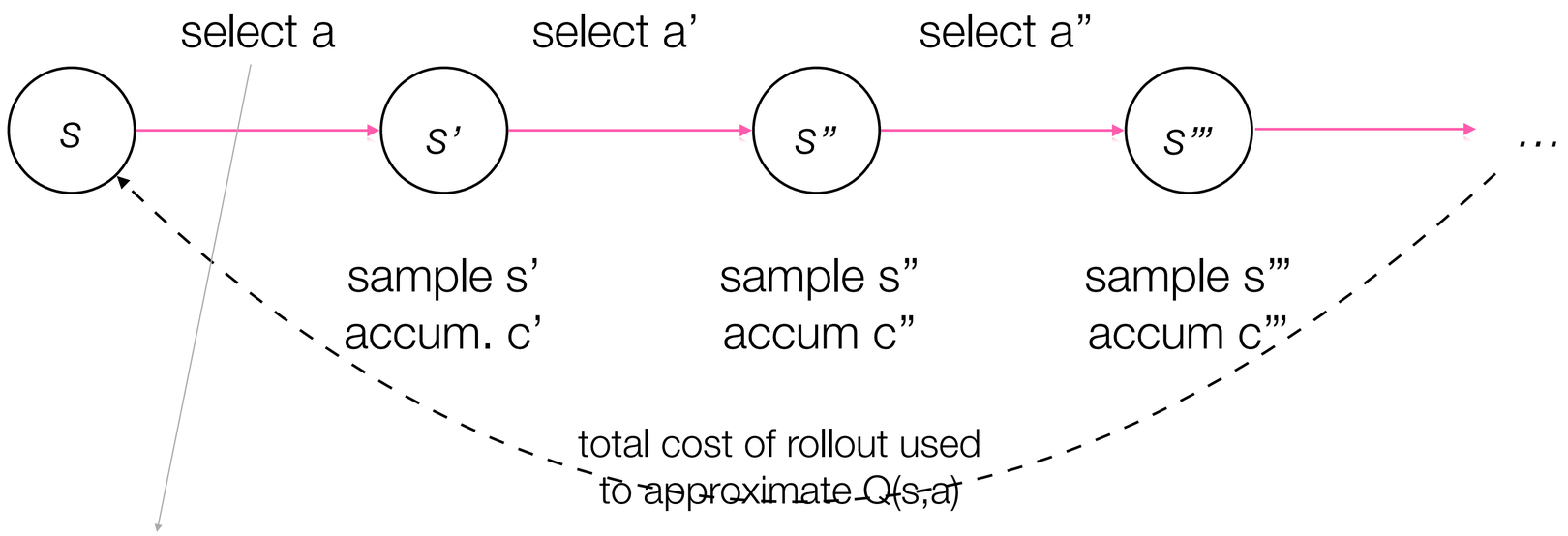


Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

UCT (Upper Confidence Bounds applied to Trees)

In cases where it is expensive or difficult to **enumerate** states, or we don't have an **explicit transition function,** but can **simulate** the transition between states, we can use **a Monte Carlo (i.e sampling-based)** approach to approximate the value function.

The Q function is **approximated** through random rollouts through the model.



$$a' = \underset{a \in \mathcal{A}}{\operatorname{argmin}} \left\{ \hat{Q}(s,a) - C\sqrt{\frac{\ln(n_s)}{n_{s,a}}} \right\}$$

n_s: visit count of state s

$n_{s,a}$: times a was selected in s

forces exploration by making under-selected actions look cheaper

Mausam and Kolobov. *Planning with Markov Decision Processes: An AI Perspective.* 2012.

MCTS

MCTS

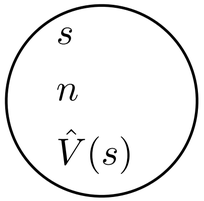
- In many cases, enumerating the state-space is not feasible
  - Approximation techniques are required

MCTS

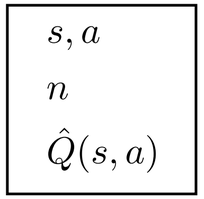
- In many cases, enumerating the state-space is not feasible
  - Approximation techniques are required
- MCTS is a trial-based tree search algorithm that has been extremely successful approximating solutions
  - See AlphaGo
  - Allows for online (interleaving planning and execution) or offline planning
  - PAC guarantees - “with probability 0.95 the solution from x trials is within 5% of optimal”

MCTS

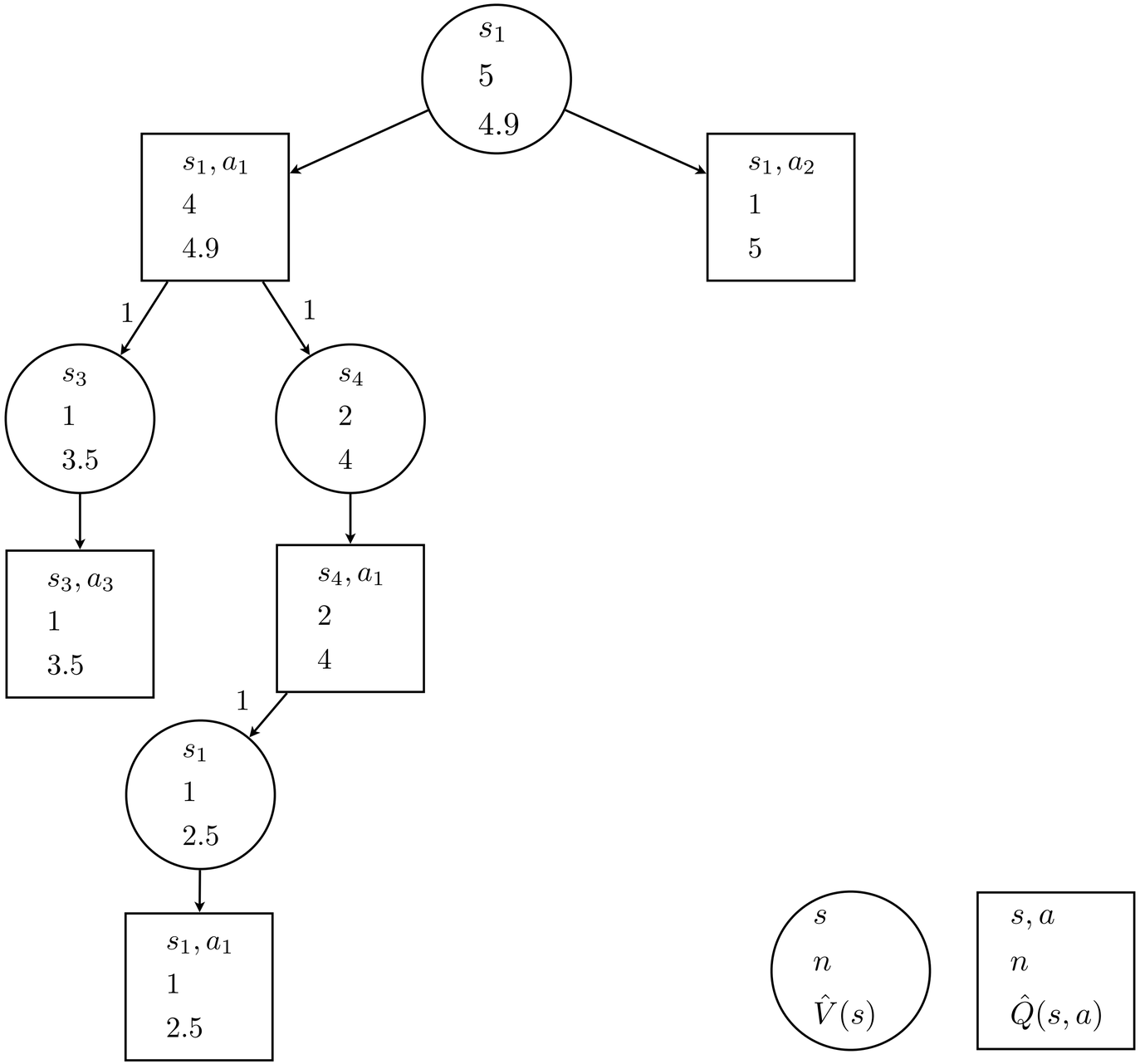
- Two types of search nodes
  - Decision nodes - correspond to states and are use to keep estimate of V(s)



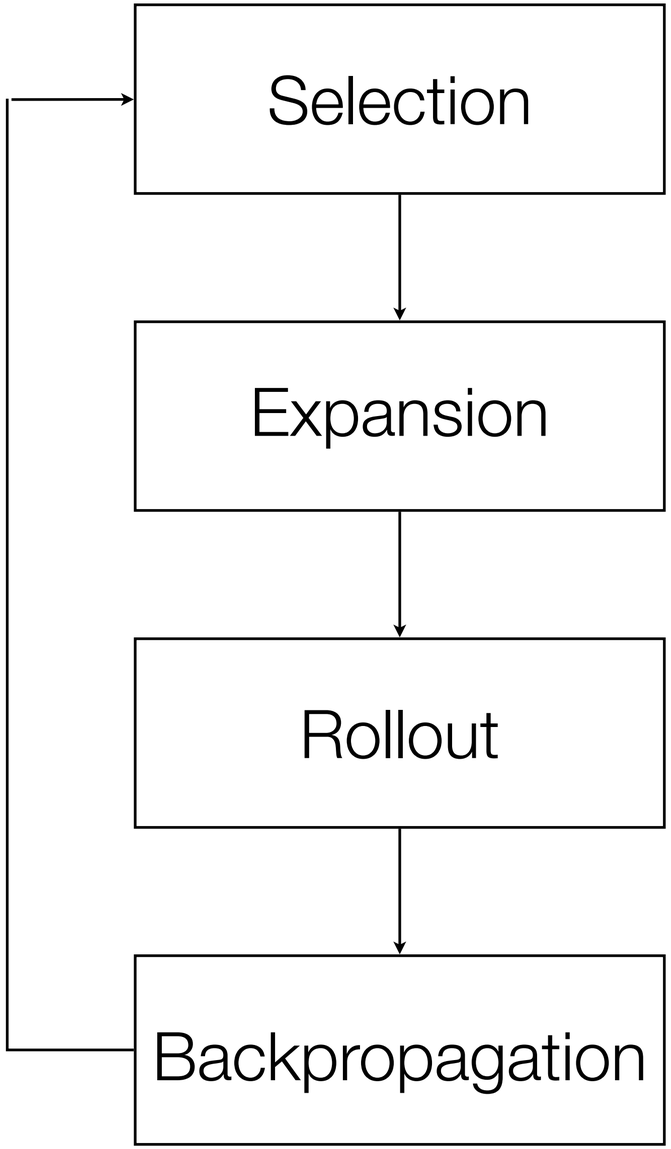
  - Chance nodes - correspond to state-action pairs and are used to keep estimate of Q(s,a)

