Chapter 10: Multiagent systems

DIT410/TIN172 Artificial Intelligence

Peter Ljunglöf
modifed from slides by Poole & Mackworth

8 May, 2015

Outline

Outline

Multiple agents

Let's consider multiple agents, where:

- the agents select actions autonomously
- each agent has its own information state
  - they can have different information (even conflicting)
- the outcome depends on the actions of all agents
- each agent has its own utility function (that depends on the total outcome)

Types of agents

There are two extremes of multiagent systems:

Cooperative: The agents share the same utility function
Example: Automatic trucks in a warehouse

Competetive: When one agent wins all other agents lose
A common special case is when $\sum_a u_a(o) = 0$ for any outcome o. This is called a *zero-sum game*.
Example: Most board games

Most multiagent systems are between these two extremes.

Example: Long-distance bike races are usually both *cooperative* (the bikers usually form clusters where they take turns in leading the group), and *competetive* (only one of them can win in the end).

Outline

Outline

Normal-form games

This is the most basic representation of a game. It consists of:

- a finite set of agents, $I = \{1, \ldots, n\}$
- a set of actions A_i for each agent $i \in I$
  - an action profile $\sigma = \langle a_1, \ldots, a_n \rangle$, means that every agent $i \in I$ carries out action a_i
- a utility function $u(\sigma, i)$ which gives the expected utility for agent i when all agents follow the action profile σ

Rock-papers-scissors

Here's a *payoff matrix* of the game "rock-paper-scissors":

	rock	*paper*	*scissors*
rock	0, 0	−1, 1	1, −1
paper	1, −1	0, 0	−1, 1
scissors	−1, 1	1, −1	0, 0

Note: This is a zero-gum game.

Prisoner's dilemma

Two criminals are arrested for a crime, and each of them is given the following opportunities:

- betray the other one by saying that (s)he did it
- remain silent

If both betray each other they will serve 2 years in prison, but if both remain silent they will only serve 1 year.

However, if one betrays and the other says nothing, the silent person will werve 3 years while the other one will be set free.

Prisoner's dilemma, payoff matrix

Here is the payoff matrix for the prisoner's dilemma:

	B stays silent	*B betrays A*
A stays silent	−1, −1	−3, 0
A betrays B	0, −3	−2, −2

Assuming that both criminals are independent and rational agents, what will be their actions?

Prisoner's dilemma, payoff matrix

Here is the payoff matrix for the prisoner's dilemma:

	B stays silent	*B betrays A*
A stays silent	−1, −1	−3, 0
A betrays B	0, −3	−2, −2

Assuming that both criminals are independent and rational agents, what will be their actions?

- if B stays silent, then it's better for A to betray
- if B betrays, then it's better for A to betray

Prisoner's dilemma, payoff matrix

Here is the payoff matrix for the prisoner's dilemma:

	B stays silent	*B betrays A*
A stays silent	−1, −1	−3, 0
A betrays B	0, −3	−2, −2

Assuming that both criminals are independent and rational agents, what will be their actions?

- if B stays silent, then it's better for A to betray
- if B betrays, then it's better for A to betray

Therefore, the rational action would be to betray the other one.

Prisoner's dilemma, payoff matrix

Here is the payoff matrix for the prisoner's dilemma:

	B stays silent	*B betrays A*
A stays silent	−1, −1	−3, 0
A betrays B	0, −3	−2, −2

Assuming that both criminals are independent and rational agents, what will be their actions?

- if B stays silent, then it's better for A to betray
- if B betrays, then it's better for A to betray

Therefore, the rational action would be to betray the other one.

So, both will get the 2 year sentence since both are rational agents and will come to the same decision.

Outline

Extensive form

The normal form of a game can only represent games that are momentaneous, i.e., that have no representation of time.

