Distributed Software Development
Multiagent Systems II

Chris Brooks

Department of Computer Science
University of San Francisco
We were talking about how to achieve coordination among groups of intelligent agents.

We’ll start by talking about groups of cooperative agents.

We started with contract net, which is a nice algorithm for task allocation.
Distributed Planning

- Contract Net works nicely for environments that have discrete one-shot tasks.
- In many cases, domains are more complex.
- Agents are specialized.
- A ’task’ may require a sequence of steps.
- These steps may be accomplished in several ways.
- This leads into the realm of distributed planning.
A plan is a sequence of operations meant to accomplish a goal.

The goal is specified declaratively: at(luggage, airport), at(brooks, airport), at(students, airport)

Actions are ways of accomplishing parts of a plan

They have preconditions and effects.
- Preconditions must be true to perform the action
- Effects must hold after the action is taken

PutLuggageInTrunk. pre: holding(luggage) effect: in(luggage, trunk)

Planning is the process of finding a sequence of actions that accomplishes a goal.
20-3: Planning example

(From Russell and Norvig)

Notice that planning operators can be fully described by their preconditions and effects.

Planning involves putting these operators into a partial order that accomplishes the conditions in the goal state.

Challenges: conflicts, including disjunctive effects, allowing flexibility at run time.
Many algorithms exist for building and repairing plans.

Issues:
- Ordering constraints
- Dealing with failure
- Adapting to changes in the world
- Scaling

A common way to deal with large planning problems is to construct a hierarchical plan.
Many times, aspects of a problem can be solved independently.

Example: taking a trip to Peru can be decomposed into:
- Buying tickets
- Getting everyone to the airport
- Getting on the plane and flying there.

I can figure out how to solve each of these problems more or less independently.

Each subproblem can be represented as an AND/OR graph

Some decisions made at runtime.
- Caveat: decisions made in one subproblem may affect possible choices in other subproblems.
This is an example from a military logistics scenario.

Resources must be moved C1 to C2.

There are many ways to accomplish some of the subtasks.

Planners would like to leave as much flexibility as possible.
Distributed planning comes about when multiple agents (usually with hierarchical plans) must share resources. 

- A communication channel, a bridge, a power supply

Alternatively, there may be opportunities for synergy.

- Both agents plan to deliver packages to the same location.

How can agents synchronize their plans?
20-8: Distributed Planning

6 Solution 1: Submit plans to a centralized coordinator.
   △ Doesn’t scale
   △ Agents may not be willing to share more information than is needed.

6 Solution 2: broadcast top-level constraints to each other.
   △ This allows agents to detect whether there is a top-level conflict.
   △ Plans will either be totally serialized or totally parallel.
A better solution:

- Detect whether there is:
  - No problem: all possible interactions may be interleaved.
  - No solution: plans must be serialized.
  - Some solution: We then ‘step down a level’ in the plans and force agents to commit to particular alternatives.

- Tradeoff: Deeper level requires more communication and interleaving (an exponential problem), but produces finer-grained coordination.
Agents begin by exchanging top-level plans.

As conflicts are detected, they:
- Eliminate possible alternatives
- If necessary, drill down to more detailed plans.

If no solutions exist, agents must move back up and coordinate at a more abstract level.
Societies of agents

- Contract net and distributed planning work for tens of agents.
- How can we govern environments with thousands (or more) agents?
- These are often referred to as agent societies
  - Still a research area
  - Inspiration drawn from human society, Internet-scale protocols.
Research in this area can be divided into descriptive and proscriptive domains:

- **Descriptive:** “Given a structure or behavior on the world, what is the outcome?”
- **Proscriptive:** “If a structure or behavior is enforced, what outcomes result?”

There is also a vigorous debate about whether participants in an Internet-level agent society should be treated as self-interested, cooperative, or a mix of the two.

- Cooperation potentially allows for more beneficial outcomes, if participants can be trusted.

Many of the same issues as P2P systems arise.
The agent society approach can be used to construct teams of agents, each with very simple behavior, who can collectively solve a difficult task.

Ant algorithms

Problem: Explore an unknown area and locate high-resource areas
Avoid obstacles.

If you are not holding a resource, wander randomly. If you sense 'pheromones', weight random selection towards them.

If you find resources, pick them up and begin dropping pheromones. Follow a beacon back home.

