ME 456: Intelligent Robotics

Week 8, Lecture 2
Multi-Robot Coordination

Announcements:

HW 4 due on 11/25
No class on 11/26

Motivation

Towards a Swarm of Nano Quadrotors

Alex Kushleyev, Daniel Mellinger, and Vijay Kumar
GRASP Lab, University of Pennsylvania
And Another One

Multiple Robot Systems

- Context, motivation
- Multiagent Systems
- Application to Robotics

Pay close attention to
what the robots see and know
where the computation is taking place
Multiple Robots: Motivations

• Exploration in hazardous environments
  - Under water
  - On distant planets
  - Inside damaged buildings

• Tasks beyond the limits of single robots
  - Cooperative lifting
  - Assembling large complex structures

• Tasks that can be completed more rapidly with multiple robots
  - Collecting trash in a large office building
  - Searching for mines

• Distributed sensing (thousands of sensors)
  - Studying complex ecosystems (tree tops)
  - Detecting temperature changes in the ocean

Historical Notes

• Electromechanical tortoises, 1950, W. G. Walter
  - Vacuum tube technology
  - Moved toward a light when there was a light

• Multiple manipulators, 1980s
  - Two arms grasp the same object
  - Actions of one robot arm constrain the actions of the other

• Multiple manipulators part 2, 90s and beyond
  - Mobile robots grasp an object without grasping (Stanford)
  - Moving a sofa
  - Box pushing

• Behavior based robots, Mataric
• Robocup, 1998 onward
Control issues

- Centralized and hierarchical
  - Army
  - Factory
  - Advantage: well defined
  - Disadvantage: no redundancy

- Decentralized and local control
  - Ants
  - Advantage: fault tolerant, role redistribution
  - Disadvantage: difficult to control
Centralized Control

• Applicable when the controllers can be placed in a position to observe and communicate with all robots

• Useful when:
  – Individual robots would have to be larger than practical
  – Overall positional sensing is limited
  – Manufacturing costs are high

• Example: Warehouse applications
  – Electronic assembly using robots

Distributed Control

• Applicable when the robots will need to take independent actions

• Useful when:
  – Separation in space
  – Time lag
  – Redundancy is relevant
  – Cost of single point of failure outweights cost of robots

• Example: Space exploration
  – Planetary exploration rovers
Communication among Robots

- Point to point communication
  - Individual robots communicate with one another
  - Expensive (power, computation)
  - Information overload

- Broadcast
  - Robots broadcast information
  - Only robots within a range receive broadcast
  - Broadcaster may not know who received information

- Communication via the environment
  - Messages implicit
  - Turn of a light after reaching it
  - Leave trail on the path

Robot Soccer

- Simulation League
  - 2D
  - 3D
- Small size robot league (18 cm, 5 per team)
- Middle size robot league (50 cm, 4 per team)
- Standard size robot
  - Identical platforms (Software competition)
  - Was four legged competition in previous case
- Humanoid league
  - Currently: penalty kick, 2 vs. 2

By the year 2050, develop a team of fully autonomous humanoid robots that can win against the human world soccer champion team.
Multiple Robot Systems

• Context, motivation

• Multiagent Systems

• Application to Robotics

Multi-Robot Coordination

• Consider a large multiagent system where
  - Each robot has a *private objective* it is trying to optimize; and
  - There is a *system objective* function measuring the full system’s performance

• Key Questions:
  - *How to set robot objective functions?*
  - *How to update them (team formation)?*
  - *How to modify them to changing objectives (reconfiguration)?*
  - *What happens when robots can’t compute those objectives?*
  - *What happens when information is missing?*
  - *What happens when some robots start to fail?*
Analogy: A company

- Full System
- System objective
- Agents
- Agent objectives

Company

Valuation of company

Employees

Compensation packages

- Design problem (faced by the board):
  - How to set/modify compensation packages (agent objectives) of the employees to increase valuation of company (system objective)
    - Salary
    - bonuses
    - Benefits
    - Stock options
  - Note: Board does not tell each individual what to do. They set the “incentive packages” for employees (including the CEO).
Key Concepts for Coordinated MAS

- **Factoredness**: Degree to which an agent’s objective is “aligned” with the system objective
  - e.g. stock options are factored w.r.t. company valuation.

- **Learnability**: Based on sensitivity of an agent’s private objective to changes in its state (signal-to-noise).
  - e.g., performance bonuses increase learnability of agent’s objective

- Interesting question: If you could, would you want everyone’s objective to be valuation of company?
  - Factored, yes; but what about learnability?

