Algorithm and Complexity Issues in Discrete Multistage Games

Michael L. Littman

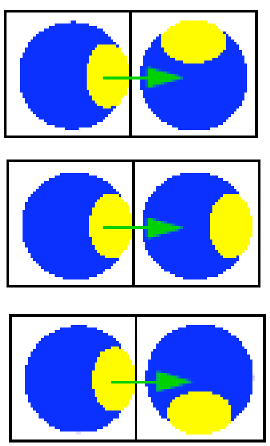
Rutgers University
Department of Computer Science
Rutgers Laboratory for Real-Life Reinforcement Learning

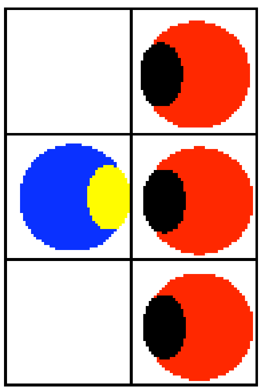
Simple Game: “Sharks”

6x4 grid, two players (with directionality)

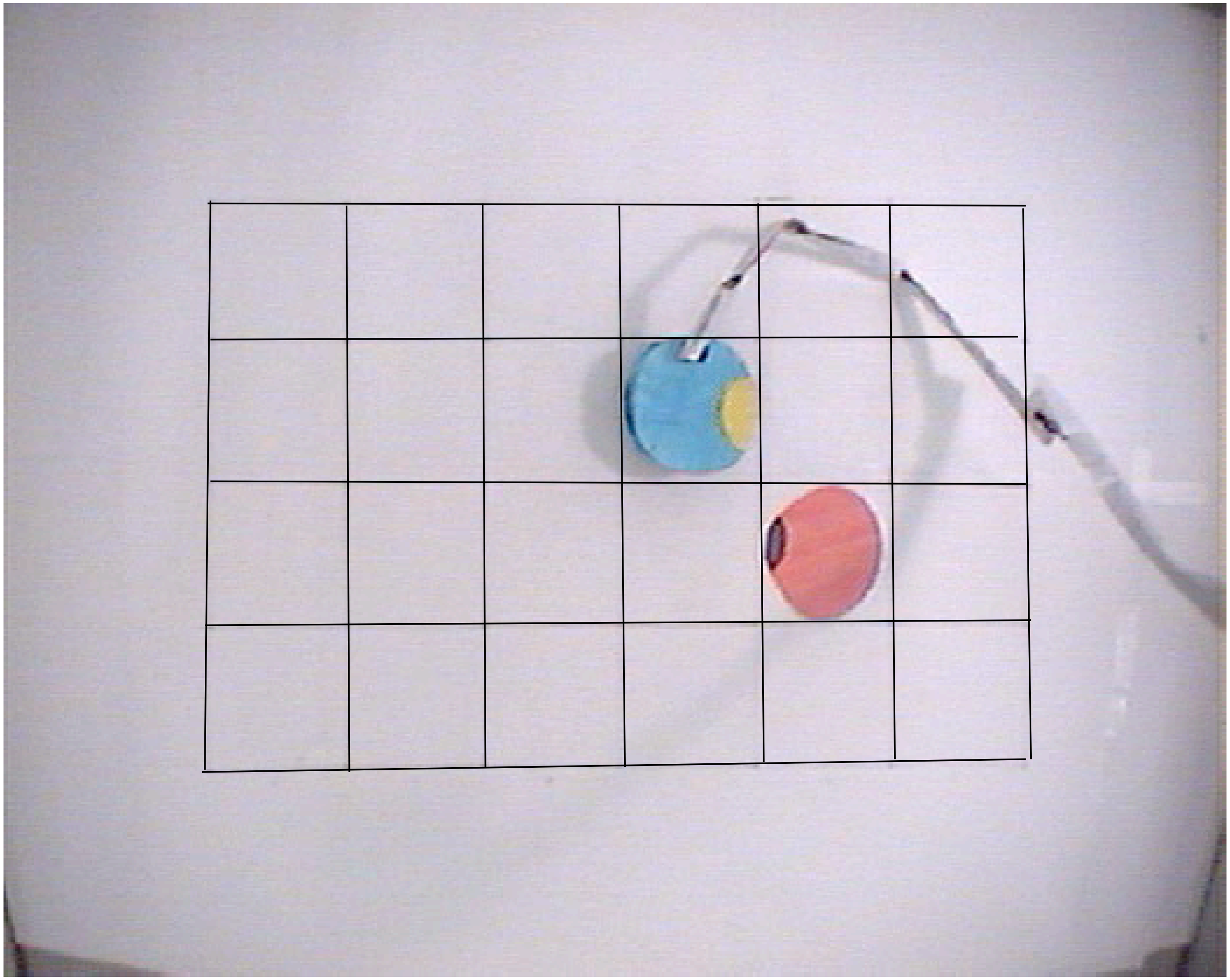
Movement: Step forward, option of turning right or left (or no turn).

