2019 IEEE RAS Summer School on Multi-Robot Systems

Introduction to Multi-Robot Systems and Collective Movement

Lecture 1

Dr. Amanda Prorok

Assistant Professor, University of Cambridge

asp45@cam.ac.uk

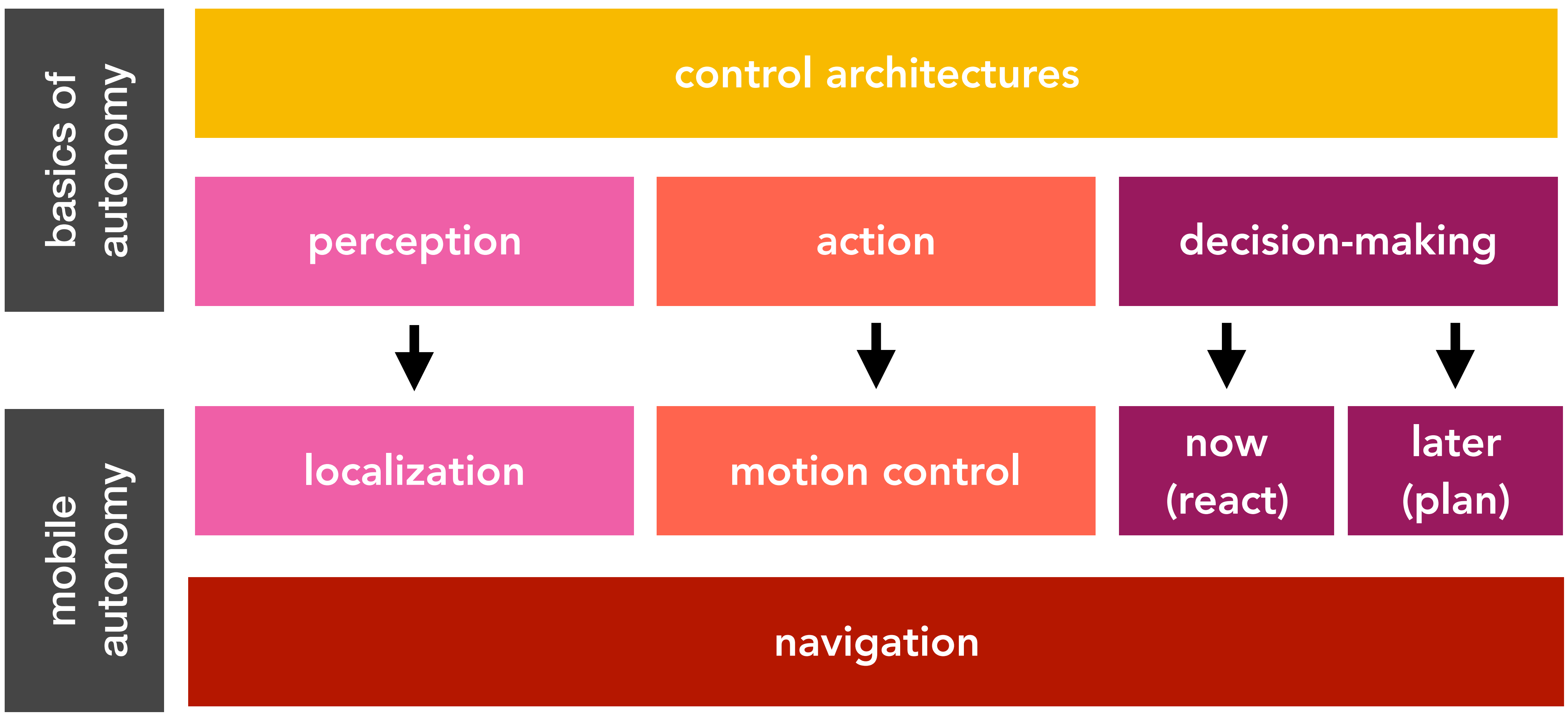
www.proroklab.org

UNIVERSITY OF CAMBRIDGE

In this Lecture

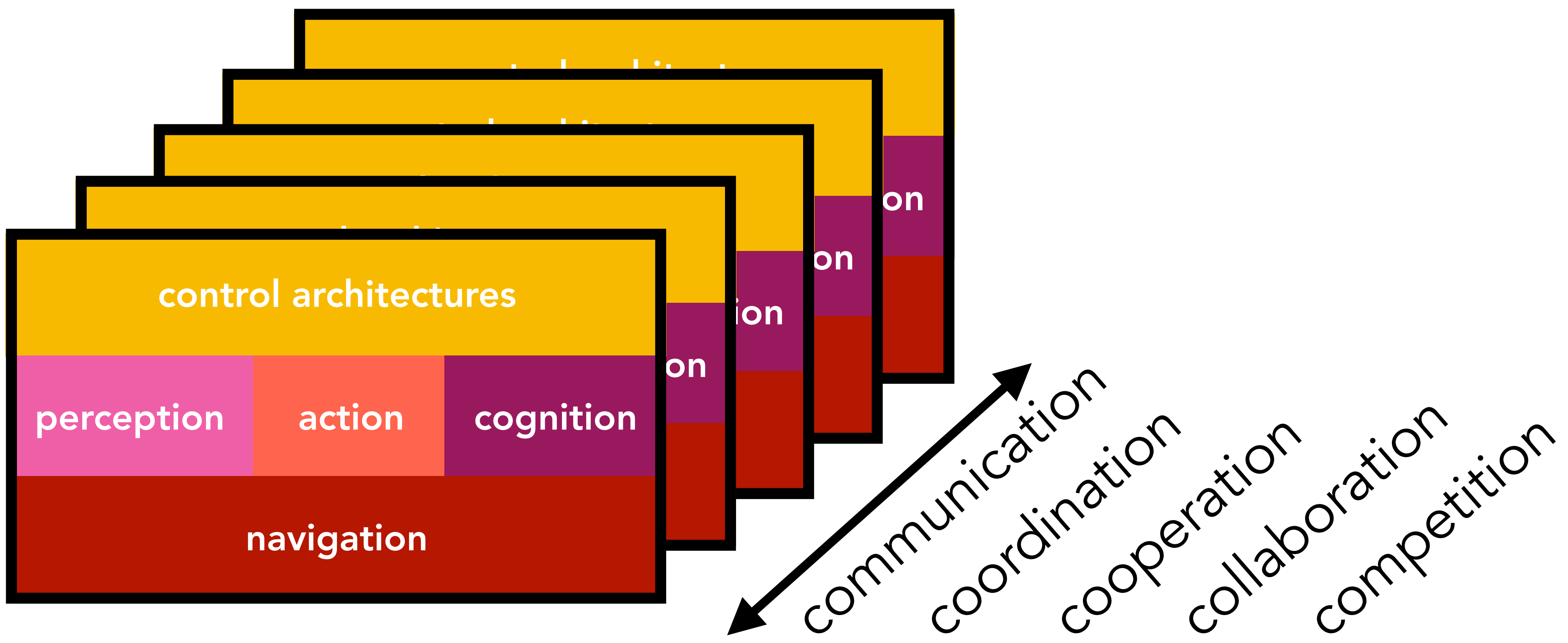
- Introduction to multi-robot systems
- Taxonomy
- Collective movement
  - Flocking (2 example methods)
  - Formations (2 example methods)

From Single to Multi-Robot Systems



From Single to Multi-Robot Systems

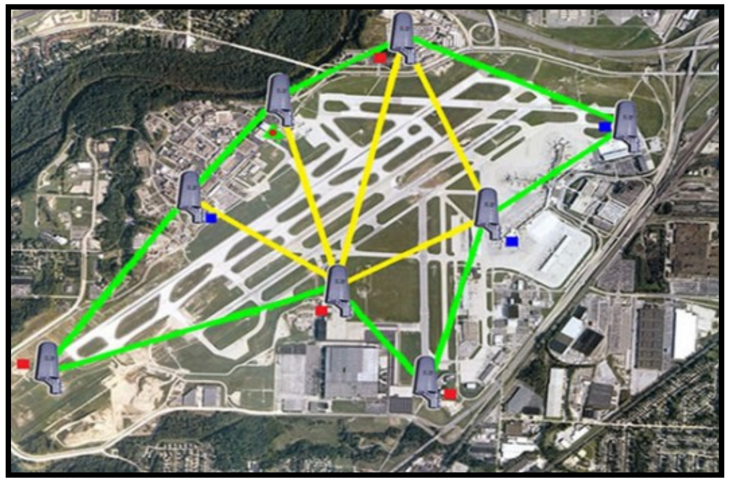
- **Multiple** mobile robots → **multi-robot systems**
- Higher-order goals
- Coordination facilitated through communication



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?)

