

# Distributed Software Development

## Cooperative MAS

Chris Brooks

Department of Computer Science  
University of San Francisco

Department of Computer Science — University of San Francisco — p. 177

### 22-2: Outline

- What is an agent?
- What are MAS/DAI problems?
- Recommender Systems
- Coordination
  - Contract Net
  - Distributed Planning
- Societies of agents

Department of Computer Science — University of San Francisco — p. 277

### 22-3: Distributed AI

- Distributed AI (more often called Multiagent Systems these days) is the study of how multiple agents can and do interact with each other.
  - May include both human or computational agents.
  - May be a closed or an open system
  - Agents may be cooperative or self-interested.

Department of Computer Science — University of San Francisco — p. 377

### 22-4: Qualities of an agent

- Autonomy
- Adaptation
- Goal-directed behavior
- Has “beliefs” and “intentions”
- Proactive
- Situated within an environment

Department of Computer Science — University of San Francisco — p. 477

### 22-5: Autonomy

- Autonomy is a quality often attributed to agents.
- An autonomous agent is able to rely on its percepts and past experience to make decisions, rather than asking a human for help.
- This is a thorny area - most agents will not have complete autonomy.
  - When might we not want an agent to have complete autonomy?
- Challenge: Designing an agent that can reason about its own autonomy and know when to ask for help.

Department of Computer Science — University of San Francisco — p. 577

### 22-6: Agent-oriented Programming

- We can also think of agents as a programming paradigm.
  - The next logical step after objects.
  - “Objects do it for free, agents do it for money.”
- Objects are *receivers* of actions, agents are actors.
- It's less useful to think of agent as an *objective* label than as a *subjective* description.
- Agency is a useful abstraction for us as programmers.
  - Allows us to think about a program at a higher level.
- Treat something as an agent if that helps to understand, predict, or explain its behavior.
  - Thermostats as agents

Department of Computer Science — University of San Francisco — p. 677

#### 22-7: Usefulness of the agent metaphor

- Why bother with all this? We already know how to write programs.
- Agents tend to be *open-ended* programs
  - Difficult to specify in advance what they should do in all cases.
- It's helpful to talk about them as *if* they were intelligent.
- "The robot wants to find the power supply."
- "The server believes that the client has reset."
- This assigning of mental states to programs is called the *intentional stance*.

#### 22-8: Agents and Services

- Service-oriented computing is a direct extension of the agent metaphor
- Idea: An agent is able to provide one or more services
  - Specified at a high level, extracted from implementational details.
  - A service might actually involve gathering information from other agents, delegating a task to another agent, or recombining information.

#### 22-9: Why solve DAI problems?

- So a DAI or MAS problem involves a number of agents interacting with each other.
- Why not construct a centralized system?
  - Complexity
  - Geographic or temporal separation of components
  - Separate ownership
  - Dynamic or changing system

#### 22-10: Some examples

- NASA
  - Coordinating Mars rovers
  - multiple orbiting satellites
- Scheduling
  - Robots in factories
  - Power generation and distribution
  - Troop movement and deployment
- Financial markets
- P2P networks
- etc

#### 22-11: Recommender Systems

- A problem that can be considered in a multiagent context concerns the gathering and combination of information from multiple sources.
- For example, Amazon, Netflix, MovieFinder, etc.
- There are two different approaches to this problem:
  - Content-based Filtering
  - Collaborative Filtering

#### 22-12: Content-based Filtering

- In Content-based filtering, a model of each user is constructed based on their expressed preferences.
  - Things you bought, pages you looked at, things you said you liked.
- A model of each product is constructed.
- Those products that are "closest" to your preference model are recommended.
- Approach: Match users to items.

#### 22-13: Content-based Filtering

- Advantages of Content-based Filtering:
  - Every product in the system can be recommended.
  - Recommendations will (in theory) correspond directly to a user's preferences.
- Disadvantages of Content-based Filtering:
  - A potentially complex product model must be constructed.
  - Users may have to evaluate a large number of products to build an accurate model of their preferences.
    - Getting users to tell you their preferences is hard.

#### 22-14: Collaborative Filtering

- Collaborative Filtering takes the opposite approach.
  - Matches users to users.
  - Intuition: If Bob likes A, B, and C, and Mary likes A and B, she'll probably also like C.
    - In other words, she's assumed to be similar to Bob.
  - Products are typically just atomic items.
- Approach: match users with users

#### 22-15: Collaborative Filtering

- A user is represented by an  $n$ -dimensional vector
  - $n$  is the number of products in the system
- A clustering or nearest-neighbor algorithm is used to group users.
- A user is then recommended items that are unseen but highly-rated by users who are "close".

