
Multiple Agents

Why can't we all just get along? (Rodney King)

Assumptions

Introduction

▷ Assumptions

Definitions

Games

Two-Agent
Zero-Sum Games

Partially Observable
Multiagent
Reasoning

- Each agent can act autonomously.
- Each agent has its own information about the world.
- Each agent can have its own utility function.
- A *mechanism* specifies how the actions of the agents lead to outcomes, e.g., rules of chess.
- A rational agent acts *strategically*; its actions are based on utility.
- Nature* can be defined as an agent with no utility and no strategy.

Definitions

Introduction

Assumptions

▷ Definitions

Games

Two-Agent
Zero-Sum Games

Partially Observable
Multiagent
Reasoning

- Agents can be *fully cooperative*; they share the same utility.
- Agents can be *fully competitive*; they have the opposite utility.
- In a *zero-sum game*, the sum of the utilities for the agents is zero for every outcome.
- *Game theory* studies what agents should do in a multi-agent setting.

Normal Form of a Game

Introduction

Games

▷ Normal Form

Simple Example

Extensive Form

Example 1

Example 2

Decision Networks

Example

Two-Agent

Zero-Sum Games

Partially Observable

Multiagent

Reasoning

The *strategic form* or *normal form* of a game contains:

- a finite set I of agents, $\{1, \dots, n\}$.
- a set of strategies for each agent.
- A *strategy profile* $s = (s_1, \dots, s_n)$ means that agent i follows strategy s_i .
- utility functions $u_i(s)$ gives the expected utility for agent i when all agents follow strategy profile s .
- An *outcome* is produced when all the agents follow a strategy profile.

Simple Example: Rock-Paper-Scissors

Introduction

Games

Normal Form

▷ Simple Example

Extensive Form

Example 1

Example 2

Decision Networks

Example

Two-Agent

Zero-Sum Games

Partially Observable

Multiagent

Reasoning

- In the game of rock-paper-scissors, there are two agents, each choosing one of three actions {rock, paper, scissors}.
- For each combinations of actions, a *payoff matrix* can be used to specify the utilities.

		Agent 2		
		rock	paper	scissors
Agent 1	rock	0, 0	-1, 1	1, -1
	paper	1, -1	0, 0	-1, 1
	scissors	-1, 1	1, -1	0, 0

Extensive Form of a Game

Introduction

Games

Normal Form

Simple Example

▷ Extensive Form

Example 1

Example 2

Decision Networks

Example

Two-Agent

Zero-Sum Games

Partially Observable

Multiagent

Reasoning

- The *extensive form of a game* or a *game tree* is a finite tree where the nodes are states and the edges are actions.
- Each internal node is *controlled* by an particular agent.
- Each edge out of a node controlled by agent i corresponds to an action for agent i .
- Each node controlled by nature has a probability distribution over its children.
- The leaves represent final outcomes and are labeled with a utility for each agent.