search & rescue



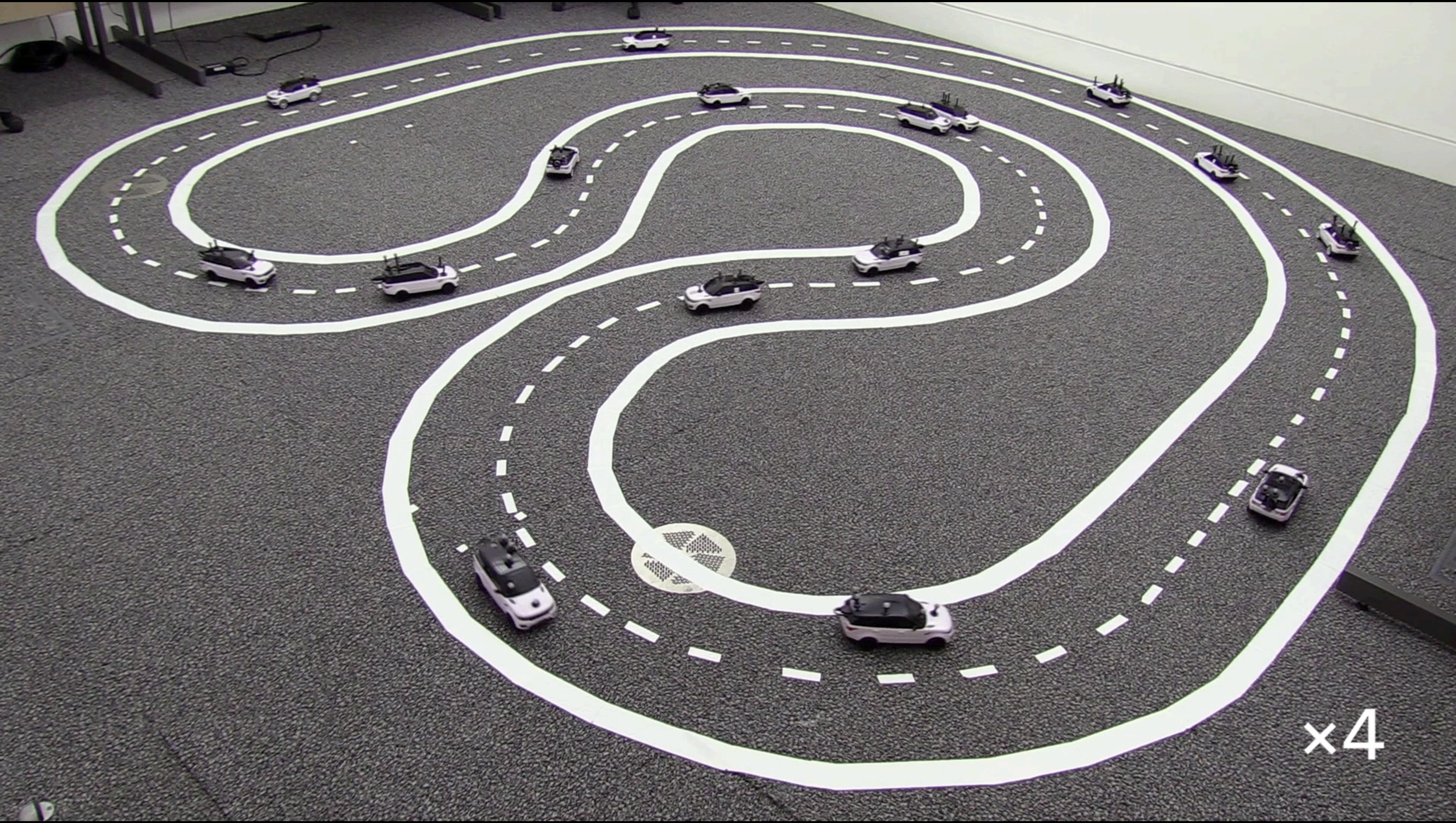
surveillance / monitoring

product pickup / delivery

Example 1: Coordination

* movie credit: Omur Arlsan; 2018

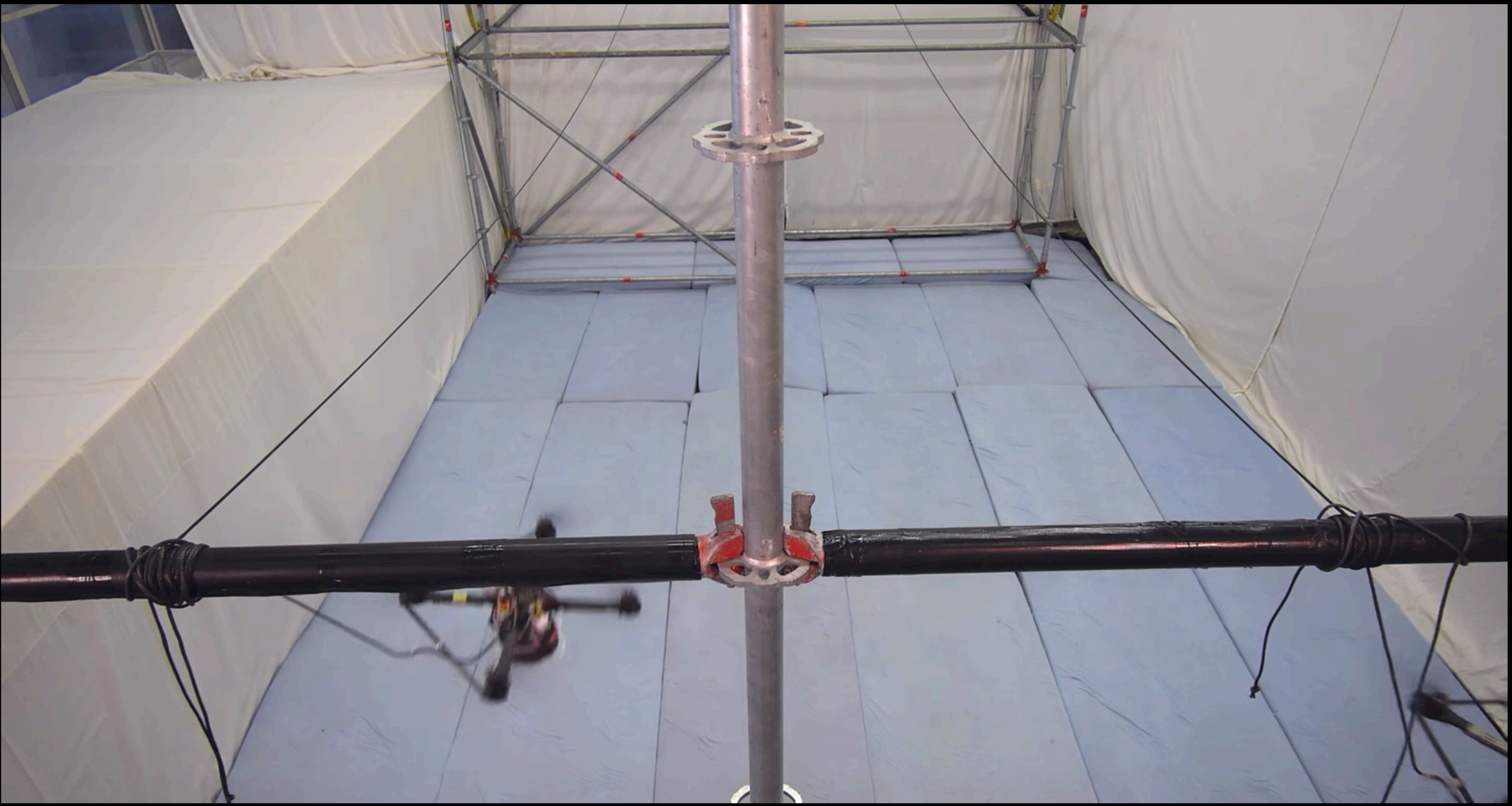
Example 2: Cooperation



Hyldmar, He, Prorok; IEEE ICRA 2019

https://youtu.be/2oJFQnbN5CA

Example 3: Collaboration



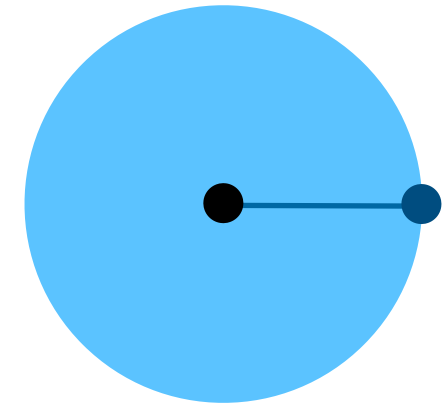
* movie credit: R. D'Andrea et al.

