Introduction to Multi-Robot Systems and Collective Movement

Lecture 1

Dr. Amanda Prorok
Assistant Professor, University of Cambridge
asp45@cam.ac.uk
www.proroklab.org
In this Lecture

• Introduction to multi-robot systems
• Taxonomy
• Collective movement
  ‣ Flocking (2 example methods)
  ‣ Formations (2 example methods)
From Single to Multi-Robot Systems

control architectures

perception → action → decision-making

localization → motion control → now (react) → later (plan)

mobile autonomy

basics of autonomy

navigation
From Single to Multi-Robot Systems

- **Multiple** mobile robots → multi-robot systems
- Higher-order goals
- Coordination facilitated through communication
Multi-Robot Systems

- Terms used: robot swarms / robot teams / robot networks
- Why?
  - Distributed nature of many problems
  - Overall performance greater than sum of individual efforts
  - Redundancy and robustness
- Numerous commercial, civil, military applications
- How to coordinate, cooperate, collaborate, (compete?)

Application Domains

- Sensor and communications networks
- Multi-agent robotics
- Coordinated control
- Biological networks
  - surveillance / monitoring
  - product pickup / delivery
  - search & rescue
Example 1: Coordination
Example 2: Cooperation
Example 3: Collaboration

* movie credit: R. D’Andrea et al.
Taxonomy

- **Architecture**: centralized vs. decentralized
  - Centralized: one control/estimation unit communicates with all robots to issue commands; requires synchronized, reliable communication channels; single-point failures
  - Decentralized: scalable, robust to failure; often asynchronous; sub-optimal performance (w.r.t centralized)

- **Communication**: explicit vs. implicit
  - Implicit: observable states (e.g., in the environment); information exchanged through common observations
  - Explicit: unobservable states; need to be communicated explicitly

- **Heterogeneity**: homogenous vs. heterogeneous
  - Robot teams can leverage inter-robot complementarities
Communication Topologies

- Robot configurations / topologies are often defined by the maximum range of the available communication module (careful!).
- A disc model can be used to represent the communication range (very crude approximation)

- fully connected
  - centralized / decentralized coordination

- star topology
  - centralized / decentralized coordination

- random mesh
  - decentralized coordination
Centralization vs Decentralization

- **Centralized control.** The controller computes actions based on knowledge of the global state.
- **Centralized estimation.** The unit fuses partial information.

- **Decentralized control.** A robot’s control input is based on interactions with its neighbors.
- **Decentralized estimation.** The robot’s estimate is based on relative observations.
Centralization vs Decentralization

automated warehouses
- min. time to product dispatch

search & rescue / surveillance
- max. area coverage / min. time to target

automated mobility-on-demand
- min. time to passenger pickup

connected autonomous vehicles
- max. throughput / min. collision probability
Decentralization

- **Goal:** Achieve similar (or same) performance as would be achievable with an ideal, centralized system.

- **Challenges:**
  - Communication: delays and overhead
  - Input: asynchronous; with rumor propagation
  - Sub-optimality with respect to the centralized solution

- **Advantages:**
  - No single-point failure
  - Can converge to optimum as time progresses
  - ‘Any-comm’ algorithms exist (graceful degradation under failing comms)
  - ‘Any-time’ algorithms exist (continuous improvement of solution)
Collective Movement

In nature:

- flock of birds
- flock of geese
- school of fish
- herd of mammals
Collective Movement

• Collective movement in natural societies:
  ‣ Properties: no collisions; no apparent leader; tolerance of loss or gain of group member; coalescing and splitting; reactivity to obstacles; different species have different flocking characteristics
  ‣ Benefits: energy saving (e.g., geese extend flight range by 70%); signs of better navigation accuracy

• Engineered flocking - decentralized:
  ‣ Reynolds’ virtual agents (Boids)
  ‣ Graph-based distributed control for spatial consensus

• Engineered flocking - centralized:
  ‣ E.g.: Controls for each robot computed off-board, in the cloud
Flocking with Boids

- In 1986, Craig Reynolds (computer animator) wanted to create a computationally efficient method to animate flocks
- Goal: $O(N)$; current best was $O(N^2)$

- A boid reacts only to its neighbors
- Neighborhood defined by distance and angle (region of influence)
- Each boid follows **3 steering rules** based on positions and velocities of neighbors. Recipe: compute 3 components, then combine to form motion (vector)
Flocking with Boids

• Sensory system: idealized, but local:
  ‣ almost omni-directional
  ‣ no delays (in sensing)
  ‣ no noise (in range and bearing)