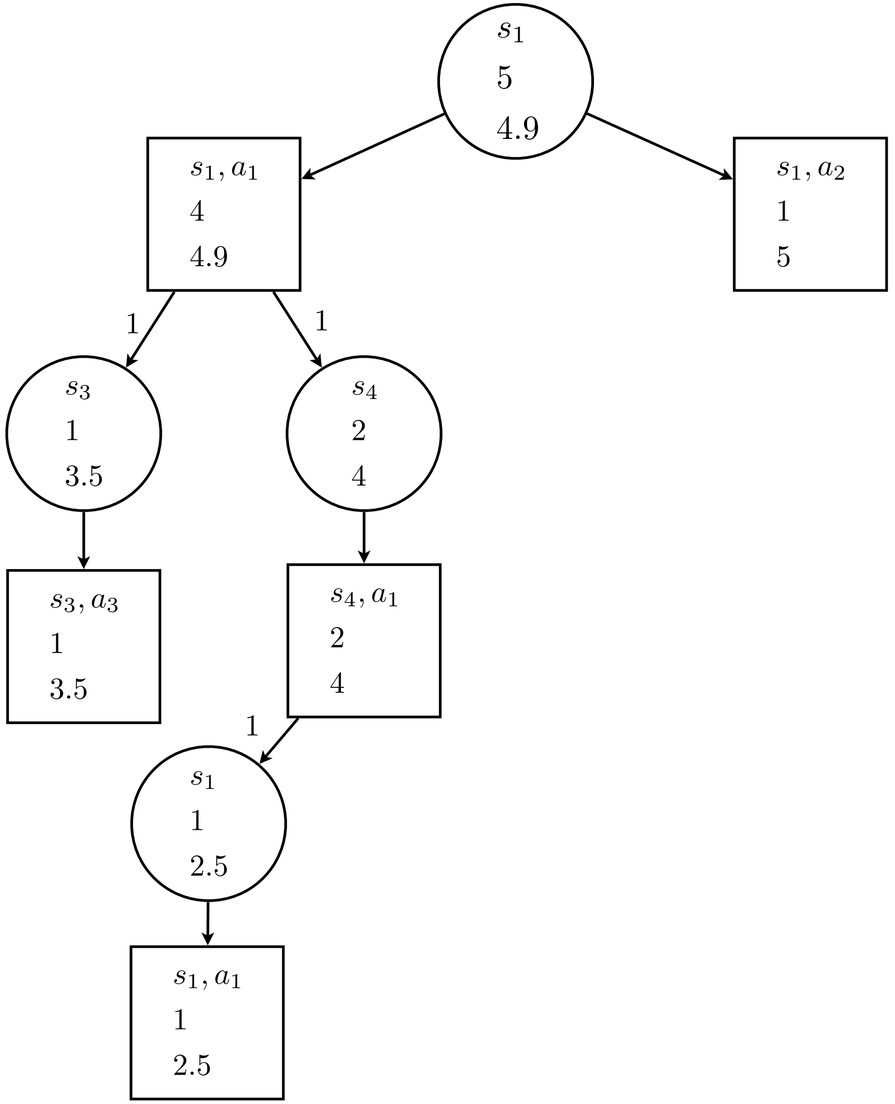


MCTS

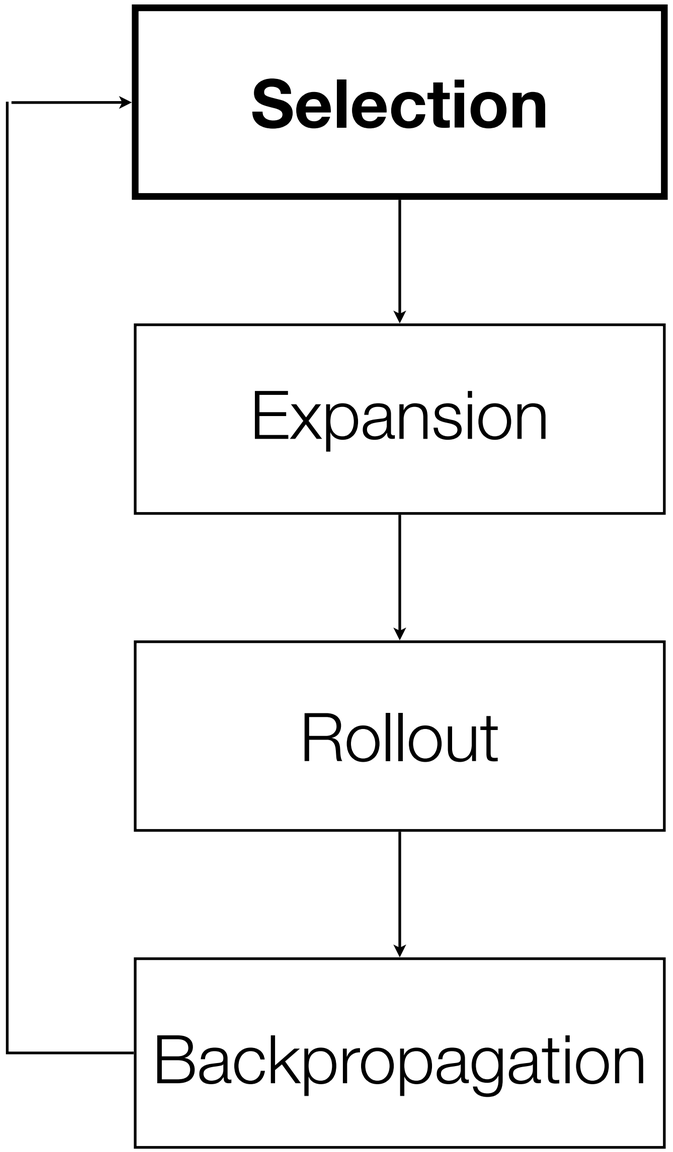

s_1
5
4.9
s_1, a_1
4
4.9
s_1, a_2
1
5
1
1
s_3
1
3.5
s_4
2
4
s_3, a_3
1
3.5
s_4, a_1
2
4
1
s_1
1
2.5
s_1, a_1
1
2.5
s
n
$\hat{V}(s)$
s, a
n
$\hat{Q}(s, a)$

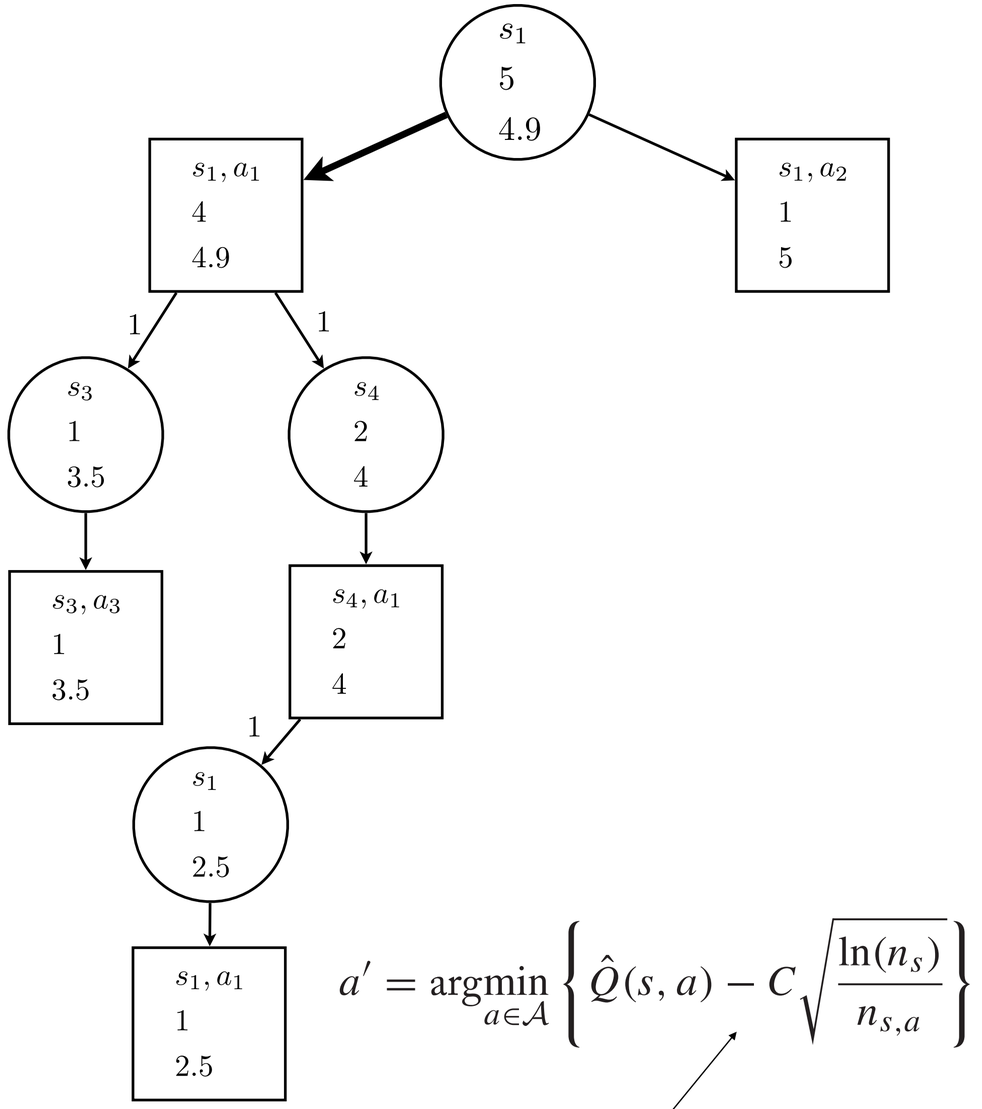
MCTS


Selection
Expansion
Rollout
Backpropagation


s_1
5
4.9
s_1, a_1
4
4.9
s_1, a_2
1
5
1
1
s_3
1
3.5
s_4
2
4
s_3, a_3
1
3.5
s_4, a_1
2
4
1
s_1
1
2.5
s_1, a_1
1
2.5

MCTS



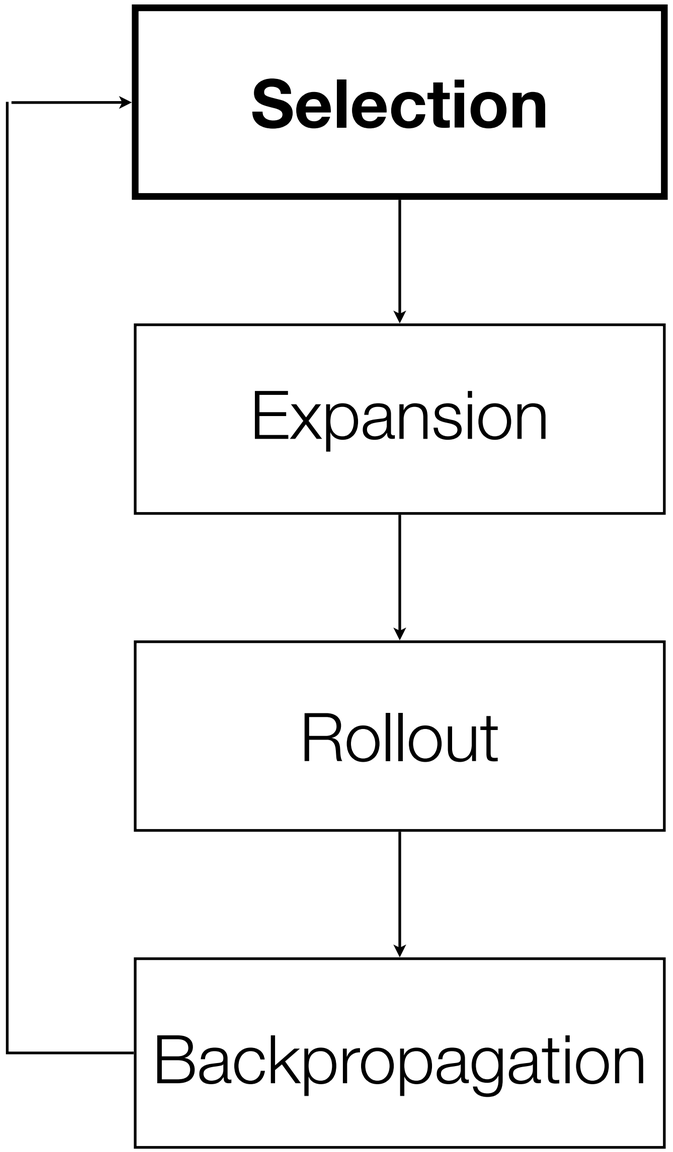


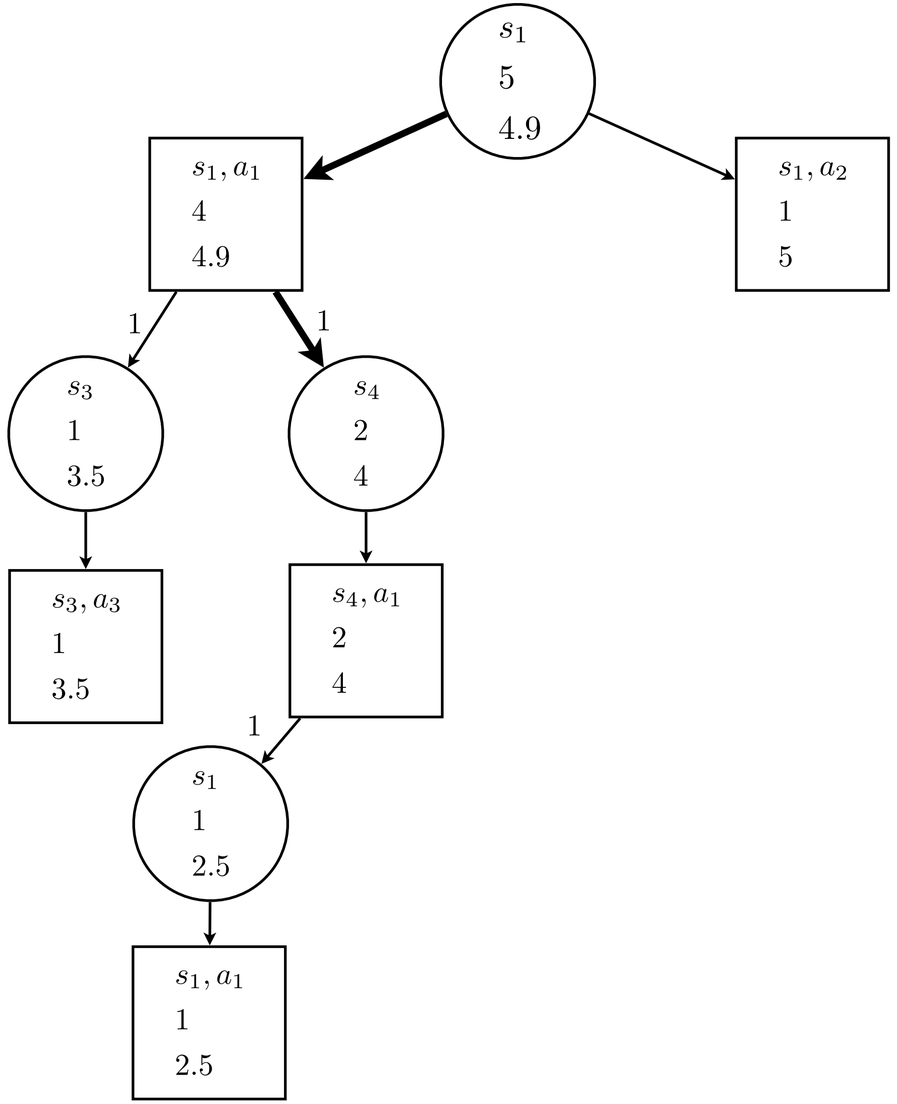
$$a' = \underset{a \in \mathcal{A}}{\operatorname{argmin}} \left\{ \hat{Q}(s, a) - C\sqrt{\frac{\ln(n_s)}{n_{s,a}}} \right\}$$

forces exploration by making under-selected actions look cheaper

Upper confidence bound applied to trees (UCT)