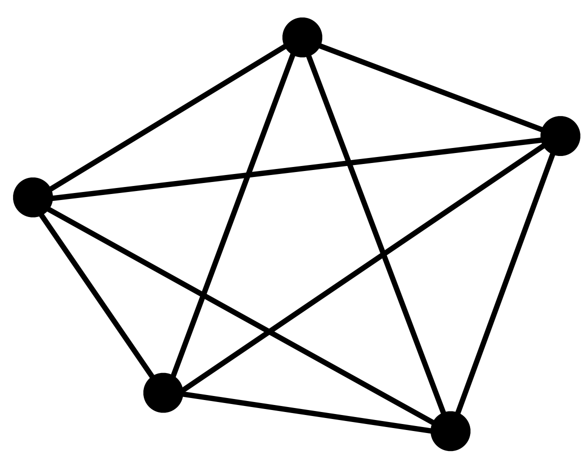
Taxonomy

- **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**: homogenenous vs. heterogeneous
  - Robot teams can leverage inter-robot complementarities

Communication Topologies

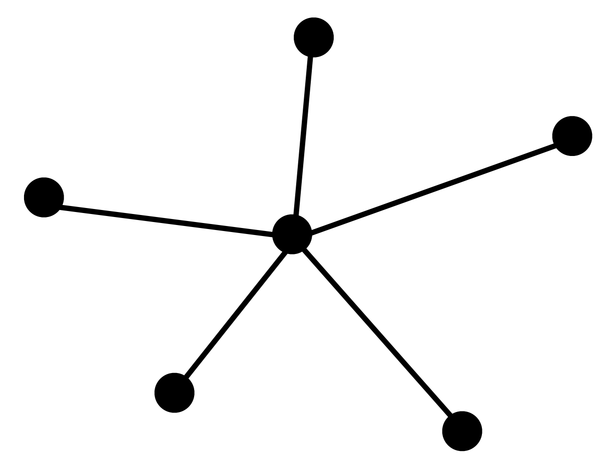
- Robot configurations / topologies are often defined by the maximum range of the available communication module (careful!).
- A disc model can be used to represent the communication range (very crude approximation)





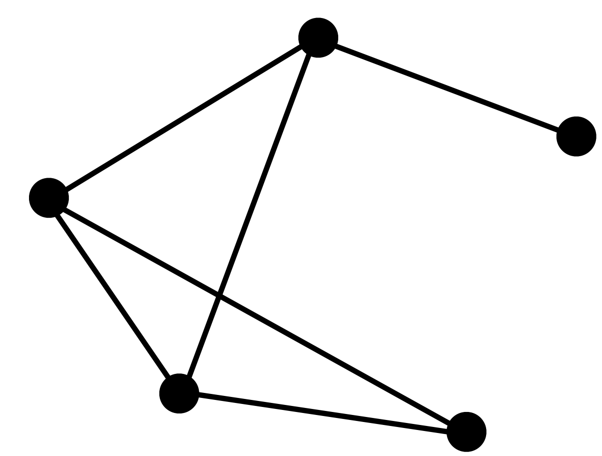
fully connected

centralized / decentralized coordination



star topology

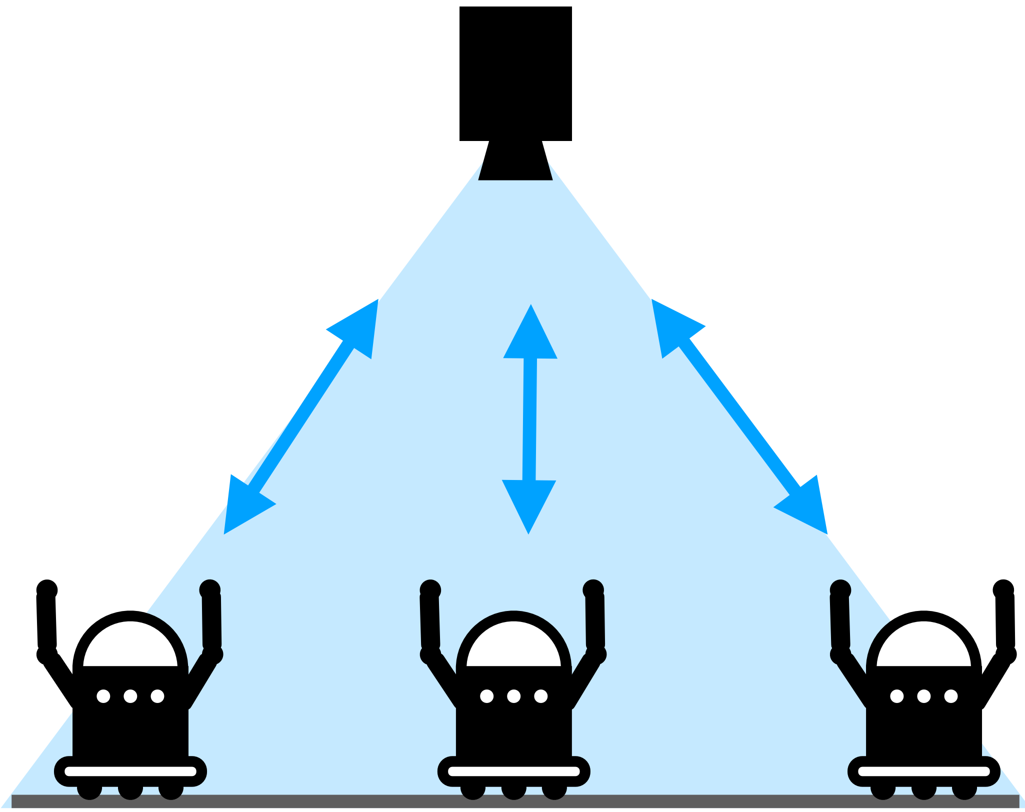
centralized / decentralized coordination



