Artificial Intelligence

Multiagent Systems (AIMA 18.1, MAS 1-2)

Christopher Simpkins

Kennesaw State University

Multiagent Systems

- Multiagent environments
- (Noncooperative) game theory

Multiagent Environments

Dealing with multiagent environments:

- Single agent with other agents considered part of the environment
- Multiple actors, but one decision maker controlling all actors
- Multiple actors, each of which makes its own decisions

One Decsion Maker

Rests on **benevolent agent assumption**: agents will do what they are told.

Action synchronization:

- Sumultaneous/joint actions
- Mutually exclusive actions executed at different times
- Sequential actions A before B when A's postconditions are B's preconditions

Architectures:

- **Multieffector planning**: multiple concurrently acting effectors
- **Multibody planning**: physically decoupled effectors
  - Centralized planning: sensor information from each body is pooled
  - Decentralized planning: plan execution partially or fully decoupled, explicit communication to share information when possible (e.g., back in comms range)

Multiple Decision Makers

All agents, a.k.a. **counterparts**, make decisions. Two categories:

1. Agents have a **common goal**.
   - Agents **coordinate** to accomplish common goal, e.g., workers in a company, players on a team
2. Agents have differeent goals.
   - Goals may be unrelated, diametrically opposed (e.g., zer-sum games), or anythin in between.

Game Theory

Multiple decision makers pursuing their own preferences.

- An agent must take into account preferences of other agents.
- These other agents also take into account preferences of other agents, and so on.

Game theory: the theory of **strategic decision making**.

- *Strategic* because decisions must take into account how other players act.
- Strategic aspect distinguishes game theory from decision theory.

Just as decision theory provides the theoretical foundation for single-agent decision making, game theory provides the theoretical foundation for multiagent decision making.

Game Theory in AI

In AI, game theory can be used in two main ways:

1. **Agent design**: analyzing decisions in multiagent environments.
   - Enumerate possible decisions
   - Compute expected utility of each decision
2. **Mechanism design**: deisgn multiagent environment in such a way that when agents act selfishly to maximize utility, it has the effect of maximizing some collective good.
   - Example: protocols for Internet routers
   - Example: criminal legal system

Cooperative vs Noncooperative Games

Two broad categories of game models:

1. Cooperative games:
   - Binding agreement between agents ensuring cooperation.
2. Noncooperative games:
   - Not necessarily competitive.
   - Simply means no binding agreement to cooperate.

Often mix models:

- Package company: centralized planning for routes and trucks, decentralized execution via autonomous decisions of drivers and pilots responsing individually to real-time conditions.
- Company: individual **incentives** for employees designed to bring their behavior into alignment with company's goals.

Multiagent Planning

Use generic term **actor** for effectors, bodies and agents. Need to define:

- transition models
- correct plans
- planning algorithms

Agents must take into account the way in which their own actions interact with the actions of other agents.

- Example: Agent A's action might clobber the precoditions of the action Agent B planned to execute.

Concurrency Models

Three approaches to dealing with concurrency:

1. **Interleaved execution**. Given Agents A and B with plans $[a_1, a_2]$ and $[b_1, b_2]$ there are six ways to execute them concurrently:

$$[a_1, a_2, b_1, b_2]$$
$$[b_1, b_2, a_1, a_2]$$
$$[a_1, b_1, a_2, b_2]$$
$$[b_1, a_1, b_2, a_2]$$
$$[a_1, b_1, b_2, a_2]$$
$$[b_1, a_1, a_2, b_2]$$

   - Plans must be correct with respect to all possible interleavings.
   - Popular in OS concurrency on single CPU.
   - Does not model true concurrency, i.e., simultaneous action.
   - Number of sequences grows exponentially with numbers of agents and actions.
2. **True concurrency**: leave plans *partially ordered* and don't create a fully serialized ordering.
3. **Synchronization**: global clock, each action takes one time step, joint actions execute simultaneously, no-ops in case "waiting" is needed.

Multiagent Transition Models

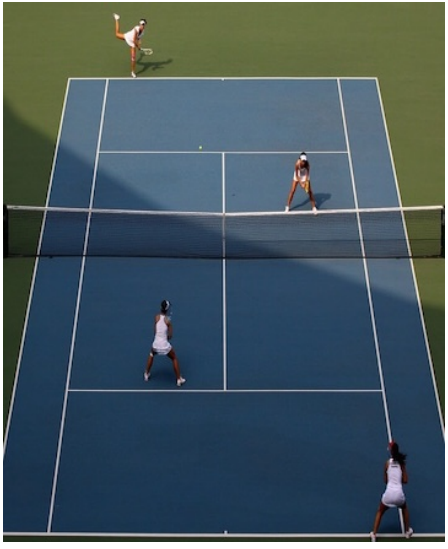
Deterministic single-agent case: `RESULT(s, a)` with b choices for the action.

Multiagent case: with n actors, joint action $\langle a_1, \ldots, a_n \rangle$ where a_i is actoin taken by ith actor.