Nomenclature

- $z$: State of full system
- $z_i$: State of agent $i$
- $z_{-i}$: State of full system, except agent $i$
- $c_i$: Fixed vector (independent of agent $i$)
- $z_i + c_i$: Full state with “counterfactual” agent $i$
- $G(z)$: Reward/Objective for full system
- $g_i(z)$: Reward/Objective for agent $i$
Factoredness

**Factoredness**: Degree to which an agent’s objective function is “aligned” with the system objective

\[
F_{g_i} = \sum_{z'} u[ ((g_i(z) - g_i(z'))(G(z) - G(z'))] \sum_{z'} 1
\]
\[
(z'_{-i} = z_{-i})
\]

For continuous states:

\[
F_{g_i} = \int_z \int_{z'} u[ (g_i(z) - g_i(z'))(G(z) - G(z'))] dz' dz
\]

\[
\int_z \int_{z'} dz' dz
\]

---

**Factoredness**

**Factoredness**: Degree to which an agent’s objective function is “aligned” with the system objective

\[
F_{g_i} = \sum_{z'} u[ ((g_i(z) - g_i(z'))(G(z) - G(z'))] \sum_{z'} 1
\]
\[
(z'_{-i} = z_{-i})
\]

\[
\boxed{F_{g_i} = \text{Actions of } i \text{ that improve/deteriorate } g, \text{AND } G \text{ All actions of } i}
\]
Learnability

**Learnability**: Degree to which an agent’s objective function is sensitive to its own actions, as opposed to the "background" noise of other agents’ actions

\[ L(g_i, z, z') = \frac{\| g_i(z) - g_i(z - z_i + z'_i) \|}{\| g_i(z) - g_i(z'_i - z_i + z_i) \|} \]

\[ L(g_i, z) = \frac{\sum_{z'} L(g_i, z, z')}{\sum_{z'} 1} \]

\[ \Delta g_i = \frac{\text{Change in } g_i \text{ as a result of } i' \text{'s actions}}{\text{Change in } g_i \text{ as a result of other agents' actions}} \]

General Solution

- To get agent objective with high factoredness and learnability, start with:

\[ g_i(z) = G(z) - G(z_{-i} + c_i) \]

- If G differentiable, then:

\[ \frac{\partial G(z_{-i} + c_i)}{\partial z_i} = 0 \]

\[ \frac{\partial g_i(z)}{\partial z_i} = \frac{\partial G(z)}{\partial z_i} \]
Analogy: A company

- Full System
- System objective
- Agents
- Agent objectives
- Company
- Valuation of company
- Employees
- Compensation packages

- Give every employee a “difference objective”

- What’s good for the employee is good for the company
  - Stock options vesting at particular times??

- How to tune the learnability of \( g_i(z) \)?

---

How about Learnability

- Two examples for \( c_i \):

  - \( c_i = 0 \)

  \[
  g_i(Z) = G(Z) - G(Z_{-i})
  \]

  “world without me”

  - \( c_i = E[z_i] \)

  \[
  g_i(Z) = G(Z) - G(Z_{-i} + E[z_i])
  \]

  “world with average me”
Analogy: A company

- Full System \leftrightarrow \text{Company}
- System objective \leftrightarrow \text{Valuation of company}
- Agents \leftrightarrow \text{Employees}
- Agent objectives \leftrightarrow \text{Compensation packages}

- Give every employee a “difference objective”
- Make sure the employee can measure his/her impact
  - Stock options vesting at particular times??
  - Sliding scale of stock options based on impact of product??

Research Issues:

- In general agents may not be able to compute $g$:
  - Limited Observability
  - Restricted Communication
  - Temporal separation
  - Spatial separation
  - Limited Computation

- Solutions:
  - Estimate missing information
  - Leverage local information
  - Approximate $G$ or $z$
  - Trade-off factoredness vs. learnability
Multiple Robot Systems

- Context, motivation
- Multiagent Systems
- Application to Robotics

Recall: Rover Coordination

- Rovers observe points of interest (POIs)
  - POIs vary in value, time and place
  - Get more information closer to POI
  - Only primary observation counts
- Learning problem
  - Rovers learn in single trial (non-episodic)
    - Dynamic: POIs appear/disappear
  - Rovers reset at regular intervals (episodic)
    - Static: POIs the same in each episode
    - Dynamic: POIs different in each episode

\[
G = \sum_i \sum_j \min \left[ V_j \delta(L_{j,i}, L_{i,i}) \right] \\
\delta(x, y) = \min \{ ||x - y||^2, d^2 \}
\]
Challenges

• How to design an adaptive control mechanism for problems with:
  - Continuous state spaces
    • Need to generalize
  - Dynamic environment
    • Need to learn sensor to actions mapping
Challenges

• How to design an adaptive control mechanism for problems with:
  - Continuous state spaces
    - Need to generalize
  - Dynamic environment
    - Need to learn sensor to actions mapping
  - Noisy sensors/actuators
    - Need to be robust
  - Limited communication/observation
    - Need to use local information effectively
Challenges

• How to design an adaptive control mechanism for problems with:
  • Continuous state spaces
    • Need to generalize
  • Dynamic environment
    • Need to learn sensor to actions mapping
  • Noisy sensors/actuators
    • Need to be robust
  • Limited communication/observation
    • Need to use local information effectively
  • Multiple agents to coordinate
    • Collective action needs to optimize system objective