Lose on your turn if facing a wall or your opponent.





“Sharks” Robot



Class of Games Considered

Finite states: Position on the board matters, history does not (Markov property).

Finite actions: Agents' decisions from finite set of choices (say forward, turn left 90 degrees, stop).

Discrete time: Decisions take place in rounds.

Stochastic transitions: Actions (like turn) change state, but not deterministically (perhaps a wheel slips).

Perfect observations: State perceived by all players.

Objective: Maximize total discounted expected reward.

Formal Models (Zero-sum)

Stochastic Game

- S: finite set of states (starting state $s_0 \in S$)
- A_1, A_2: finite sets of actions for the two players.
- $\Pr(s' | s, a_1, a_2)$ probability of state $s' \in S$ after joint action choice for the two players from state $s \in S$.
- $R(s, a_1, a_2)$: one-step reward for first player after joint action choice in state s (opponent: $1 - R(s, a_1, a_2)$).

"Alternating" Game: For all s, a_1, a_2, s', a' either

- $\Pr(s' | s, a_1, a') = \Pr(s' | s, a_1, a_2)$, $R(s, a_1, a') = R(s, a_1, a_2)$, or
- $\Pr(s' | s, a', a_2) = \Pr(s' | s, a_1, a_2)$, $R(s, a', a_2) = R(s, a_1, a_2)$

MDP (One-player game): For all s, a_1, a_2, s', a'

- $\Pr(s' | s, a_1, a') = \Pr(s' | s, a_1, a_2)$, $R(s, a_1, a') = R(s, a_1, a_2)$

Solution Concept: Minimax

Algorithmic problem:

- given game (S, A_1, A_2, Pr, R)
- find a strategy for selecting actions (policy)
- that maximizes total expected discounted reward (value): $\Sigma_t \gamma^t R(s^t, a_1^t, a_2^t)$
- assuming opponent knows the strategy and chooses actions to minimize value.

Helpful fact:

- There is a stationary optimal policy: $\pi^*(s)$.

Complexity Concerns

Assume rational-valued transitions, rewards.

Stochastic Game

- optimal value, policy can be irrational (Vrieze 87)
- approximations feasible, though

"Alternating" Game

- optimal deterministic stationary policy
- optimal value always rational numbers
- in NP $\cap$ co-NP, not known to be in P (Condon 93)

