COSC 4P78

Multi-Agent Systems

Week 7

(Work based on Matarić)

Brock University

When One Robot Isn't Enough...

Use TWO Robots!

For various reasons, some tasks aren't suitably addressed with a single robot, no matter how effective it is.

For those cases, we consider using multiple robots.

When we combine the use of multiple robots for the same task, we're creating a *society*, or a *multi-agent system*.

In this context, agents follow the same basic principle as agents in AI.

Before We Start...

Before we start, we should realize that we're going to be solving one problem, but introducing new concerns instead:

- Inherently dynamic environment
  - Additional robots change the environment, and also effectively become part of it
- Complex local and global interaction
- Increased Uncertainty
  - Each robot already had a degree of uncertainty due to imperfect knowledge from sensors, effectors, etc.; adding additional players to the field increases that uncertainty
- The need for coordination
  - As well as cooperation (not the same thing!)
- The need for communication
  - (Asterisk)

Why Do It?

What's in it for us?

- Some tasks are simply impossible for a single robot
  - Moving a large or awkward object
  - Automated assembly
  - Simply put, if teamwork makes sense for humans, it may also make sense for robots
- Parallelism
  - Suppose you were lost in the woods. Which would you realistically want searching for you?
    - A single "Lassie"
    - A pack of 40 non-tv dogs
  - Sometimes, additional agents means additional brains trying to solve a problem
  - Sometimes, it simply means a larger presence. On that note...
- Omnipresence
  - Related to parallelism, but distinct, *sensor-actuator networks* might be helpful for, for example, habitat monitoring
  - Track animals, fish, etc.; provide security; never have to go more than eleven seconds without a martini

Why Do It?
(cont)

- Cost
  - A *swarm* of very simplistic robots might actually be cheaper than a single, military-grade wunderkind
- Robustness and Redundancy
  - Robustness is the ability to resist failure. Even if a component (or agent) breaks down, the system can continue
  - Redundancy refers to redundant agents
    - In Computer Science, we're used to "redundancy" being a bad thing, but in this case, it means that, when an agent ceases to function, another one may be there to take its place, improving robustness
  - Note that having multiple agents doesn't necessarily imply redundancy. If they're all different, then agents may not be able to fill in for each other

So...

It sounds like the benefits outweigh the drawbacks, right?

Actually, we haven't covered those yet. What we discussed earlier were simply issues to consider before even starting...

We still have a few more concerns to address...

Challenges of Teamwork

Help *me* help *you* help *me*

- Interference
  - Agents already need to deal with imperfect knowledge, a poor understanding of their environment, and trying to react quickly to changes
  - Adding additional agents means that environment becomes more complicated and dynamic
  - What's more, two agents could end up interfering directly with each other if they both try to occupy the same space, or move the same object
    - ★ Imagine two agents pushing the same box from opposite sides, or trying to get through a doorway at the same time!

Challenges of Teamwork
(cont)

- Communication
  - Assume you've decided on explicit communication. Whose turn is it? How about now? What about now?
  - What happens when two agents try to 'talk' at the same time wirelessly?
  - Will a message sent by one agent be received by *all* agents?
  - Where wireless is used, it's not uncommon to require that either an agent can act as a coordinator, or to use a more powerful networking protocol (e.g. one that allows for packets, *mesh networks*, etc)

Challenges of Teamwork
(still cont)

- Uncertainty and Agreement
  - Each agent has (typically) imperfect knowledge of their environment
  - Even more simple than the problem if them potentially not fully sharing that knowledge, what happens when their knowledge conflicts?
    - e.g. What happens when one agent hasn't yet realized that a phase of the job is done yet?
    - What happens when some agents think there's a path, but one thinks there's an obstruction?
  - Differing opinions of the world may not be a problem, but lack of agreement on what to do next *is*

Challenges of Teamwork
(stiiiillll cont)

- Expense
  - Though several simple robots can be more expensive than a single deluxe model, they're still more expensive than a single, equally simple robot
  - Also, with each additional robot, there's another set of parts to potentially fail
    - Maintenance can get costly/time-consuming

Types of Groups and Teams

Consider the task of designing a team of robots to play soccer (e.g. RoboCup)

Would it be sufficient to simply program a single robotic soccer agent, and then duplicate that? How well do you think that'd work?