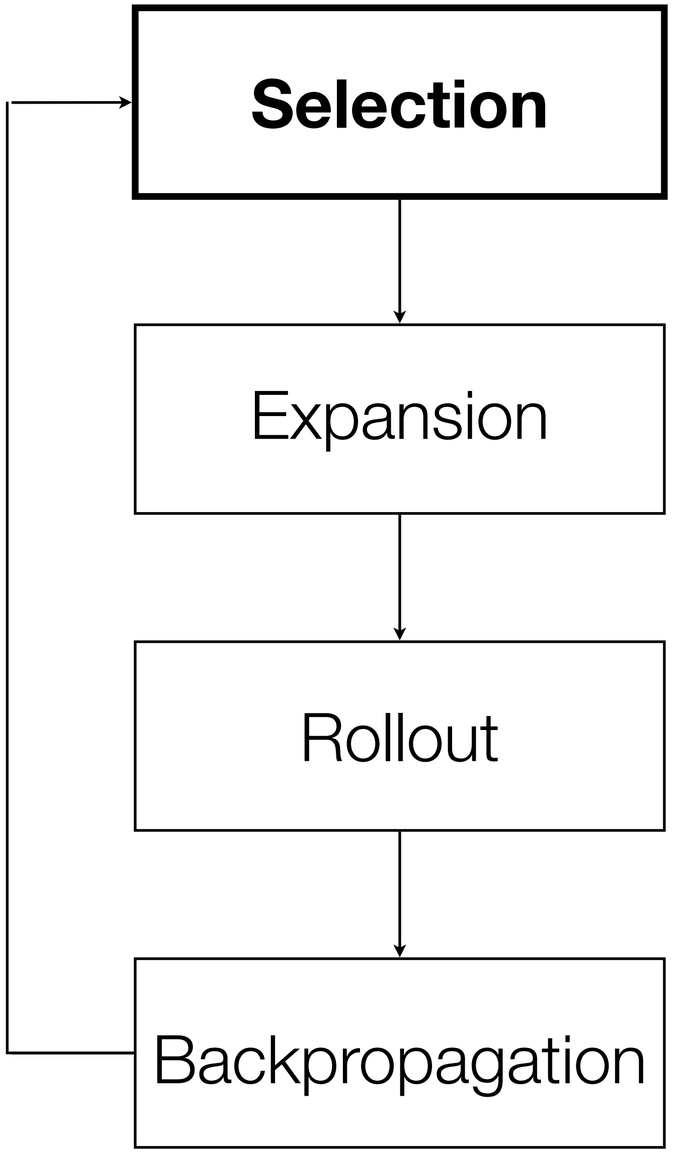
MCTS

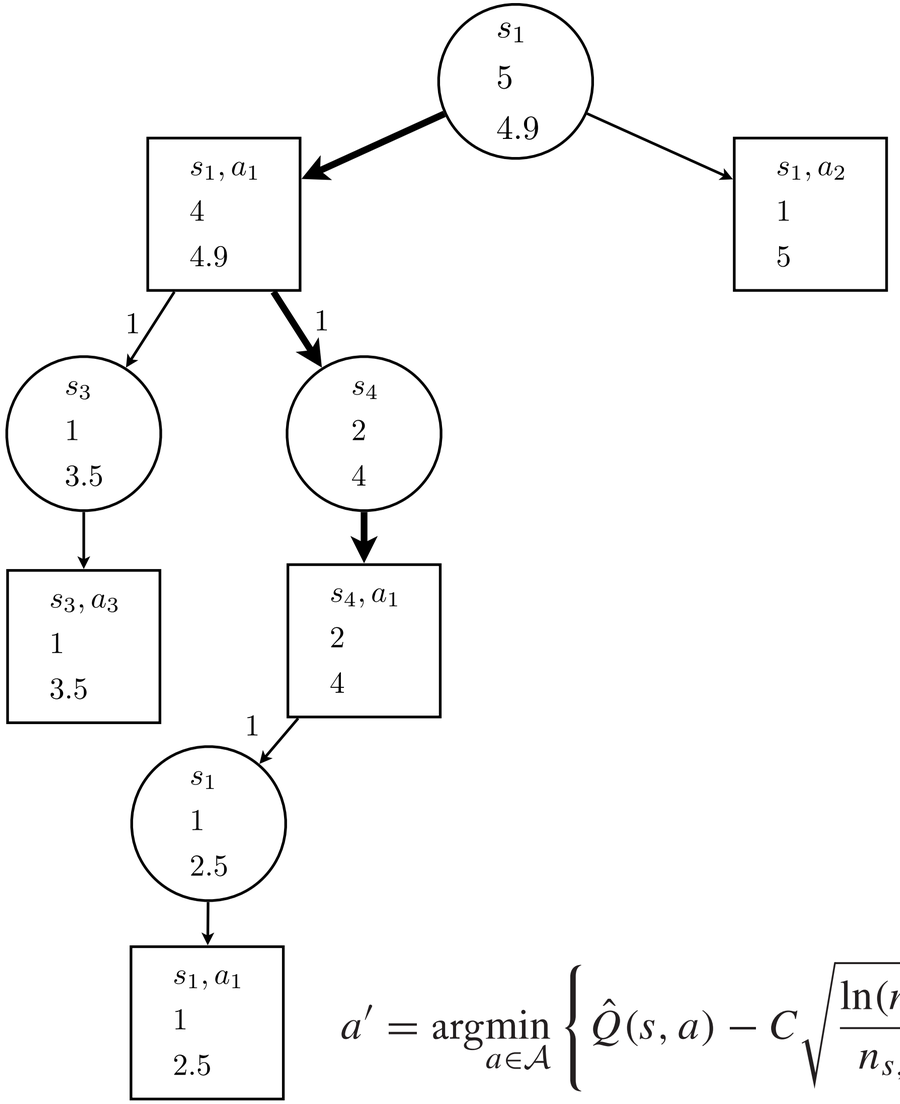




Sample successor (either according to transition function or a simulator)

MCTS



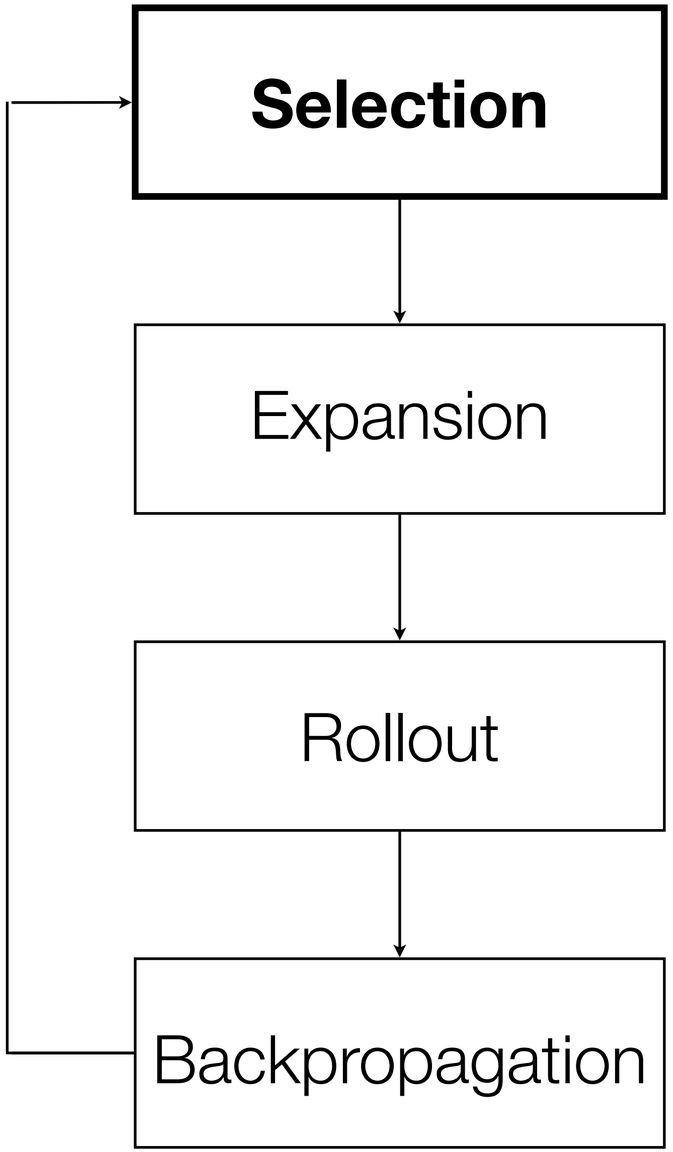


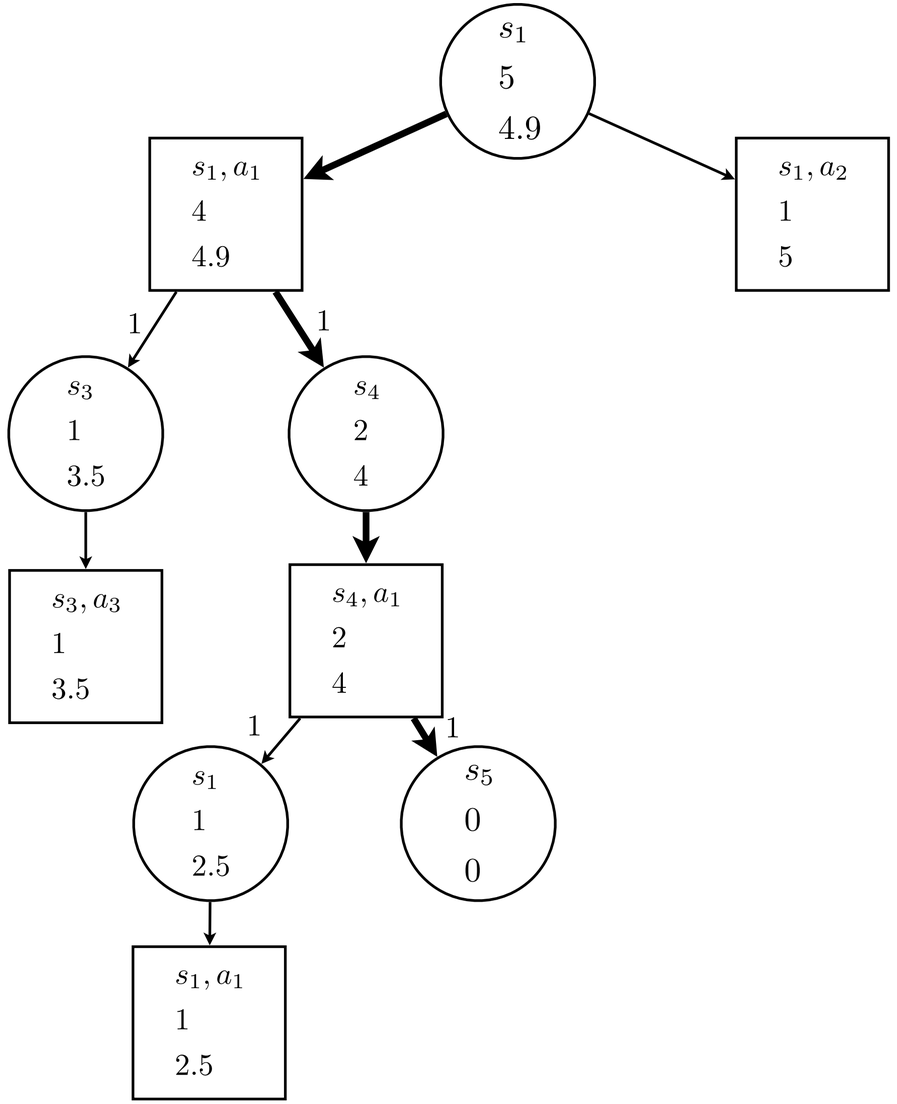
$$a' = \underset{a \in \mathcal{A}}{\operatorname{argmin}} \left\{ \hat{Q}(s,a) - C\sqrt{\frac{\ln(n_s)}{n_{s,a}}} \right\}$$

forces exploration by making under-selected actions look cheaper

Upper confidence bound applied to trees (UCT)