Perfect Information Example

Introduction

Games

Normal Form

Simple Example

Extensive Form

▷ Example 1

Example 2

Decision Networks

Example

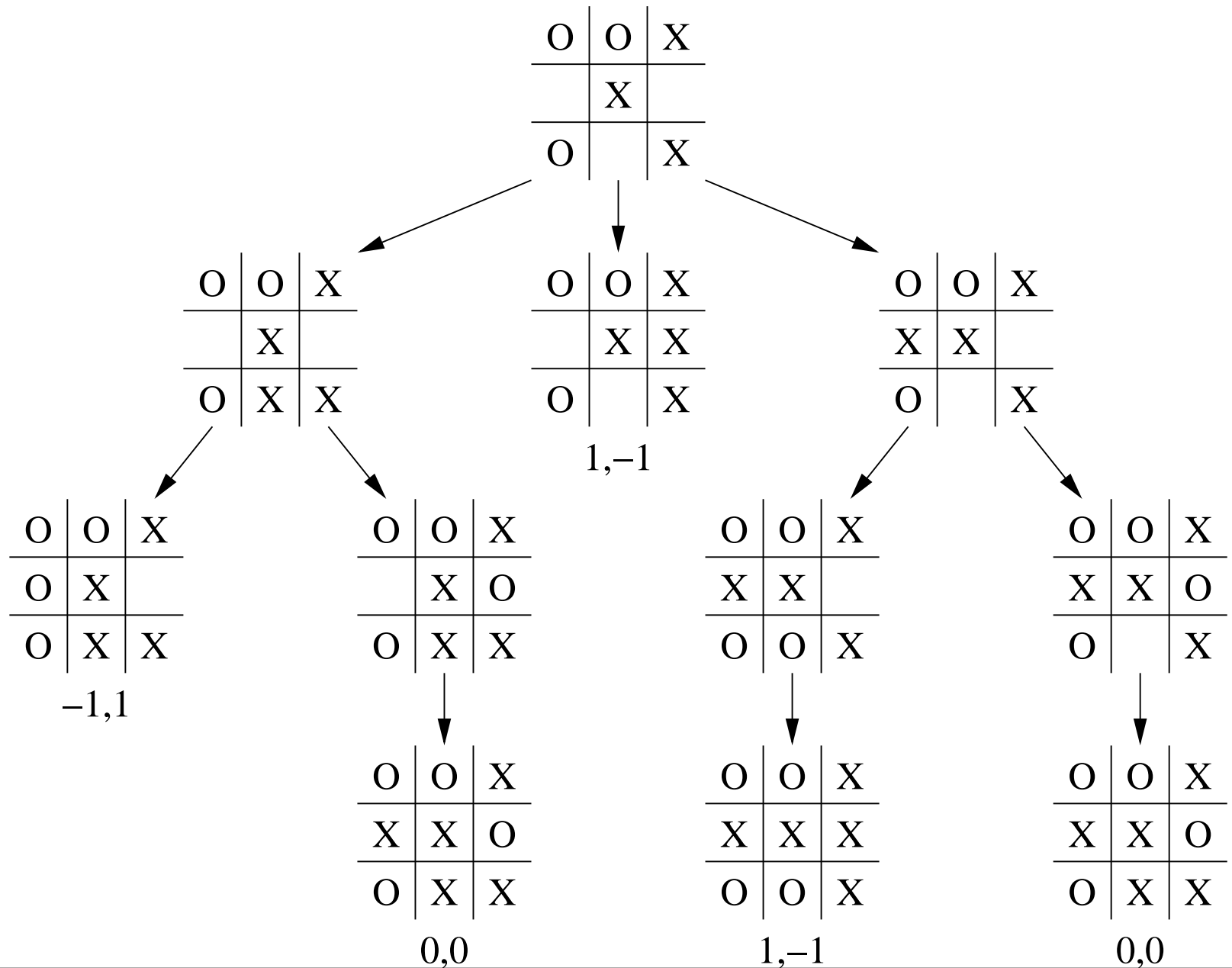
Two-Agent

Zero-Sum Games

Partially Observable

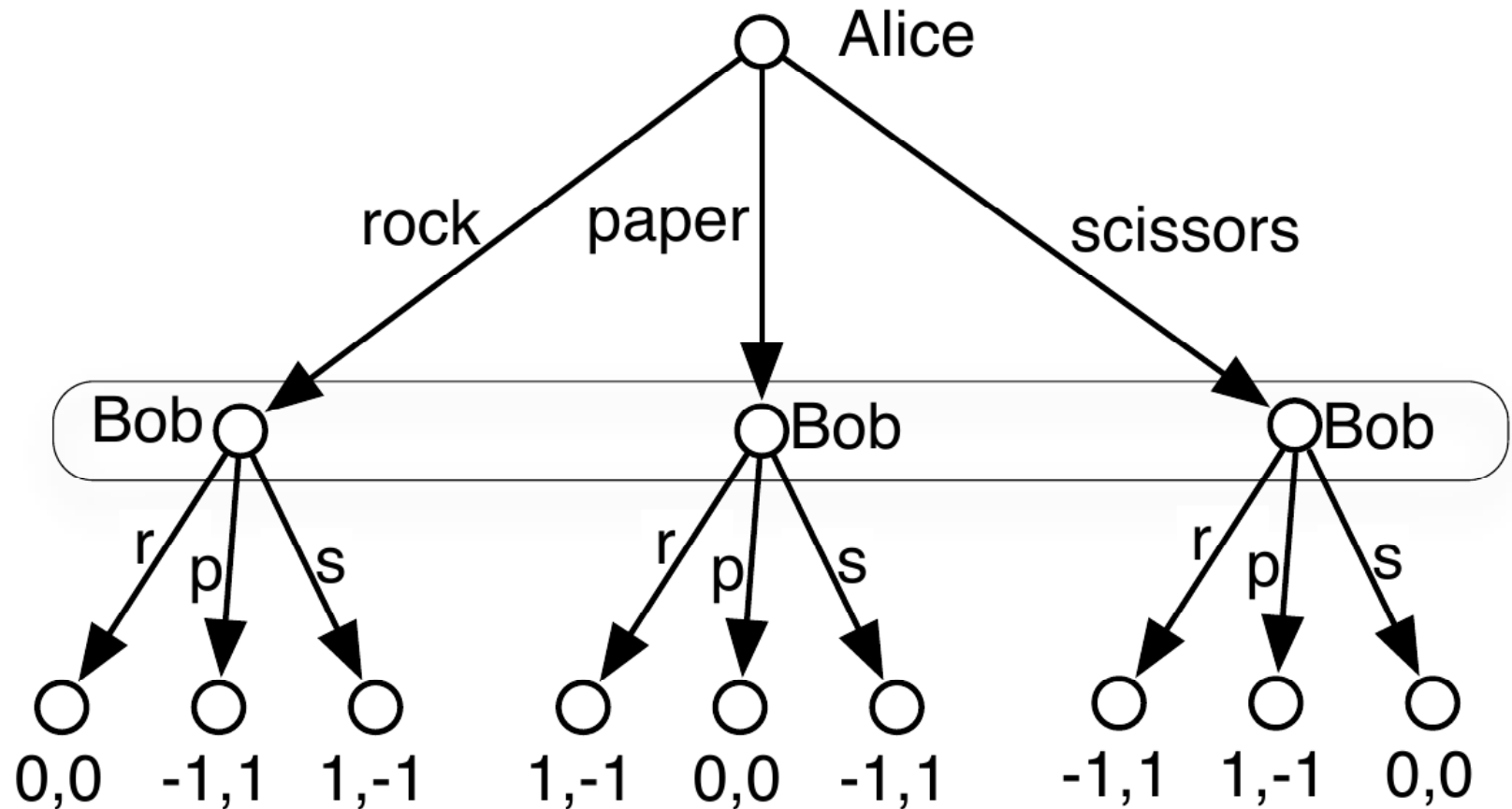
Multiagent

Reasoning



Imperfect Information Example

An agent cannot distinguish the nodes in an *information set*.