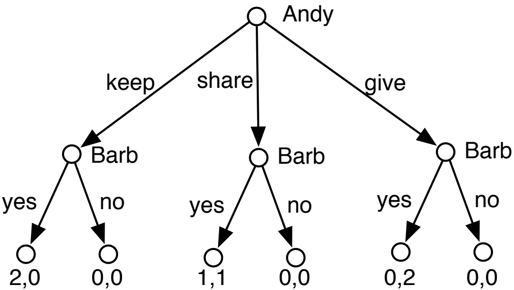
- With the *extensive form* of a game we can represent the unfolding of a game through time.
- For now we assume that the game is fully observable.

A *perfect information game* in extensive form is also known as a *game tree*.

Extensive form for perfect information games

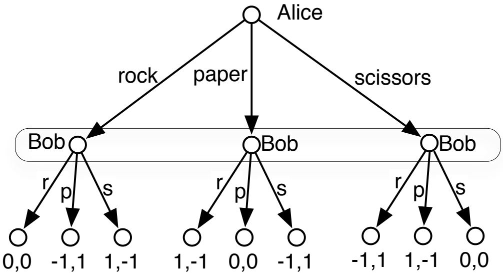
A *perfect information game* in extensive form is a finite tree where the nodes are states and the arcs correspond to actions by the agents, in particular:

- Each internal node is labeled with an agent.
- Each outgoing arc corresponds to an action for its agent.
- The leaves represent final outcomes and are labeled with utilities.



Imperfect information games

Rock-paper-scissors is not a perfect information game:



In this example, Bob cannot know which of the nodes he is in, since he doesn't know which action Alice will perform.

Outline

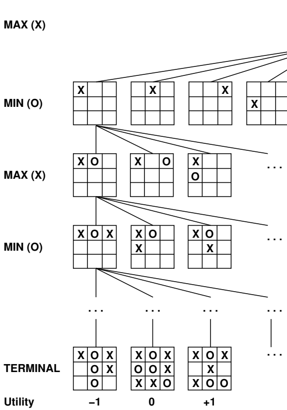
1. *Multiple agents (10.1)*
2. *Games (10.2)*
   - Normal form of games (10.2.1)
   - Extensive form of games (10.2.2)
3. *Zero-sum games (10.3)*

Perfect information games

- Perfect information games are solvable in a manner similar to fully observable single-agent systems. We can either do it backward using dynamic programming or forward using search.
- If two agents are competing so that a positive reward for one is a negative reward for the other agent, we have a two-agent *zero-sum game*.
- The value of a game zero-sum game can be characterized by a single number that one agent is trying to maximize and the other agent is trying to minimize.
- This leads to a *minimax* strategy:
  - A node is either a MAX node (if it is controlled by the maximising agent),
  - or is a MIN node (if it is controlled by the minimising agent).

Minimax search: tic-tac-toe

Here is an example of a minimax search tree for tic-tac-toe:

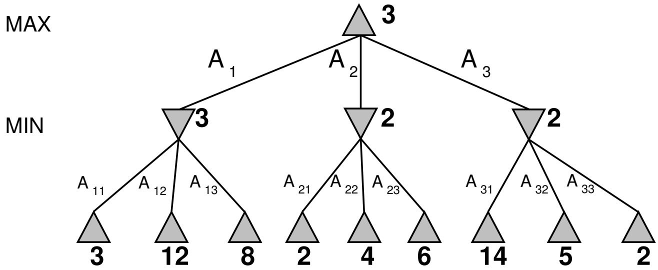


$\alpha - \beta$ *pruning*

- Suppose, we reach a node t in the game tree which has leaves t_1, $\ldots$, t_k corresponding to moves of player MIN.
- Let α be the best value of a position on a path from the root node to t.
- Then, if any of the leaves evaluates to $u(t_i) \leq \alpha$, we can discard t, because any further evaluation will not improve the value of t.
- Analogously, define β values for evaluating response moves of MAX.

