EH2750 Computer Applications in Power Systems, Advanced Course.

Lecture 6

Professor Lars Nordström, Ph.D.
Dept of Industrial Information & Control systems, KTH
larsn@ics.kth.se

Acknowledgement

- These slides are based largely on a set of slides provided by:

  Professor Michael Wooldridge, Oxford University, England

  and

  Dr. Georg Groh, TU-München, Germany.

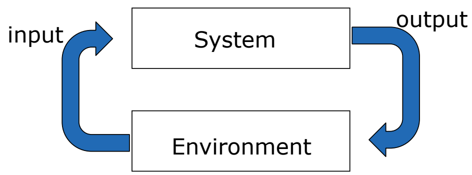
- Available at the Student companion site of the Introduction to Multi Agent Systems book

Outline of the Lecture

- Repeating where we are right now
  - Intelligent Agents of various types
  - How to make agents think and plan
  - Constratin Satisfaction Problems
  - Communicating & Cooperating
- Allocating Scarce Resources - Auctions

What is an Intelligent Agent?

- The main point about agents is they are *autonomous*: capable of acting independently, exhibiting control over their internal state
- Thus: *an intelligent agent is a computer system capable of flexible autonomous action in some environment in order to meet its design objectives*

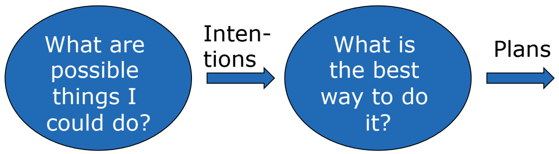


The discussion so far

- Chapter 2 describes the idea of agents that perform tasks in an environment and sets some definitions
- Chapters 3, 4, & 5 describe three different approaches to describing and developing the apparent Intelligence in the agents.
  - Chapter 3 – Deductive Reasoning Agents
  - Chapter 4 – Practical Reasoning Agents
  - Chapter 5 - Reactive (and Hybrid Agents)
- In the Excerpt from the AI book used in Lecture #4 we took a look at planning and searching
- At lecture #5 we looked at multi-agent encounters and rational ways of collaboration

Practical Reasoning

- Human practical reasoning consists of two activities:
  - *deliberation*
    deciding *what* state of affairs we want to achieve
  - *means-ends reasoning*
    deciding *how* to achieve these states of affairs
- The outputs of deliberation are *intentions*



Problem Formulation

- Before starting the search for a solution, we need to define the problem we are trying to solve
- A Problem formulation has the following parts:
  - An initial state
  - Actions possible in terms of **successor** function, that is a list of tuples:
    - `(Action, Successor)`
  - A goal state and a test if we are at the goal
  - A path cost related to the cost of a path/action*

*It is easy to think of the steps along the path as separate actions, this is OK, but formally not correct at this stage.

Practical Reasoning Agent

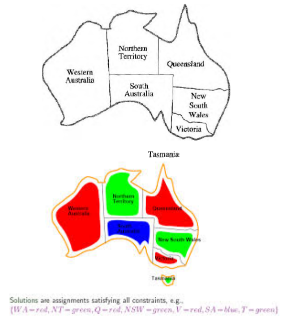
```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
  inputs: percept, a percept
  static: seq, an action sequence, initially empty
          state, some description of the current world state
          goal, a goal, initially null
          problem, a problem formulation

  state ← UPDATE-STATE(state, percept)
  if seq is empty then do
     goal ← FORMULATE-GOAL(state)
     problem ← FORMULATE-PROBLEM(state, goal)
     seq ← SEARCH(problem)
  action ← FIRST(seq)
  seq ← REST(seq)
  return action
```

Constraint Satisfaction problems

- Formally, a Constraint Satisfaction Problem (CSP) is
  - A set of variables $x_1, x_2, \ldots x_n$
  - All within a domain $d_1, d_2, \ldots d_n$
  - A set of constraints $c_1, c_2, \ldots c_m$
- A set of assigned values (to one or more of) the variable(s) is a state.
  - E.g.
    - $x_1 = 23$, $x_2 = 3$ is the state {23,3}