#### 22-16: Collaborative Filtering

- Advantages of collaborative filtering
  - Can combine information from multiple sources
    - Each user potentially has to rate fewer items
  - No involved modeling needed.
- Disadvantages of Collaborative Filtering
  - How to deal with new items?
    - If no one has rated an item, it can't be recommended.
  - Tends to "push toward the middle" and recommend popular items.
- Much of the current work on recommender systems involves combining these approaches.

#### 22-17: Coordination

- Coordination turns out to be one of the most fundamental problems in MAS.
- Coordination can occur because:
  - Agents must work together to solve some larger problem.
  - There is the possibility of conflict
  - There is the possibility of shared goals.

#### 22-18: Mechanisms for Coordination

- Organizational structure
  - Little communication needed
  - Agents do not need to know about the entire population
  - Structure must be agreed on in advance
  - Can be brittle, or inefficient.

#### 22-19: Mechanisms for Coordination

- Negotiation mechanisms
  - More communication needed
  - Agents do not have to be cooperative
  - Challenges: designing a negotiation protocol that is fair and efficient.

#### 22-20: Mechanisms for Coordination

- Markets and auctions
  - Can deal with relatively large numbers of agents.
  - Agents can be self-interested
  - Under certain conditions, efficient outcomes are reached.
  - If those conditions are not met, outcomes are less predictable.
  - Can include NP-hard problems, which limits the size of the population.
  - Potential expressivity problems.

#### 22-21: Mechanisms for Coordination

- Social norms/laws
  - Designed for Internet-level population sizes
  - Rules are defined for how one "should act."
  - Violators are (ideally) discouraged from behaving poorly.
  - Challenge: enforcement.

#### 22-22: Cooperative environments

- We'll start with cooperative environments.
- Distribution due to spatial/geographic reasons or complexity.
- Agents are assumed to tell the truth, and act to achieve a global goal.

#### 22-23: Contract Net

- Contract Net is one of the oldest and most widely-used coordination mechanisms.
  - Simple, easy to implement.
- Works best in cooperative environments, but can be used in self-interested environments.
- Used to allocate tasks.
  - Who will agree to do what?

#### 22-24: Contract Net

- Begin with an agent known as the *manager*
  - Allocates tasks
- A set of agents act as *contractors*
  - Potentially perform these tasks
- The manager announces a task to (a subset of) the contractors
- Any agent who can complete the task responds with a bid.
- Manager evaluates all bids and assigns the task.
- Awardee now has a *commitment* to accomplish this task.
  - It may further contract this task out.
- Note that there's an implicit research problem in here: constructing languages that allow agents to describe and negotiate about tasks.

#### 22-25: Example: distributed sensor management

- A set of cameras located around campus can track people
- Cameras each have an area they can see.
- A camera is tracking a person. As the person leaves its line of sight, it announces a bid to continue tracking.
- The first agent to reply wins the contract and continues tracking.
- Since all agents are cooperative, the 'bidding' is quite simple.

#### 22-26: What's interesting about contract net?

- Contract net seems very simple; why is it interesting?
- Decisions about how tasks are allocated are made by both the manager and the contractor.
  - Manager cannot force an agent to take on a task.
- Formal bidding models can easily be added onto the basic protocol.
- Works nicely in environments where tasks appear dynamically.

#### 22-27: 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*

#### 22-28: Planning in Five Minutes

- 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.

#### 22-29: Planning in Five Minutes

- 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.

#### 22-30: Hierarchical Planning

- 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.

#### 22-31: Distributed Planning

- 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?

#### 22-32: Distributed Planning

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

#### 22-33: Distributed Planning

- 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.

#### 22-34: 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.

#### 22-35: Societies of agents

- 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.

#### 22-36: Example

- 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

#### 22-37: Rules

- 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.

#### 22-38: Issues

- Achieving macro-level behavior from microlevel rules.
  - How do you guarantee outcomes? Is there an efficient way to synthesize these sorts of rules?
- 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?
- Mechanisms for trust and reputation
  - How can noncompliance be enforced?

#### 22-39: Summary

- Multiagent problems arise due to spatial, privacy or scaling constraints.
- Combining of information is a fundamental task.
- Coordination is also a fundamental problem
  - Contract net
  - Distributed planning
  - Ant Algorithms

#### 22-40: Next time

- Lying liars and the lies they tell.
- or,
- What to do when agents are out for themselves.
- Applying economics to coordination and allocation problems.