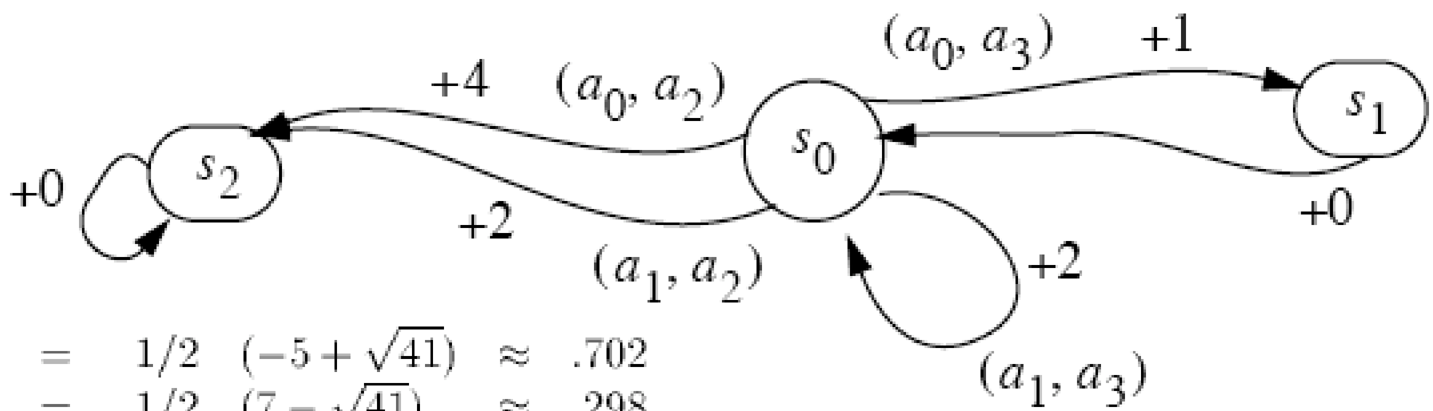
MDP

- P-complete (Papadimitriou and Tsitsiklis 87)

Deterministic Models

$\Pr(s' \mid s, a_1, a_2) = 0$ or 1

Stochastic Game



$$
\begin{aligned}
\pi_1(s_0, a_0) &= 1/2\ (-5+\sqrt{41}) \approx .702\\
\pi_1(s_0, a_1) &= 1/2\ (7-\sqrt{41}) \approx .298\\
\pi_2(s_0, a_2) &= 1/12\ (-1+\sqrt{41}) \approx .450\\
\pi_2(s_0, a_3) &= 1/12\ (13-\sqrt{41}) \approx .550
\end{aligned}
$$

(Littman 96)

"Alternating" Game: Open (Zwick & Paterson 96)

MDP: in NC (P & T 87), $O(|A||S|^3)$ (Littman 96)

Planning in Markov Models

Basic idea, need to know *value* of s_0:

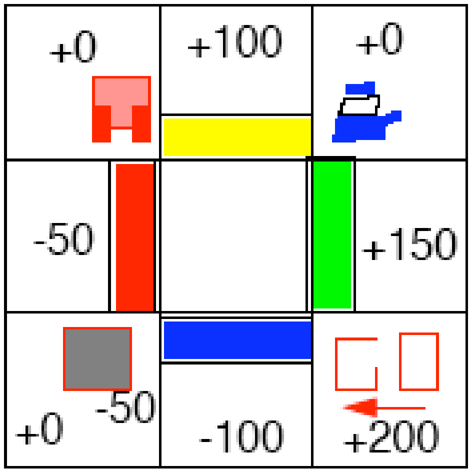
Total expected discounted reward of best policy starting from state s_0.