CSP in discrete finite domains

- Classic example – map coloring

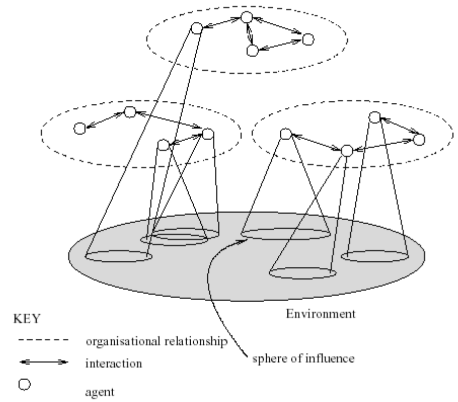


- Color the map of Australia
- Using the colors Red, Green, Blue
- No neighbours can have the same color
- CSP formulation
  - x_i = color of state i
  - D = {Red, Green, Blue, Null}
  - $x_i \neq x_j$ if $x_i = N(x_j)$

So, why all this?

- CSPs can be seen as search problems
  - States are defined by values assigned this far
    - Initial state: empty assignment {}
    - Successor function:
      - Assign value to a variable that is OK with constraints
    - Goal test: complete assignment with all constraints satisfied
- Note that every solution appears at depth n
  - ➔use depth-first search

Multi-agent Systems



Working Together

- Why and how do agents work together?
- Important to make a distinction between:
  - *benevolent agents*
  - *self-interested agents*

Multiagent Encounters

- We need a model of the environment in which these agents will act…
  - agents simultaneously choose an action to perform, and as a result of the actions they select, an outcome in Ω will result
  - the *actual* outcome depends on the *combination* of actions
  - assume each agent has just two possible actions that it can perform, *C* ("cooperate") and *D* ("defect")
- Environment behavior given by *state transformer function*:

$$\tau : \underbrace{Ac}_{\text{agent } i\text{'s action}} \times \underbrace{Ac}_{\text{agent } j\text{'s action}} \rightarrow \Omega$$

Payoff Matrices

- We can characterize the previous scenario in a *payoff matrix*:

		i defect	i coop
j	defect	1 / 1	4 / 1
j	coop	1 / 4	4 / 4

- Agent i is the *column player*
- Agent j is the *row player*

The Prisoner's Dilemma

- This apparent paradox is *the fundamental problem of multi-agent interactions*.
  It appears to imply that *cooperation will not occur in societies of self-interested agents*.
- Real world examples:
  - nuclear arms reduction ("why don't I keep mine. . . ")
  - free rider systems — public transport;
- The prisoner's dilemma is *present everywhere*.
- Can we recover cooperation?
  - Well, yes we can introduce auctions, negotiations and argumentation. More on this next lecture!

Outline of the Lecture

- Repeating where we are right now
  - Intelligent Agents of various types
  - How to make agents think and plan
  - Constratin Satisfaction Problems
  - Communicating & Cooperating
- Allocating Scarce Resources - Auctions

Outline of the Lecture

- Repeating where we are right now
  - Intelligent Agents of various types
  - How to make agents think and plan
  - Constratin Satisfaction Problems
  - Communicating & Cooperating
- Allocating Scarce Resources - Auctions

Allocating Scarce Resources

- Allocation of scarce resources amongst a number of agents is central to multiagent systems.
- Resource might be:
  - a physical object
  - the right to use land
  - computational resources (processor, memory, . . .)
  - Network capacity
  - Amount of energy
  - ….

Reaching Agreements

- The extreme case of a Multiagent encounter is the zero-sum. (*Profit only at expense of others*)
- Normal case is the "Win-win-situation" where mutually beneficial agreement is possible
- Reaching Agreements is fundamental for social intelligence and society building in general
- Reaching Agreements is a result of Negotiation, Auctions and/or Argumentation
- How do you construct algorithms these types of interactions

Algorithm Design Criteria

- We want to have a design of the algorithm that has certain properties:

1. Guaranteed success - Agreement is certain
2. Maximizing social welfare - Agreement maximizes sum of utilities of all participating agents
3. Computationally efficient

Algorithm Design Criteria, continued

4. Pareto efficiency: iff there exists no other agreement which increases utility of at least one agent while not decreasing the utility of the other agents
5. Individual Rationality: Following protocol is in best interest of all agents (no incentive to cheat, deviate from protocol etc.)
6. Stability: Protocol gives agents incentive to behave in a certain way. (→ e.g. by establishing Nash-Eq.)
7. Simplicity: Protocol makes for the agent appropriate strategy „obvious". (Agent can tractably determine optimal strategy)
8. Distribution: no single point of failure; minimize communication

What are auctions?