Multiagent Decision Networks

Introduction

Games

Normal Form

Simple Example

Extensive Form

Example 1

Example 2

▷ Decision
Networks

Example

Two-Agent

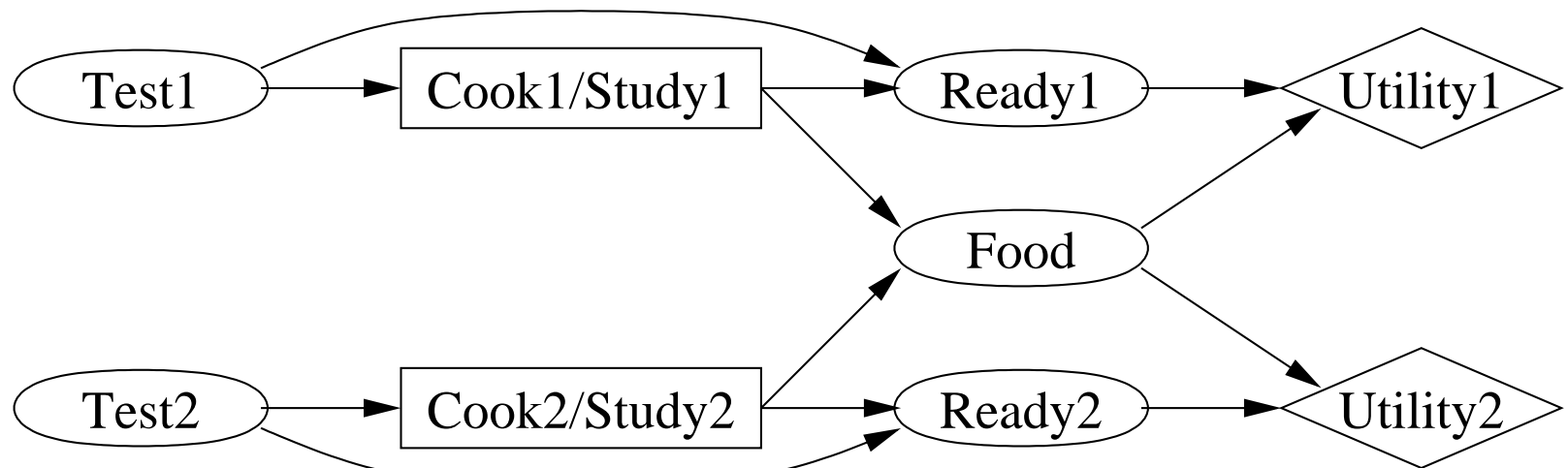
Zero-Sum Games

Partially Observable
Multiagent
Reasoning

- A multiagent decision network is a factored representation of a multiagent decision problem.
- Each decision node is labeled with an agent that makes the decision for the node.
- Each agent has a utility node.
- As with a decision network, the parents of a decision node are the information for making the decision.

Example Multiagent Decision Network

The scenario is that two roommates each might have a test to study for, choosing then to study or cook some food for both of them to eat.



Introduction

Games

Normal Form

Simple Example

Extensive Form

Example 1

Example 2

Decision Networks

▷ Example

Two-Agent

Zero-Sum Games

Partially Observable

Multiagent

Reasoning

Two-Agent Zero-Sum Games

Introduction

Games

Two-Agent
Zero-Sum Games

▷ Zero-Sum Games

Example

Minimax Procedure

Eval Functions

Example

Minimax with Eval

Idea of α - β Pruning

Example

α - β with Eval

Performance

Other Issues

Partially Observable

Multiagent

Reasoning

- A *two-agent zero-sum game* is when a positive reward for one agent is an equally negative reward for the other agent.
- The utility can be characterized by a single number that one agent is trying to maximize and the other agent is trying to minimize.
- Having a single value for a two-agent zero-sum game leads to a *minimax* strategy.
- Each node is either a MAX node, if controlled by the maximizing agent, or a MIN node if controlled by the minimizing agent.
- Treat agent currently in control (whose turn it is to move) as MAX.