Recall: Robot Control

• Robots observe environment through some sensors
• Sensors are inputed into a neural network
• Output of neural network determines direction/velocity of rover
Recall: Rover Control

1. At $t=0$ initialize $N=10$ controllers
2. Pick a controller using $e$-greedy alg ($e=.1$)
3. Randomly modify controller
4. Use controller on this agent for 15 steps
5. Evaluate controller performance
6. Re-insert controller into pool
7. Remove worst controller from pool
8. Go to step 2
**Objective Functions**

\[ G = \sum_t \sum_j \frac{V_j}{\min_i \delta(L_j, L_{i,t})} \]  
Global  
(Fully Factored, Low Learnability)

\[ P_i = \sum_t \sum_j \frac{V_j}{\delta(L_j, L_{i,t})} \]  
Selfish / Individual  
(Low Factoredness, \(\infty\) Learnability)

\[ D_i = \sum_t \left[ \sum_j \frac{V_j}{\min_{i'} \delta(L_j, L_{i',t})} - \sum_j \frac{V_j}{\min_{i' \neq i} \delta(L_j, L_{i',t})} \right] \]  
Difference  
(High Factoredness, High Learnability)

**Episodic Learning**

Dynamic Environment  
30 Rovers

Dynamic Environment  
Scaling
Non-Episodic Learning

Dynamic Environment (30 Agents)

Dynamic Environment (Scaling)

Communication Limitations

Non-Episodic Dynamic Environment 70x75 Unit Env (30 Agents)
Communication Limitations

Non-Episodic Dynamic Environment 70x75 Unit Env (30 Agents)

Beyond Basic Coordination

- Recall key questions:
  - What if you don’t know how to set robot objective functions?
  - How to update them (team formation)?
  - How to modify them to changing objectives (reconfiguration)?
  - What happens when robots can’t compute those objectives?
  - What happens when information is missing?
  - What happens when some robots start to fail?
Approximating Difference Evaluations

• Computing difference evaluations requires:
  - Mathematical form of $G(z)$
  - Global knowledge about the state and actions of all agents

• We typically don’t have access to all this information!

• Assume an agent only knows:
  - Local state
  - Action taken
  - Value, but not functional form, of $G(z)$; team game assumption

Approximating Difference Evaluations

• Idea: each agent maintains a private approximation of $G(z)$
  - Inputs: agent state and action taken
  - Outputs: approximate value of $G(z)$

• At each time step
  - Record local state and action
  - Receive broadcast of $G(z)$ value
  - Use this information to update function approximation

  - Think of this as using a “linear” approximation to a nonlinear function
  - Accuracy depends on “sensitivity to locality” and approximation region
Approximating Difference Evaluations

- The approximation of difference evaluation is then computed:

\[
\tilde{D}_i(z) = G(z) - \tilde{G}(z_{-i})
\]

- Use broadcast for first term
- Use approximation for second term

Approximations in Rover Domain

- Each agent maintains neural network mapping local state and action to \(G(z)\)

\[
\tilde{G}_i(z) \equiv F(s_i, a_i, \bar{w}, f, h)
\]

- \(f(\cdot)\) : Neural network output
- \(s_i\) : state of agent \(i\) (more generally, states known to \(i\))
- \(a_i\) : action of agent \(i\) (more generally, actions known to \(i\))
- \(w\) : vector of weights
- \(f\) : activation functions (more generally, parameters of approximator)
- \(h\) : structure of neural network
Approximations in Rover Domain

• Each agent maintains neural network mapping local state and action to $G(z)$

• At each time step:
  - Record state and action
  - Record broadcast value of $G(z)$
  - Use back-propagation to update neural network for estimate of $G$

  - Now we have “new” estimate of $G(z)$

  - Use that to approximate $D_i(z)$:
Approximating $D_1(z)$ in Rover Domain

Approximating $D_1(z)$ in Rover Domain
Approximating $D_i(z)$ in Rover Domain

What About Alignment?

- Recall difference evaluations are 100% aligned with system evaluation
  - When we approximate $D_i(z)$, we lose the guarantee of alignment

- What is the relationship between alignment and performance?
  - 10 Agents: 94% Alignment $\rightarrow$ 88% of the performance of $D_i(z)$
  - 100 Agents: 78% Alignment $\rightarrow$ 79% of the performance of $D_i(z)$

- Intuition:
  - Approximation’s alignment and relative performance to $D_i(z)$ are coupled
  - Better alignment leads to better performance

Better approximation man lead to better performance
Key Issues in Multi-Robot Systems

- Communication among robots
- Homogeneity vs. heterogeneity
- Task assignment and specialization
- Computational limitations
- Reliability
- Robustness
- Control architecture
- Localization (with respect to other robots)
- Formation control
- Scalability

Summary

- Loosely coupled systems:
  - Focus on what each robot should do
  - Then let each robot do its thing

- Tightly coupled systems:
  - Focus on what each robot should do
  - Make sure the robots stick to their tasks

- Key: Don’t try to figure out what each robot has to do
  Just put their “incentives” right and let them work
Next Time

- Tensegrity Robots