• Behavior-based with priorities (cf Brooks’ subsumption):
  ‣ Low priority acceleration request towards a point or in a direction (to direct flock)
  ‣ Highest priority to obstacle avoidance (‘steer-to-avoid’ with a different sensory system)
Flocking with Boids

more info on http://www.red3d.com/cwr/boids/
Flocking with Consensus

1 leader robot; robots apply consensus algorithm to agree on heading
Aim of consensus:
  - Reach decentralized agreement
  - Purely based on local interactions

Consensus
  - Based on a graph-topological definition of multi-robot system
  - Applications: motion coordination; cooperative estimation; synchronization

Discrete time consensus update:

\[ x_i[t + 1] = \frac{1}{|\mathcal{N}_i| + 1} (x_i[t] + \sum_{j \in \mathcal{N}_i} x_j[t]) \]

Consensus outcome:
  - All robots converge to average of initial values (convergence rate is exponential):

\[ t \to \infty, \quad x_i[t] = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} x_i[0] \]
Flocking with Consensus

Holonomic robot: \( \dot{x} = u \) with \( x_i = [x_i, y_i] \)

Consensus on heading \( \theta_i \) with a leader agent

Note: Collision avoidance and connectivity maintenance are needed in addition to agreement on direction of motion.
Other Consensus Applications

- rendezvous
- cyclic pursuit
- flocking
- configuration
Formation Control

• Formations (versus flocks): **specific** geometric configurations
• Some applications benefit from multiple robots navigating as a group:
  ‣ Transport (vehicle formations; platooning); scout platoons for reconnaissance and search; environmental monitoring; lawn mowing
• Generally required: information on state (e.g. pose) of all robots
• Challenges:
  ‣ Noisy sensors; delay in sensing / actuation
  ‣ Anonymous robots (no IDs)
  ‣ Non-holonomicity
• Variants:
  ‣ Behavior-based (Balch et al., 1999) (recall: reactive control paradigm)
  ‣ Closed-loop control (Das et al., 2002) (recall: error-based control paradigm)

e.g.: diamond formation
Formation Control

• Referencing schemes:
  
  ‣ **Unit-center-referenced**: obtained by averaging positions of all robots. A robot determines its position relative to this center.
  
  ‣ **Leader-referenced**: robots determine pose relative to leader, which does not attempt to maintain the formation.
  
  ‣ **Neighbor-referenced**: robots attempt to maintain relative pose to one (or a select group) of neighboring robots.

![Diagram showing unit-center, leader, and neighbor referencing schemes](image)

• How is positioning information obtained?
  
  ‣ Each robot estimates its own pose, and communicates this to other robots.
  
  ‣ Or: robots estimate their relative pose via sensor observations

*image credit: Balch 1999*
Behavior-Based Formation Control

- Method based on ‘Motor-Schema’ [Balch, Arkin; 1999]
- Different motor schemes are defined; each generates a vector representing a behavioral response (direction and magnitude of movement) as a function of sensor stimuli (recall lecture on architectures)
- A gain value is used to attribute relative importance of schemes

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>avoid-static-obstacle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gain</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>sphere of influence</td>
<td>50</td>
<td>meters</td>
</tr>
<tr>
<td>minimum range</td>
<td>5</td>
<td>meters</td>
</tr>
<tr>
<td>avoid-robot</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gain</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>sphere of influence</td>
<td>20</td>
<td>meters</td>
</tr>
<tr>
<td>minimum range</td>
<td>5</td>
<td>meters</td>
</tr>
<tr>
<td>move-to-goal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gain</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td>noise</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gain</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>persistence</td>
<td>6</td>
<td>time steps</td>
</tr>
<tr>
<td>maintain-formation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gain</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>desired spacing</td>
<td>50</td>
<td>meters</td>
</tr>
<tr>
<td>controlled zone radius</td>
<td>25</td>
<td>meters</td>
</tr>
<tr>
<td>dead zone radius</td>
<td>0</td>
<td>meters</td>
</tr>
</tbody>
</table>

*image credit: Balch 1999*
Behavior-Based Formation Control

**maintain-formation**: decomposed into two parts

**maintain-formation-speed**

\[ V_{speed} = R_{mag} + K \times \delta_{speed} \]

**maintain-formation-steer**

\[ H_{desired} = F_{dir} - \delta_{heading} \]

\[ V_{steer} = H_{desired} - R_{dir} \]

- \( R_{pos}, R_{dir} \): the robot’s present position and heading.
- \( R_{mag} \): the robot’s present speed.
- \( F_{pos} \): the robot’s proper position in formation.
- \( F_{dir} \): the direction of the formation’s movement; towards the next navigational waypoint.
- \( F_{axis} \): the formation’s axis, a ray passing through \( F_{pos} \) in the \( F_{dir} \) direction.
- \( H_{desired} \): desired heading, a computed heading that will move the robot into formation.
- \( \delta_{heading} \): the computed heading correction.
- \( \delta_{speed} \): the computed speed correction.
- \( V_{steer} \): steer vote, representing the directional output of the motor behavior, sent to the steering arbiter.
- \( V_{speed} \): speed vote, the speed output of the motor behavior, sent to the speed arbiter.