Minimax Example

Introduction

Games

Two-Agent
Zero-Sum Games

Zero-Sum Games

▷ Example

Minimax Procedure

Eval Functions

Example

Minimax with Eval

Idea of α - β Pruning

Example

α - β with Eval

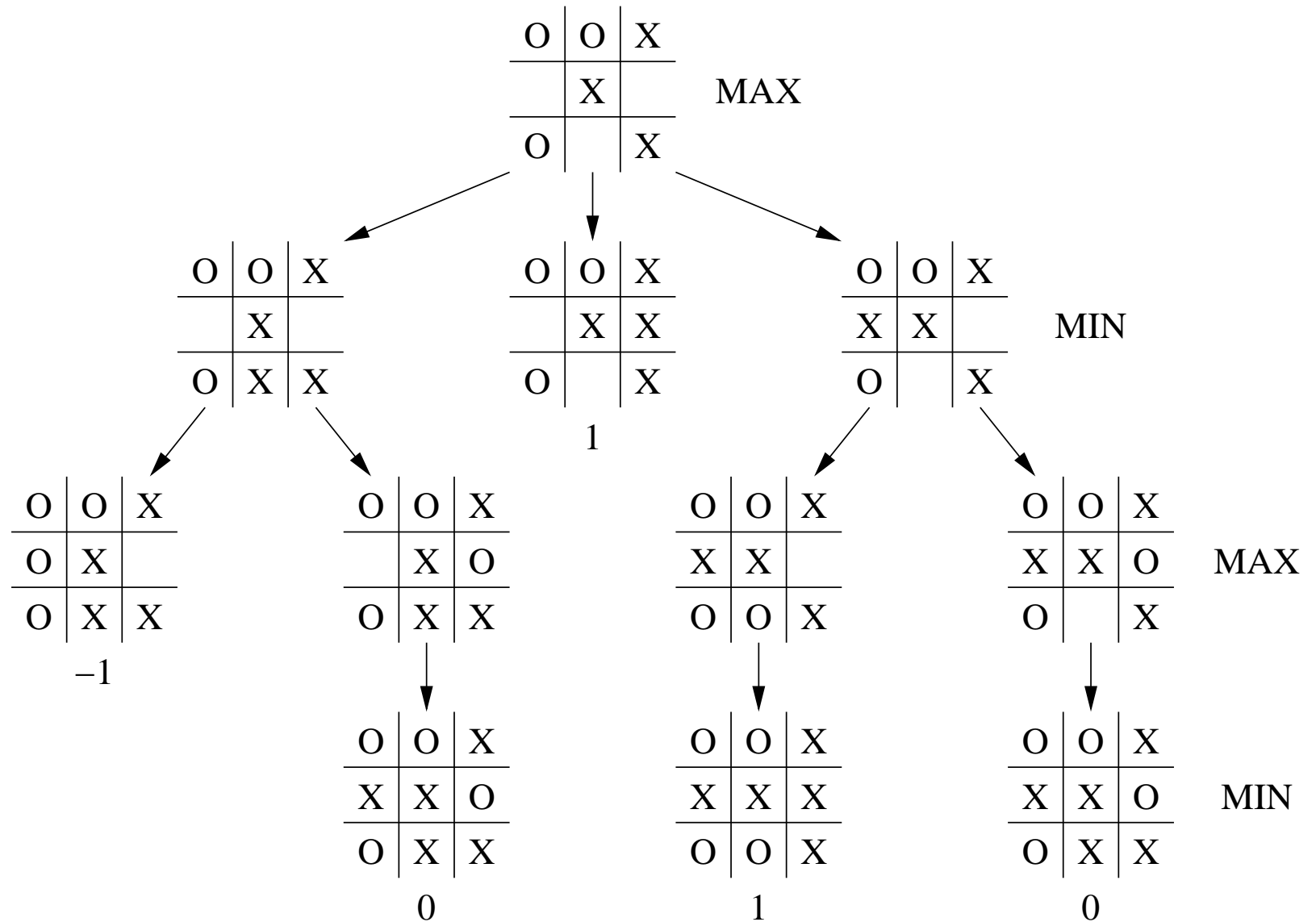
Performance

Other Issues

Partially Observable

Multiagent

Reasoning



Minimax Procedure

Introduction

Games

Two-Agent

Zero-Sum Games

Zero-Sum Games

Example

▷ Minimax
Procedure

Eval Functions

Example

Minimax with Eval

Idea of α - β Pruning

Example

α - β with Eval

Performance

Other Issues

Partially Observable

Multiagent

Reasoning

Procedure *Minimax*(N)

Inputs: a node in a game tree

if N is a leaf node, then

$$v \leftarrow \text{value of } N$$

else if N is a MAX node

$$v \leftarrow -\infty$$

for each child C of N

$$v \leftarrow \text{maximum of } v \text{ and } \textit{Minimax}(C)$$

else if N is a MIN node

$$v \leftarrow +\infty$$

for each child C of N

$$v \leftarrow \text{minimum of } v \text{ and } \textit{Minimax}(C)$$

return v

Evaluation Functions

Introduction

Games

Two-Agent
Zero-Sum Games

Zero-Sum Games

Example

Minimax Procedure

▷ Eval Functions

Example

Minimax with Eval

Idea of α - β Pruning

Example

α - β with Eval

Performance

Other Issues

Partially Observable

Multiagent

Reasoning

The game tree of many games is too large to search. An alternative is:

- Search as deeply as possible given time requirement.
- Use an *evaluation function* to estimate the values of nodes at the fringe.
- Use minimax to combine the values into an overall evaluation.