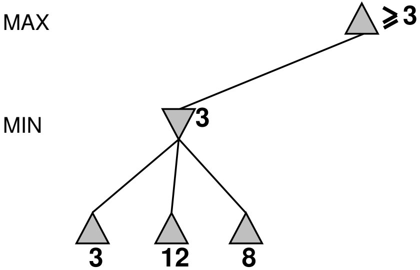
Minimax example

Assume the following minimax search tree:



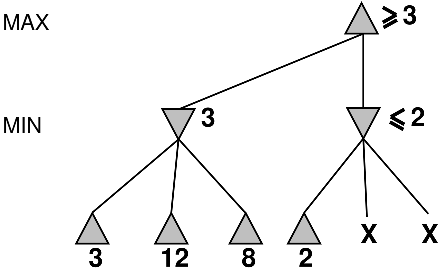
Minimax example, with $\alpha - \beta$ pruning

This is how $\alpha - \beta$ pruning might work:



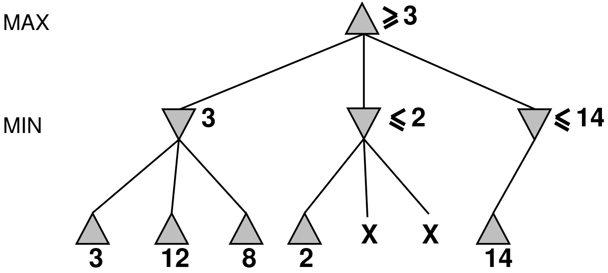
Minimax example, with $\alpha - \beta$ pruning

This is how $\alpha - \beta$ pruning might work:



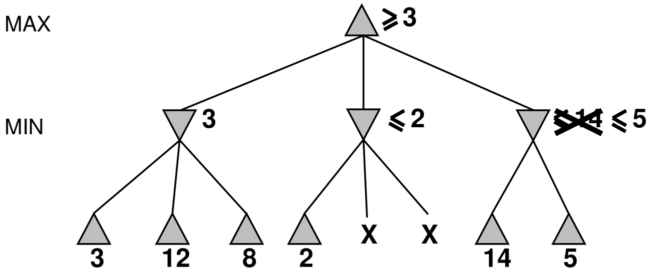
Minimax example, with $\alpha - \beta$ pruning

This is how $\alpha - \beta$ pruning might work:



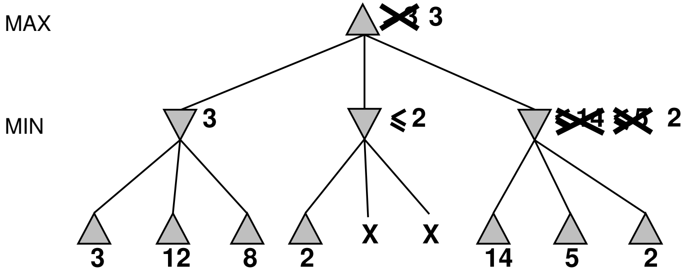
Minimax example, with $\alpha - \beta$ pruning

This is how $\alpha - \beta$ pruning might work:



Minimax example, with $\alpha - \beta$ pruning

This is how $\alpha - \beta$ pruning might work:



How efficient is $\alpha - \beta$ pruning?

The amount of pruning provided by the $\alpha - \beta$ algorithm depends on the ordering of the children of each node.

- It works best if a highest-valued child of a MAX node is selected first and if a lowest-valued child of a MIN node is returned first.
- In implementations of real games, much of the effort is made to try to ensure this outcome.

Minimax and real games

Most real games are too big to carry out minimax search, even with α-β pruning.

- For these games, instead of only stopping at leaf nodes, it is possible to stop at any node.
- The value returned at the node where the algorithm stops is an estimate of the value for this node.
- The function used to estimate the value is an evaluation function.
- Much work goes into finding good evaluation functions.
- There is a trade-off between the amount of computation required to compute the evaluation function and the size of the search space that can be explored in any given time.