[Balch, Arkin; 1999]
Behavior-Based Formation Control

Example of results, for leader-referenced scheme [Balch ’99]:

Assumptions:
• fully networked system; robots have IDs (non-anonymous)
• robot positioning with little noise and delay
• straight-forward implementation for holonomic (point-) robots

*image credit: Balch 1999
Formation Control

- Non-holonomic robots:
  - Proposed method: *fore-aft / side-side corrections*
  - Separate motor behaviors a generated for steering / speed. *Arbiters* accept votes from the motor schemas to compute speed / steering values.
  - Combined with a rule-based program that selects final speed / steering value.

- Issues:
  - Behavior-based methods have no guarantees:
    - Convergence to desired formation? Stability of formation?
    - **Need for more principled approaches**

- Introduction of control-theoretic principles to provide these guarantees
  - One of the first such approaches presented by Das et al., 2002

*image credit: Das 2002*
Closed-Loop Control for Formations

- Method based on feedback linearization [Das et al., 2002]
- Basic case: leader-referenced control based on separation distance and relative bearing: \( z_{ij} = [l_{ij}, \psi_{ij}]^T \)

Control input: \( u_j = [v_j, \omega_j]^T \) (forwards and rotational velocities)

**Aim:** Find \( u_j \) such that desired separation \( l^d_{ij} \) and desired bearing \( \psi^d_{ij} \) are reached, and stably maintained.
Closed-Loop Control for Formations

Dynamical system model: \[ \dot{z}_{ij} = G \, u_j + F \, u_i \]

with:

\[
G = \begin{bmatrix}
\cos \gamma_{ij} & d \sin \gamma_{ij} \\
-d \sin \gamma_{ij} & d \cos \gamma_{ij}
\end{bmatrix}
\]

\[
F = \begin{bmatrix}
-cos \psi_{ij} & 0 \\
\sin \psi_{ij} & -1
\end{bmatrix}
\]

where relative orientation is: \[ \beta_{ij} = \theta_i - \theta_j \quad \text{and} \quad \gamma_{ij} = \beta_{ij} + \psi_{ij} \]

Proportional control law: \[ \dot{z}_{ij} = k(z_{ij}^d - z_{ij}) \]

Control: \[ u_j = G^{-1} \left( k(z_{ij}^d - z_{ij}) - Fu_i \right) \]

This guarantees convergence to desired relative state \( z_{ij}^d \)

(Stability is proven in paper.)
Closed-Loop Control for Formations

By an analysis similar to Theorem 2, the control system will show that the closed-loop system is stable and the robots converge to a neighborhood of the origin. Robust control theory provides a systematic way of approaching this problem analytically. As can be shown to be stable (see [20] for details), the internal dynamics of a robot, where the relative velocities and orientations of individual robots and sensor errors are bounded, the system errors are also bounded. 

Robots need individual gains, and back linearization is applied to the desired value for nonholonomic robots with input–output feedback linearized along the path that minimizes the length of leader-follower chains (we prefer control assignments that are otherwise similar, we prefer the one a simple heuristic: when deciding between two formation constructions, we prefer the one that minimizes the formation shape errors for the control graph (a).

In Section II, we have shown that under certain assumptions, the orientation error for nonholonomic robots with input–output feedback linearized can be shown to be bounded under the assumptions of the theorem.
Closed-Loop Control for Formations

Four robots with omnidirectional cameras:

An obvious concern regarding stability of the formation arises when the formation geometry is compromised by an obstacle. To resolve this, the formation shape is achieved and the robots successfully negotiate the obstacle. During the course of this, an equilateral triangle is formed by exploiting the triangle constraint. However, we need a representation of an environment. We use directed graphs to accomplish this.

Consider the case of a triangular formation approaching a narrow passage through obstacles shown in Fig. 10. A formation shape controller (C1) is designed to follow the team leader (C0), and select controllers based on the absolute scale of the robots.
A Figure 8 with Range & Bearing

*movie credit: Gowal, Martinoli, EPFL
Further Reading

Papers:

• Behavior-Based Formation Control for Multi-Robot Teams; T Balch, R Arkin; 1999

• A Vision-Based Formation Control Framework; A K. Das, R Fierro, R. V Kumar, J P. Ostrowski, J Spletzer, C J. Taylor; 2002

• Consensus and cooperation in networked multi-agent systems; Olfati-Saber, Fax, Murray; 2007
Task Assignment in Multi-Robot Systems

Dr. Amanda Prorok
Assistant Professor, University of Cambridge
asp45@cam.ac.uk
www.proroklab.org
In this Lecture

- Motivation: task allocation in nature
- Assignment algorithms:
  - Hungarian method
  - Swarm distribution mechanisms
  - Market-based
  - Threshold-based
- Credit:
  - Threshold-based example from A. Martinoli’s course at EPFL
In nature: Physical castes

In *Pheidole guilelmuelleri* the minors show ten times as many different basic behaviors as the majors.