Example

- Introduction
- Games
- Two-Agent Zero-Sum Games
- Zero-Sum Games
- Example
- Minimax Procedure
- Eval Functions
- ▷ Example
- Minimax with Eval
- Idea of α - β Pruning
- Example
- α - β with Eval
- Performance
- Other Issues
- Partially Observable Multiagent Reasoning

evaluation = +10 if MAX has 3 in line

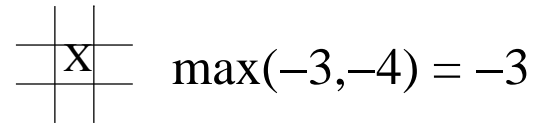
-10 if MIN has 3 in line

+1 for each potential 3 in line for MAX

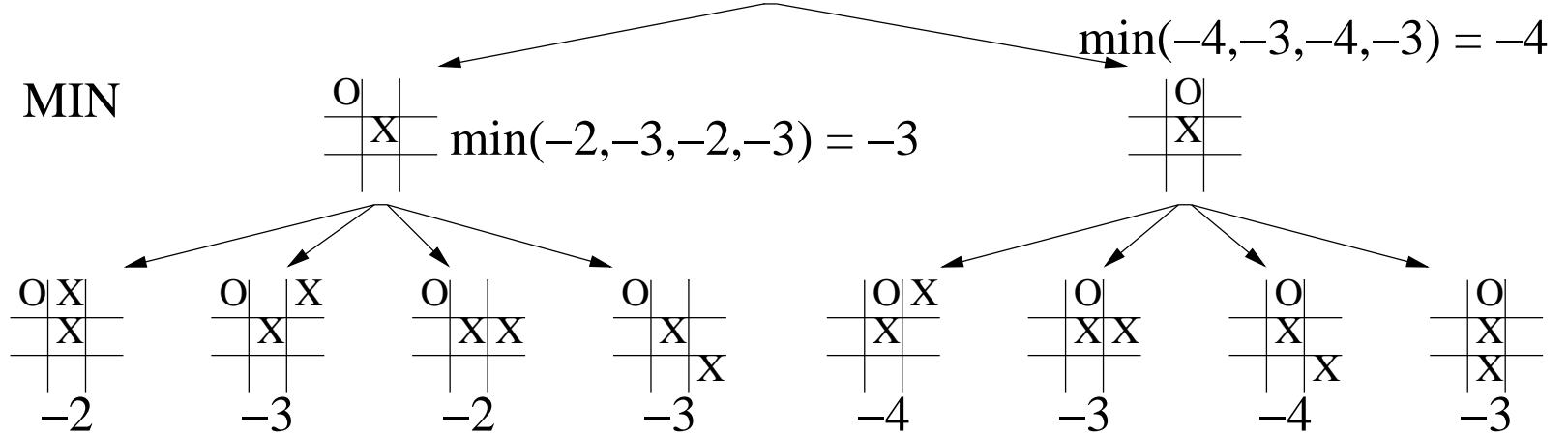
-1 for each potential 3 in line for MIN

MAX is O; MIN is X

MAX



MIN



Minimax Procedure with Evaluation Function

Introduction

Games

Two-Agent

Zero-Sum Games

Zero-Sum Games

Example

Minimax Procedure

Eval Functions

Example

▷ Minimax with
Eval

Idea of α - β Pruning

Example

α - β with Eval

Performance

Other Issues

Partially Observable

Multiagent

Reasoning

Procedure *Minimax*(N, d, f)

Inputs: game tree node, search depth, eval. fn.

if N is a leaf node, then return value of N

else if $d = 0$, then return $f(N)$

else if N is a MAX node

$v \leftarrow -\infty$

for each child C of N

$v \leftarrow$ maximum of v and *Minimax*($C, d-1, f$)

else if N is a MIN node

$v \leftarrow +\infty$

for each child C of N

$v \leftarrow$ minimum of v and *Minimax*($C, d-1, f$)