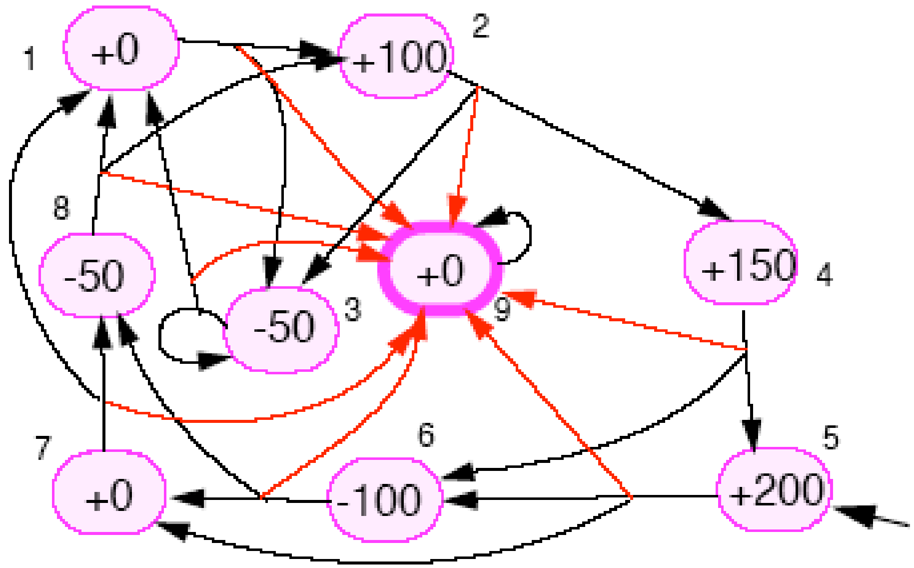
"Simple machines" of Markov model planning:

- Search: Create a tree of possible sequences.
- Simulation: Use weighted sampling to handle stochastic transitions.
- Dynamic programming: Calculate value from all states in S simultaneously.

Can be combined in various ways.