Behavioral repertoire of majors and minors: In *Pheidole guilelmuelleri* the minors show ten times as many different basic behaviors as the majors.

*Image credit: Alcherio Martinoli*
In nature: temporal polyethism

Behavioral change in worker bees as a function of age; young individuals work on internal tasks (brood care and nest maintenance), older workers forage for food and defend the nest.

*image credit: Alcherio Martinoli*
Task Allocation vs Division of Labor

In robotics:

- Monitoring
- Mobility on Demand
- Warehousing and Product Delivery
- Situational Awareness
Assignment Problems
Assignment Problems
Assignment Problems
Assignment Problems

[Kumar et al.; UPenn]
The Assignment Problem

• Which robot goes where? Which robot does what?

• What is a task?
  ‣ Discrete: e.g., pickup parcel X from location Y, ...
  ‣ Continuous: e.g., monitor building X, search area Y...
  ‣ Key assumption: task independence
    (dependent tasks $\rightarrow$ scheduling)

• Assignment methods are drawn from multiple fields:
  ‣ operations research, economics, scheduling, network flows, combinatorial optimization.

• Classical problem formulation: bipartite graph matching
The Assignment Problem

• What is to be optimized? **Utility**: an individual robot knows the value of executing a certain action.

• Utility, depending on context: value, cost, fitness. Knowing the true (exact) utility is key to finding an optimal assignment.

• Various formulations exist. For example:

\[
U(R, T) = \begin{cases} 
Q_{RT} - C_{RT} & \text{if } R \text{ is capable of executing } T \text{ and } Q_{RT} > C_{RT} \\
0 & \text{otherwise}
\end{cases}
\]

![Diagram](U11_U21_R1_R2_T1.png)
The Linear Assignment Problem

• In an optimal assignment problem, maximize the system performance:

\[
\mathcal{U} = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} U(i, j)
\]

subject to

\[
\sum_{i=1}^{m} x_{ij} = 1, \quad 1 \leq j \leq n
\]

\[
\sum_{j=1}^{n} x_{ij} = 1, \quad 1 \leq i \leq m
\]
The Hungarian Algorithm

- Published by Kuhn in 1955, based on the earlier works of two Hungarian mathematicians: Dénes Kőnig and Jenő Egerváry.
  - $O(n^3)$ running time is possible.
- Steps (input is an $n \times n$ by matrix with non-negative elements):
  - **Step 1**: Subtract row minima; For each row, find the lowest element and subtract it from each element in that row.
  - **Step 2**: Subtract column minima; Similarly, for each column, find the lowest element and subtract it from each element in that column.
  - **Step 3**: Cover all zeros with a minimum number of lines; Cover all zeros in the resulting matrix using a minimum number of horizontal and vertical lines. If $n$ lines are required, an optimal assignment exists among the zeros. The algorithm stops. If less than $n$ lines are required, continue with Step 4.
  - **Step 4**: Create additional zeros; Find the smallest element (call it $k$) that is not covered by a line in Step 3. Subtract $k$ from all uncovered elements, and add $k$ to all elements that are covered twice. Go to Step 3.
The Hungarian Algorithm - Example

Step 0: robot-task assignment costs

Step 1: subtract row minima

Step 2: subtract column minima

Step 3: cover all zeros with a minimum of lines

Step 4: create additional zeros

Stop: An optimal assignment exists.

-6: unmarked elements
+6: twice marked elements

(find smallest uncovered element)

*Example from www.hungarianalgorithm.com
Application: Vehicle-to-Passenger Assignment

Goal: find optimal assignment matrix $A^*$

$$A^* = \arg\min_A \sum_{i=1}^{N} \sum_{j=1}^{M} c_{i,j} a_{i,j}$$

Publicly available data:
- OpenStreetMap for whole area
- Convert to graph (4302 vertices, 9414 edges)
- Cost of an assignment ~ distance (time)
- NYC public taxicab dataset

*see Prorok et al. IROS 2017 for an example*
The Hungarian Algorithm

• Assumptions when using an assignment algorithm such as the Hungarian method:
  ‣ Costs (utilities) are known at a centralized computation unit.
  ‣ Costs (utilities) are deterministic (no noise).
  ‣ Costs (utilities) do not change (constant).
  ‣ 1-to-1 assignment (one robot per task, one task per robot).

