Multiagent Problem Formulation

José M Vidal

Department of Computer Science and Engineering, University of South Carolina

January 5, 2010

Abstract

We cover the most popular formal models for representing agents and multiagent problems.

Why Study Multiagent Systems?

- Multiagent systems everywhere!

Why Study Multiagent Systems?

- Multiagent systems everywhere!
- Internet: peer-to-peer programs (bittorrent), web applications (REST), social networks, the routers themselves.

Why Study Multiagent Systems?

- Multiagent systems everywhere!
- Internet: peer-to-peer programs (bittorrent), web applications (REST), social networks, the routers themselves.
- Economics: just-in-time manufacturing and procurement, sourcing, ad auctions.

Why Study Multiagent Systems?

- Multiagent systems everywhere!
- Internet: peer-to-peer programs (bittorrent), web applications (REST), social networks, the routers themselves.
- Economics: just-in-time manufacturing and procurement, sourcing, ad auctions.
- Political Science and Sociology: negotiations among self-interested parties.

Why Study Multiagent Systems?

- Multiagent systems everywhere!
- Internet: peer-to-peer programs (bittorrent), web applications (REST), social networks, the routers themselves.
- Economics: just-in-time manufacturing and procurement, sourcing, ad auctions.
- Political Science and Sociology: negotiations among self-interested parties.
- Nanofabrication and MEMS: sensor networks.

Why Study Multiagent Systems?

- Multiagent systems everywhere!
- Internet: peer-to-peer programs (bittorrent), web applications (REST), social networks, the routers themselves.
- Economics: just-in-time manufacturing and procurement, sourcing, ad auctions.
- Political Science and Sociology: negotiations among self-interested parties.
- Nanofabrication and MEMS: sensor networks.
- Biology: social insects, ontogeny, neurology.

- Science: How stuff works.

- Science: How stuff works.
- Engineering: How to build stuff.

- Science: How stuff works.
- Engineering: How to build stuff.
- Multiagent Systems: We want to build systems of, mostly, artificial agents. To do this we need to understand the science and math.

Fundamentals of Multiagent Systems

- Theory: Game Theory, Economics, Sociology, Biology, AI, Multiagent algorithms.
- Practice: NetLogo

History

- 1970s AI Boom

History

- 1970s AI Boom
- 1980s AI Bust, Blackboard Systems, DAI

History

- 1970s AI Boom
- 1980s AI Bust, Blackboard Systems, DAI
- 1990s The Web, Multiagent Systems

History

- 1970s AI Boom
- 1980s AI Bust, Blackboard Systems, DAI
- 1990s The Web, Multiagent Systems
- 2000s Ad auctions, Algorithmic Game Theory, Social Networks, REST

History

- 1970s AI Boom
- 1980s AI Bust, Blackboard Systems, DAI
- 1990s The Web, Multiagent Systems
- 2000s Ad auctions, Algorithmic Game Theory, Social Networks, REST
- 2010s ?

Grading

- Problem sets.
- Tests.

Our Model: The Utility Function

$$u_i : S \rightarrow \Re$$

Utility Requirements

- reflexive: $u_i(s) \geq u_i(s)$
- transitive: If $u_i(a) \geq u_i(b)$ and $u_i(b) \geq u_i(c)$ then $u_i(a) \geq u_i(c)$.
- comparable: $\forall_{a,b}$ either $u_i(a) \geq u_i(b)$ or $u_i(b) \geq u_i(a)$.

Utility is Not Money

Which one do you prefer

1. 50/50 chance at winning $10 dollars,
2. $5 dollars?

Utility is Not Money

Which one do you prefer

1. 50/50 chance at winning $1,000,000 dollars,
2. $500,000 dollars?

Expected Utility

$$E[u_i, s, a] = \sum_{s' \in S} T(s, a, s') u_i(s'),$$

Expected Utility

$$E[u_i, s, a] = \sum_{s' \in S} T(s, a, s') u_i(s'),$$

$T(s, a, s')$ probability of reaching s' from s by taking action a.