If you make it home, drop the resource.

Over time, pheromone paths are built up between the home and the resource.
## Issues

6. Achieving macro-level behavior from microlevel rules.
   - How do you guarantee outcomes? Is there an efficient way to synthesize these sorts of rules?

6. Imposition of social norms or laws
   - What outcomes can be guaranteed for a given set of norms or laws? What language is necessary to describe norms or laws?

6. Mechanisms for trust and reputation
   - How can noncompliance be enforced?
Ant-type approaches lead us to think about how we can build systems that produce the effects we want.

“Given that agents will act in a particular way, how can we constrain the environment to achieve a desirable outcome?”

This method of problem solving is best applied to problems involving self-interested agents.
Agents will typically have preferences over outcomes

- This is declarative knowledge about the relative value of different states of the world.
- “I prefer ice cream to spinach”

Often, the value of an outcome can be quantified (perhaps in monetary terms.)

This allows the agent to compare the utility (or expected utility) of different actions.

A rational agent is one that maximizes expected utility.

Self-interested agents each have their own utility function.
By treating participants as rational agents, we can exploit techniques from game theory and economics.

Assume everyone will act to maximize their own payoff.

How do we structure the rules of the game so that this behavior leads to a desired outcome?

This approach is called *mechanism design*. 
Imagine a communications network. We want to know the shortest path between two nodes. (x and y)

- Edges between nodes are agents that can forward messages.
- Each agent has a private cost $t$ to forward a message along its edge.
- If all agents will truthfully reveal their $t$, we can run a shortest-path algorithm.

Agents may have an incentive to lie. (reduce traffic through their network).

How can we get self-interested agents to reveal their $t$?
20-20: An example

- Each agent reveals a $t$ and the shortest path is computed.

- Costs are accumulated
  - If an agent is not on the path, it is paid 0.
  - If an agent is on the path, it is paid the cost of the shortest path that doesn’t include it - (cost of best path - t)

\[ P = g_{next} - (g_{best} - t) \]

- For example, if I bid 10 and am in a path with cost 40, and the best solution without me is 60, I get paid 60 - (40 - 10) = 30

- Agent compensated for its contribution to the solution.
20-21: An example

6 If an agent overbids:
   △ If it was on the shortest path, still on the shortest path.
      - Payment lower, so no benefit to lying.
   △ Was on the shortest path, now not on the shortest path.
      - This means the lie was greater than $g_{\text{next}} - g_{\text{best}}$
      - Agent will receive 0.
      - But it would rather get a positive amount than 0!

6 If an agent underbids:
   △ Not on the shortest path, but now are.
   △ Underbidding leads to being paid at the lower amount, but still incurring higher cost.
   △ Truth would be better!
So how do we evaluate an algorithm or protocol involving self-interested agents?

Some solutions may be better for some agents and worse for others.
- Example: cake-cutting problem

We know that each agent will try to maximize its own welfare

What about the system as a whole?
There are a number of potential solution concepts we can use:

- **Social welfare** - sum of all agent utility.
- **Pareto efficiency**
  - Is there a solution that makes one agent better off without making anyone worse off?
- **Individual rationality**
  - An agent who participates in the solution should be better off than if it hadn’t participated.
- **Stability**
  - The mechanism should not be able to be manipulated by one or more agents.

It’s not usually possible to optimize all of these at the same time.
Ideally, we can design mechanisms with dominant strategies.

- A dominant strategy is the best thing to do no matter what any other agent does.
- In the previous example, truth-telling was a dominant strategy.
- We would say that the mechanism is non-manipulable. (lying can’t break it.)

Unfortunately, many problems don’t have a dominant strategy.

Instead, the best thing for agent 1 to do depends on what agents 2, 3, 4, ... do.
This leads to the concept of a *Nash equilibrium*

A set of actions is a Nash equilibrium if, for every agent, given that the other agents are playing those actions, it has no incentive to change.