• Complications:
  ‣ Uncertainty around true utility \( U(i,j) \) **
  ‣ Dynamic environment (changes in utility / agents)
  ‣ Robot / task dependencies (robot heterogeneity / redundancy).

• Consequences:
  ‣ Sub-optimality
  ‣ Problems can become NP-hard (for combinatorial matching problems)
  ‣ Practically infeasible (centralized solutions may not be possible)

**see Prorok, DARS 2018 for a solution

all of these issues are very common in robotics!!
Assignment of Robot Coalitions

Some tasks require more than 1 robot.

How many ways to partition \( n \) robots into \( k \) non-empty subsets?

Given by the Stirling number of the second kind.

E.g.: 10 robots, 5 tasks: \( S(10,5) = 42'525 \)
Assignment of Robot Coalitions

The problem of forming **robot coalitions**:

- $E$ is the ground set (all robots) and $X$ is a family of subsets.

\[ y \cap z = \emptyset \quad \forall y, z \in X, y \neq z \quad \text{robot subsets are mutually disjoint} \]

\[ \bigcup_{x \in X} x = E \quad \text{the union of subsets is equivalent to the ground set.} \]

**Set Partitioning Problem:** Given a finite set $E$, a family $F$ of acceptable subsets of $E$, and a utility function $u : F \mapsto \mathbb{R}_+$, find a maximum-utility family $X$ of elements in $F$ such that $X$ is a partition of $E$.

The set-partitioning problem is **strongly NP-hard**. [Garey and Johnson; 1978]

... One potential solution: *relaxation of the problem to the continuous domain.*
Countable vs Uncountable Systems

- Difference between a multi-robot system and a robot swarm?
- Swarms are larger, but how large...?
- The method is the key!

- robot-to-task allocation
  - method: **combinatorial approach**
  - exact, but computationally demanding

- redistribution of robots among tasks
  - method: **mean-field approach**
  - approximative, but fast
Redistribution of a Swarm of Robots

Example: monitor geographical sites

- task 1
- task 2
- task 3
- task 4
- task 5

translating frequency
Redistribution of a Swarm of Robots

Model: connected tasks

- Task 1
- Task 2
- Task 3
- Task 4
- Task 5
Redistribution of a Swarm of Robots

Model: connected tasks

What proportion of robots of each kind?

*note: for the purpose of this lecture, assume non-overlapping robot traits
Redistribution of a Swarm of Robots

Insight: we can model the distribution dynamics of the robot swarm as a linear dynamical system!

System state, e.g.: \( \mathbf{x} = [0.3, 0.2, 0.1, 0.1, 0.3]^T \)

Note: if matrix \( \mathbf{K} \) has certain properties, this system is stable.
Redistribution of a Swarm of Robots

Robot distribution dynamics:

\[ \dot{x}^{(s)} = K^{(s)} x^{(s)} \]

\( \rightarrow \)

rates \( M \times M \)
robots \( M \times 1 \)

Solution:

\[ x^{(s)}(t) = e^{K^{(s)} t} x_0^{(s)} \]

---

**Given** a desired robot distribution \( x^{(s)*} \)

**Find** transition rates \( K^{(s)*} \) that are fastest to satisfy \( x^{(s)*} \)

Methods:
1. Explicit optimization; [Prorok 2016]
2. Approximation of \( K \); semi-definite programming [Berman 2009]
Controller Synthesis

- Probabilistic controller is immediate
- Deterministic controller can also be derived
- Architecture: both open-loop and closed-loop possible

We extract rates for task-to-task transitions $k_{ij}^{(s)}$, and directly infer the switching probability.
Redistribution of a Heterogeneous Swarm

Time elapsed: 2.40

Species 0: [Colors]
Species 1: [Colors]
Species 2: [Colors]
Species 3: [Colors]

[Prorok et al.; ICRA 2016; T-RO 2017]
Redistribution of a Heterogeneous Swarm

[Prorok et al.; ICRA 2016]
Market-Based Coordination

- Robots: “self-interested agents that operate in a virtual economy”
- Tasks: “commodities of measurable worth that can be traded”

Example scenario: three robots exploring Mars. The robots need to gather data around the craters; they need to visit the 7 highlighted sites. Which robot visits each site?