Evaluation Example: Coinopoly

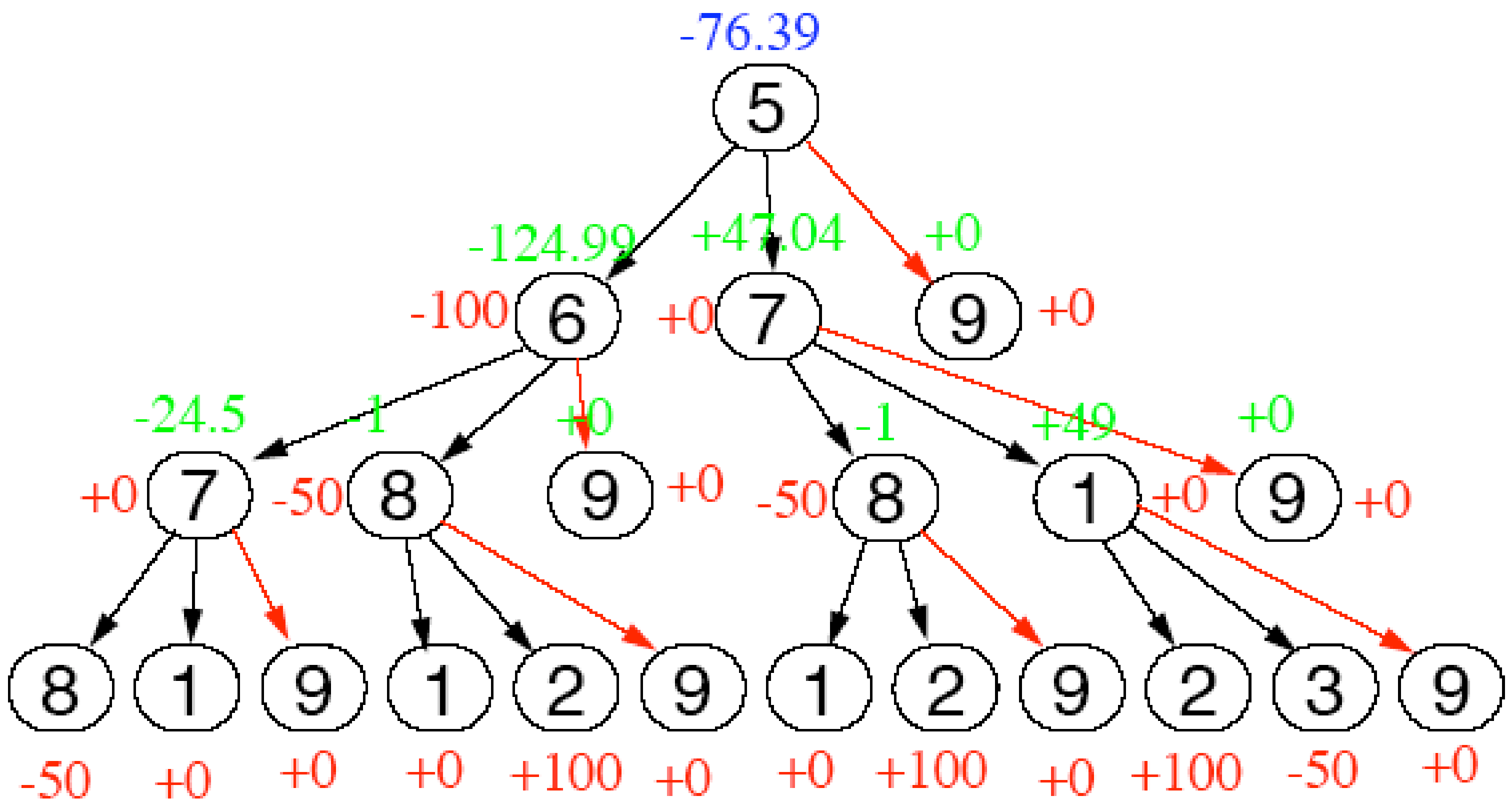




Start at “Go”. Flip: heads move 1, tails 2. Reward based on landing space. “Go To Jail” (“tails” to get out). Before each move, game terminated with probability 0.02 (γ=0.98).

MONOPOLY® is a trademark of Parker Brothers. (COINOPOLY isn’t.)

Search



Expand tree out from start state, sum values up tree.

Search: Analysis

Strengths:

- Simple.
- Focus computation on reachable states.
- Combines well with heuristics.

Weaknesses:

- May consider the same state multiple times (resulting in a huge search tree).
- Results in an approximation for cyclic models.

Search: Running Time

Given a discount factor of γ, expanding tree to a depth of

$D = O(1/(1-\gamma) (\log(1/(1-\gamma))+\log(1/\varepsilon)))$

is sufficient to get an ε-optimal policy.

If B is the branching factor, then $O(B^D)$ time is sufficient for near-optimal decision making.

Note: No dependence on $|S|$.

Simulation

Run it *m* times, take the average.

- 5 7 1 3 3 1 3 3 1 3 3 3 1 3 1 3 1 2 4 5 6 8 2 3 9: 100
- 5 6 8 1 2 4 5 7 1 3 1 3 9: +200
- 5 7 8 2 3 1 3 9: -50
- 5 7 8 2 9: +50
- 5 6 7 1 3 3 3 3 1 3 1 2 3 3 1 2 9: -250
- ...

1,000 times, mean: +269.65.

Simulation: Analysis

Strengths:

- Simple.
- Focus computation on *likely* reachable states.
- Can get good approximation with little work.
- Applicable without explicit transition probabilities.
- “PAC”-style guarantees with tail bounds.

Weaknesses:

- Only approximate guarantee.
- May require many samples.
- Somewhat wasteful of data.

Simulation: Running Time

From Hoeffding bounds, $m = O(1/\varepsilon^2 \log(1/\delta))$ samples sufficient to estimate a random quantity to within an accuracy of ε with probability $1-\delta$.

To get an approximate value, each sample should be D steps, so $m\ D$ time altogether.

Note: No dependence on $|S|$ *or* branching factor B. However, fails with some prob.

Dynamic Programming

Fundamental idea: Instead of just the value of the start state, we find the value of *all* states (the "value function").

1	2	3	4	5	6	7	8	9
277.41	297.65	218.49	288.96	218.10	271.60	273.51	330.78	0.00

Insight: By the linearity of expectations, $V(s) = \Sigma_{s' \text{ in } S} \Pr(s' | s)\, \gamma\, (R(s') + V(s'))$

System of simultaneous linear equations.

Dynamic Programming: Analysis

Strengths:

- Calculation exact even for cyclic problems.
- Calculation is relatively efficient.

Weaknesses:

- More complicated to represent & compute values.
- Can "overcompute" in that values are computed for states that don't matter.