return v

Idea of Alpha-Beta Pruning

Introduction

Games

Two-Agent
Zero-Sum Games

Zero-Sum Games

Example

Minimax Procedure

Eval Functions

Example

Minimax with Eval

▷ Idea of α - β
Pruning

Example

α - β with Eval

Performance

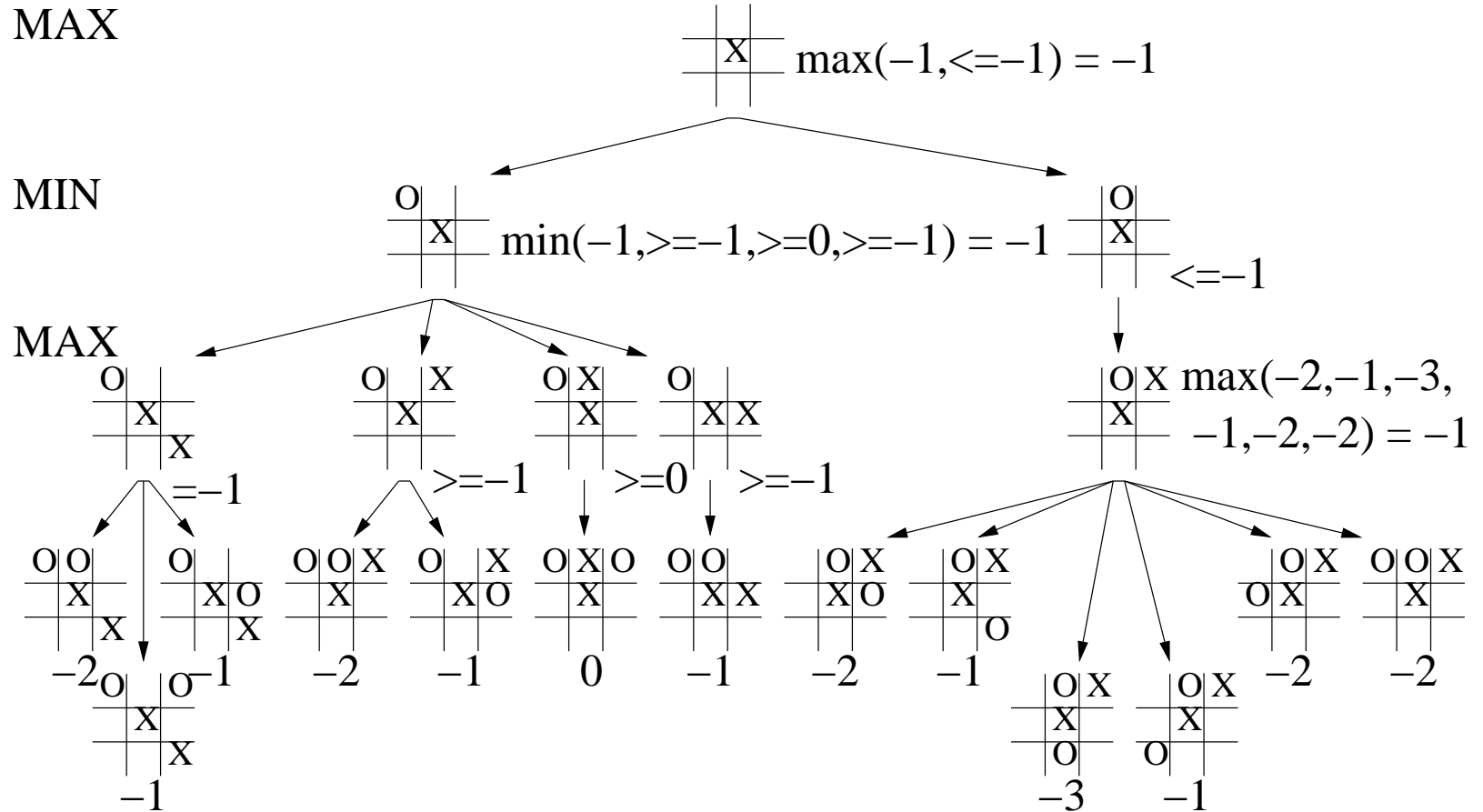
Other Issues

Partially Observable
Multiagent
Reasoning

- Alpha-beta pruning avoids search that won't change the minimax evaluation.
- Example: If MAX has a move with value 3, stop searching other moves known to be ≤ 3 .
- General Principle: Consider a node N .
 α = largest v in MAX ancestors of N .
 β = smallest v in MIN ancestors of N .
If $\alpha \geq \beta$, processing N cannot change eval.
- Proof: Let $v = N$'s minimax value.
 $\alpha \geq \beta$ implies $\alpha \geq v$ or $v \geq \beta$.
 $\alpha \geq v$ implies v can't change ancestor with α .
 $v \geq \beta$ implies v can't change ancestor with β .
This implies v cannot propagate to the top.

Example

- Introduction
- Games
- Two-Agent Zero-Sum Games
- Zero-Sum Games
- Example
- Minimax Procedure
- Eval Functions
- Example
- Minimax with Eval
- Idea of α - β Pruning
- ▷ Example
- α - β with Eval
- Performance
- Other Issues
- Partially Observable
- Multiagent Reasoning



Alpha-Beta Procedure with Eval Function

Introduction

Games

Two-Agent

Zero-Sum Games

Zero-Sum Games

Example

Minimax Procedure

Eval Functions

Example

Minimax with Eval

Idea of α - β Pruning

Example

▷ α - β with Eval

Performance

Other Issues

Partially Observable

Multiagent

Reasoning

Procedure *Alpha-Beta*(N, d, f, α, β)

if N is a leaf node, then return value of N

else if $d = 0$, then return $f(N)$

else if N is a MAX node

for each child C of N

$\alpha \leftarrow \max(\alpha, \text{Alpha-Beta}(C, d-1, f, \alpha, \beta))$

if $\alpha \geq \beta$ return α

else if N is a MIN node

for each child C of N

$\beta \leftarrow \min(\beta, \text{Alpha-Beta}(C, d-1, f, \alpha, \beta))$

if $\alpha \geq \beta$ return β

if N is a MAX node, then return α , else return β

Performance of Minimax and Alpha-Beta

Introduction

Games

Two-Agent
Zero-Sum Games

Zero-Sum Games

Example

Minimax Procedure

Eval Functions

Example

Minimax with Eval

Idea of α - β Pruning

Example

α - β with Eval

▷ Performance

Other Issues

Partially Observable

Multiagent

Reasoning

- b = branching factor
 d = depth of search
- Minimax visits every state from level 0 to d .
$$\sum_{i=0}^d b^i = \frac{b^{d+1}-1}{b-1} \in O(b^d)$$
- Alpha-Beta visits as few as $\Omega(b^{d/2})$ states.
Depends on a good ordering of children.
Actual programs approach the minimum bound.
- Alpha-beta pruning allows programs to look ahead nearly twice as many moves as minimax.