random mesh

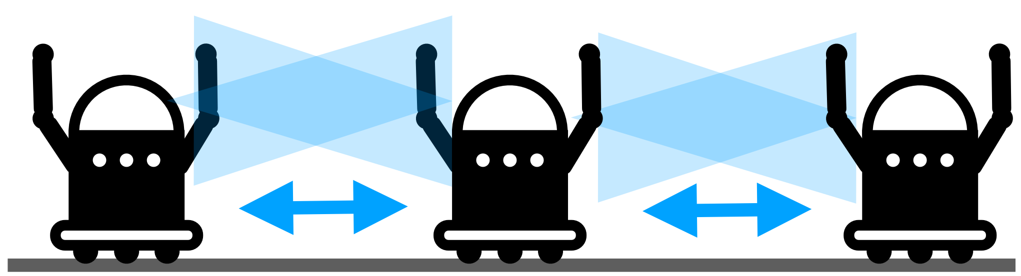
decentralized coordination

Centralization vs Decentralization



centralized

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



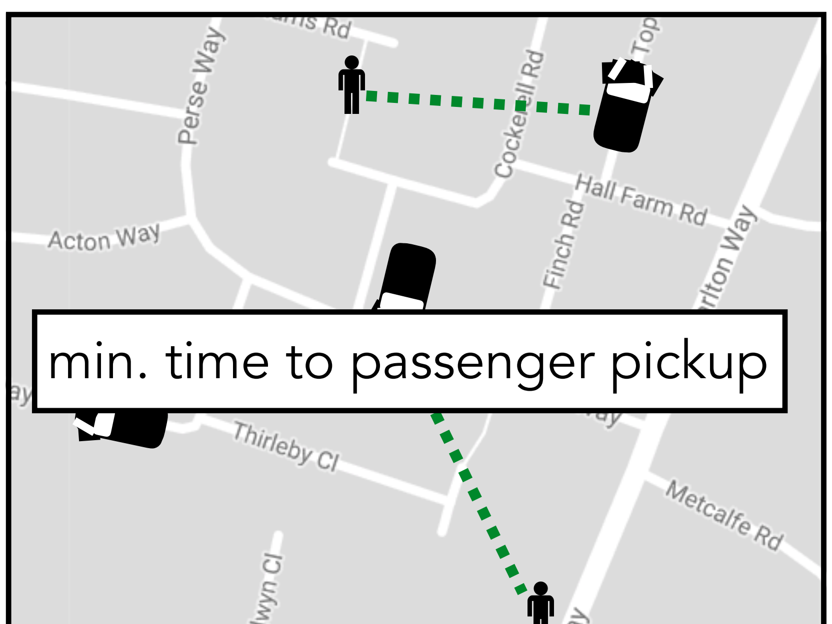
decentralized

- 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

search & rescue / surveillance



automated mobility-on-demand

connected autonomous vehicles

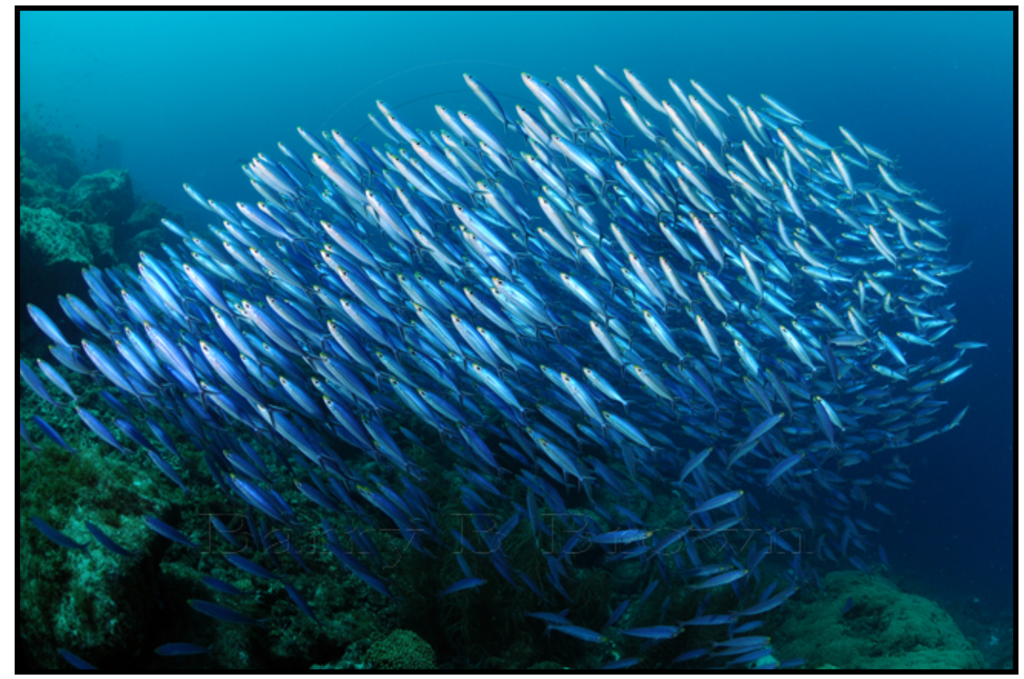
Decentralization

- 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)

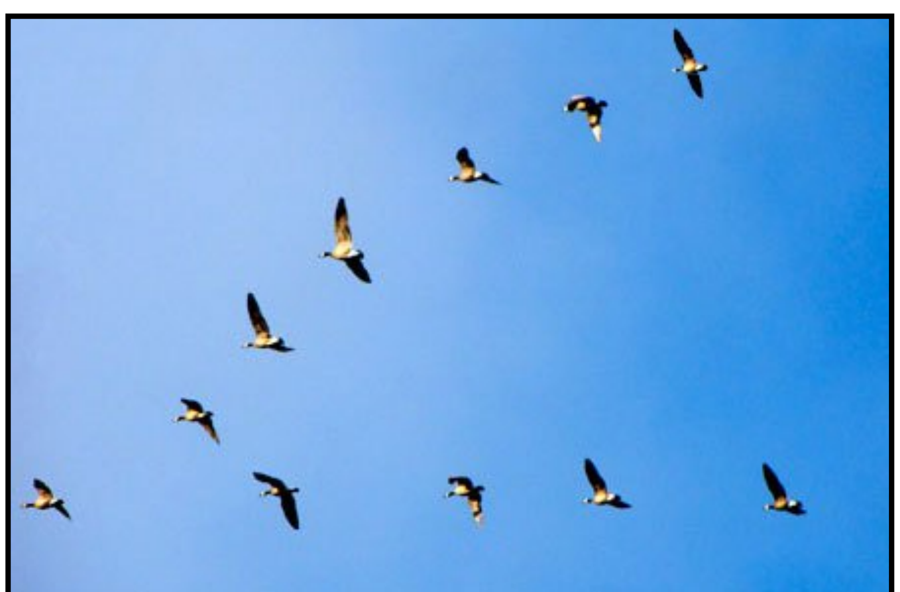
Collective Movement

In nature:

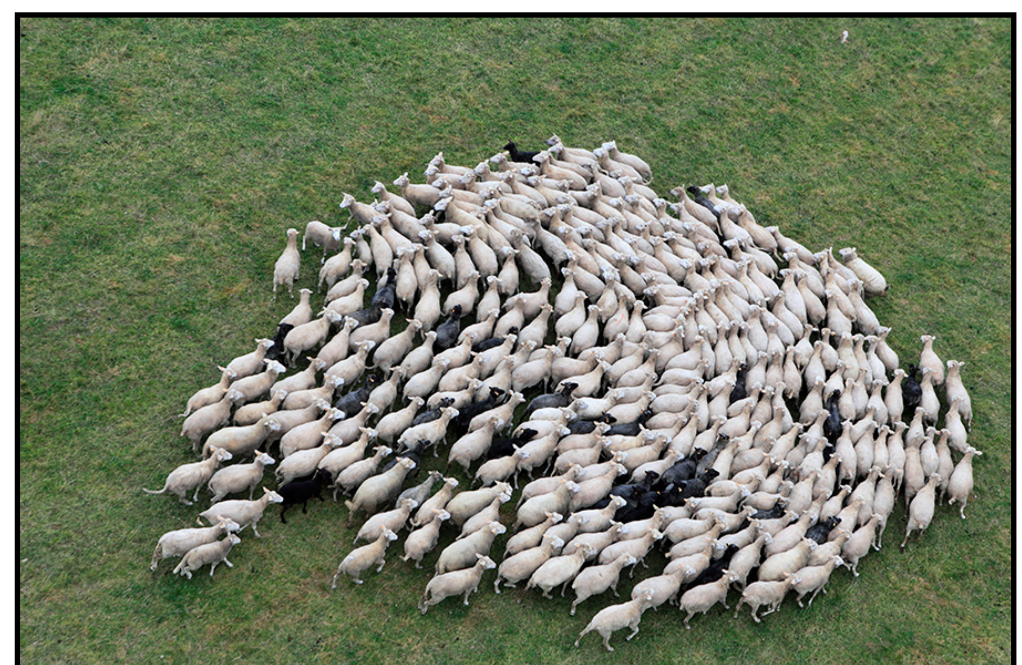
flock of birds



school of fish



flock of geese



herd of mammals

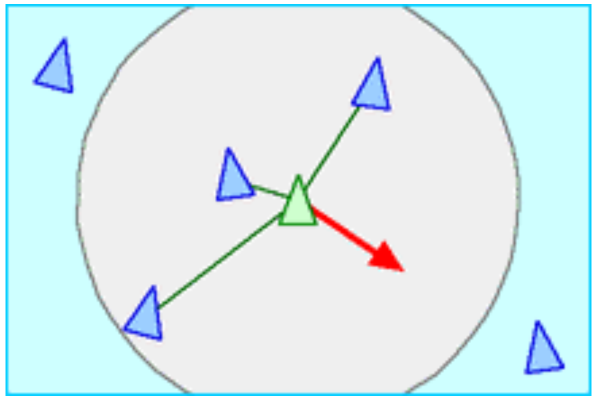
Collective Movement

- 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

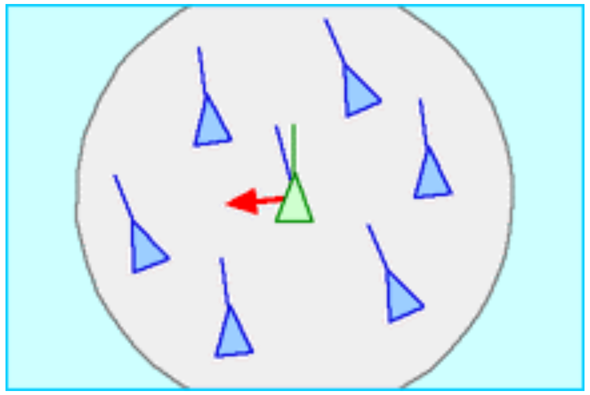
Intel drone swarm

Flocking with Boids

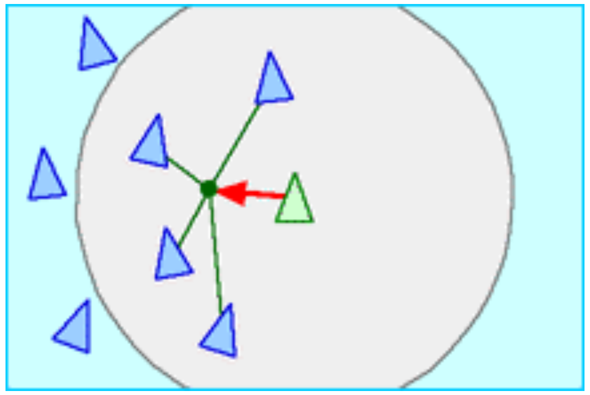
- In 1986, Craig Reynolds (computer animator) wanted to create a computationally efficient method to animate flocks
- Goal: $O(N)$; current best was $O(N^2)$