Maximum Expected Utility

$$\pi_i(s) = \arg\max_{a \in A} E[u_i, s, a]$$

Value of Information

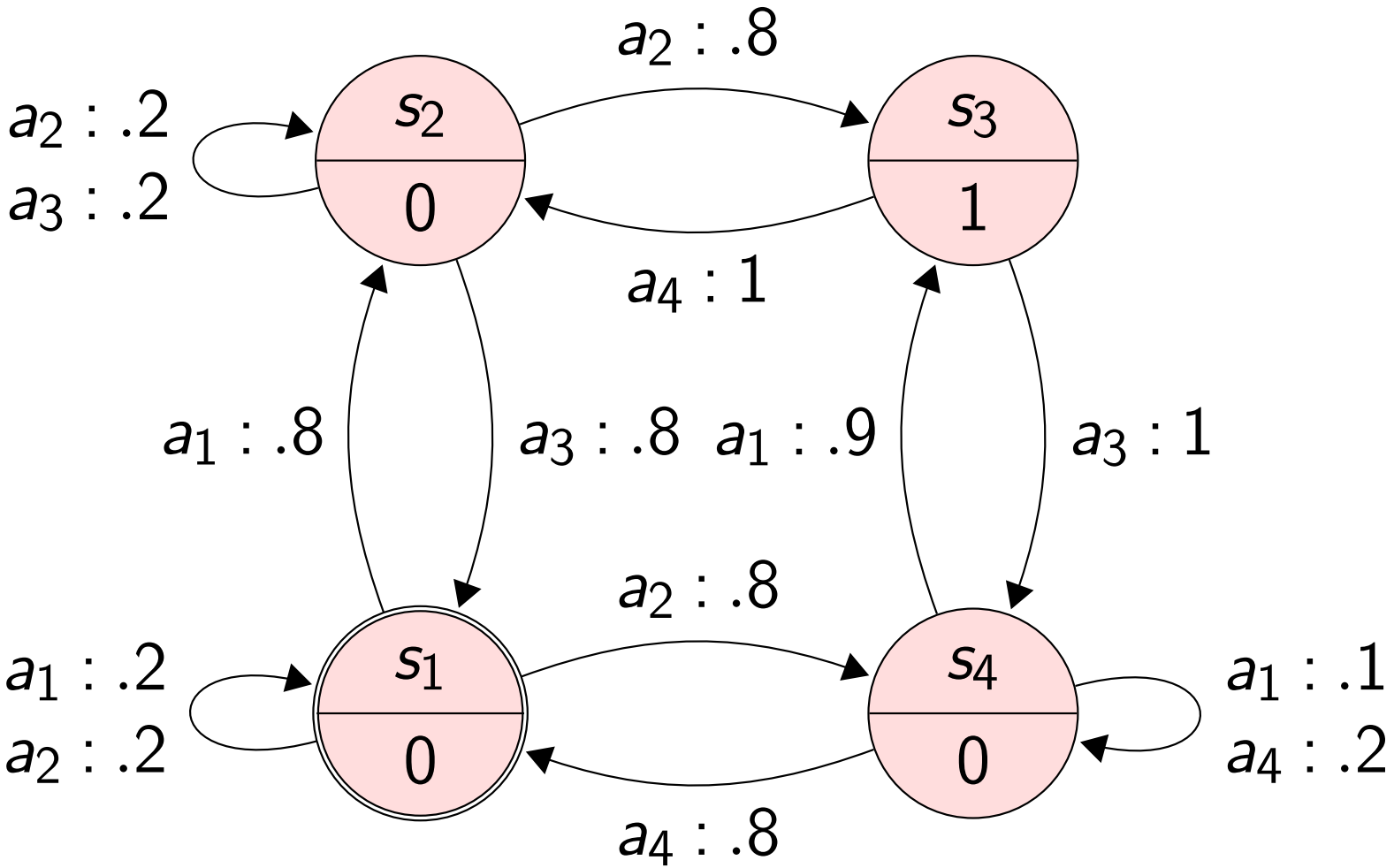
Value of information that tells agent it is not in s but is in t instead:

$$E[u_i, t, \pi_i(t)] - E[u_i, t, \pi_i(s)]$$

Markovian Assumption

Andrey Markov. 1856–1922.

Markov Decision Process



What To Do?

- Reward when arriving at each state.
- Must take action each time.

What To Do?

- Reward when arriving at each state.
- Must take action each time.
- Take a high reward now or go to states with higher reward?

Discount Future Rewards

Let γ be a discount factor, then reward at s_0 is

$$\gamma^0 r(s_0) + \gamma^1 r(s_1) + \gamma^2 r(s_2) + \cdots$$

Define Utility

$$u(s) = r(s) + \gamma \max_a \sum_{s'} T(s, a, s') u(s')$$

Define Utility

$$u(s) = r(s) + \gamma \max_a \sum_{s'} T(s, a, s') u(s')$$

Then its easy to calculate:

$$\pi^*(s) = \arg\max_a \sum_{s'} T(s, a, s') u(s')$$

Define Utility

$$u(s) = r(s) + \gamma \max_a \sum_{s'} T(s, a, s') u(s')$$

Then its easy to calculate:

$$\pi^*(s) = \arg\max_a \sum_{s'} T(s, a, s') u(s')$$

How do we calculate $u(s)$?