*image credit: Dias et al.*
Market-Based Coordination

• Underlying mechanism: **auctions**
• Auctioneer: offers items (tasks or resources) in announcement
• Participants (robots) submit bids to negotiate allocation of items
  ‣ sealed-bid vs. open-cry
  ‣ first-price vs. Vickrey auction

• **Single-item** auction:
  ‣ highest bidder wins task
  ‣ if no bid beats reserve-price, then auctioneer can retain item

• **Combinatorial** auction:
  ‣ multiple items, robots bid on bundles
  ‣ a bid expresses synergies between items

• **Multi-item** auction:
  ‣ a robot can win at most one item apiece
  ‣ special case of combinatorial auction for bundle of size 1
A simple example (multi-item auction)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robot 1</td>
<td><img src="" alt="image" /></td>
<td>50</td>
</tr>
<tr>
<td>Robot 2</td>
<td>100</td>
<td>70</td>
</tr>
</tbody>
</table>

reward: 120
profit: 70 = 120-50
profit: 80 = 150-70

allocation cost

system cost: 50+70 = 120

Running time: $O(NRM)$ (greedy) or $O(N^2R)$ (optimal) [T. Sandholm; 2002]
Market-Based Allocation Frameworks

- Murdoch [Gerkey, Mataric; 2002]
  - loosely coordinated tasks
  - demonstrated on box pushing
  - demonstrated robustness, fast auctioning

- TraderBots [Dias et al.; 2004]
  - loosely coordinated tasks
  - demonstrated on exploration tasks
  - demonstrated robustness, scalability, auction types, task trees

- Hoplites [Kalra, Stentz; 2005]
  - tightly coordinated spatial tasks
  - robots auction plans not tasks
  - demonstrated on perimeter sweeping, constrained exploration
Centralized assignment. Cost estimates are known at a central point (computational unit). The unit performs the assignment and communicates with all robots.

Decentralized assignment. Robots do not have global knowledge of each other’s costs. They locally negotiate assignments.

Hybrid mechanisms: locally defined robot cliques can elect ‘leader’ robots and perform centralized mechanisms.
Threshold-Based Assignment

- Fully **decentralized** mechanism.
- Each robot has an **activation threshold** for each task that needs to be performed.
- A **stimulus**:
  - reflects the urgency of a task
  - continuously perceived **locally** by each individual robot
- Example: threshold-based control of *aggregation* [Agassounon, Martinoli; 2002]
  - Goal: aggregate all sticks into 1 cluster
  - End criterion: robots should stop working once task is achieved
Threshold-Based Assignment

- **Stimulus**: time needed to find a stick to manipulate (the longer the time, the lower the stimulus associated with the task).
- **Threshold** is self-calibrated (fully decentralized).
- Key: The number of manipulation sites (either end of line of sticks) decreases as global task nears completion.
- If time to find next stick goes beyond threshold $T$, then agent switches to resting behavior.

$$T = f \cdot \frac{1}{K} \sum_{k=1}^{K} t_k$$

*image credit: Agassounon et al.*
## Overview of Allocation Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>centralized vs decentralized</th>
<th>optimality</th>
<th>completeness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hungarian method</td>
<td>centralized</td>
<td>optimal</td>
<td>guaranteed</td>
</tr>
<tr>
<td>Mean-field approach</td>
<td>centralized or decentralized</td>
<td>approximative</td>
<td>The system converges. With high probability, completeness is guaranteed</td>
</tr>
<tr>
<td>Market-based approach</td>
<td>centralized or decentralized</td>
<td>greedy (sub-optimal) or optimal</td>
<td>depends on reserve price</td>
</tr>
<tr>
<td>Threshold-based approach</td>
<td>decentralized</td>
<td>suboptimal</td>
<td>not guaranteed</td>
</tr>
</tbody>
</table>

Lecture 2: Task Assignment in Multi-Robot Systems
Further Reading

Nice overview of the classical problem:
http://www.assignmentproblems.com/

Seminal papers:

- M. B. Dias et al; “Market-Based Multirobot Coordination: A Survey and Analysis”; 2006
- N. Kalra, A. Martinoli, “Comparative study of market-based and threshold-based task allocation”; 2006

Some new approaches for those interested:

- **Redundant robot assignment under uncertainty**: A. Prorok, Redundant Robot Assignment on Graphs with Uncertain Edge Costs, 14th International Symposium on Distributed Autonomous Robotic Systems (DARS), 2018
- **Assignment under privacy constraints**: A. Prorok, V. Kumar, Privacy-Preserving Vehicle Assignment for Mobility-on-Demand Systems, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017
Multi-Robot Navigation and Path Planning

Lecture 3

Dr. Amanda Prorok
Assistant Professor, University of Cambridge
asp45@cam.ac.uk
www.proroklab.org
In this Lecture

• Taxonomy of MR path planning problems
• MR path planning methods:
  ‣ Discrete
  ‣ Continuous
• Concurrent assignment and path planning
Taxonomy of Multi-Robot Path Planning Problems