separation



alignment

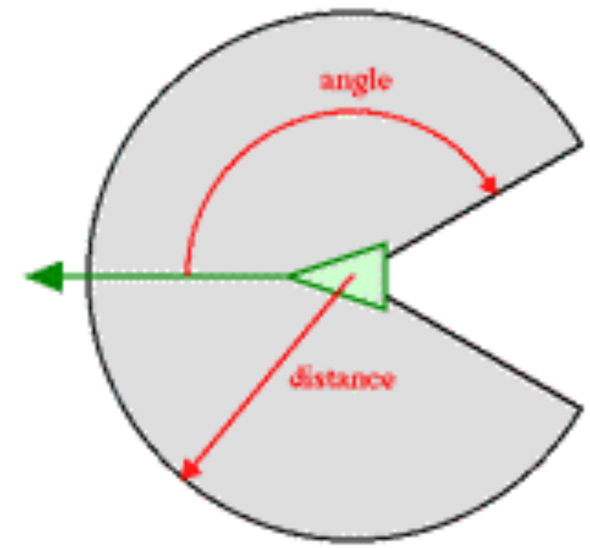


cohesion

- 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

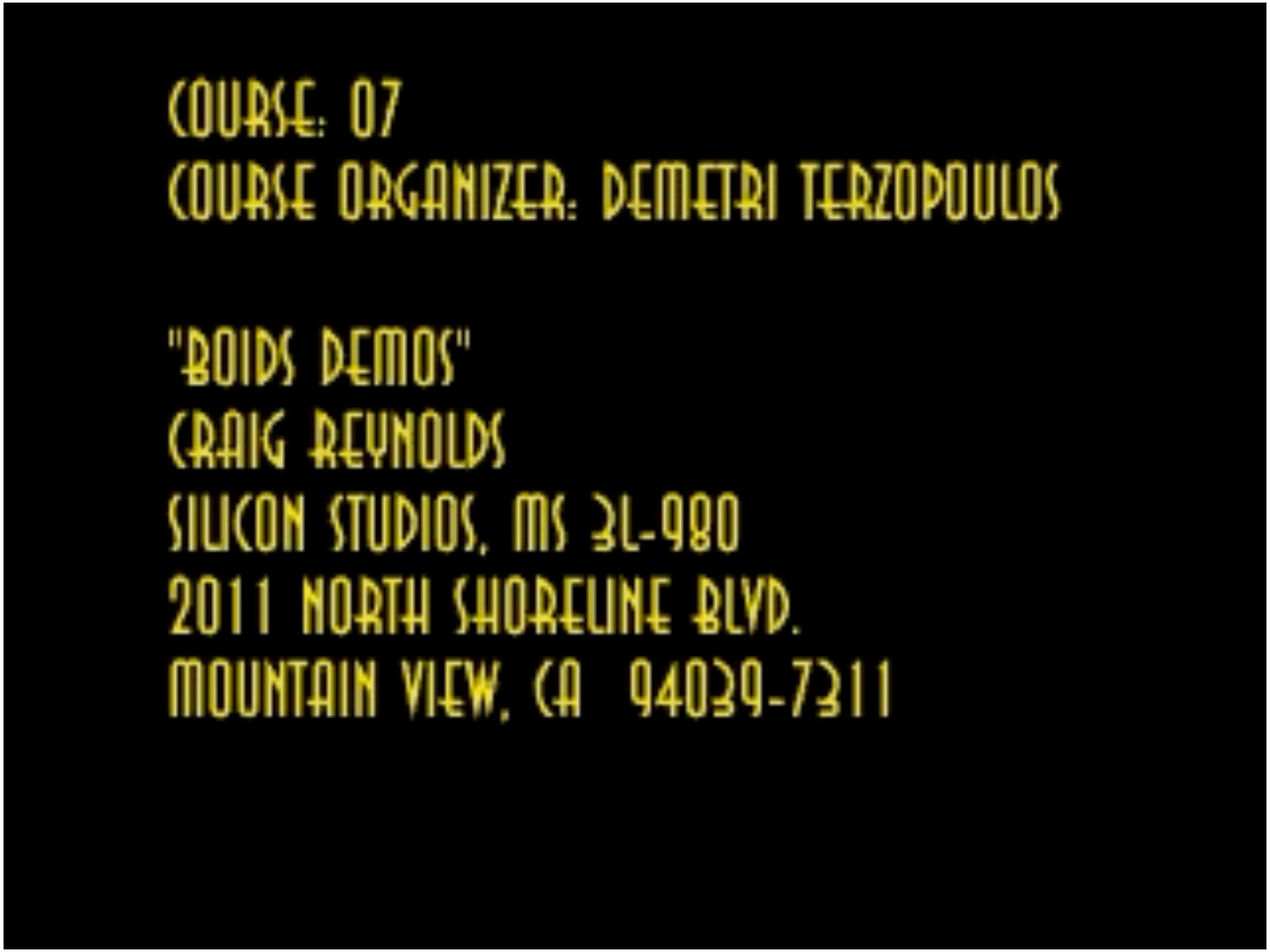
- Sensory system: idealized, but local:
  - almost omni-directional
  - no delays (in sensing)
  - no noise (in range and bearing)



2D representation of boid neighborhood

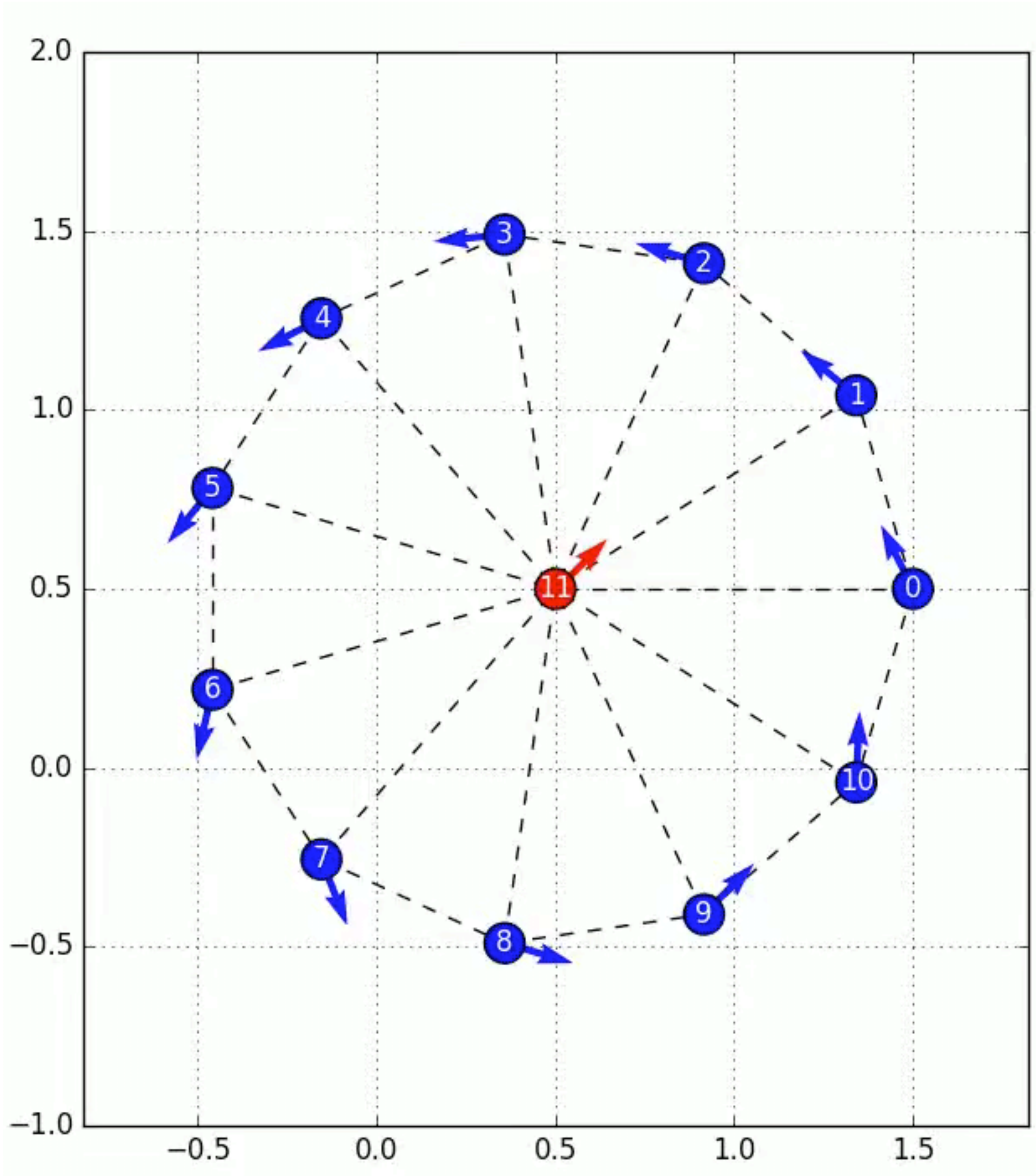
- Behavior-based with **priorities** (cf Brooks' subsumption):
  - Low priority acceleration request towards a point or in a direction (to direct flock)
  - Highest priority to obstacle avoidance ('steer-to-avoid' with a different sensory system)