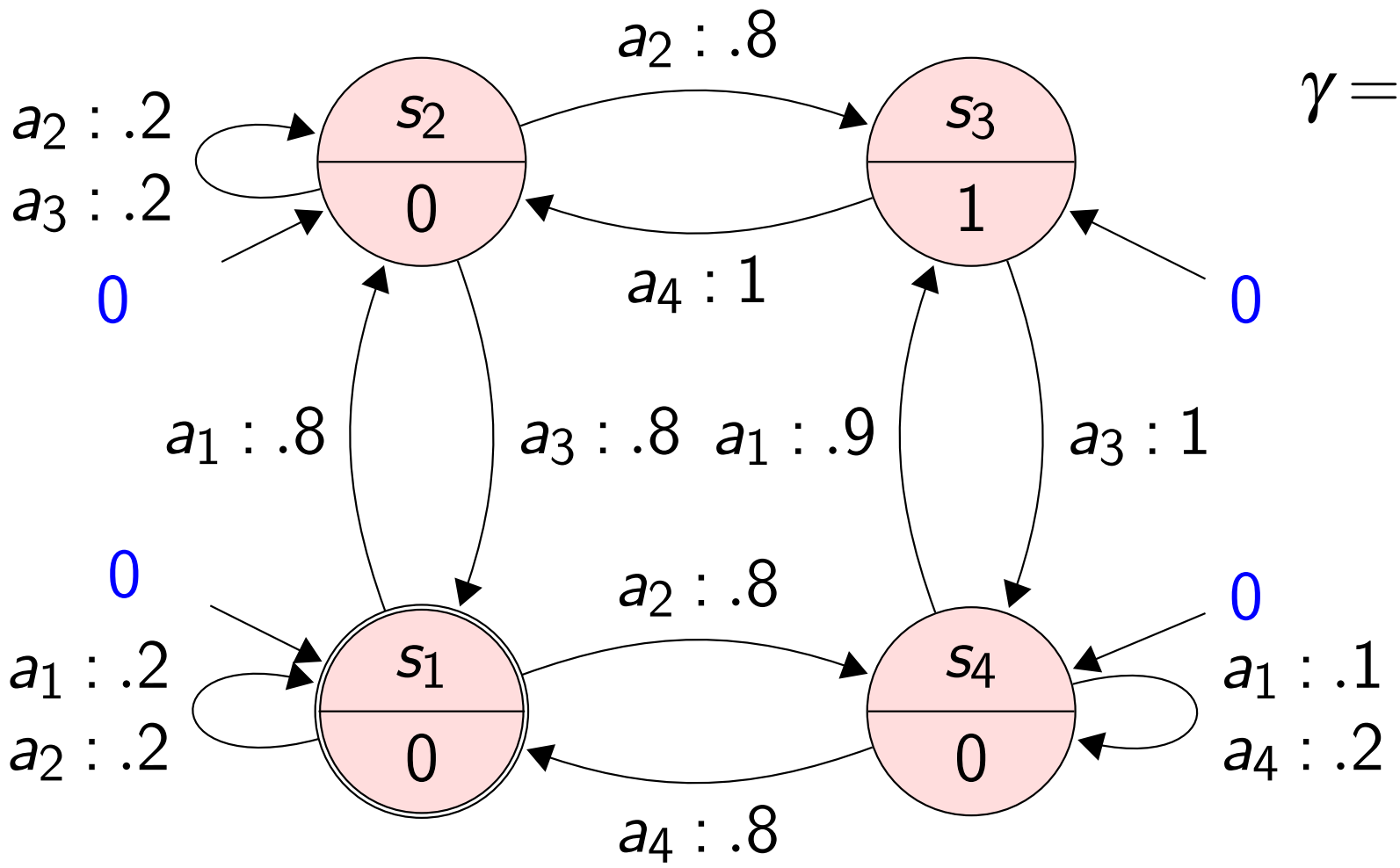
Bellman Update

Richard Bellman. 1920–1984. Inventor of dynamic programming.