Example: big monkey and little monkey

- Monkeys usually eat ground-level fruit
- Occasionally they climb a tree to get a coconut (1 per tree)
- A Coconut yields 10 Calories
- Big Monkey expends 2 Calories climbing the tree. (net 8 calories)
- Little Monkey expends 0 Calories climbing the tree. (net 10 calories)
If BM climbs the tree
△ BM gets 6 C, LM gets 4 C
△ LM eats some before BM gets down

If LM climbs the tree
△ BM gets 9 C, LM gets 1 C
△ BM eats almost all before LM gets down

If both climb the tree
△ BM gets 7 C, LM gets 3 C
△ BM hogs coconut

How should the monkeys each act so as to maximize their own calorie gain?
Assume BM decides first
- Two choices: wait or climb

LM has four choices:
- Always wait, always climb, same as BM, opposite of BM.
6 What should Big Monkey do?
6 If BM waits, LM will climb (1 is better than 0): BM gets 9
6 If BM climbs, LM will wait :BM gets 4
6 BM should wait.
6 What about LM?
   ▲ LM should do the opposite of BM.
6 This is a Nash equilibrium. For each monkey, given the other’s choice, it doesn’t want to change.
6 Each monkey is playing a best response.
Nash equilibria are nice in systems with rational agents.

If I assume other agents are rational, then I can assume they’ll play a best response.

I only need to consider Nash equilibria.

They are efficient (in the Pareto sense).

Problems:
- There can be many Nash equilibria.
- Some games have no Nash equilibrium.
- There may be ways for groups of agents to cheat.
An auction is a negotiation mechanism where:
- The mechanism is well-specified (it runs according to explicit rules)
- The negotiation is mediated by an intermediary
- Exchanges are market/currency-based

Agents place bids on items or collections of items.

An auctioneer determines how goods are allocated.

Desiderata: the auction should be fair, efficient, easy to use, and computationally efficient.

We’ll need to trade these against each other.
Private-value auctions are easier to think about at first.

In this case, the value agent A places on a job has nothing to do with the value that agent B places on the object.

△ For example, an hour of computing time.

In common-value auctions, the value an agent places on an item depends on how much others value it.

△ Example: art, collectibles, precious metals.
An English (or first-price) auction is the kind we’re most familiar with.

Bids start low and rise. All agents see all bids.

May be a reserve price involved.

Dominant strategy: bid \( \epsilon \) more than the highest price, until your threshold is reached.

Problems: requires multiple rounds, not efficient for the seller, requires agents to reveal their valuations to each other.

There may be technical problems to solve with making sure all agents see all bids within a limited period of time.
20-33: First-price sealed-bid auction

- Each agent submits a single sealed bid. Highest wins and pays what they bid.
  △ This is how you buy a house.
- Single round of bidding. All preferences remain private.
- Problems: No Nash equilibrium - agents need to counterspeculate. Item may not go to the agent who valued it most. (inefficient).
20-34: Dutch auction

- Prices start high and decline.
- First agent to bid wins.
- Strategically equivalent to first-price sealed-bid.
- In practice, closes quickly.
20-35: Vickrey auction

- The Vickrey, or second-price, auction, has a number of appealing aspects from a computational point of view.
- Single round of bidding.
- Efficient allocation of goods.
- Truth-telling is the dominant strategy.
- Rule: each agent bids. Highest bid wins, but pays the second price.
  - (the example we used earlier is isomorphic to the Vickrey auction).
Angel, Buffy and Cordelia are bidding on a sandwich.
- Angel is willing to pay $5, Buffy $3, and Cordelia $2.
- Each participant bids the amount they’re willing to pay.
- Angel gets the sandwich and pays $3.
Let’s prove that truth-telling is a dominant strategy.

Angel:
- If he overbids, he still pays $3. No advantage.
- If he bids between $3 and $5, he still pays $3. No advantage.
- If he bids less than $3, then he doesn’t get the sandwich - but he was willing to pay $5, so this is a loss.
Buffy (the same reasoning will hold for Cordelia)

- If she bids less than $3, she still doesn’t get the sandwich. (notice that we assume she doesn’t care how much Angel pays.)
- If she bids between $3 and $5, she still doesn’t get the sandwich. No benefit.
- If she bids more than $5, she gets the sandwich and pays $5. But she was only willing to pay $3, so this is a loss.
Because of these properties, Vickrey auctions have been adopted for:
- Allocation of computer resources
- Distribution of electrical power
- Bandwidth allocation
- Scheduling problems.

Interestingly, they are not widely used in human auctions.
- Perhaps people are not rational ...
20-40: Next time

- Dealing with common-value auctions.
- Combinations of goods.
- Computational markets and double auctions.