Dynamic Prog.: Running Time

Using Gaussian elimination, solution found in $O(|S|^3)$.

Note: Big dependence on $|S|$, but exact answer and no branching or γ dependence.

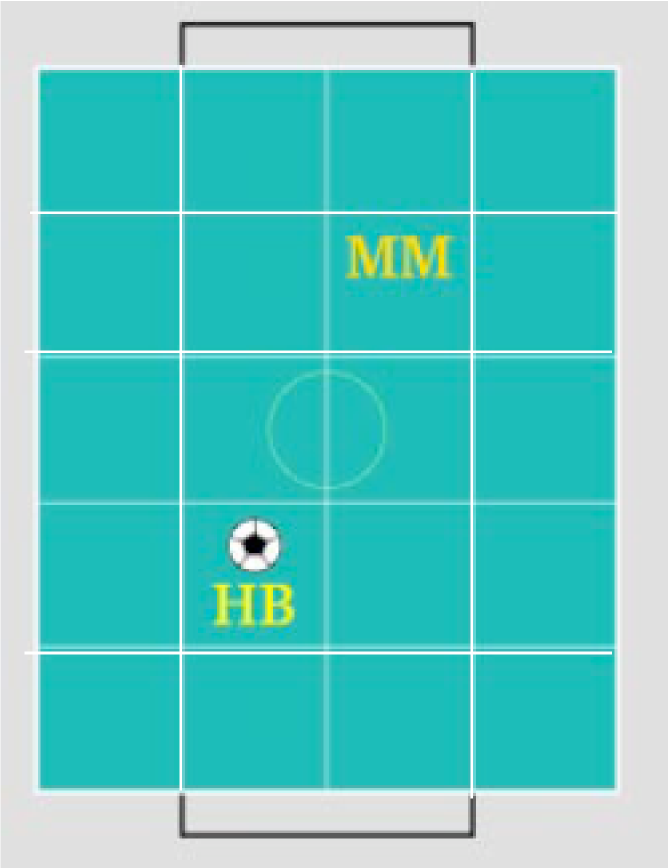
Extending to Games

The evaluation problem doesn't deal with decision making.

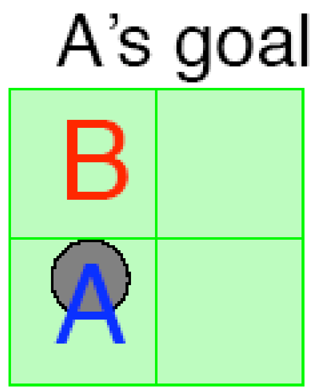
We'll next discuss more complex decision making using a more general game scenario.

Grid Soccer Example

N,S,E,W,X; lose ball if hit other; order randomized



Soccer Dilemma



payoff to A

		A: N	A: E
B	E	1	0
B	X	-1	0.5

What should A do in this situation?

Depends on what B will do!

Randomness Can Help

payoff to A

		A: N	A: E	A: .2N/.8E
B	E	1	0	0.2
	X	-1	0.5	0.2

If A chooses N with probability .2 and E with .8:

- B goes E: 0.2
- B stays put: 0.2

Any other choice for A does worse.

Equivalent to LP; solvable in P (Khachiyan 79).

Bellman Equation for Games

Stochastic Game

For all $s \in S$,

$$Q^*(s,a_1,a_2) = R(s,a_1,a_2) + \gamma \sum_{s' \in S} \Pr(s' | s,a_1,a_2)\, V^*(s')$$

$$V^*(s) = \max_{\rho \in \Pi(A1)} \min_{a2 \in A2} \sum_{a1 \in A1} \rho(a_1)\, Q^*(s,a_1,a_2)$$

The ρ that achieves the max in s is the optimal choice.