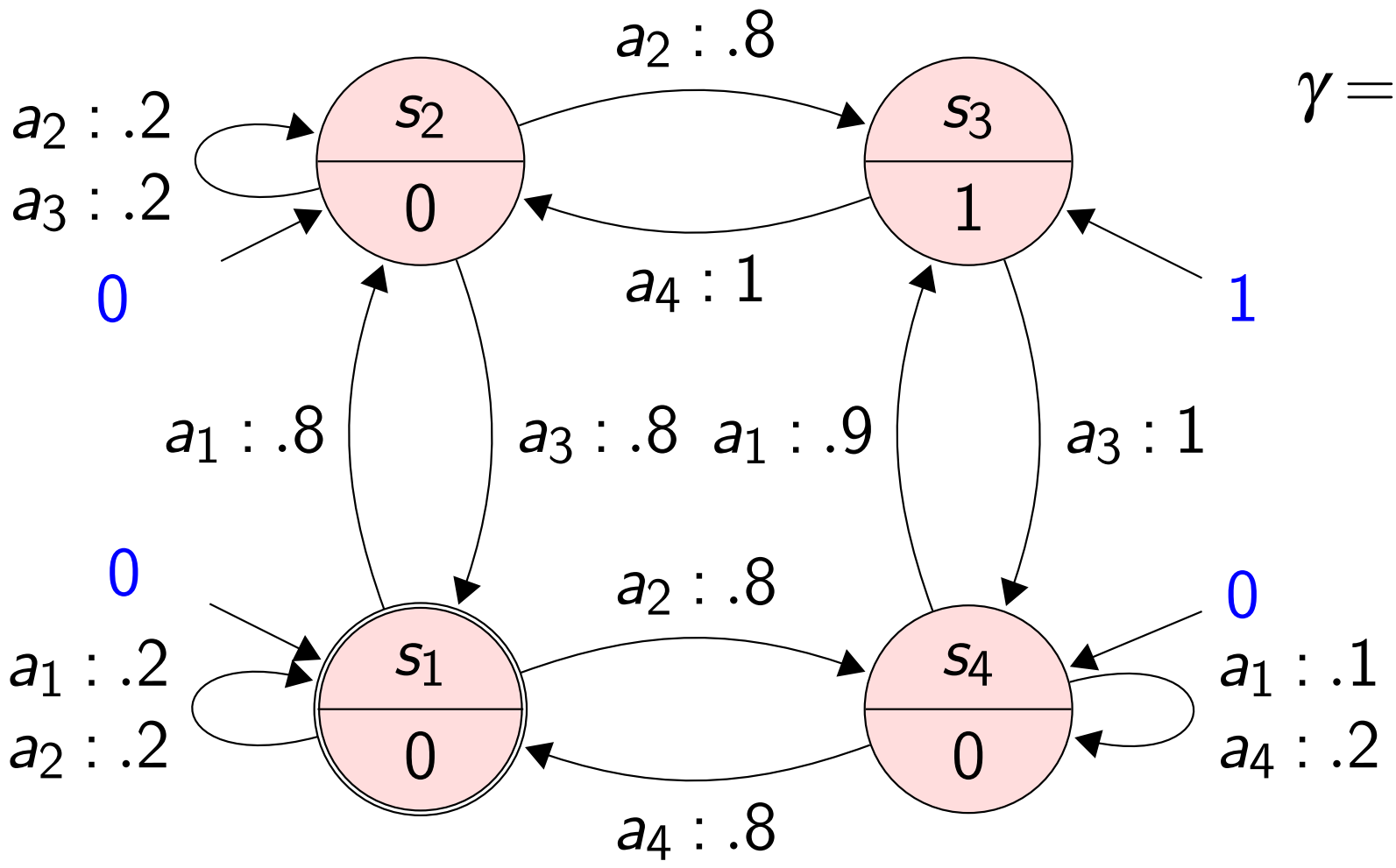
VALUE-ITERATION$(T, r, \gamma, \varepsilon)$

1 **repeat**
2 $\quad u \leftarrow u'$
3 $\quad \delta \leftarrow 0$
4 $\quad$ **for** $s \in S$
5 $\quad\quad$ **do** $u'(s) \leftarrow r(s) + \gamma \max_a \sum_{s'} T(s, a, s') u(s')$
6 $\quad\quad\quad$ **if** $|u'(s) - u(s)| > \delta$
7 $\quad\quad\quad$ **then** $\delta \leftarrow |u'(s) - u(s)|$
8 **until** $\delta < \varepsilon(1-\gamma)/\gamma$
9 **return** u