Flocking with Boids



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

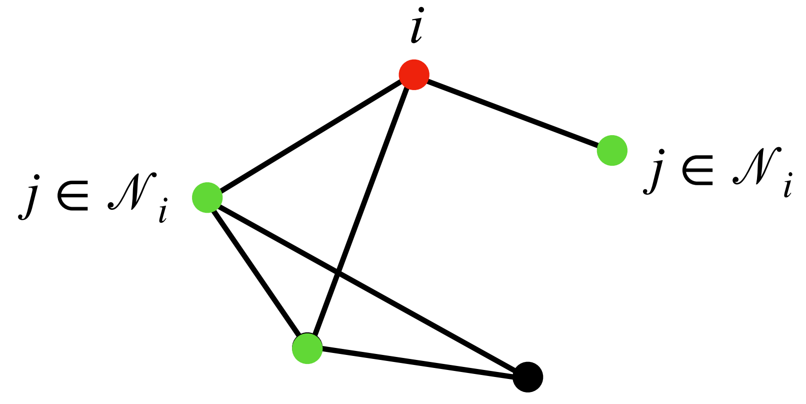
Flocking with Consensus



1 leader robot; robots apply consensus algorithm to agree on heading

The Consensus Algorithm

- 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:

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

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

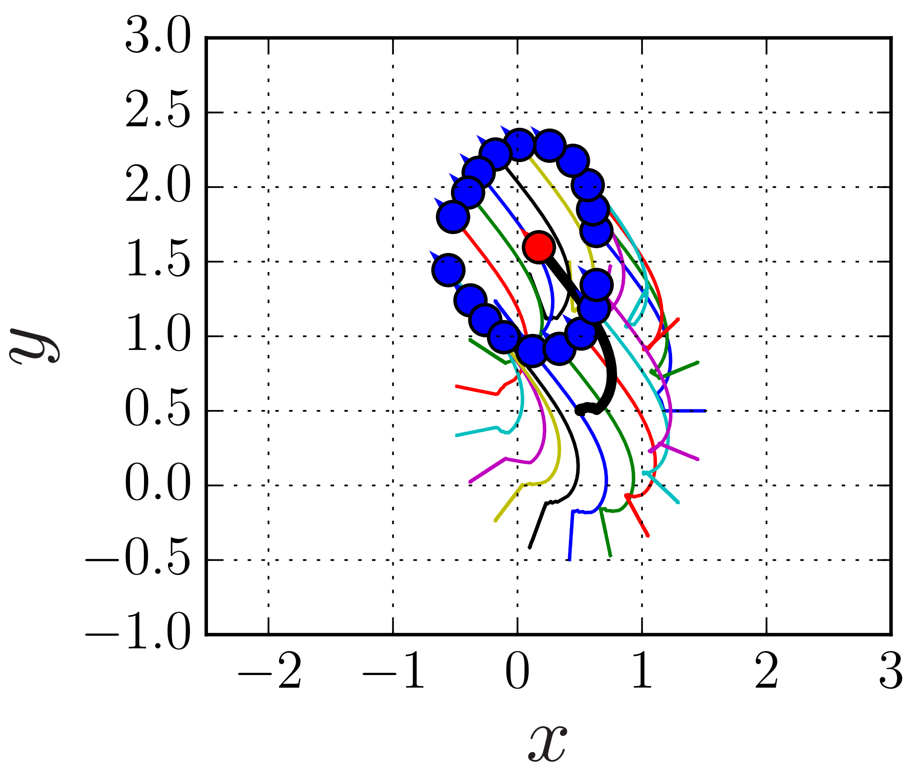
$$t \to \infty, \qquad x_i[t] = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} x_i[0]$$

Flocking with Consensus

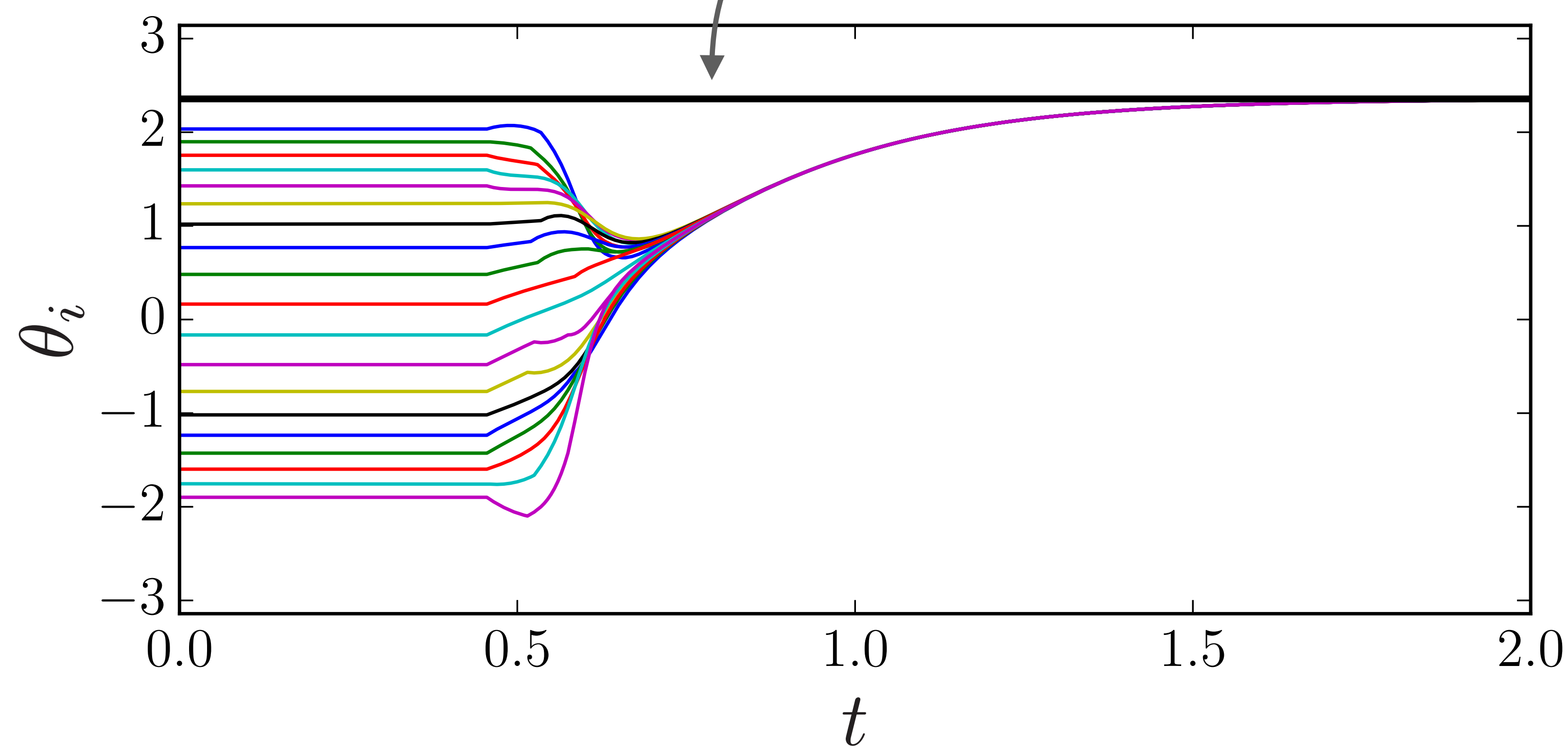
Holonomic robot: $\dot{\mathbf{x}} = \mathbf{u}$ with $\mathbf{x}_i = [x_i, y_i]$