- Concerned with traders and their allocations of:
  - Units of an indivisible good; and
  - Money, which is divisible.
- Assuming some initial allocation
- Exchange is the free alteration of allocations of goods and money between traders

Auctions

- Auctions are simple → easy to implement
- Auction =consists of (Auctioneer, Bidders, Good);
  Goal of the Auctioneer is to maximize price for good;
  Goal of the Bidders is to minimize price for good;
  Each bidder has personal price maximum
- Auctioneer: Tries to reach goal by choosing appropriate auction mechanism
- Bidders: Try to reach goal by choosing appropriate strategy
- Auction algorithms differ by:
  - Winner determination,
  - Secrecy of bids,
  - Auction procedure

Single vs. Multi-dimensional auctions

- Single dimensional auctions
  - The only content of an offer are the price an quantity of some specific type of good.
  - "I'll bid $200 for those 2 chairs"
- Multi dimensional auctions
  - Offers can relate to many different aspects of many different goods.
  - "I'm prepared to pay $200 for those two red chairs, but $300 if you can deliver them tomorrow."

Value of the goods

Good has a public (common) value: Good has the same value for all bidders. (E.g.: One-Dollar-Bill)

Good has private value: Good has different value for each agent. (E.g.:)

Good has correlated value: Value of good depends on own private value and private value for other agents. (E.g.: Buy sth. with intention to sell it later)

Winner Determination

- First price: Highest bid wins, Winner pays his bid
- Second price: Highest bid wins. Winner pays second-highest bid
- General case, highest bid wins, pays n-k bid.

Secrecy of the Bids

- Open cry: All agent's know all agent's bids.
- Sealed bid: No agent knows other agent's bids

Auction Procedure

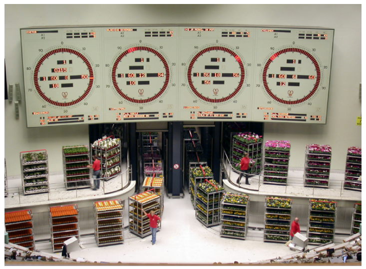
- One shot: Only one bidding round
- Ascending
  - Auctioneer begins at minimum price, bidders increase bids
  - Also known as English Auction
- Descending
  - Auctioneer begins at price over value of good and lowers the price at each round
  - Also known as Dutch auction

English Auctions

- Most common form (human world)
- **Open cry, first price, ascending**.
- Dominant strategy: Bid slightly more than current bit, withdraw if bid reaches personal valuation of good
- If uncertainty of (private or public) value of good exists:
  - "Should you be happy that you won the good?"
  - "Why did the other bidders not bid more?"
- Possibly: Although winning bid is below personal valuation of winner, the "true value" may be less than bid → "Winner's curse". (E.g. bidding for gold-mine)

Dutch Auctions

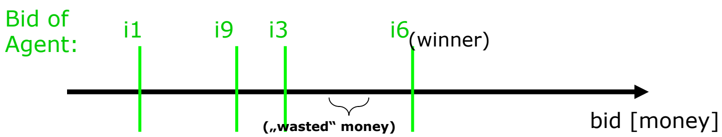
- **Open cry, first-price, descending**.
- No dominant strategy. Winner's curse also possible.



Sealed-bid First-price One-shot

- "Pseudo"-dominant strategy: (Assuming that others bid their true valuation): Bid less than true valuation.

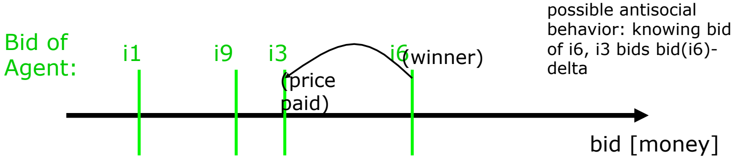
How much less → ??