Other Issues

Introduction

Games

Two-Agent

Zero-Sum Games

Zero-Sum Games

Example

Minimax Procedure

Eval Functions

Example

Minimax with Eval

Idea of α - β Pruning

Example

α - β with Eval

Performance

▷ Other Issues

Partially Observable

Multiagent

Reasoning

- Horizon problem
- Quiescence
- Data bases of openings and end games
- Games of chance
 - *Expectiminimax* extends minimax to include chance nodes in the game tree.
 - Example: Roll of dice in Backgammon
 - Evaluate chance node by summing probability times child's value.

Example of Partially Observable Multiagents

What should the kicker and goalie do?



		goalie	
		left	right
kicker	left	0.6	0.2
	right	0.3	0.9

Introduction

Games

Two-Agent
Zero-Sum Games

Partially Observable
Multiagent
Reasoning

▷ Example

Example

Strategy Profiles

Nash Equilibriums

Multiple NEs

Prisoners' Dilemma

Tragedy Commons

Computing NEs

Learning

Example of Partially Observable Multiagents

Introduction

Games

Two-Agent
Zero-Sum Games

Partially Observable
Multiagent
Reasoning

Example

▷ Example

Strategy Profiles

Nash Equilibriums

Multiple NEs

Prisoners' Dilemma

Tragedy Commons

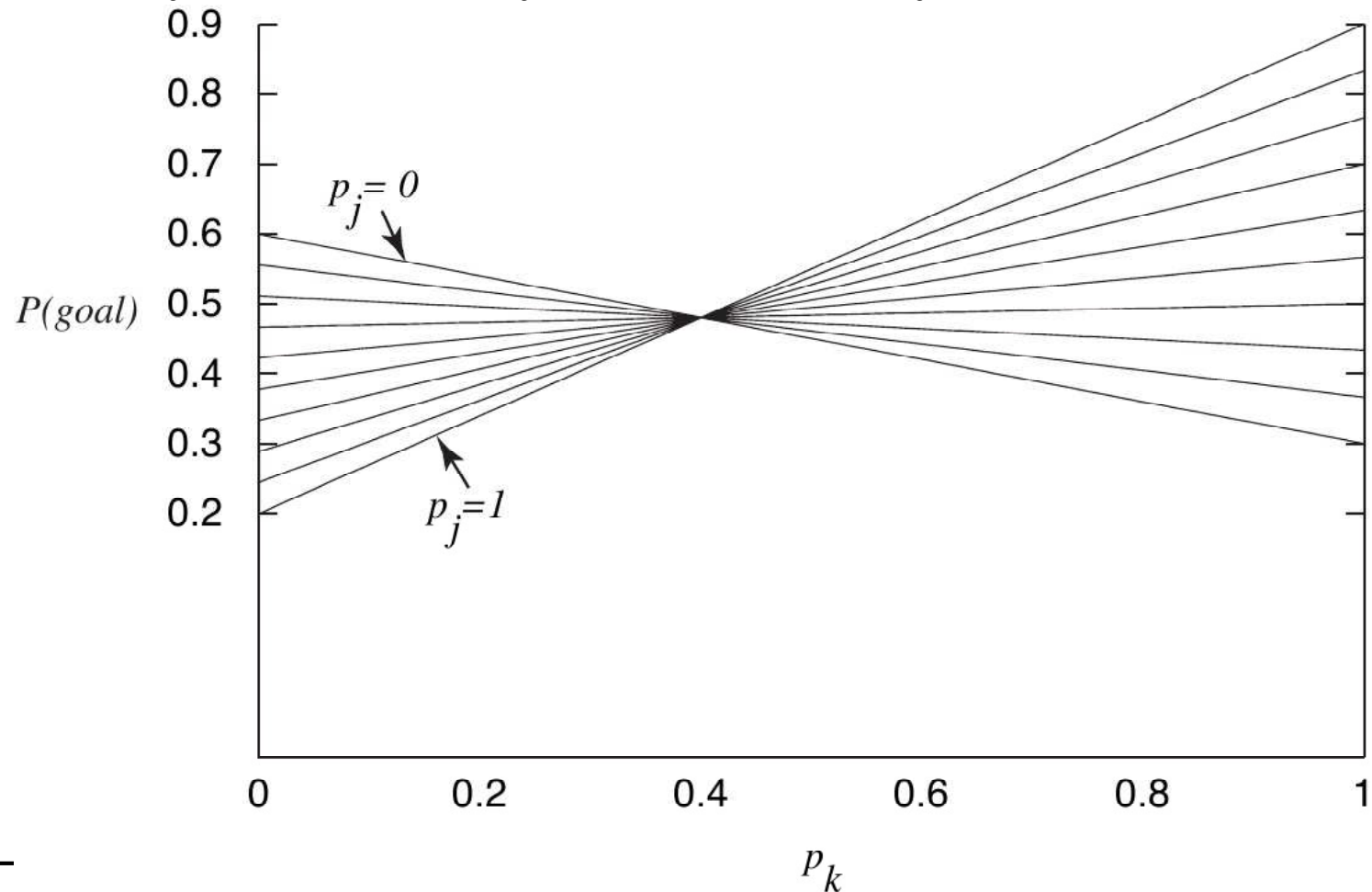
Computing NEs

Learning

p_k = kicker kicks right, p_j = goalie jumps right

$P(goal) =$

$$0.9p_k p_j + 0.3p_k(1-p_j) + 0.2(1-p_k)p_j + 0.6(1-p_k)(1-p_j)$$



Strategy Profiles

Introduction

Games

Two-Agent
Zero-Sum Games

Partially Observable
Multiagent
Reasoning

Example

Example

▷ Strategy Profiles

Nash Equilibriums

Multiple NEs

Prisoners' Dilemma

Tragedy Commons

Computing NEs

Learning

- Assume a general n -player game,
- A *strategy* for an agent is a probability distribution over the actions for this agent.
- A strategy profile is an assignment of a strategy to each agent.
- A *strategy profile* s has a utility for each agent. Let $utility(s, i)$ be the utility of strategy profile s for agent i .
- If s is a strategy profile:
 s_i is the strategy of agent i in s ,
 s_{-i} is the set of strategies of the other agents.
Thus s is $s_i s_{-i}$

Nash Equilibriums

Introduction

Games

Two-Agent
Zero-Sum Games

Partially Observable
Multiagent
Reasoning

Example

Example

Strategy Profiles

▷ Nash
Equilibriums

Multiple NEs

Prisoners' Dilemma

Tragedy Commons

Computing NEs

Learning

- s_i is a best response to s_{-i} if for all other strategies s'_i for agent i :

$$utility(s_i s_{-i}, i) \geq utility(s'_i s_{-i}, i)$$

- A strategy profile s is a *Nash equilibrium* if, for each agent i , strategy s_i is a best response to s_{-i} . A Nash equilibrium is a strategy profile such that no agent can be better by changing its strategy.
- Theorem [Nash, 1950] Every finite game has at least one Nash equilibrium.
- In the soccer example, $p_k = 0.4$ and $p_j = 0.3$ is a Nash equilibrium.

Multiple Nash Equilibriums