"Alternating" Game

$$V^*(s) = \max_{a1 \in A1} Q^*(s,a_1,a_2), \text{ or}$$

$$V^*(s) = \min_{a2 \in A2} Q^*(s,a_1,a_2)$$

MDP

$$V^*(s) = \max_{a1 \in A1} Q^*(s,a_1,a_2)$$

Value Iteration

Use approximate values to improve approximation.

Let $V_0(s) = 0$ for all $s \in S$.

Let t=1.

Do {

For all $s \in S$, $a_1 \in A_1$, $a_2 \in A_2$,

$$Q_t(s,a_1,a_2) = R(s,a_1,a_2) + \gamma \sum_{s' \in S} Pr(s' | s,a_1,a_2)\, V_{t-1}(s'))$$

For all $s \in S$,

$$V_t(s) = \max_{\rho \in \Pi(A1)} \min_{a2 \in A2} \sum_{a1 \in A1} \rho(a_1)\, Q_t(s,a_1,a_2)$$

t=t+1

} Until ($\max_s |V_{t-1}(s) - V_{t-2}(s)| \leq \varepsilon$)

Value Iteration Analysis

Stochastic Game

After D iterations: $V_{t-2}(s_0) \approx V_{t-1}(s_0) \approx V^*(s_0)$.

Each iteration:

$|S|^2 |A_1||A_2| + |S| \text{ poly}(|A_1|,|A_2|)$

So, an ε-optimal policy can be found in time $\text{poly}(|S|,|A_1|,|A_2|,1/\varepsilon,1/(1-\gamma))$.

"Alternating" Game, MDP

Optimal value polynomial-precision rational, there is an ε for which the policy is exactly optimal.

Overall: $\text{poly}(|S|,|A_1|,|A_2|,1/(1-\gamma))$.

Linear Programming

To express:

$V(s) = \max_a \left(R(s,a) + \gamma \sum_{s' \in S} \Pr(s' | s,a)\, V(s') \right)$ for $s \in S$.

$V(s)$ are vars. "Max" is smallest upper bound.

$\min_{V(s)} \sum_s V(s)$

$V(s) \geq R(s,a) + \gamma \sum_{s' \in S} \Pr(s' | s,a)\, V(s')$ for s,a

Can be implemented to run in polytime (in number of bits) (Khachiyan 79).

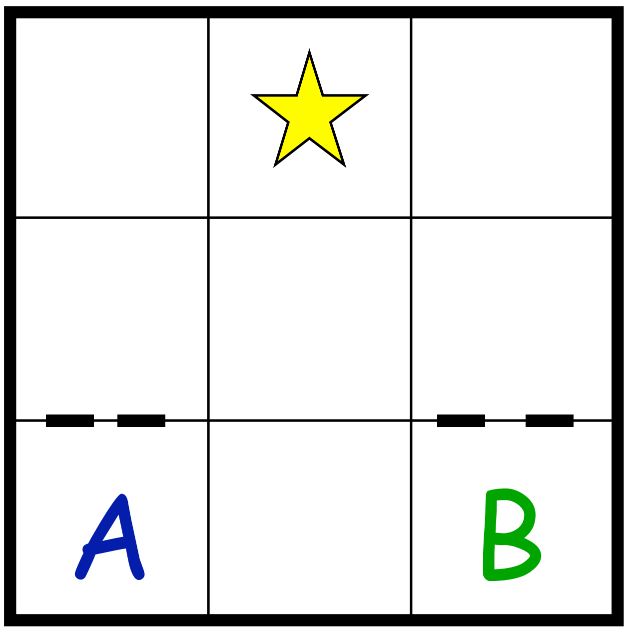
Beyond Zero-sum

Personal philosophy: Nothing in life is zero-sum.

- *War*: Can win a war and lose the peace.
- *Baseball*: Can win a game, but damage the sport.
- *Family dynamics*: Can win the argument, but wreck the relationship.

The art of the deal is finding the win-win.