- Must describe transition model for b^n joint actions.
- Joint planning problem with branching factor of b^n.

Standard solution: pretend problems are decoupled and fix the interactions as they arise, i.e., writing action schemas as if actors acted independently. Let's see how this works with an example ...

Example: Doubles Tennis Planning



$Actors(A, B)$
$Init(At(A, LeftBaseline) \wedge At(B, RightNet) \wedge$
$\quad Approaching(Ball, RightBaseline) \wedge Partner(A, B) \wedge Partner(B, A)$
$Goal(Returned(Ball) \wedge (At(x, RightNet) \vee At(x, LeftNet))$
$Action(Hit(actor, Ball),$
$\quad$ PRECOND:$Approaching(Ball, loc) \wedge At(actor, loc)$
$\quad$ EFFECT:$Returned(Ball))$
$Action(Go(actor, to),$
$\quad$ PRECOND:$At(actor, loc) \wedge to \neq loc,$
$\quad$ EFFECT:$At(actor, to) \wedge \neg At(actor, loc))$

Here's a 2-step **joint plan** that works:

$$\text{PLAN 1 }: A : [Go(A, RightBaseline), Hit(A, Ball)]$$
$$B : [NoOp(B), NoOp(B)].$$

Concurrent Action Constraints

What if both players try to Hit at the same time? Add a **concurrent action constraint** to prevent:

$$
\begin{aligned}
&Action(Hit(actor, Ball),\\
&\quad \text{CONCURRENT:} \forall b, b \neq Actor \implies \neg Hit(b, Ball)\\
&\quad \text{PRECOND:} Approaching(Ball, loc) \wedge At(actor, loc)\\
&\quad \text{EFFECT:} Returned(Ball))
\end{aligned}
$$

Some tasks require concurrent action, e.g., carrying a large cooler:

$$
\begin{aligned}
&Action(Carry(actor, cooler, here, there),\\
&\quad \text{CONCURRENT:} \exists b, b \neq Actor \wedge Carry(b, cooler, here, there)\\
&\quad \text{PRECOND:} At(actor, here) \wedge At(cooler, here) \wedge Cooler(cooler)\\
&\quad \text{EFFECT:} At(actor, there) \wedge At(cooler, there) \wedge \neg At(actor, here) \wedge \neg At(cooler, here)
\end{aligned}
$$

Cooperation and Coordination

There can be multiple joint plans that acheive the goal:

PLAN 1 : $A : [Go(A, RightBaseline), Hit(A, Ball)]$ PLAN 2 : $A : [Go(A, LeftNet), NoOp(A)]$
$B : [NoOp(B), NoOp(B)]$. $B : [Go(B, RightBaseline), Hit(B, Ball)]$

If both agents choose Plan 1 or 2, goal is acheived. If one agent chooses 1 and other chooses 2, goal is not acheived. How to coordinate?

- **Convention**: any constraint on selection of joint plans.
  - Example: "stick to your side of the court."
  - Widespread conventions are sometimes called **social laws**.
- **Communication**
  - Example: shout "Mine!" or "Yours!"
  - Can be implicit, based on observation – **plan recognition**.

Games with a Single Move: Normal Form Games

All players take action that are chosen simultaneously with no knowledge of other players' choices and the result of the game is based on the profile of actions that are selected in this way.

Normal form game defined by:

- **Players** or agents who will be making decisions. Two-player games have received the most attention, although n-player games for $n > 2$ are also common. We give players capitalized names, like Ali and Bo or O and E.
- **Actions** that the players can choose. We will give actions lowercase names, like one or testify. The players may or may not have the same set of actions available.
- **Payoff function** that gives the utility to each player for each combination of actions by all the players. For two-player games, the payoff function for a player can be represented by a matrix in which there is a row for each possible action of one player, and a column for each possible choice of the other player: a chosen row and a chosen column define a matrix cell, which is labeled with the payoff for the relevant player. In the two- player case, it is conventional to combine the two matrices into a single **payoff matrix**, in which each cell is labeled with payoffs for both players.

Example: Two-Finger Morra

Here is the payoff matrix for **two-finger Morra**, a game in which each player displays one or two fingers.

	O: one	O: two
E: one	E = +2, O = -2	E = -3, O = +3
E: two	E = -3, O = +3	E = +4, O = -4

- E is the **row player**, O is the **column player**.
- Each cell shows the payoffs given the players' actions.