About as well as it'd work for humans, presumably.

What we need to do is to consider *division of labour* and/or *role assignment*

- When we say division of labour, we're just talking about *divide and conquer*
- When we're talking about roles, we're saying that each agent has a specific job
    - One agent could be the goalie
    - One could play offense
    - One could play defense
    - One could get me that martini, dangit!

Division of Labour?

Is that always necessary?

Of course it isn't *always* necessary. And, even when it is, it may be to differing extents.

- Forraging may not require much cooperation at all
- Search and rescue may require only agreement on how to divide up an area so two agents aren't searching the same space
- Either case might only require additional communication after they've found something (and thus the current sub-task is complete)

Describing Teams

Homogeneity/Heterogeneity

- A team is *homogeneous* when all agents are completely identical
  - This means that both their hardware *and* software are the same
  - Any agent may be swapped with any other
  - Typically coordination is simple, and may not even be explicit (e.g. flocking)
  - This, of course, requires that all agents be either equally talented, or sufficiently general-purpose (i.e. equally mundane)
- A team is *heterogeneous* when some agents are different
  - Typically, this goes together with some agents having different *roles*
  - If some agents have different capabilities and/or roles, that typically requires more direct coordination
- Strictly speaking, heterogeneity can be defined for form *or* function (or both)
  - e.g. A wheeled robot and a crawling robot (different constructions) both pushing a box together, based on the same 'thought process' (identical roles)

Coordination Strategies

Keep your friends close, your enemies closer, and your martini closest

There are different possible levels of coordination and cooperation:

- Mere Coexistence
  - Agents might not even recognize each other as agents (but rather, moving parts of the environment/obstacles)
  - e.g. Foraging for loose debris
    - Agents would already have to have the ability to avoid obstacles in their environment. It isn't much more complicated to have them avoid each other as well
    - There would still be issues to address (e.g. trying to circumvent what could appear to be a 'moving wall', dropping off debris to the 'home base' when another agent might be doing the same, etc)

Coordination Strategies
(cont)

- Loosely Coupled
  - Agents recognize each other as being agents
  - Some simple coordination may be employed
    - e.g. moving away from each other to minimize interference
  - No explicit inter-dependency for achieving the task
  - e.g. For foraging, if an agent finds debris, it wouldn't coordinate the retrieval of all other agents to quickly harvest it, but a single agent might notice that another is holding debris, and start following it in the hopes of finding out where they got it
    - Think of it like following another customer in a grocery store when you spot that they've found something tasty. You might stalk them a bit to see where they got it, but that doesn't make you friends
    - (Or, uh, some non-creepy analogy. That works, too)

- Tightly Coupled
    - Agents carefully coordinate with each other
    - Typically extensive communication
    - Often turn-taking, or other tight coordination
    - *Usually* more powerful, but at the cost of redundancy/robustness
    - If we aren't sick of the foraging example yet, imagine the agents deciding on a shared map, explicitly designated destinations, and possible explicit communication/direction upon finding a large cache

Communication

Can you hear me now?

Communication isn't *always* necessary, but often it is.
Reasons include:

- Improving Perception
  - Each agent has limited perception
  - Sharing information gives each agent the benefit of the other agents' perception
- Synchronization of Actions
  - Imagine one arm trying to hand an item off to another. You would *not* want to do that based solely on timing or other environmental cues!
  - As the number of agents increases, anything but an explicit coordinating signal pushes the problem from burdensome to impractical
- Other Coordination and Negotiations
  - Even deciding that a handoff is necessary would be difficult
  - There may be cases where one agent has a better understanding of what another agent should do than that agent itself

Case Study: Foraging and Communication

Because we're not sick of this example yet. Nope. Not at all.

For our foraging example, what *might* we wish to communicate?