Consensus on heading θ_i **with a leader agent**

convergence to leader's value



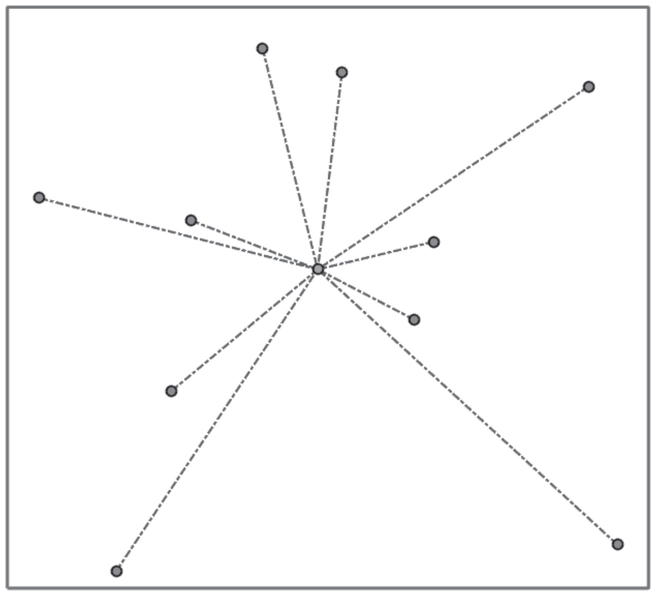
robot trajectories



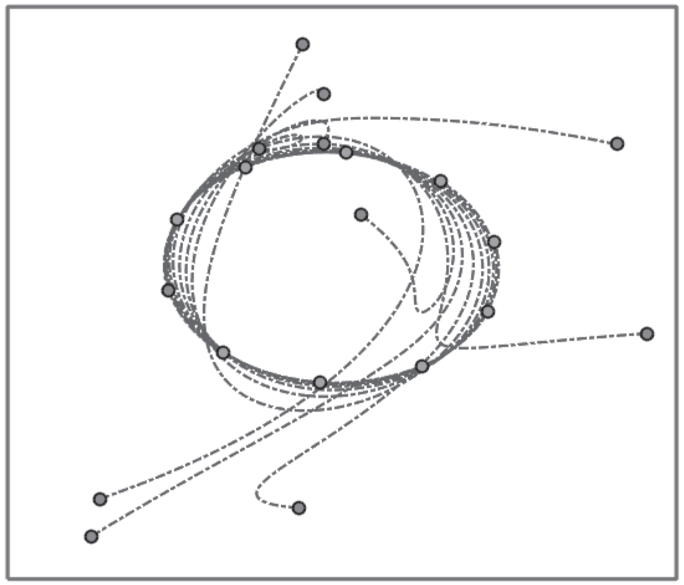
heading as a function of time

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

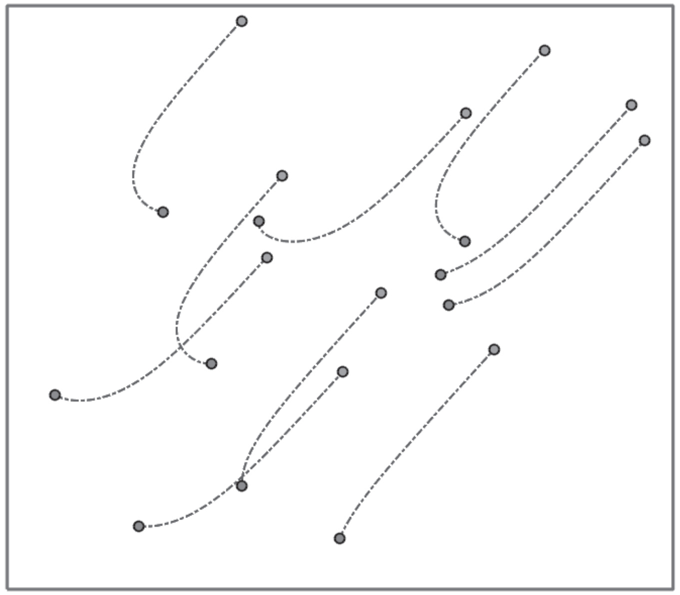
Other Consensus Applications



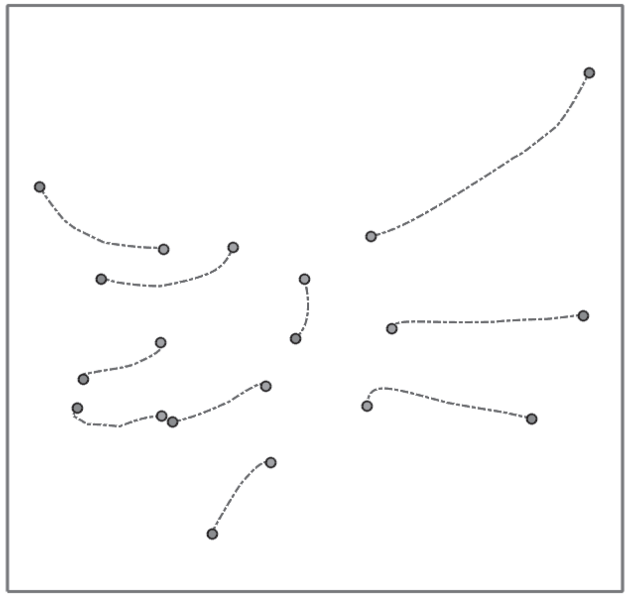
rendezvous



cyclic pursuit



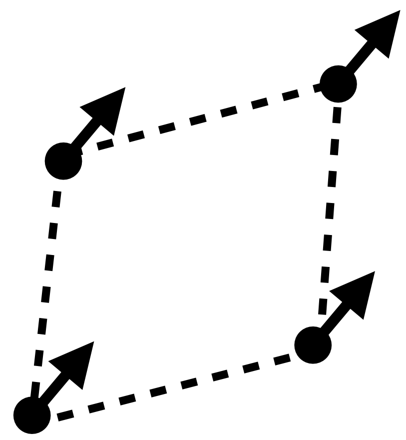
flocking



configuration

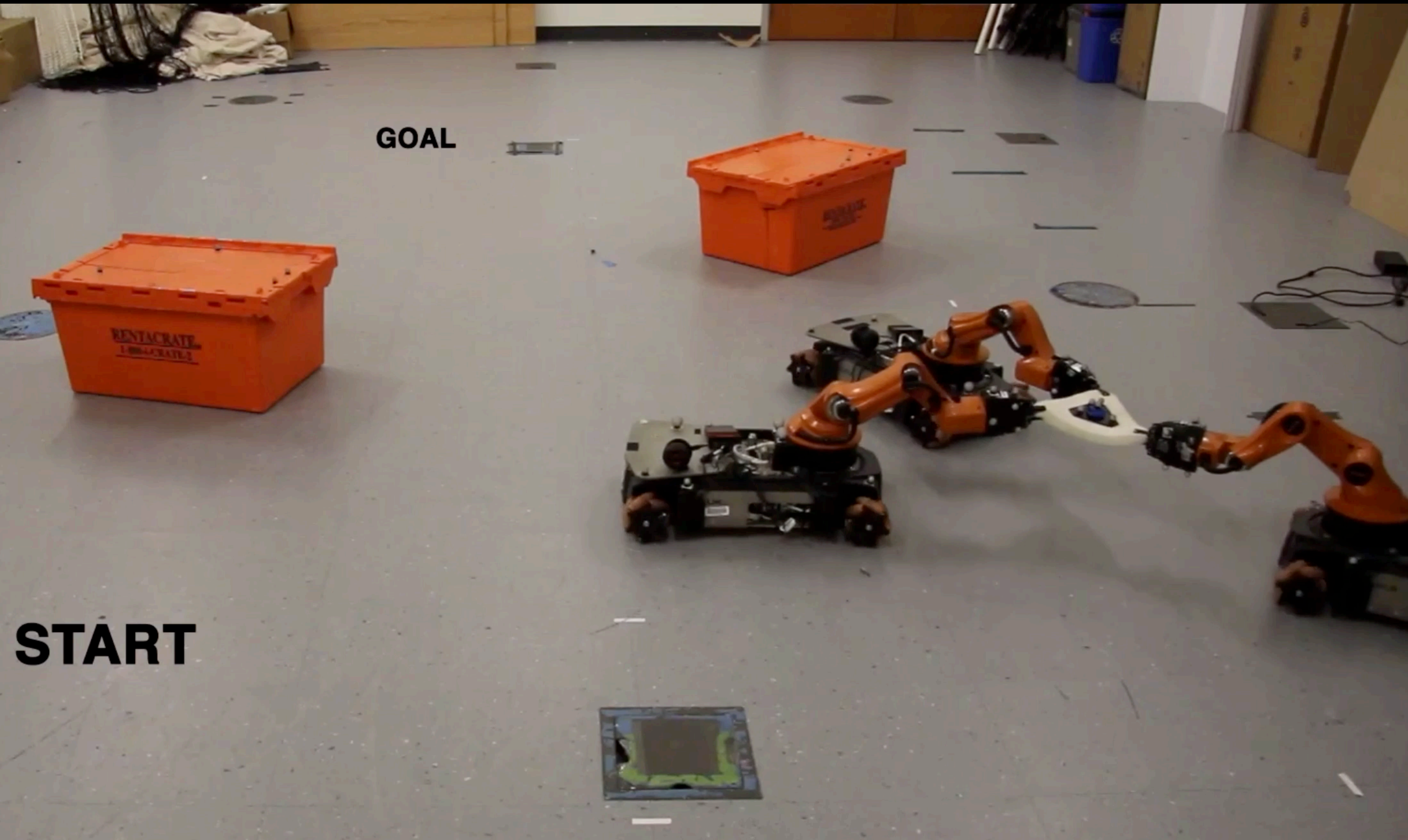
Formation Control

- 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



e.g.: diamond formation

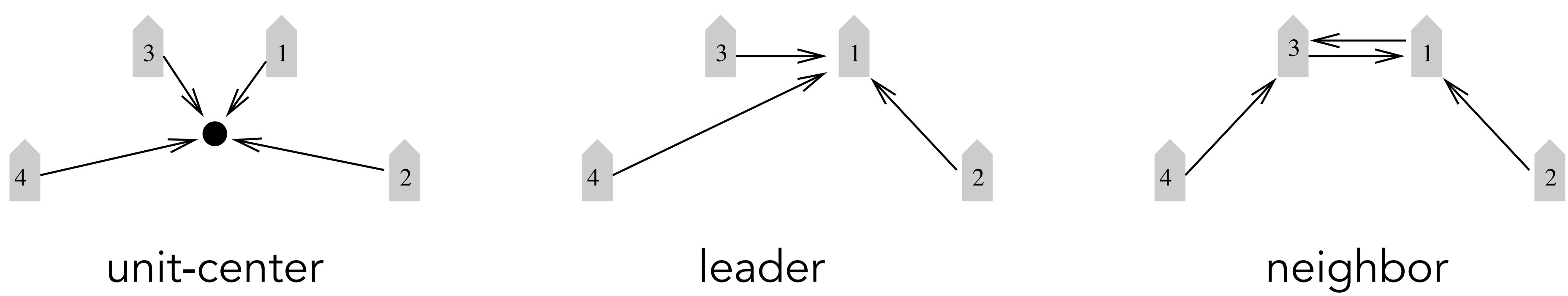
- Variants:
  - Behavior-based (Balch et al., 1999) *(recall: reactive control paradigm)*
  - Closed-loop control (Das et al., 2002) *(recall: error-based control paradigm)*



D. Rus et al.

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.