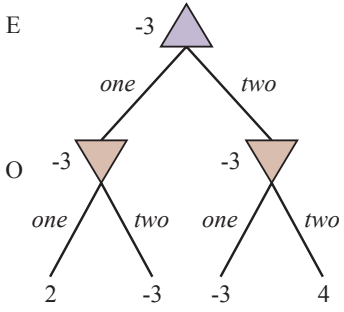
Solution Concepts

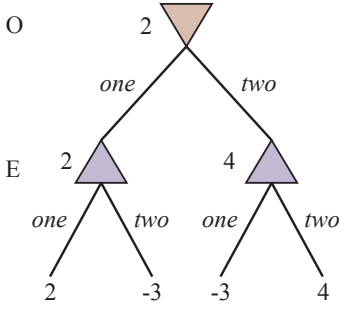
A **solution concept** is a way of choosing actions that take other players' actions into account. Some important terminology:

- **Strategy**: like a policy, but we must account for the actions of the other players.
- **Pure strategy**: a deterministic strategy/policy. For a single-move game, a single action.
- **Mixed strategy**: a randomized policy. The mixed strategy that chooses action a with probability p and action b otherwise is written $[p : a; (1 - p) : b]$.
  - Two-finger Morra example: $[0.5 : one; 0.5 : two]$
- **Strategy profile**: an assignment of a strategy to each player.
- **Outcome**: a numeric value for each player. For mixed strategies, this is expected utility.

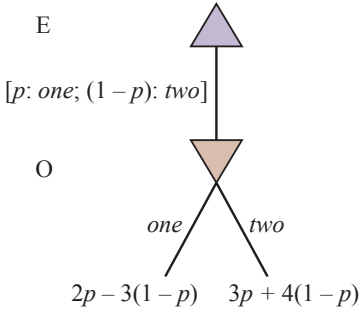
Solution concepts define rationality in games.

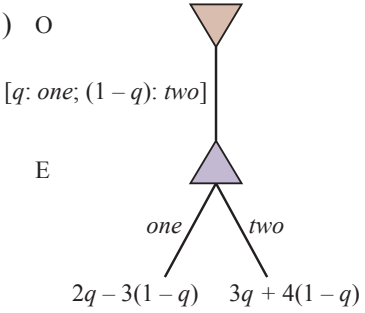
Morra Minimax Trees with Pure Strategies

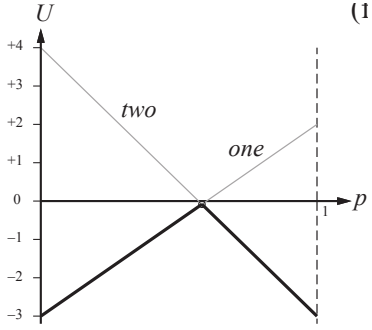


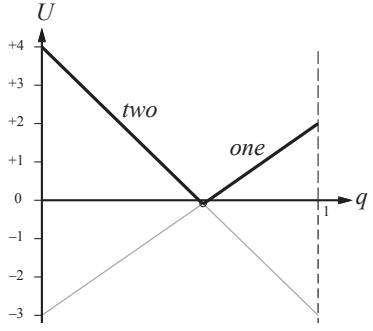


Morra Minimax Trees with Mixed Strategies

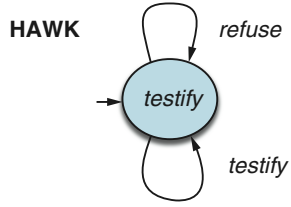




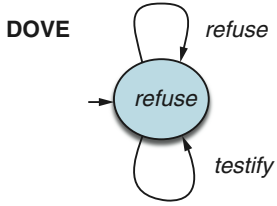

U
(1
+4
+3
+2
+1
0
−1
−2
−3
two
one
p
1


U
+4
+3
+2
+1
0
–1
–2
–3
two
one
q
1

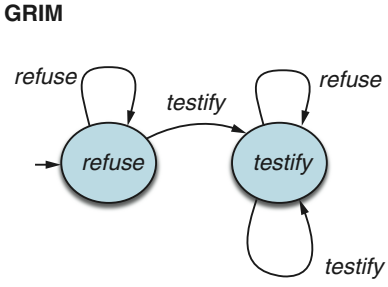
Hawk Strategy in IPD



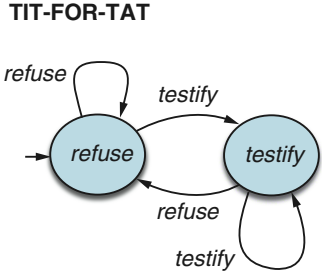
Dove Strategy in IPD



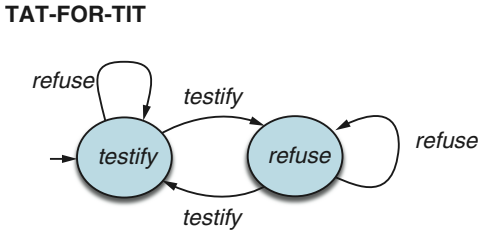
Grim Strategy in IPD

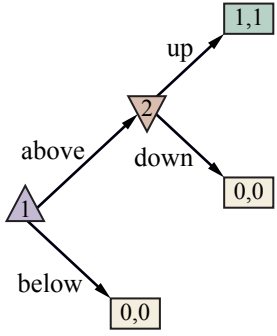


Tit for Tat Strategy in IPD



Tat for Tit Strategy in IPD




1,1
up
2
above
down
0,0
1
below
0,0

Multiagent Systems