Introduction

Games

Two-Agent
Zero-Sum Games

Partially Observable
Multiagent
Reasoning

Example

Example

Strategy Profiles

Nash Equilibriums

▷ Multiple NEs

Prisoners' Dilemma

Tragedy Commons

Computing NEs

Learning

- Hawk-Dove game ($D > R$):

		Agent 2	
		dove	hawk
Agent 1	dove	$R/2, R/2$	$0, R$
	hawk	$R, 0$	$-D, -D$

- Do what activity together?

		Agent 2	
		shopping	football
Agent 1	shopping	$2, 1$	$0, 0$
	football	$0, 0$	$1, 2$

Prisoners' Dilemma

Introduction

Games

Two-Agent
Zero-Sum Games

Partially Observable
Multiagent
Reasoning

Example

Example

Strategy Profiles

Nash Equilibriums

Multiple NEs

▷ Prisoners'
Dilemma

Tragedy Commons

Computing NEs

Learning

Two prisoners for a crime can either stay silent or talk, providing evidence against the other prisoner. The payoff is the number of years in prison.

		Prisoner 2	
		silent	talk
Prisoner 1	silent	-1, -1	-3, 0
	talk	0, 3	-2, -2

Tragedy of the Commons

Introduction

Games

Two-Agent
Zero-Sum Games

Partially Observable
Multiagent
Reasoning

Example

Example

Strategy Profiles

Nash Equilibriums

Multiple NEs

Prisoners' Dilemma

▷ Tragedy
Commons

Computing NEs

Learning

- There are 100 agents.
- There is a common environment shared by the agents.
- Each agent can choose do nothing with a 0 payoff, or do a selfish action that has a +10 payoff for the agent but includes a -1 payoff for all the other agents.
- If only one agent does a selfish action, that agent has a +9 payoff.
- If every agent does a selfish action, each agent gets a -90 payoff.

Computing Nash Equilibria

Introduction

Games

Two-Agent
Zero-Sum Games

Partially Observable
Multiagent
Reasoning

Example

Example

Strategy Profiles

Nash Equilibria

Multiple NEs

Prisoners' Dilemma

Tragedy Commons

▷ Computing NEs

Learning

To compute Nash equilibria:

- Eliminate dominated strategies.
- Determine which actions will have non-zero probabilities. This is the support set.
- Determine the probability for the actions in the support set.

		Agent 2		
		x	y	z
Agent 1	a	3, 5	5, 1	1, 2
	b	1, 1	2, 9	6, 4
	c	2, 6	4, 7	0, 8

Learning

Introduction

Games

Two-Agent
Zero-Sum Games

Partially Observable
Multiagent
Reasoning

Example

Example

Strategy Profiles

Nash Equilibriums

Multiple NEs

Prisoners' Dilemma

Tragedy Commons

Computing NEs

▷ Learning

- Issues: multiple equilibria, unknown utilities and outcomes for other agents
- Approach: Agent learns what is good for itself.
- Repeat:
 - Select action a using distribution P .
 - Do a and observe *payoff*.
 - Update $Q[a] \leftarrow Q[a] + \alpha(\textit{payoff} - Q[a])$.
 - Find action a_{best} that maximizes Q .
 - Increase $P[a_{best}]$ and normalize P .
- If other agents have fixed, possibly stochastic, strategies, this algorithm converges near a best response (as long as all actions are tried occasionally).