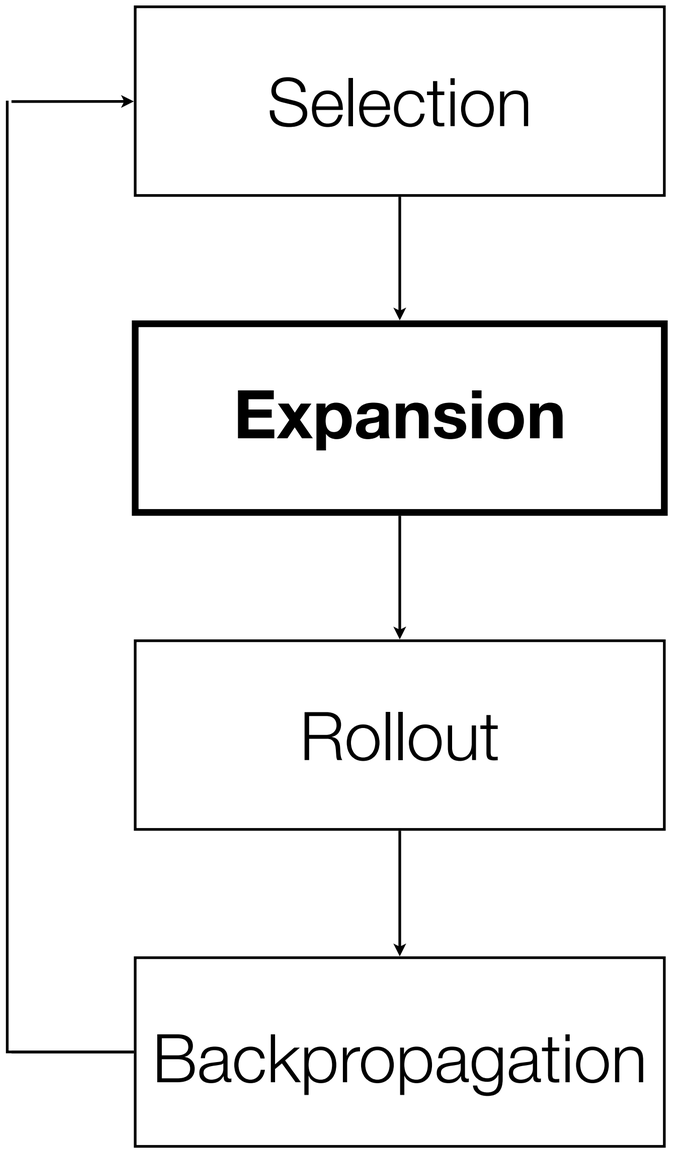
MCTS

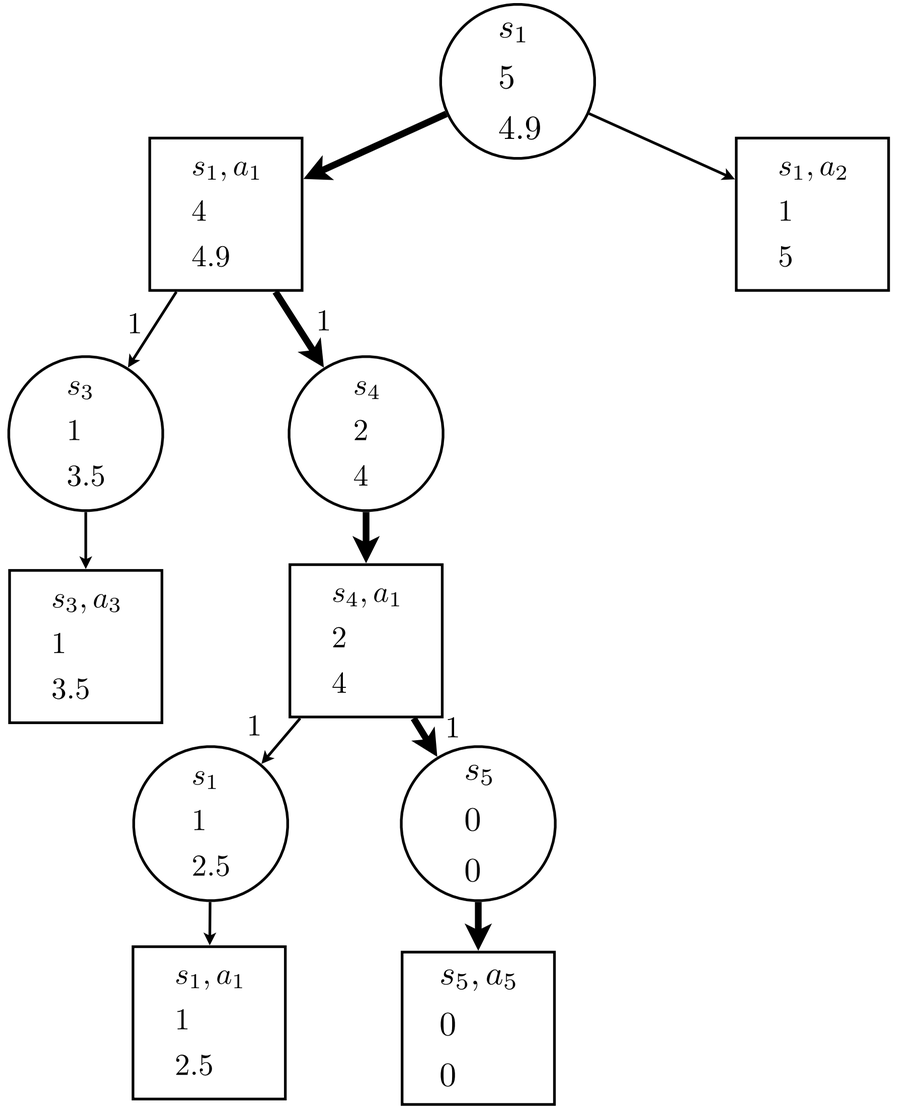




Sample successor (either according to transition function or a simulator)