• Domain: continuous vs. discrete
  ‣ Continuous: planning time-parameterized trajectories in metric space.
  ‣ Discrete: planning on graphs, or regular grids
• Goal assignment: labeled vs. unlabeled
  ‣ Labeled: each robot has a predetermined goal destination
  ‣ Unlabeled: all goals must be reached, but assignment is not predetermined
• Problem representation: coupled vs. decoupled
  ‣ Coupled: represent the joint state of all robots in the system
  ‣ Decoupled: each robot’s state represented independently
• Planning: reactive vs. deliberative
  ‣ Reactive: dynamic obstacle avoidance; plan as you go (cf. decentralized)
  ‣ Deliberative: planning for optimality (cf. centralized, coupled)
• Computation: centralized vs. decentralized
Multi-Agent Path Planning

- Multi-robot path planning $\rightarrow$ multi-agent path planning:
  - discretized environment (grids or planar graphs)
  - point robots (holonomic, no motion constraints)

- The problem:
  - Given: a number of agents at start locations with predefined goal locations, and a known environment
  - Task: find **collision-free paths** for the agents from their start to their goal locations that optimize some objective

- Generally, we assumed a **labeled** problem.

- Classical application domain: automated warehouses (e.g., Amazon)
Multi-Agent Path Planning

- Allowed motion: North, East, South, West
- Collisions:
  - Vertex-collision
  - Edge-collision
  - No collision

- Performance metrics
  - **Makespan**: time of last robot’s arrival time
  - **Flowtime**: sum of arrival times, over all robots
Coupled vs Decoupled Path Planning

- Coupled planning provides completeness.
- Decoupled path planning is not complete, in general.
Coupled Path Planning

Coupled formulation:

Robot $i$ has configuration space: $C_i$

The joint state space is given by the Cartesian product:

$$X = C_1 \times C_2 \times \ldots \times C_n$$

The dimensionality grows **linearly** w.r.t. the number of robots. Complete algorithms (such as A*) require time that is at least **exponential** w.r.t. the search space dimension!
Coupled Path Planning

Coupled formulation for $N$ robots and $M$ cells in grid-world:

For $M$ possible states in each configuration space, we have $M^N$ states in the coupled system.

E.g., worst case complexity for A*:

$$O(|E|) \approx O(|V|) = O(M^N)$$

Exponential complexity in the number of robots!

* if graph is sparse
Coupled Path Planning

- Hardness: **NP-hard to solve optimally** for makespan or flowtime minimization [Yu and LaValle; 2013]
- It is impossible to minimize both objectives simultaneously (Pareto)
- But: coupled method provides **completeness** and **optimality**
  - Lots of attention devoted to this field
  - Development of approximate solutions (see literature by Sven Koenig; Howie Choset; Maxim Likhachev)
Decoupled path planning is not complete, in general.

But: in well-formed environments, prioritized decoupled planning is complete!

- Well-formed environment: goals are distributed in such a way that any robot standing on a goal cannot completely prevent other robots from moving between any other two goals.

[Cap, Novak, Klaeiner, Selecky; 2015]
Decoupled Path Planning

- Well-formed environment:
  - There must exist a path between any two endpoints.
  - That path must have at least $R$-clearance with respect to static obstacles and at least $2R$-clearance to any other endpoint.
  - A robot is always able to find a collision-free trajectory to its goal by waiting for other robots to reach their goals, and then following a path around those occupied goals (any prioritization works!).
Decoupled Path Planning

- De-coupling the problem:
  - Each robot plans in its own space-time
  - Robots negotiate path plans as conflicts arise
  - De-confliction can be online (dynamic) or offline (a-priori)
Decoupled, Prioritized Path Planning

The red robot is prioritized and plans a space-time path that is optimal. The blue robot plans a path that does not collide with the red robot’s path.

[Received February 1, 2019.]

[Confidential. Limited circulation. For review only.]

[Wu, Bhattacharya, Prorok; arxiv 2019]
Decoupled, Prioritized Path Planning

• Key question: How to prioritize robots?

• Online, exhaustive method:
  ‣ Evaluate all $N!$ options (where $N$ is robots within communication or visibility neighborhood) [Azarm, Schmidt; 1997]

• Existing prioritization heuristics (online and offline):
  ‣ Ideal path length: Robots with longer ideal path length have higher priority. [Van den Berg et al.]
  ‣ Planning time: Robots that take longer to plan their paths get higher priority. [Velagapudi, Sycara, Scerri; 2010]
  ‣ Workspace clutter: Robots with more clutter in local vicinity have higher priority. [Clark, Bretl, Rock; 2002]
  ‣ Path prospects: Robots with fewer path options have higher priority [Wu, Bhattacharya, Prorok; 2019]
Decoupled, Prioritized Path Planning

Example of a multi-agent system where agents have heterogeneous sizes. Agents with fewer path prospects are prioritized.
The Continuous Domain