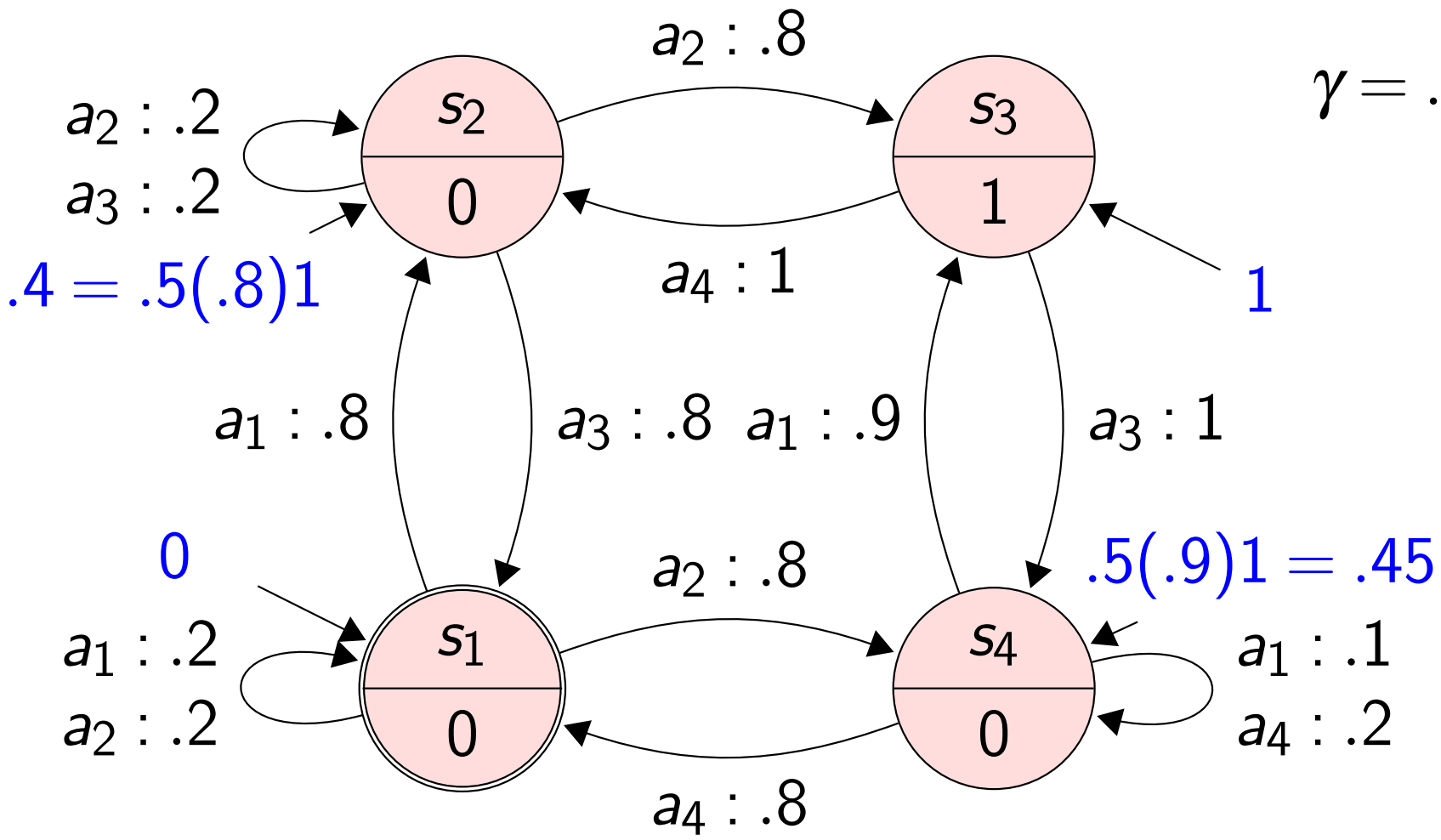
Value Iteration Example



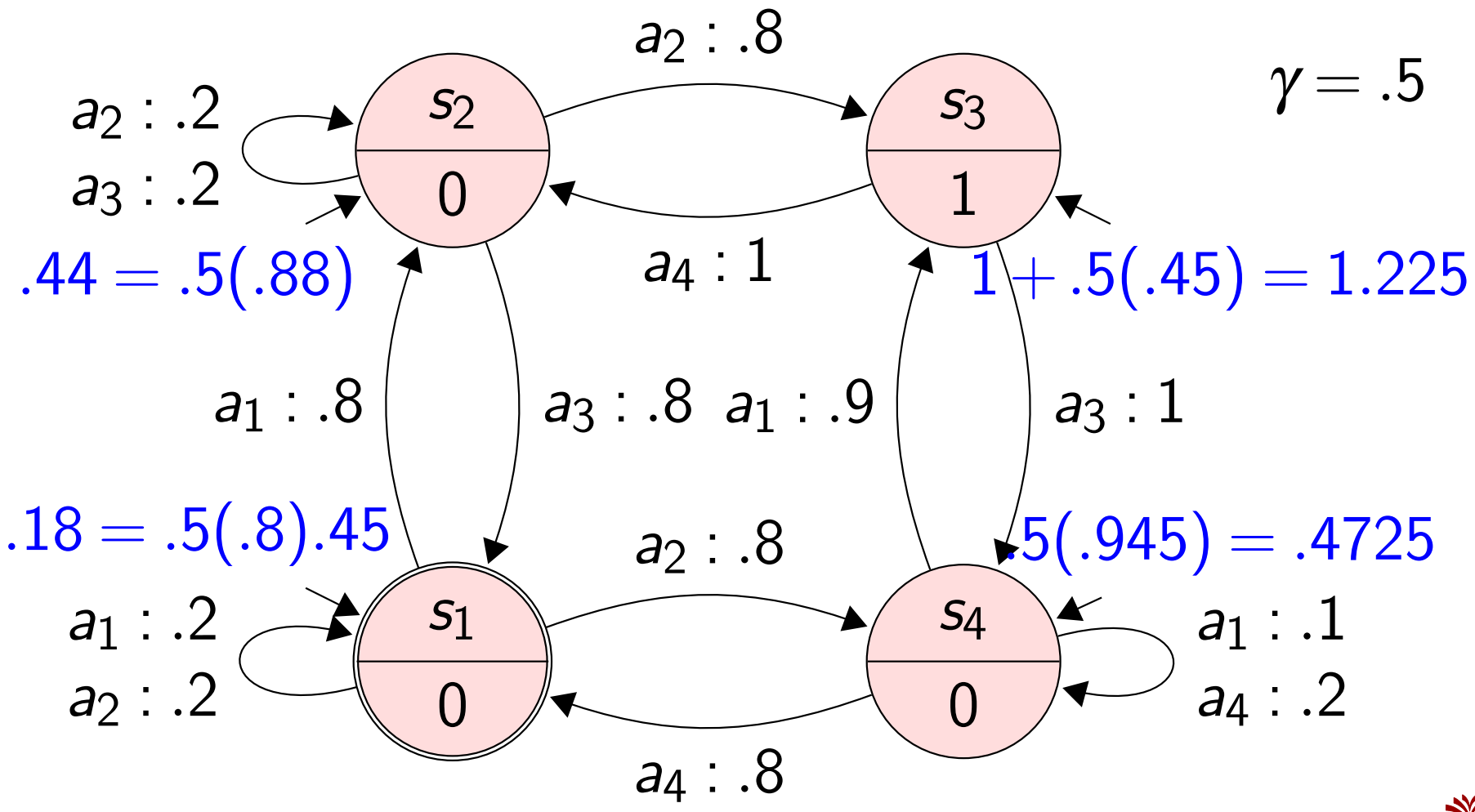
Value Iteration Example



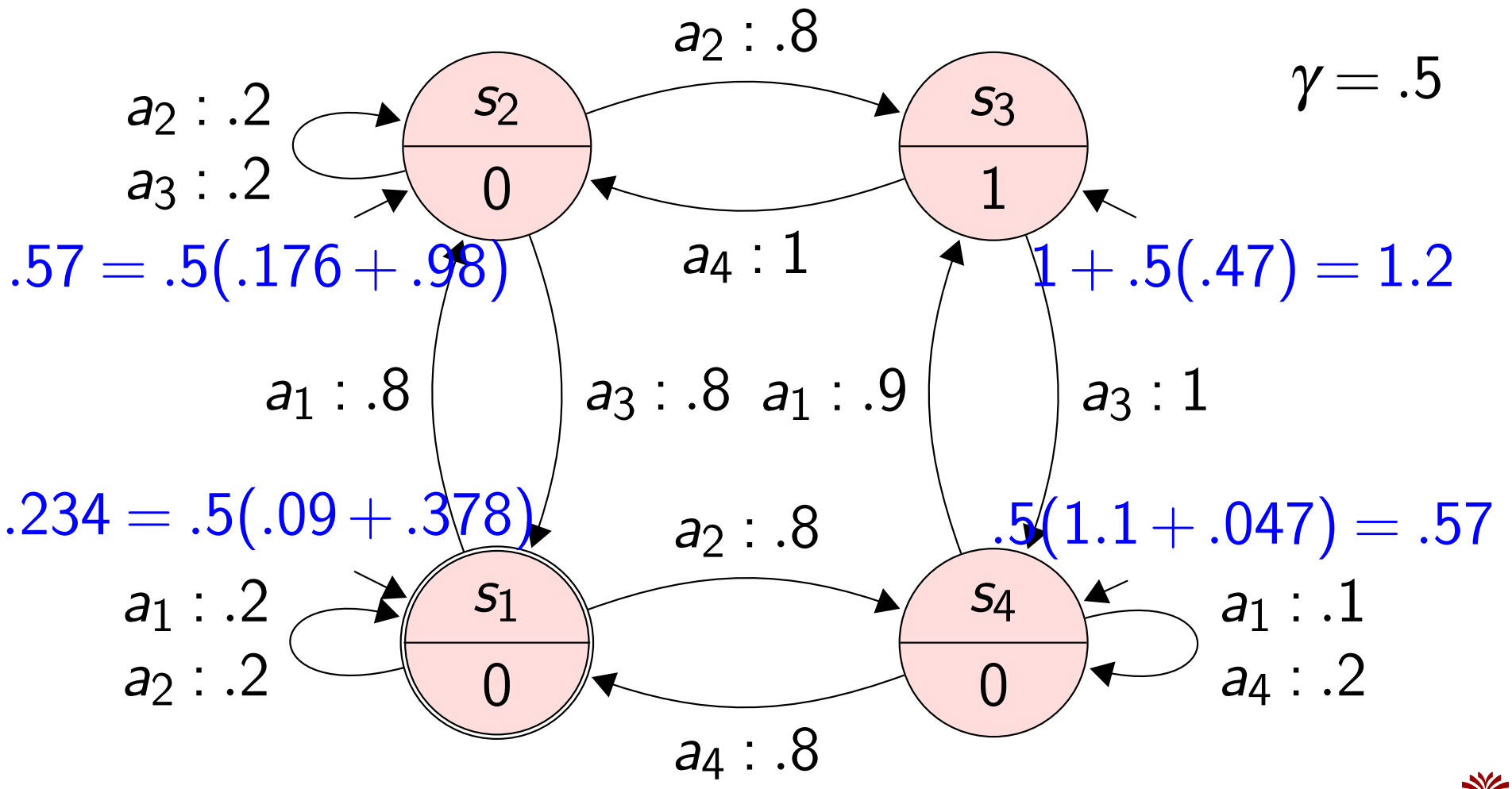
Value Iteration Example



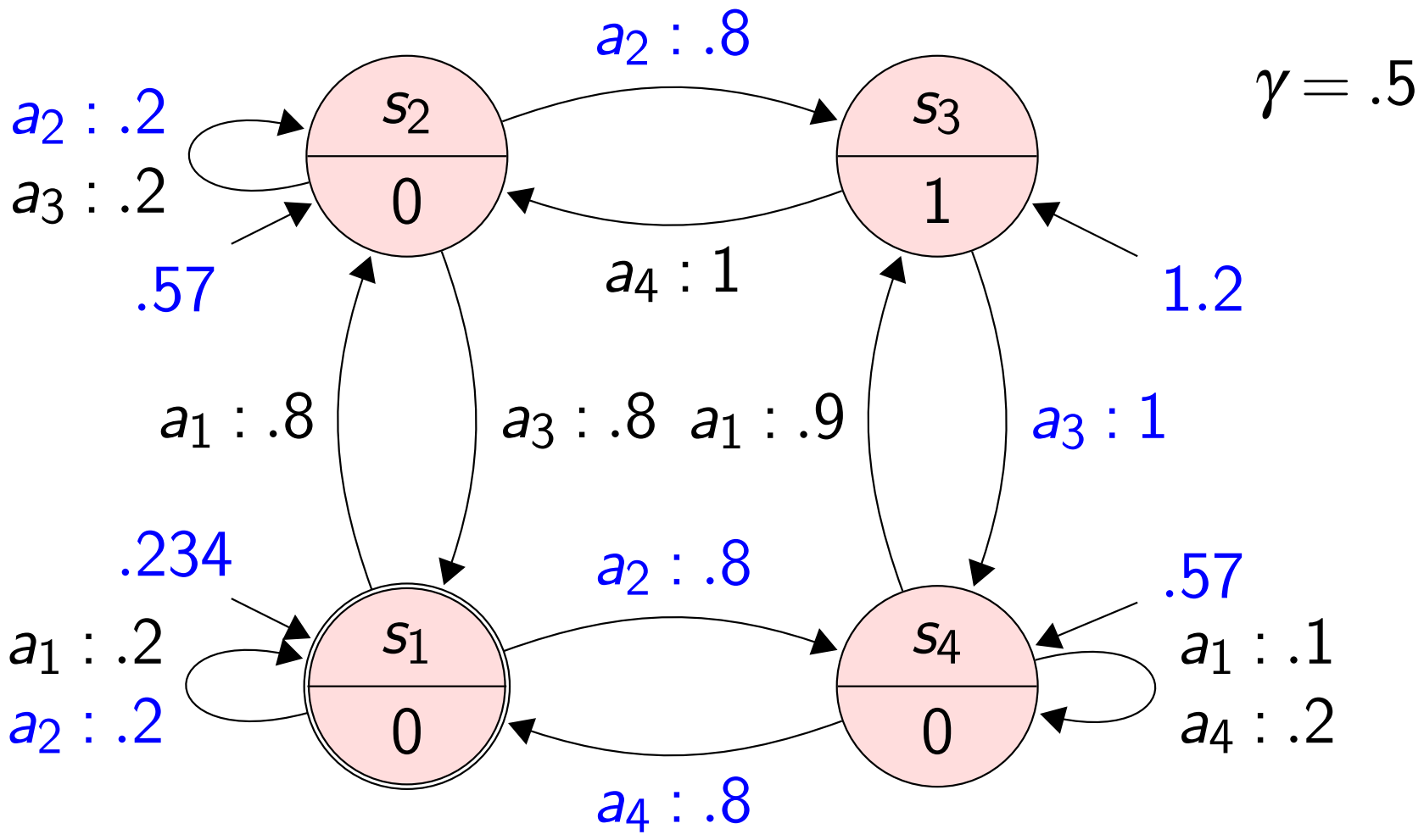
Value Iteration Example



Value Iteration Example



Value Iteration Example



Multiagent MDPs

- Instead of individual actions use a vector of actions.
- $T(s, a, s')$ becomes $T(s, \vec{a}, s')$.
- $r(s)$ becomes $r_i(s)$.

Multiagent MDPs

- Instead of individual actions use a vector of actions.
- $T(s, a, s')$ becomes $T(s, \vec{a}, s')$.