Simple General-Sum Example



(Hu & Wellman 01)
Grid Game 2

U, D, R, L, X

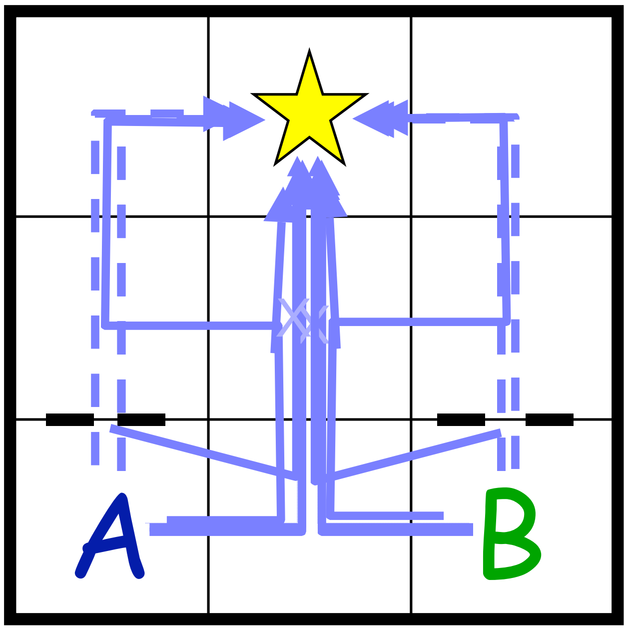
No move on collision

Semiwalls (50%)

-1 for step, -10 for collision, +100 for goal, 0 if back to initial config.

Both can get goal.

Strategies in Grid Game



Average reward:

- (32.3, 16.0), C, S
- (16.0, 32.3), S, C
- (-1.0 , -1.0), C, C
- (15.8, 15.8), S, S
- (15.9, 15.9), mix
- (25.7, 25.8), L, F
- (25.8, 25.7), F, L

What’s the Right Strategy?

Assume the worst (minimax): **side**.

Assume the best: **center**.

Assume other agent will adapt: **center**

Watch other agent and choose: opponent modeling.

Other ideas?

Nash Equilibrium

Choose strategies so no incentive to switch.

Joint best response.

Corresponds to (joint) minimax in zero-sum case.

GG2: A side, B center; A center, B side; symmetric.

How would we find such a thing? DP, simulation, search?

Failed Analogies to Zero-sum

In the general-sum case:

- optimal value function need not be unique.
- multiple incompatible policies for the same value function.
- value-iteration style algorithm doesn't converge.
- equilibrium can require randomization even in deterministic "alternating" case! (Zinkovich)
- no efficient algorithm known

Active area of research. One result: In repeated case, use threats to stabilize simple strategies. Builds on the zero-sum solutions (Littman & Stone 02).

Other Models to Know

- **RL**: Reinforcement learning, trying to find good behaviors *without* knowledge of transition and reward functions.
- **POMDPs**: Partially observable MDPs, agent doesn't know the current state (sensors).
  - exact incomputable, can approximate with VI
- **D-POMDPs**: Distributed POMDPs, set of agents, shared reward function, distributed knowledge and action!
  - way hard

Other Approaches to Know

- RTDP, reinforcement learning, neuro-dynamic programming : simulation + DP (Barto, Bradtke & Singh, 93; Bertsekas & Tsitsiklis 96)
- sparse sampling: simulation + search (Kearns, Mansour & Ng 99)
- envelope methods: search + simulation + DP (Tash & Russell, 94; Dean et al. 93)
- heuristic search: search + background knowledge (Hansen & Zilberstein, 99).
- policy search (Jaakkola et al., 93; Baird & Moore, 99; Meuleau et al., 99; etc.)
- hierarchical representations (Parr; Sutton, Precup & Singh; Dean & Lin; Dietterich 00; etc.)

Parting Thoughts

- Stochastic games (and variants) provide analytically tractible formal models for adversarial decision making
- Search, simulation, dynamic programming are the basic algorithmic ideas.
- More realistic problems can be solved building on these models and ideas.

CFP, MLJ special issue: “Special Issue on Learning and Computational Game Theory” with Amy Greenwald.