*movie credit: Gowal, Martinoli
Minkowski Sum

- In geometry, the Minkowski sum (also known as dilation) of two sets of position vectors $A$ and $B$ in Euclidean space is formed by adding each vector in $A$ to each vector in $B$, i.e., the set:

$$A \oplus B = \{a + b \mid a \in A, b \in B\}$$
Minkowski Sum

As long as reference point stays outside dilated area, there will be no collisions.
Velocity Obstacle Method

[Fiorini, Shiller; 1998]

Two robots, $A$ and $B$, translating in space. Will they collide?
Two robots, \( A \) and \( B \), translating in space. Will they collide?
Step 1: inflate robot B by area of robot A.

\[
VO_B^A(v_B = 0)
\]

\[-A \oplus B\]

\[v_B\]

\[v_A\]
Velocity Obstacle Method

Step 2: determine whether $v_A$ lies in the velocity obstacle of $B$ to $A$
If $v_A$ is outside the VO, then the robots will never collide.
Equivalence: \( v_A \) lies in the velocity obstacle of \( B \) to \( A \) \( \rightarrow \) the relative velocity \( v_A - v_B \) lies in the velocity obstacle of \( B \) to \( A \), assuming \( B \) does not move.
Compute set of admissible accelerations for robot A.
Velocity Obstacle Method

Check that new velocity is outside VO.
Velocity Obstacle Method

• Assumptions:
  ‣ Robots share their current (noise-free) position and velocity
  ‣ Robots truthfully execute reported velocities

• Complications:
  ‣ Oscillations! Scenario: Robots with current velocities \( v_A \) and \( v_B \) currently lie in each others VOs. Both robots select new \( v'_A \) and \( v'_B \) such that new velocities lie outside respective VOs. In new situation, the old velocities \( v_A \) and \( v_B \) lie outside VOs. If \( v_A \) and \( v_B \) are preferable (e.g., they lie on direct path to goal), they will be chosen again, hence, leading to oscillations.
  ‣ Solution: See reciprocal velocity obstacle method.
Reciprocal Velocity Obstacle Method

Idea: Choose a new velocity that is the average of its current velocity and a velocity that lies outside the other agent’s velocity obstacle. [Van den Berg, Lin, Manocha; 2008]

Choosing the closest velocity outside the other agent’s RVO guarantees oscillation-free navigation.

The RVO of B to A contains all the velocities of A that are the average of the current velocity $\mathbf{v}_A$ and a velocity inside the VO of B to A.

Geometric interpretation: the apex of the RVO lies at:

$$\frac{\mathbf{v}_A + \mathbf{v}_B}{2}$$

The old velocity of A is inside the new RVO of B to A, given the new velocities.
The following video shows 12 agents that move to their diametrically opposite position on the circle.

Four Corners
Two more robots are added with goals on the other diagonal.

[D. Manocha et al.]
Concurrent Assignment and Planning of Trajectories

- New problem formulation:
  - $N$ robots need to reach $N$ goal locations as efficiently as possible: we want to find the assignment as well as generate the trajectories, simultaneously.
  - Un-labeled problem (any robot may go to any goal)
  - Robots must have collision-free trajectories

- Assumptions:
  - Robots have a minimum separation distance at start / goal locations
  - Robots are holonomic and arrive simultaneously at goals
Given start and goal locations, find assignments **AND** trajectories that are optimal and collision-free
Concurrent Assignment and Planning of Trajectories

Given start and goal locations, find assignments **AND** trajectories that are optimal and collision-free
Concurrent Assignment and Planning of Trajectories

What is the **optimization objective**?

**Sum of distances:**

**Sum of distances squared:**

[Reference: Turpin et al.; IJRR 2013]
Concurrent Assignment and Planning of Trajectories

Objective:  
\[ \min_{\phi, \gamma(t)} \sum_{i=1}^{N} \int_{t_0}^{t_f} \dot{x}_i(t)^T \dot{x}_i(t) dt \]

Key result:  
If separation distance between any start and goal location is \( \Delta > 2\sqrt{2}R \) we can guarantee collision-free trajectories.

Solve assignment:  
\[ \phi^* = \arg\min_{\phi} \sum_{i=1}^{N} \sum_{j=1}^{N} \phi_{i,j} D_{i,j} \]

cost: distance squared

[Turpin et al.; IJRR 2013]
Further Reading

Fundamental planning concepts:

• Some of the planning concepts in Steven LaValle’s book.

Seminal papers:

• P. Fiorini and Z. Shiller, “Motion planning in dynamic environments using velocity obstacles”; 1998
• J. van den Berg, M. Lin, D. Manocha; “Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation”; 2008
• J. Van Den Berg, M. Overmars. "Prioritized motion planning for multiple robots." 2005

More recent papers:

• M. Turpin, N. Michael and V. Kumar; “CAPT: Concurrent assignment and planning of trajectories for multiple robots”; IJRR 2013