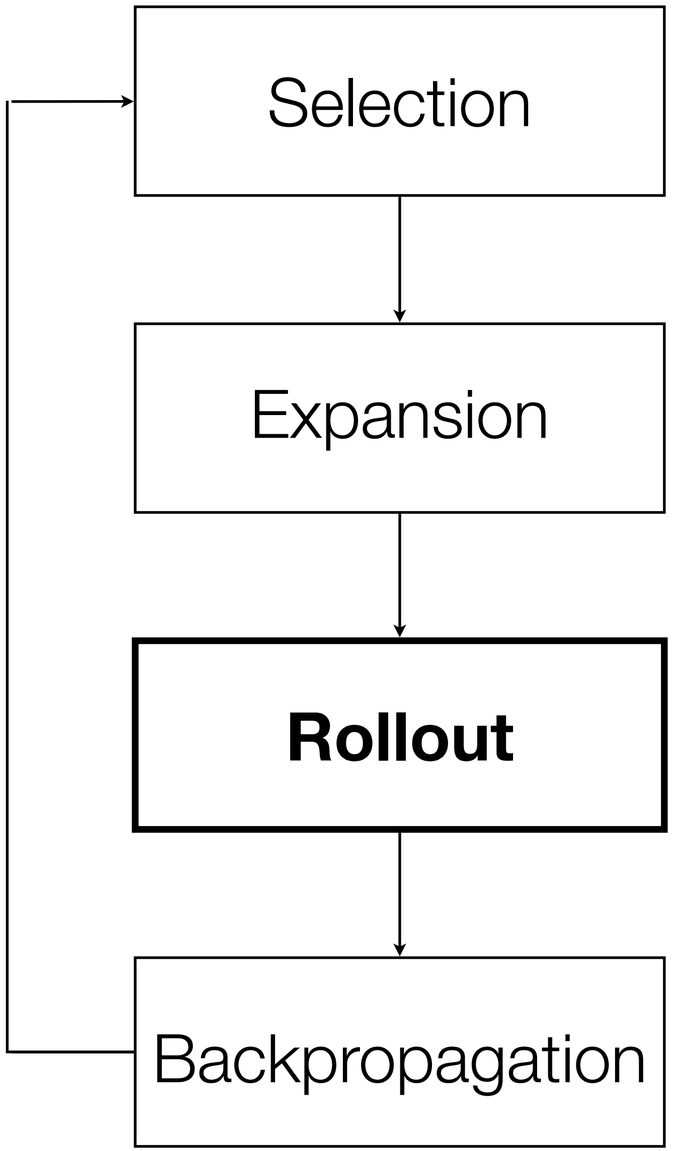
MCTS

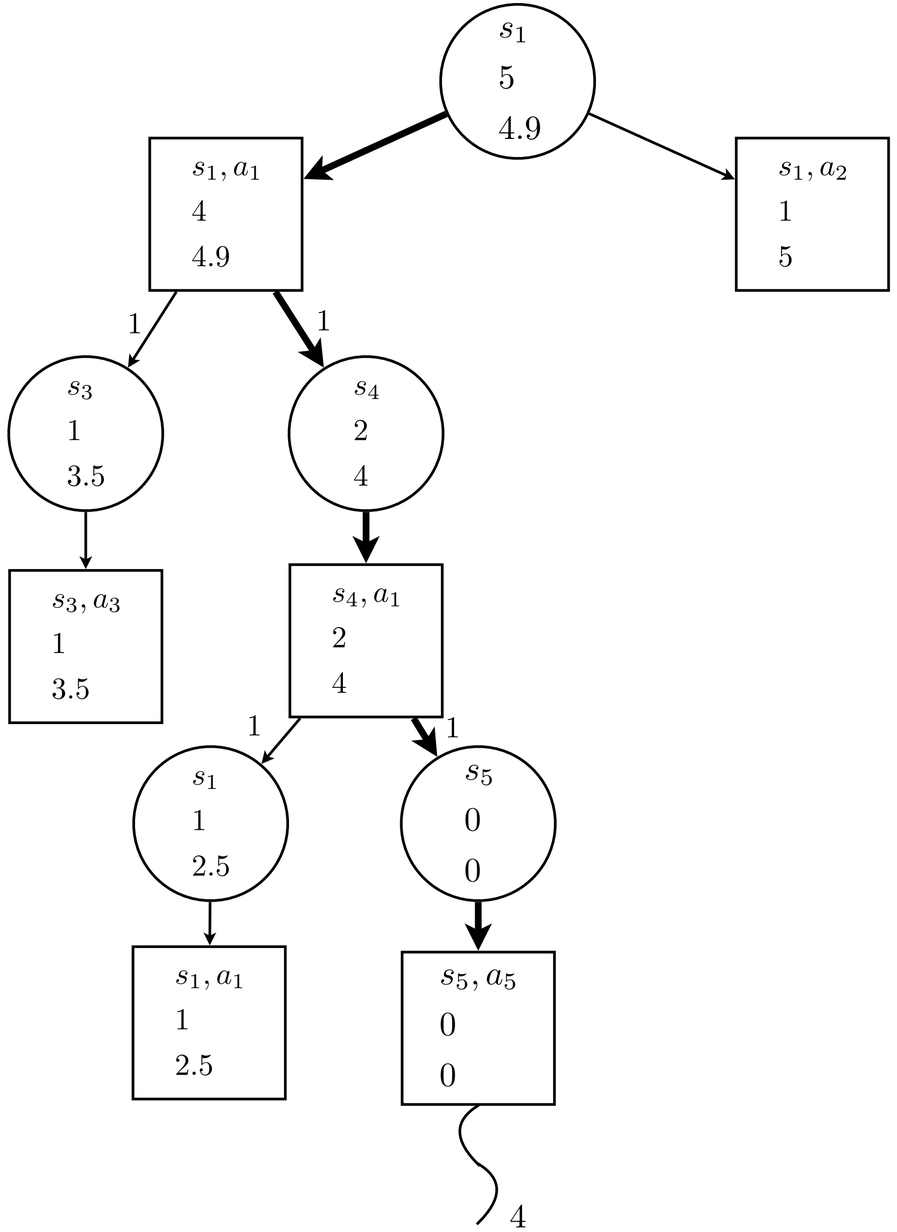




Choose random action

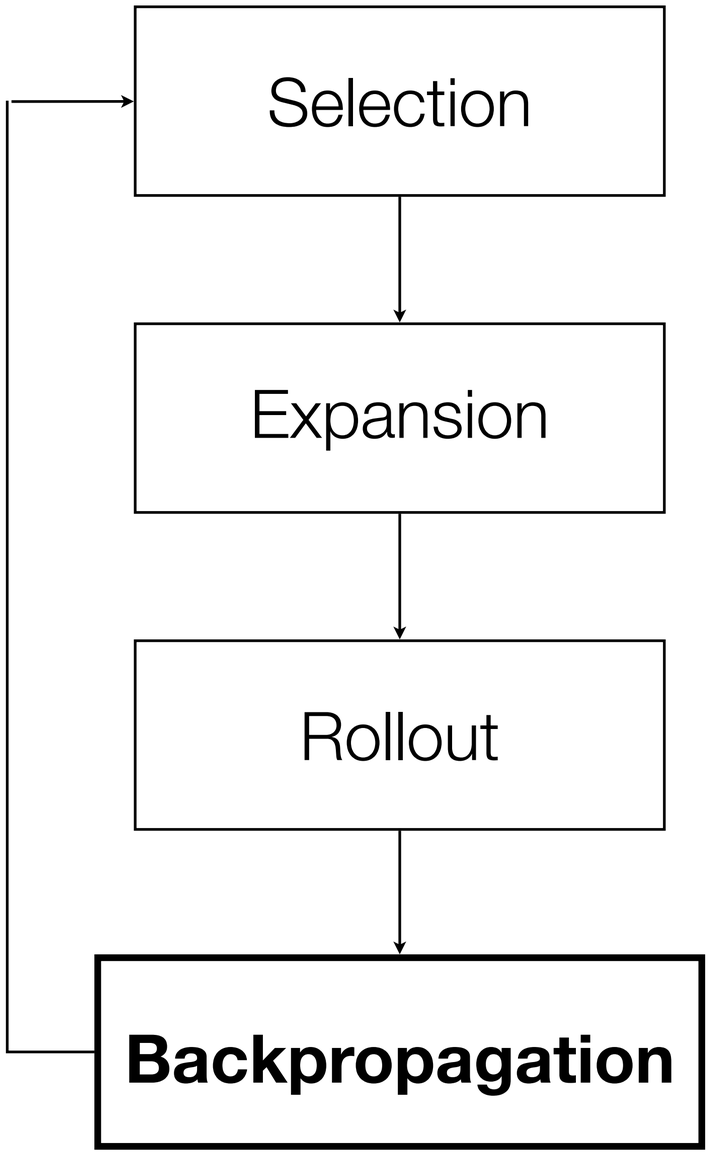
MCTS

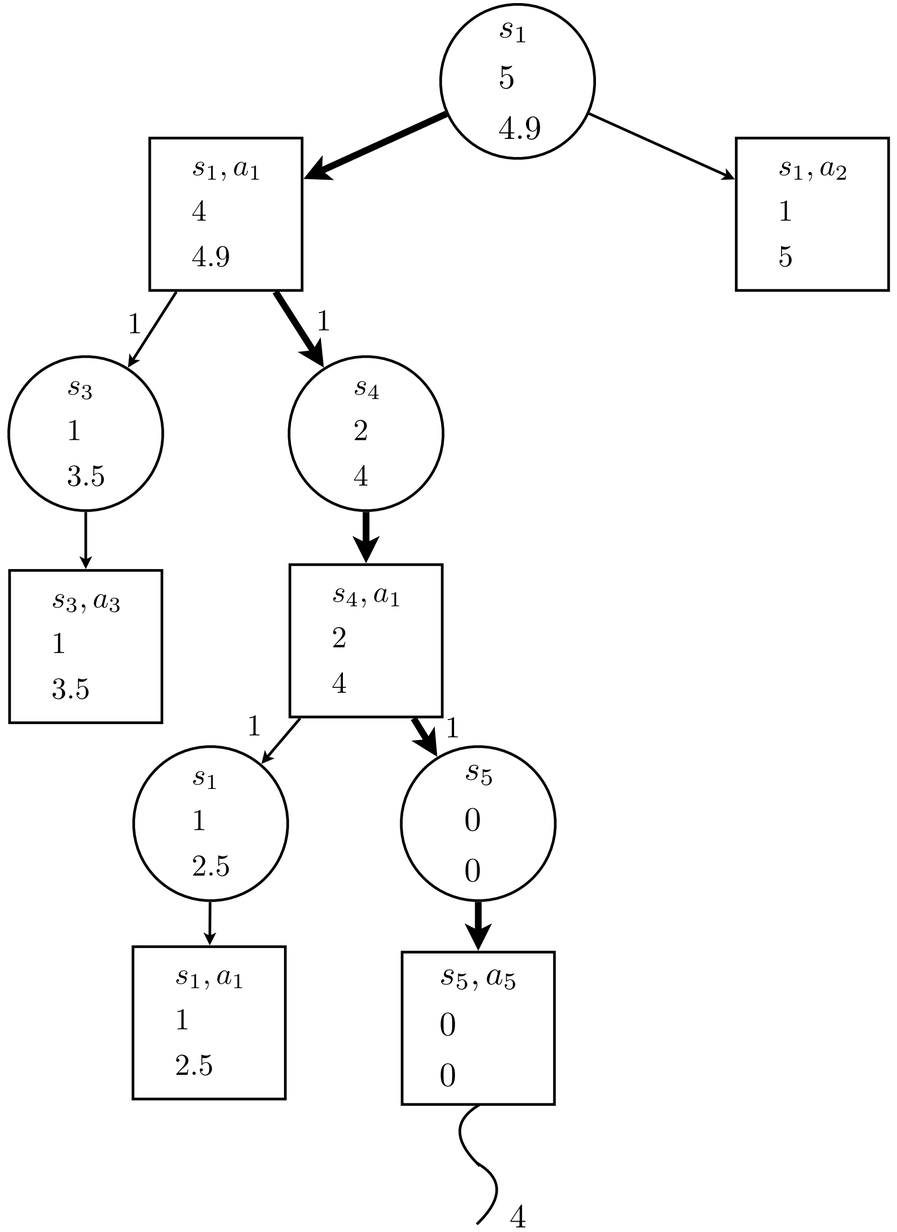




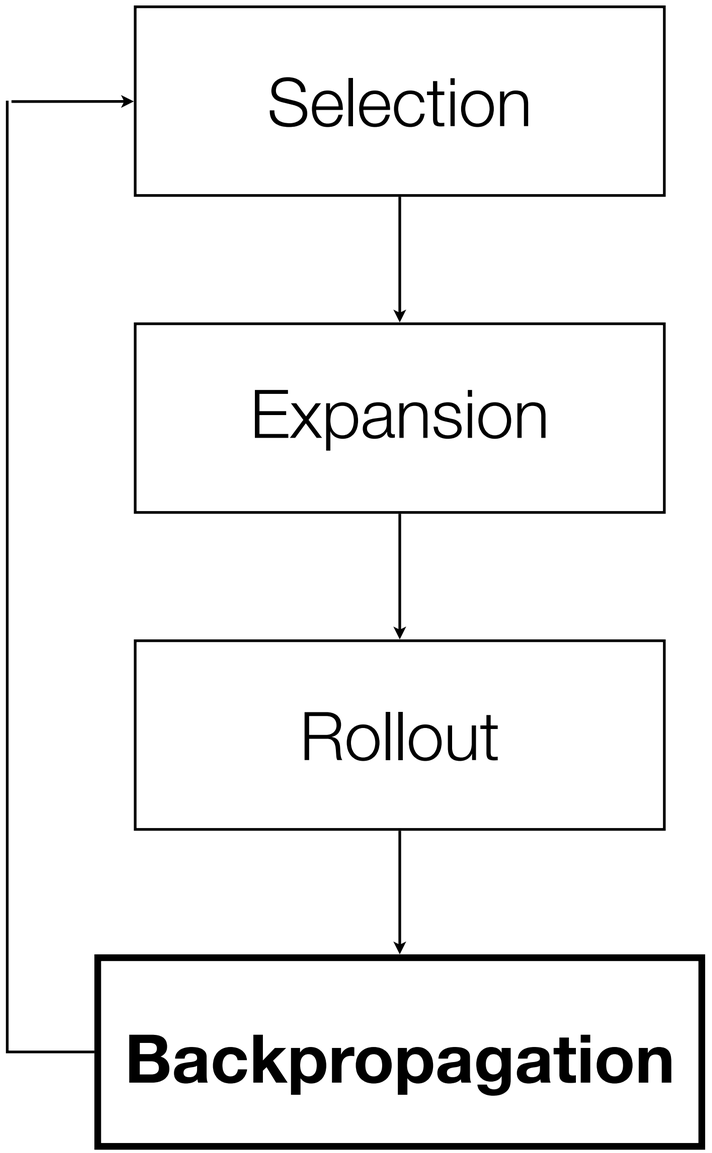
According to rollout policy - can be used to include domain knowledge

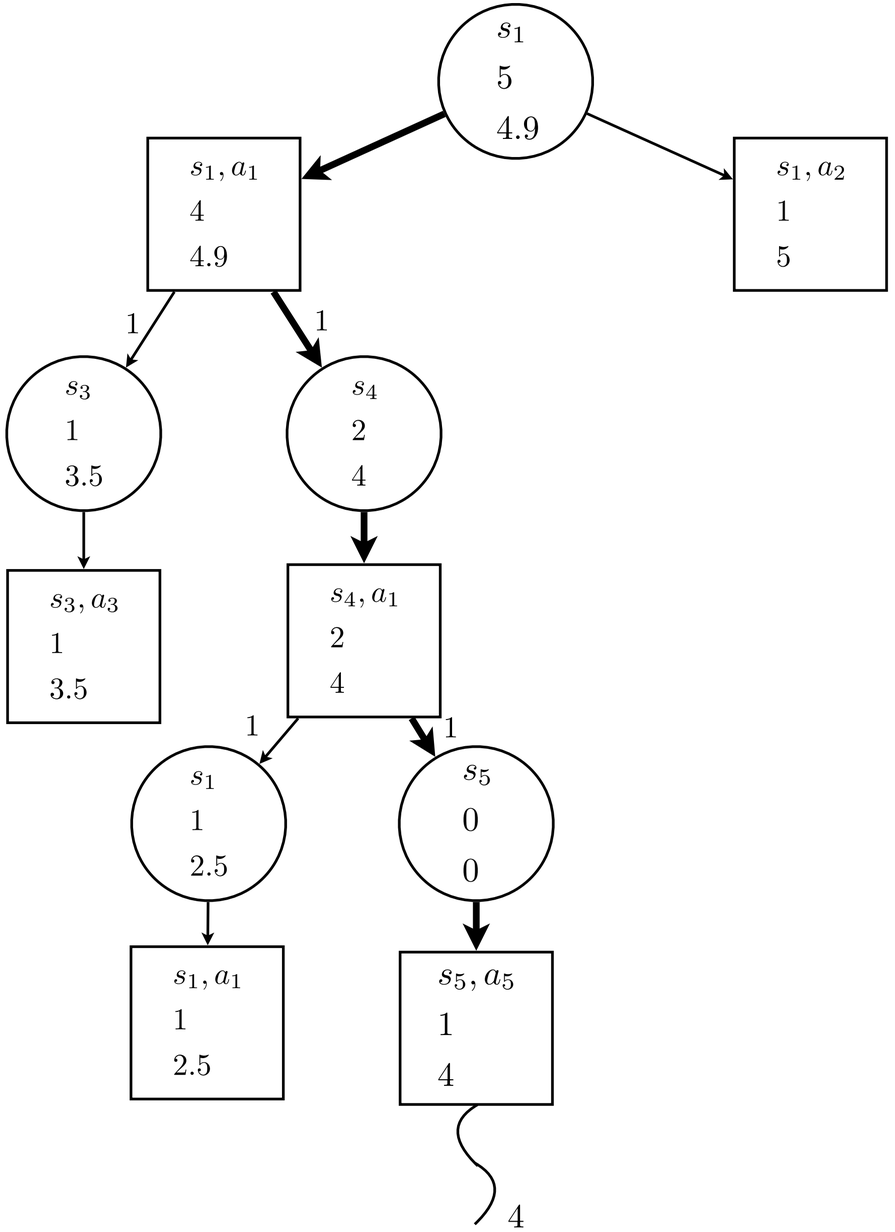
MCTS


Selection
Expansion
Rollout
Backpropagation

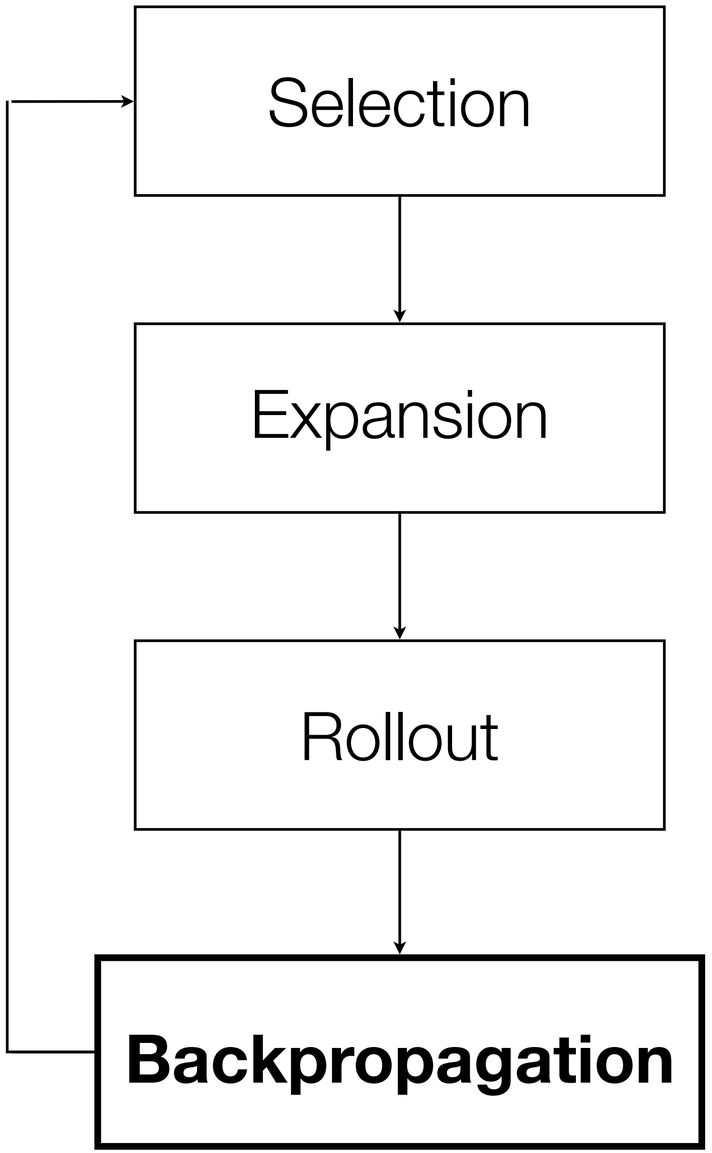

s_1
5
4.9
s_1, a_1
4
4.9
s_1, a_2
1
5
1
1
s_3
1
3.5
s_4
2
4
s_3, a_3
1
3.5
s_4, a_1
2
4
1
1
s_1
1
2.5
s_5
0
0
s_1, a_1
1
2.5
s_5, a_5
0
0
4