- $r(s)$ becomes $r_i(s)$.
- But, the other agents are messing up my rewards!

When Agent Can't See Everything

Use a Partially Observable MDP (POMDP).

- Belief state: $\vec{b}$
- Observation model: $O(s, o) \rightarrow [0, 1]$

Belief Update

An agent with beliefs $\vec{b}$ takes action a and now observes o. It can update its beliefs to

$$\forall_{s'} \; \vec{b}'(s') = \alpha O(s', o) \sum_{s} T(s, a, s') \, \vec{b}(s) \qquad (1)$$

Build MDP with Belief States

$$\tau(\vec{b}, a, \vec{b}') = \begin{cases} \sum_{s'} O(s', o) \sum_{s} T(s, a, s') \vec{b}(s) & \text{if (1) is true} \\ 0 & \text{otherwise,} \end{cases}$$

and reward function

$$\rho(\vec{b}) = \sum_{s} \vec{b}(s) r(s)$$

Still Problematic

- There is now one state for each possible $\vec{b}(s)$.
- These are continuous values.
- But, there are value iteration algorithms that use regions to solve these problems.
- Better to use dynamic decision networks.

AI Planning

- Operators with pre-conditions and effects.
- It is a special case of an MDP.
- More succinct representation (sometimes).
- Many planning algorithms exist. Turns out, most use a graphical representation.

Hierarchical Planning

- Divide and Conquer: build plans using big (general) operators then make each operator its own planning problem.
  - First plan plane,taxi,car then plan how to get from gate to taxi.
- Plan hierarchy can be used to share partial plans:
  - If agent plans to be in Charleston then the one in Columbia knows it won't bump into him: no need to know exactly where in Charleston.

Summary

- Utility functions and MDP are most used: power, analyzable in very small cases.
- Most multiagent research tries to solve the large cases.