Vickrey Auctions

- **Sealed-bid, second-price, one shot**
- Dominant strategy: (Assuming that others bid their true valuation): Bid true valuation.

Why? If you bid less than your true valuation → you only decrease your chances, but you will not influence the price you have to pay.
(As in all auctions: If you bid more than your true valuation → you risk winner's curse)



Combinatorial Auctions

- Until now, we have considered auctions for one indivisible good.
- Now consider auctions for goods thatare divisible
- Like for instance
  - Amount of available energy next hour, or
  - Power that can flow along a line

Formulation

$$\mathcal{Z} = \{z_1, \ldots, z_m\}$$

is a set of items to be auctioned, we have the usual set of agents

$$Ag = \{1, \ldots, n\},$$

and we capture preferences of agent i with the valuation function:

$$v_i : 2^{\mathcal{Z}} \mapsto \mathbb{R}$$

meaning that for every possible bundle of goods

$$Z \subseteq \mathcal{Z}, \; v_i(Z)$$

says how much Z is worth to Agent i.

Some additional facts & assumptions

If

$$v_i(\emptyset) = 0$$

then we say that the valuation function for i is normalised. Meaning that if an agent is allocated nothing it is worth nothing.

Similarly, we have that

$$Z_1 \subseteq Z_2 \text{ implies } v_i(Z_1) \leq v_i(Z_2)$$

Winner (or allocation) determination

- An allocation is a list of sets (allocations) $Z_1 \ldots Z_n$ for each agent Ag_i so that

$$Z_i \subseteq \mathcal{Z}$$

- And for all i,j in Ag such that $i \neq j$ we have $Z_i \wedge Z_j = 0$
- Valid for discrete sets of goods
- A general continuous case is similarly constrained by the sum of $Z_1 \ldots Z_n \leq Z$

How to do the allocation then?

- A reasonable assumption is to allocate in a way that maximises the social welfare, i.e. Maximizing the toal value achieved, the sum of all utilities.

$$sw(Z_1, \ldots, Z_n, v_1, \ldots, v_n) = \sum_{i=1}^{n} v_i(Z_i)$$

Cobinatorial Auction setup

- Given this, we can define a *combinatorial auction*.
- Given a set of goods $\mathcal{Z}$ and a collection of valuation functions $v_1, \ldots, v_n$, one for each agent $i \in Ag$, the goal is to find an allocation

$$Z_1^*, \ldots, Z_n^*$$

that maximizes sw, in other words

$$Z_1^*, \ldots, Z_n^* = \arg \max_{(Z_1, \ldots, Z_n) \in alloc(\mathcal{Z}, Ag)} sw(Z_1, \ldots, Z_n, v_1, \ldots, v_n)$$

- Figuring this out is *winner determination*.

Determining the allocation

- How do we do this?
- Well, we could get every agent i to declare their valuation $\hat{v}_i$
  - The hat denotes that this is what the agent *says*, not what it necessarily is.
  - The agent may lie!
- Then we just look at all the possible allocations and figure out what the best one is.

Computational efficiency?

- One problem here is *representation*, valuations are exponential:

$$v_i : 2^{\mathcal{Z}} \mapsto \mathbb{R}$$

  - A naive representation is impractical.
  - In a bandwidth auction with 1122 licenses we would have to specify 2^{1122} values for each bidder.
- Searching through them is computationally intractable.

So, how do we do it then?

- Searching through all combinations is a basic problem but intractable due to computation resources needed.
- However,his is the worst case result, so it may be possible to
- We can try to develop approaches that are optimal and run well in many cases.
- Can also forget optimality and either:
  - use heuristics; or
  - look for approximation algorithms.
- Common approach: code the problem as an integer linear program and use a standard solver – often works in practice.
- In practice a constraint satisfaction problem, that can be solved with different search mechanisms

Let's try an example

Outline of the Lecture

- Repeating where we are right now
  - Intelligent Agents of various types
  - How to make agents think and plan
  - Constratin Satisfaction Problems
  - Communicating & Cooperating
- Allocating Scarce Resources - Auctions