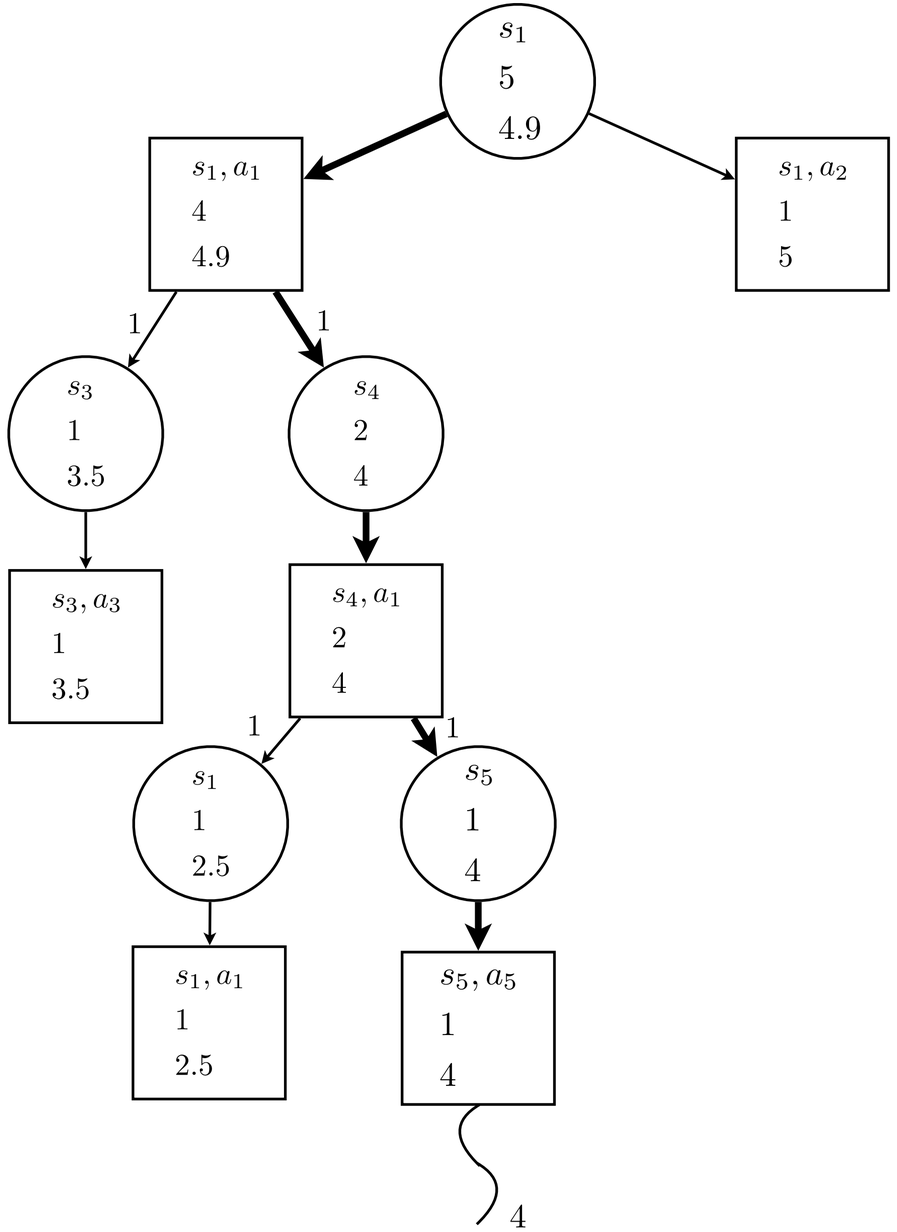
MCTS


Selection
Expansion
Rollout
Backpropagation


s_1
5
4.9
s_1, a_1
4
4.9
s_1, a_2
1
5
1
1
s_3
1
3.5
s_4
2
4
s_3, a_3
1
3.5
s_4, a_1
2
4
1
1
s_1
1
2.5
s_5
0
0
s_1, a_1
1
2.5
s_5, a_5
1
4
4