- Nothing
  - For 'mere coexistence', communication is likely unnecessary
- Task-Related State
  - The location of known objects, number of recently-seen robots, etc.
- Individual State
  - ID number, energy level, number of objects collected, etc.
  - Note that this mostly makes sense for heterogeneous systems
  - It implies roles (potentially even changing roles)
- Environment State
  - Blocked exits/paths, dangerous conditions, newly discovered paths, etc.
- Goal(s)
  - Direction to nearest object, etc.
- Intentions
  - e.g. "I'm going that way because there's debris there; I'm not going there because everyone else is already there."
  - Agent can make decisions based on the shared information, and may choose to make similar (or intentionally different) plans accordingly

Types of Communication

In addition to the obvious (previously-mentioned) wireless option, there are also others. (Consider all of the ways humans communicate for just a second. You could build a robot with muppet-arms!!!)

- Explicit Communication
  - Broadcast
    - An agent sends a message out to all other agents within range
    - Also includes the light show we'll look at later
  - Peer-to-peer
    - An agent designates a single recipient for a message
    - Particularly appropriate for coordination between two agents
  - Publish/Subscribe
    - Sends messages to a team of agents (but only those agents)
- Implicit Communication
  - Rather than *directly* communicating, leave information in the environment
  - e.g. stigmergy (changing the environment)
    - Ants and pheromones
    - Leaving marks on already-searched areas

Back to the Case Study?

Foraging for Pucks!

Have we discussed this one to death? Do we also want to talk about using communication to help with this task a bit?

Kin Recognition

We won't be discussing this one too much, but it'd be *nice* if agents can recognize each other as agents.

But, come to think of it, wouldn't that be really tricky? How would we even do that? Machine vision?

There are ways of identifying agents (either merely that they *are* agents, what role of agent they play, or which specific agent it is), but the techniques will vary by task/robot/etc.

Control

There are two basic approaches: Centralized, or Distributed

- Centralized
  - A single controller (possibly one of the agents) takes in information and delegates to other agents
  - Of course, this requires all information be accumulated/processed in one place
  - This may impose a bottleneck, and could potentially introduce other issues like mobility/range
  - So, why could it still be good?
    - It can be easier to compute an optimal solution
    - Less chances of interference (ideally, there should be *none*)
    - It may also provide a single frame of reference for collecting and integrating environmental information

Control
(cont)

- Distributed
  - Each agent has a role in planning out actions
  - This *might* reduce communication and doesn't require consolidation of information
  - Similarly, this might reduce processing bottlenecks
    - This also means the swarm can often be scaled larger without increasing the processing demands
  - Of course, coordination can become a nightmare, and interference is back on the table
  - We're looking for a *global behaviour*, and somehow need to achieve that by creating *local rules*
    - This is the *inverse problem*
    - We're trying to trigger an emergent behaviour, without explicitly creating the precise rules for that to happen
    - Compare this to inverse kinematics, even though this one's a lot harder!

Architectures for Multi-Agent Control

Just as with individual robots, we still have the same basic options: Deliberative/hierarchical, Reactive, and Hybrid (and Behaviour-based, but we haven't really talked about that one).

- Obviously, deliberative is appropriate for centralized control
- Reactive is well-suited for distributed control
- Hybrid is good for either

Hierarchies

As mentioned earlier, heterogeneous systems tend to include the idea of roles. We should give consideration to that fact when deciding on control schemes.

Without getting too bogged down in extraneous details, hierarchies ("pecking orders" in the book) allow agents to act as the 'leader'

- Either permanently, in fixed hierarchies
- Or with changing pecking orders, as with dynamic hierarchies

It may be more complicated than simply 'follower' and 'leader'. There may be multiple levels. My suggestion is to simply give it some thought.

Penultimate Slide

Industrial robots/manufacturing (including CNC and 3D printing), and multi-agent systems are two fields with a lot of promise, and plenty of room to still find new innovations. If you're interested in playing with robots as a career, it wouldn't hurt to be well-versed in both.

Neat Links

- http://www.youtube.com/watch?v=aCyAN7ILQ7c
- http://www.youtube.com/watch?v=i3ernrkZ91E
- http://www.youtube.com/watch?v=Fy1lYF53d8Y