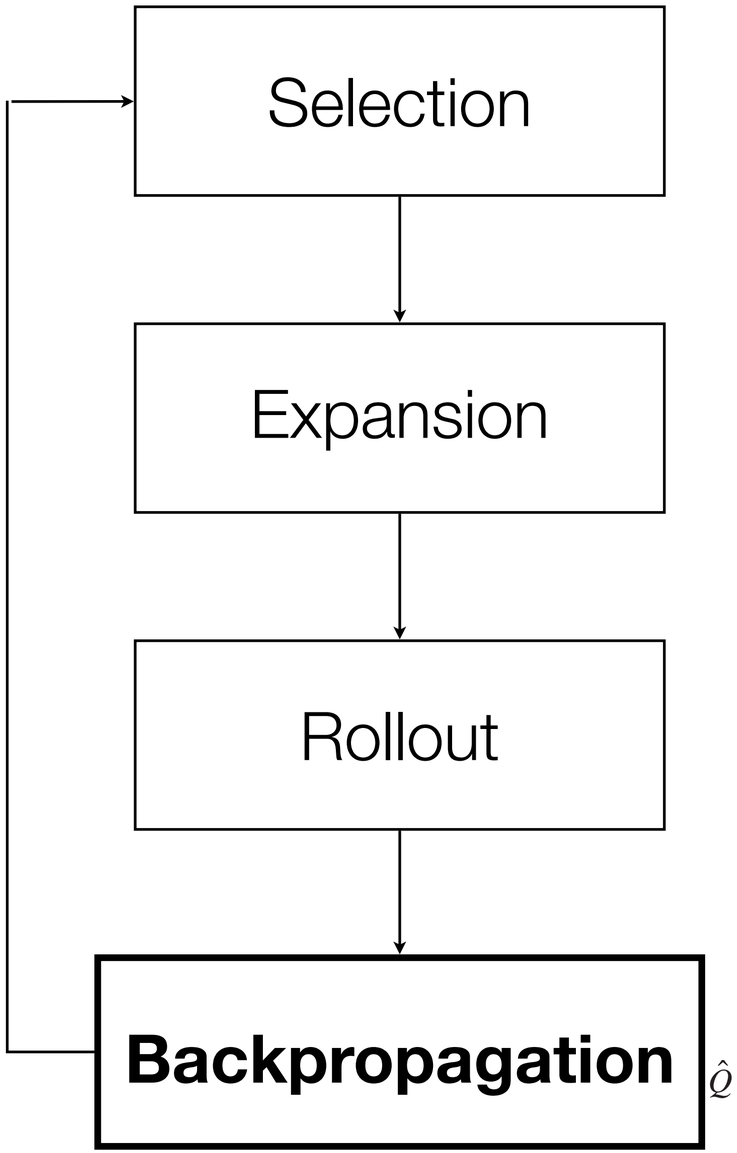
MCTS

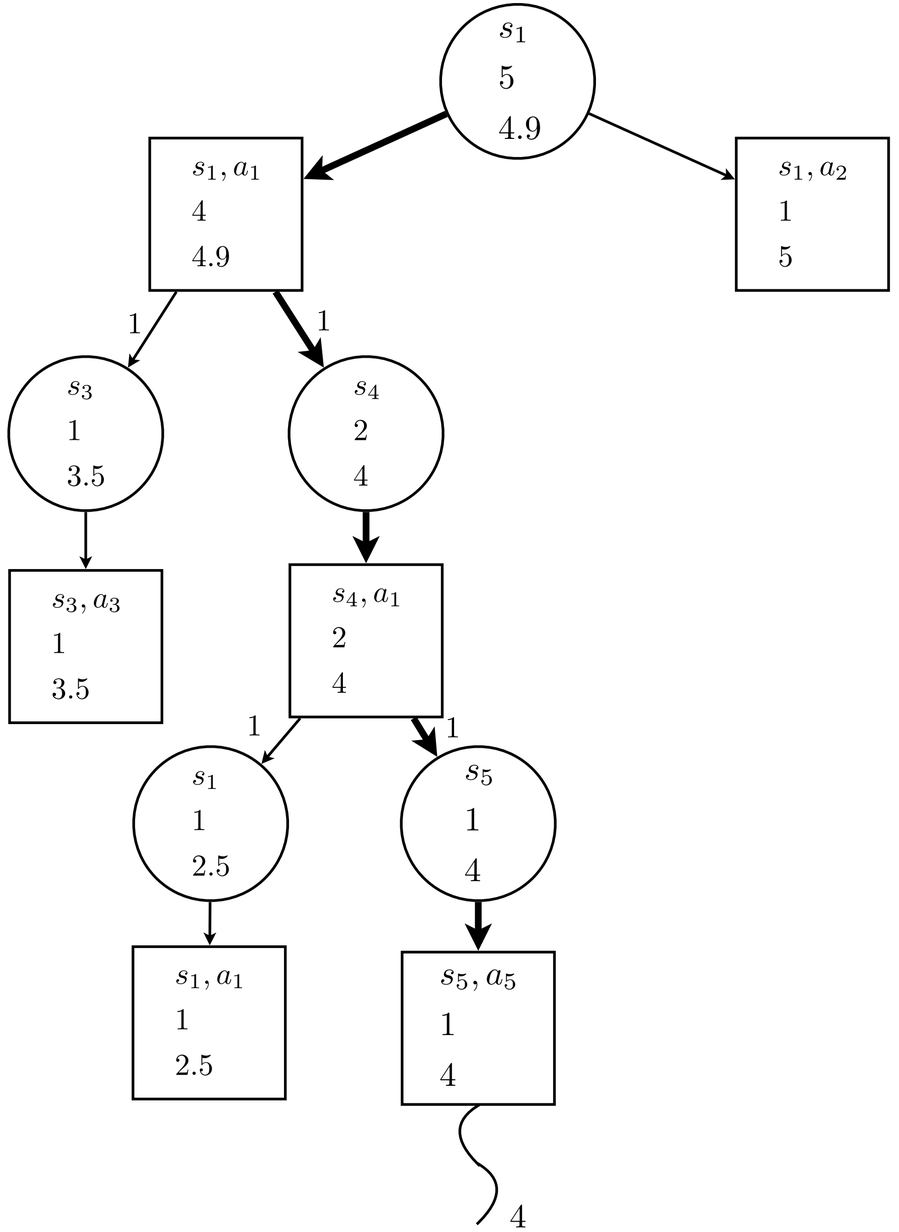




$$\hat{V}(s) = \min_a \hat{Q}(s, a)$$

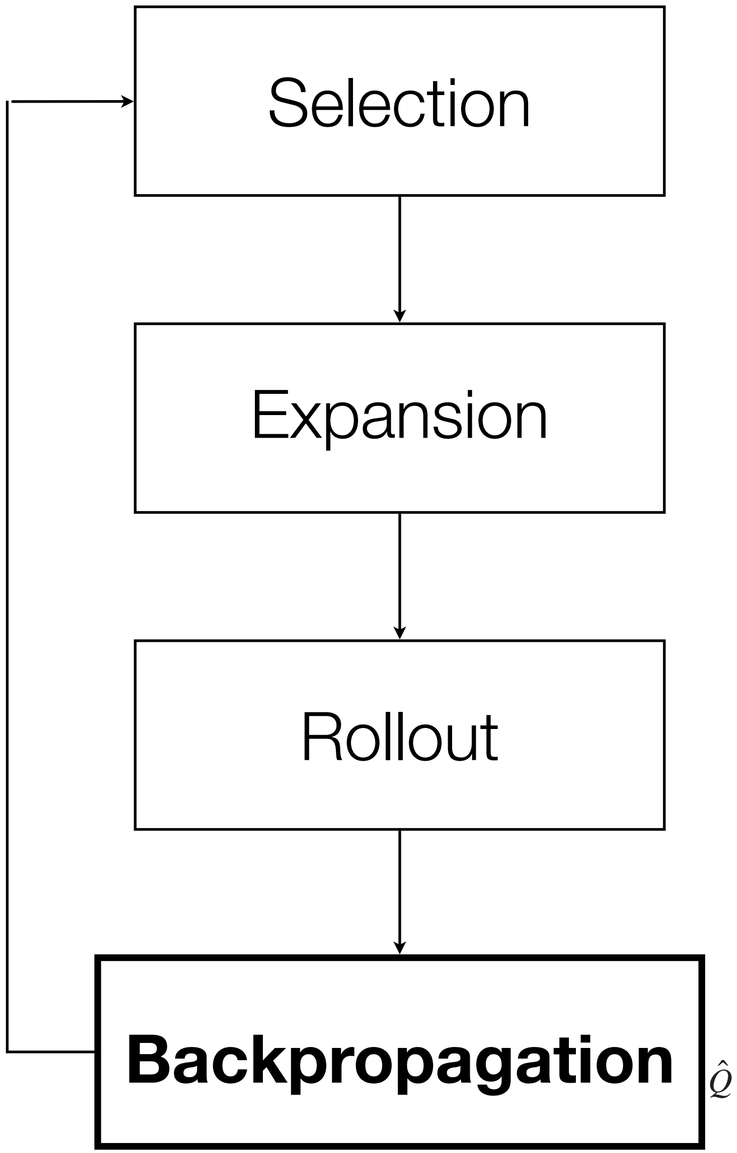
MCTS

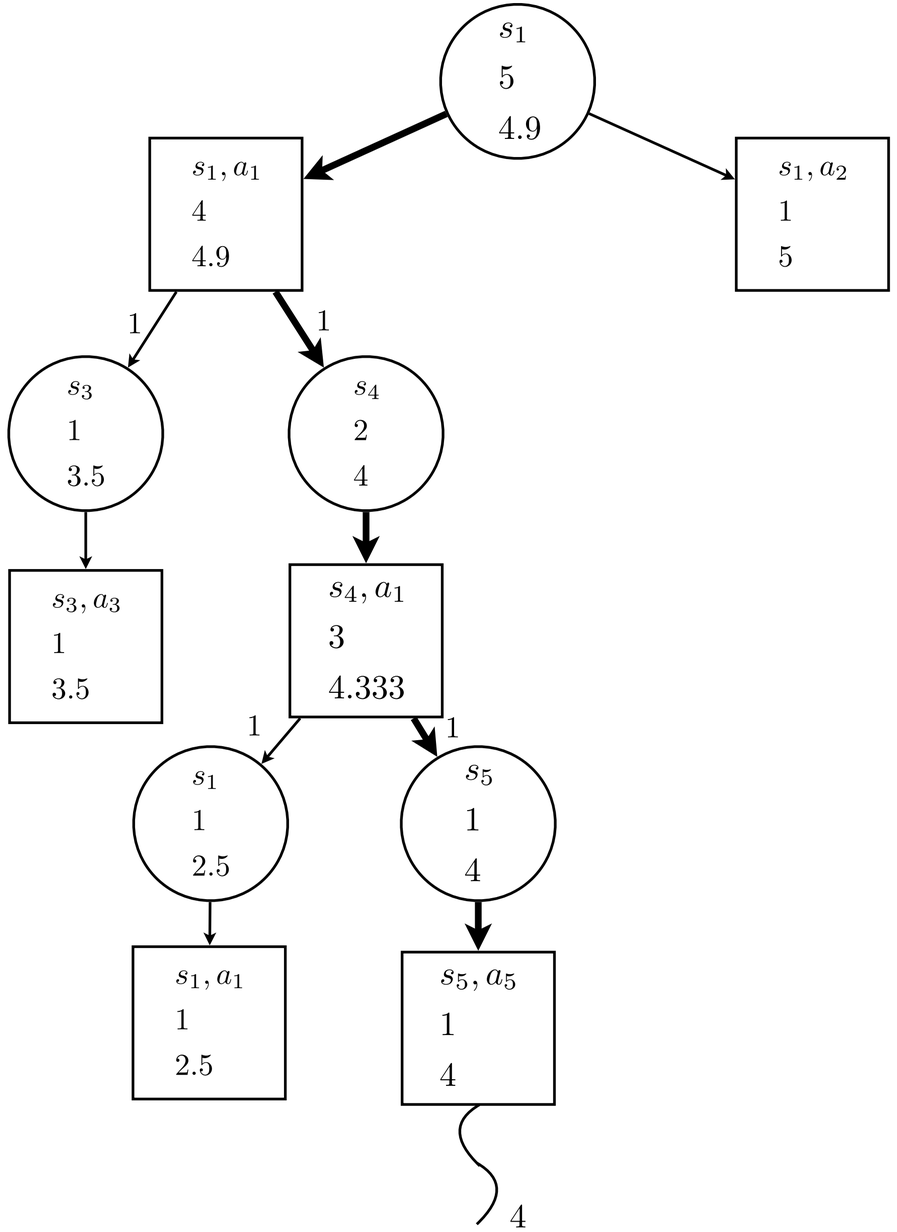




$$\hat{Q}(s_i, a_i) \leftarrow \frac{n_{s_i,a_i}\,\hat{Q}(s_i,a_i) + (cumulativeCost[maxNumSteps] - cumulativeCost[i-1])}{n_{s_i,a_i}+1}$$

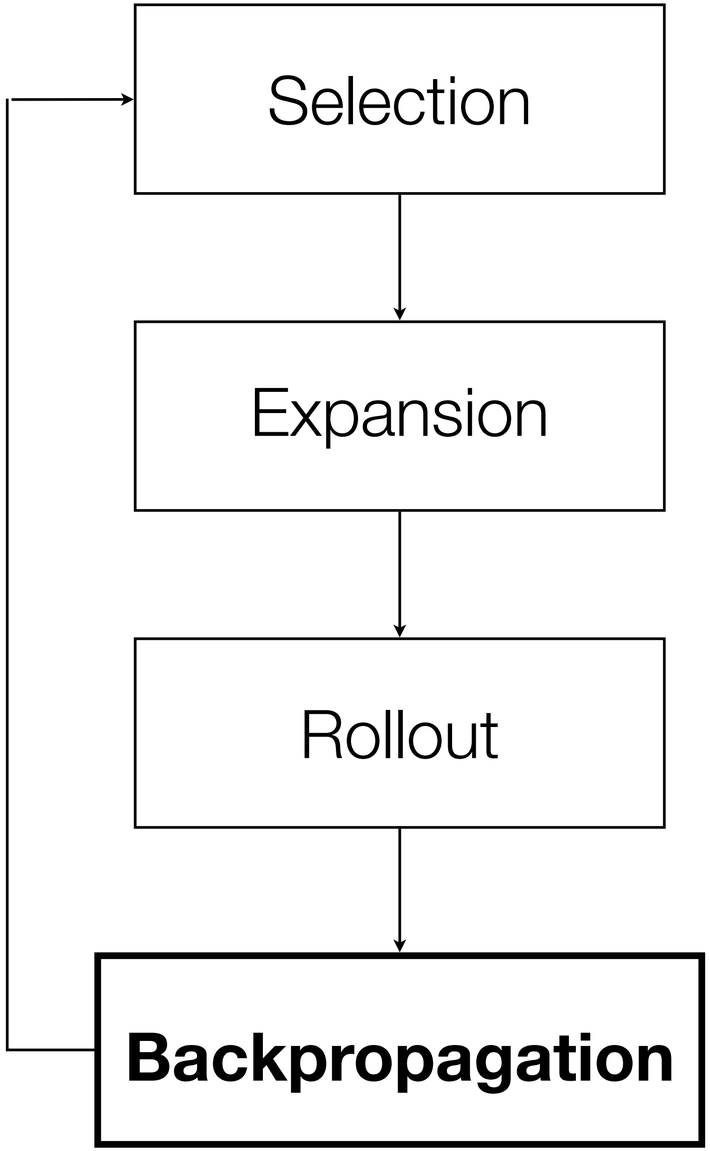
MCTS

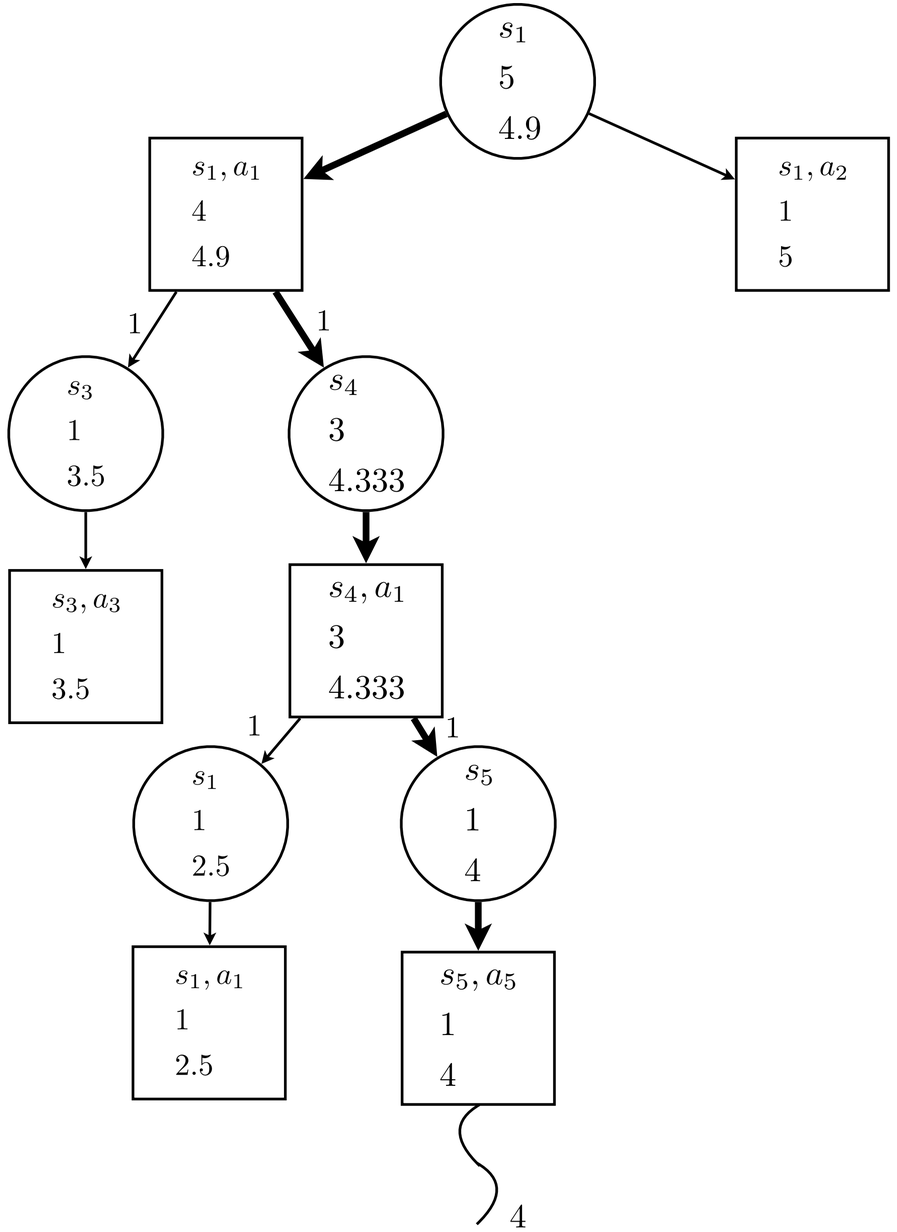




$$\hat{Q}(s_i, a_i) \leftarrow \frac{n_{s_i,a_i}\,\hat{Q}(s_i,a_i) + (cumulativeCost[maxNumSteps] - cumulativeCost[i-1])}{n_{s_i,a_i}+1}$$

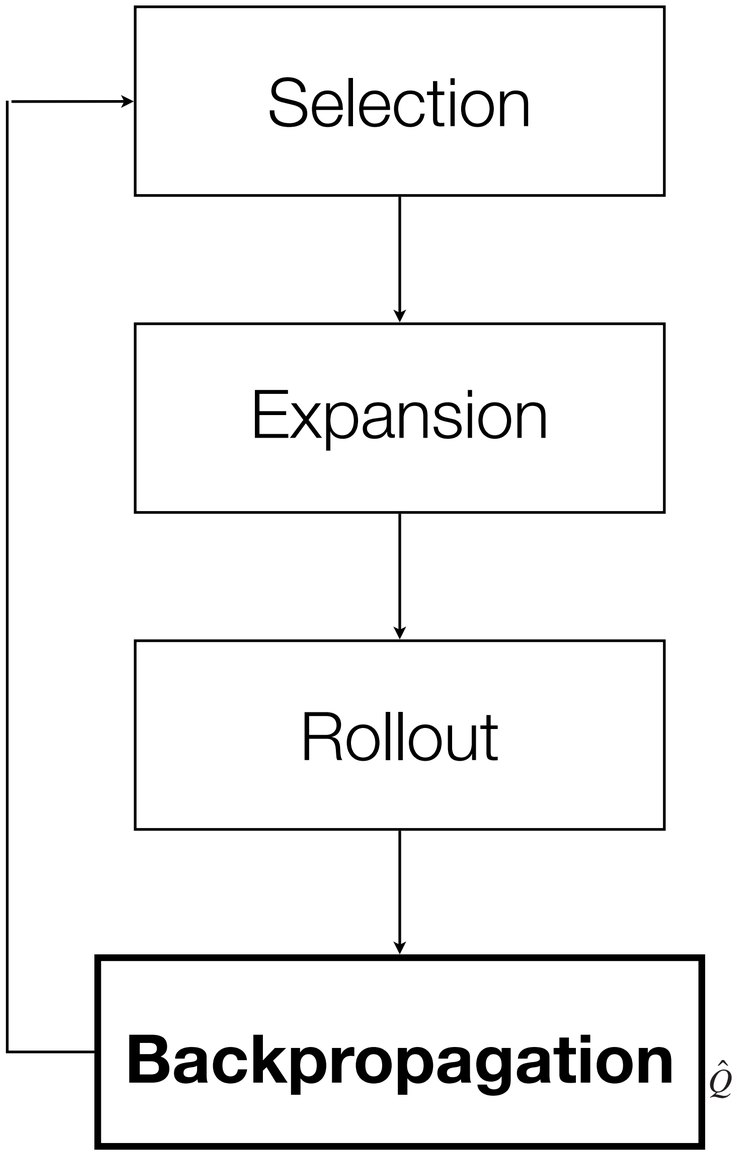
MCTS

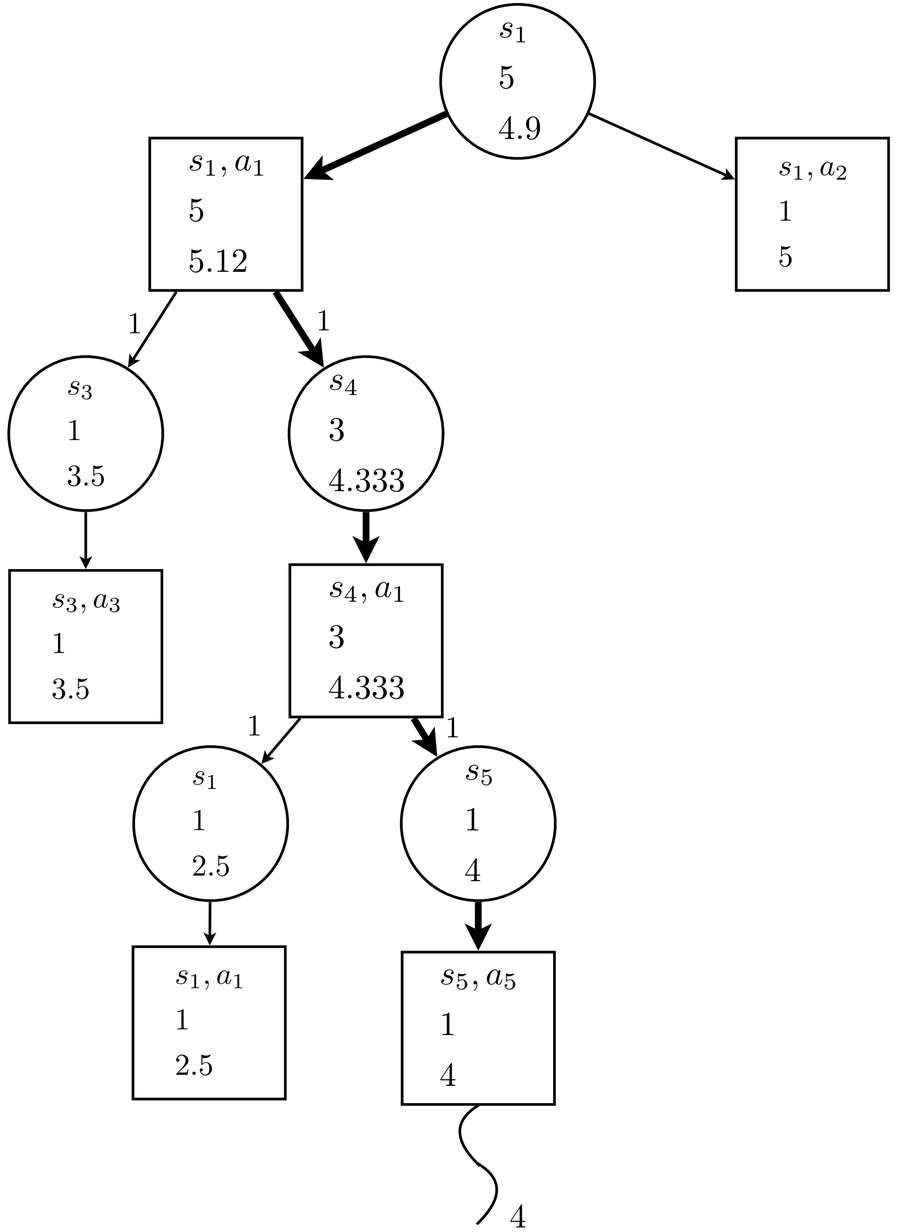




$$\hat{V}(s) = \min_a \hat{Q}(s, a)$$

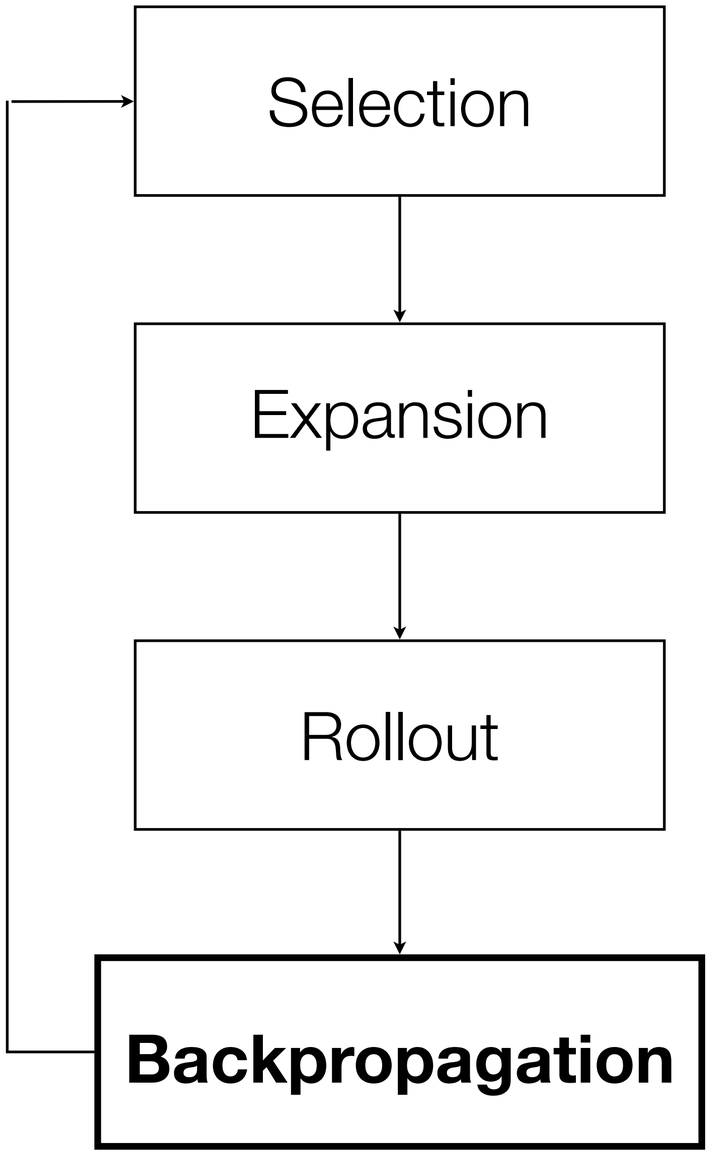
MCTS

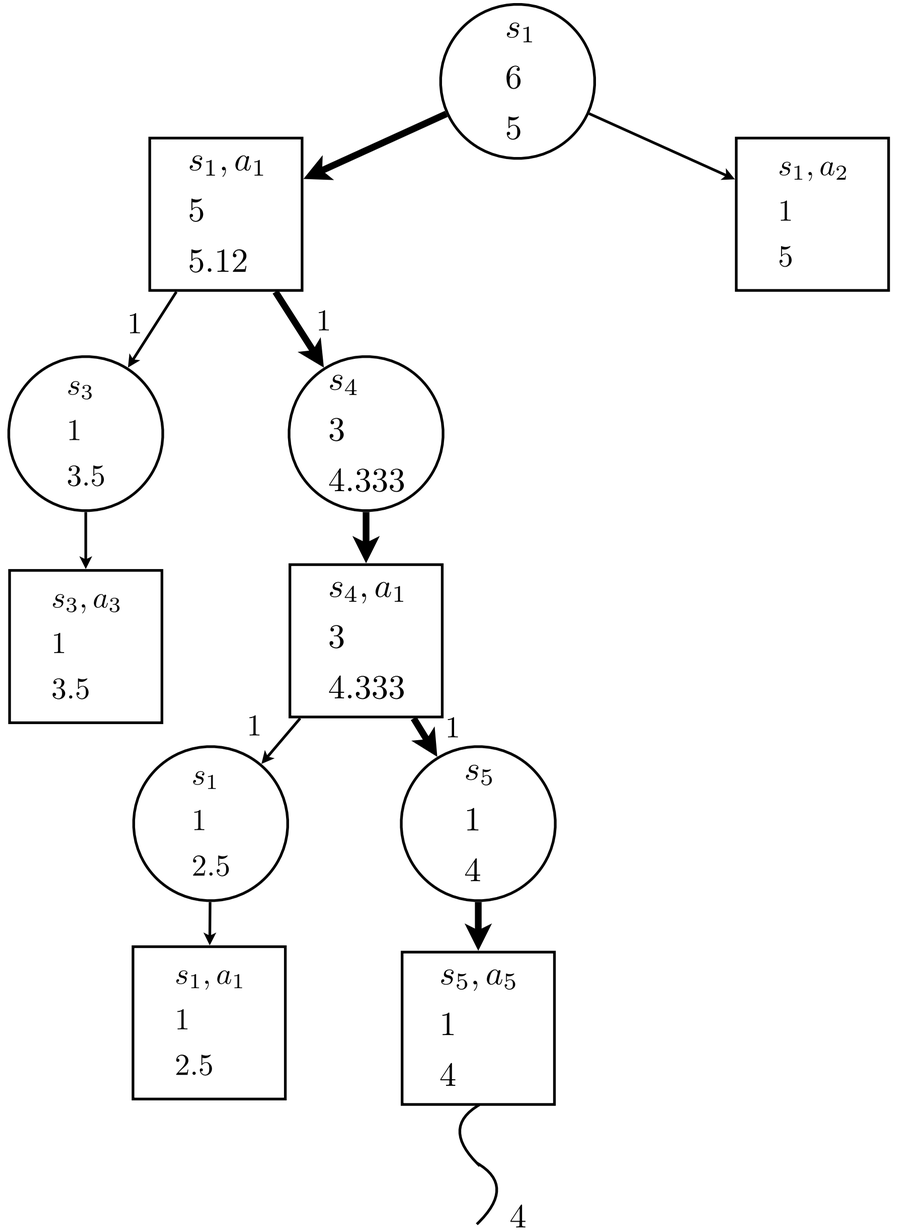




$$\hat{Q}(s_i, a_i) \leftarrow \frac{n_{s_i,a_i}\,\hat{Q}(s_i,a_i) + (cumulativeCost[maxNumSteps] - cumulativeCost[i-1])}{n_{s_i,a_i} + 1}$$

MCTS





$$\hat{V}(s) = \min_a \hat{Q}(s, a)$$

MCTS

MCTS

In the limit (i.e. given infinite samples), UCT produces the optimal value function, but can also function as an **anytime algorithm.**

MCTS

In the limit (i.e. given infinite samples), UCT produces the optimal value function, but can also function as an **anytime algorithm.**

It implicitly takes advantage of knowing the initial state, and **can be re-run throughout execution** to improve approximations as states are visited for real.

Overview

- Classical Planning to Replanning
- Planning with Non-Deterministic Models
- Planning with Probabilistic Models
  - Value Iteration
  - Beyond Value Iteration

Questions?