- 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

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

*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

$$H_{desired} = F_{dir} - \delta_{heading}$$

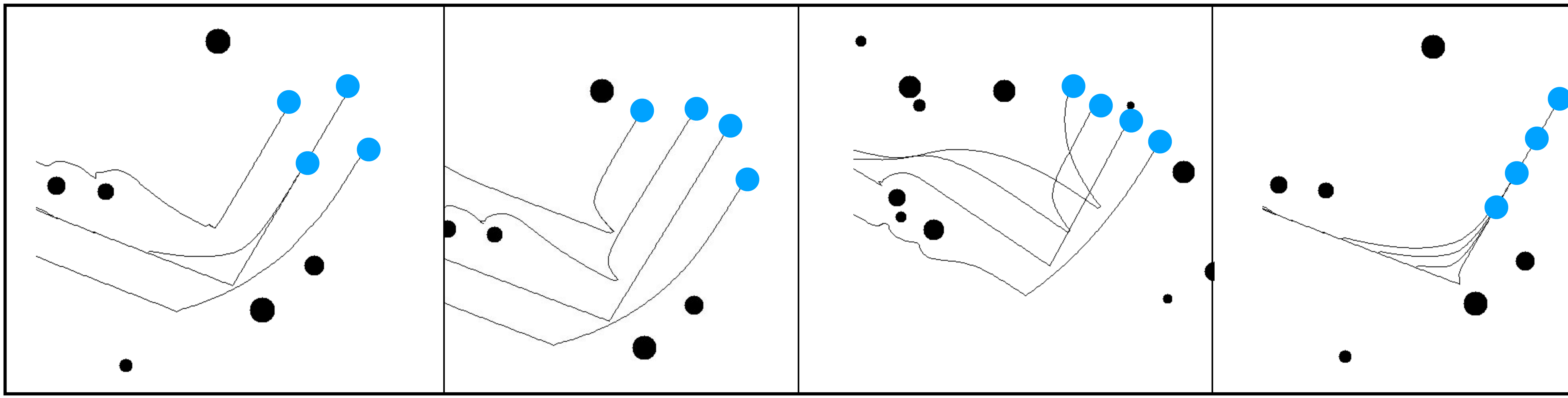
$$V_{steer} = H_{desired} - R_{dir}$$

- 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.
- δ_{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]:



diamond wedge line column

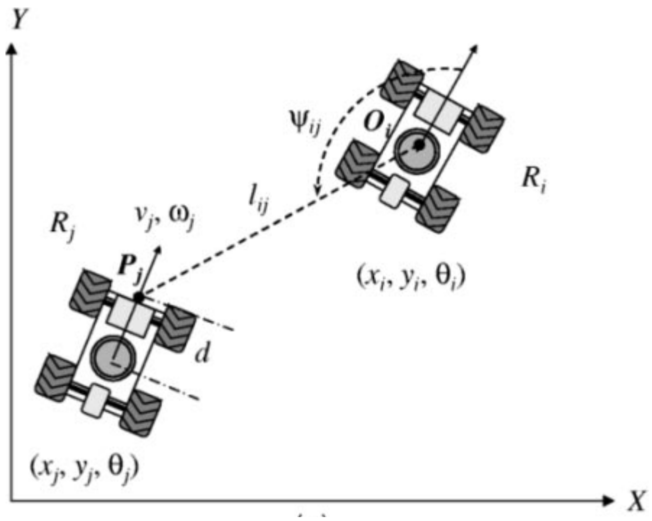
Assumptions:

- 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

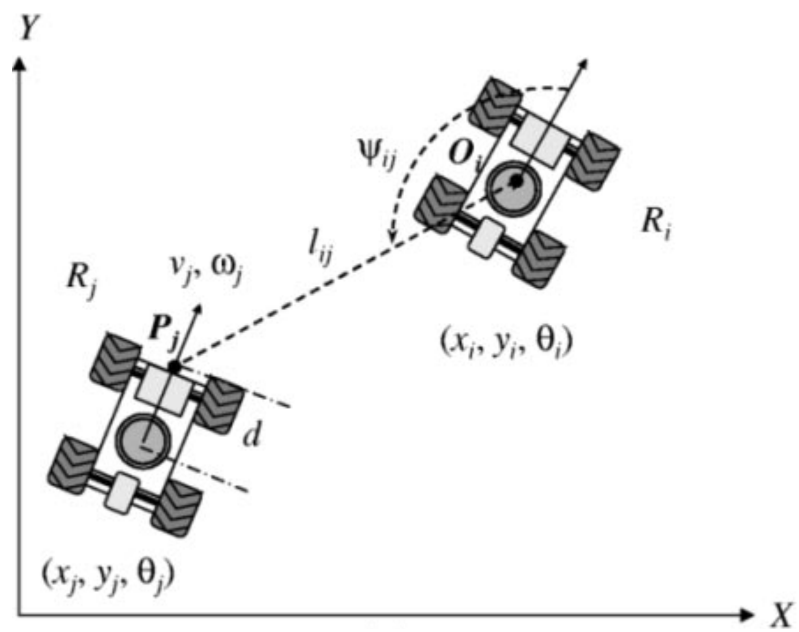
*image credit: Das 2002

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**: $\mathbf{z}_{ij} = [l_{ij}, \psi_{ij}]^\top$

  Control input: $\mathbf{u}_j = [v_j, \omega_j]^\top$ (forwards and rotational velocities)

follower, robot ***j***



leader, robot ***i***

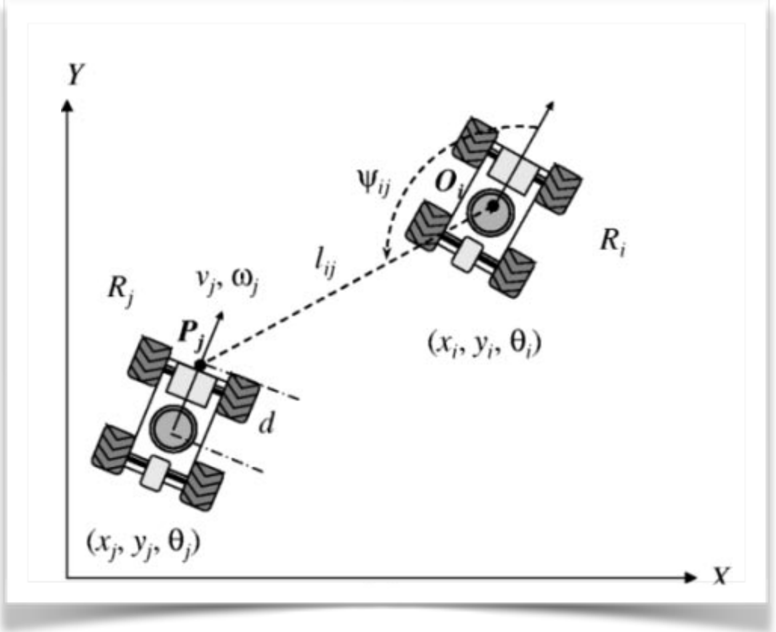
Aim: Find $\mathbf{u}_j$ such that desired separation l_{ij}^d and desired bearing ψ_{ij}^d are reached, and stably maintained.

Closed-Loop Control for Formations

Dynamical system model: $\dot{\mathbf{z}}_{ij} = G\,\mathbf{u}_j + F\,\mathbf{u}_i$

with:

$$G = \begin{bmatrix} \cos\gamma_{ij} & d\sin\gamma_{ij} \\ \dfrac{-\sin\gamma_{ij}}{l_{ij}} & \dfrac{d\cos\gamma_{ij}}{l_{ij}} \end{bmatrix} \qquad F = \begin{bmatrix} -\cos\psi_{ij} & 0 \\ \dfrac{\sin\psi_{ij}}{l_{ij}} & -1 \end{bmatrix}$$



where relative orientation is: $\beta_{ij} = \theta_i - \theta_j$ and $\gamma_{ij} = \beta_{ij} + \psi_{ij}$

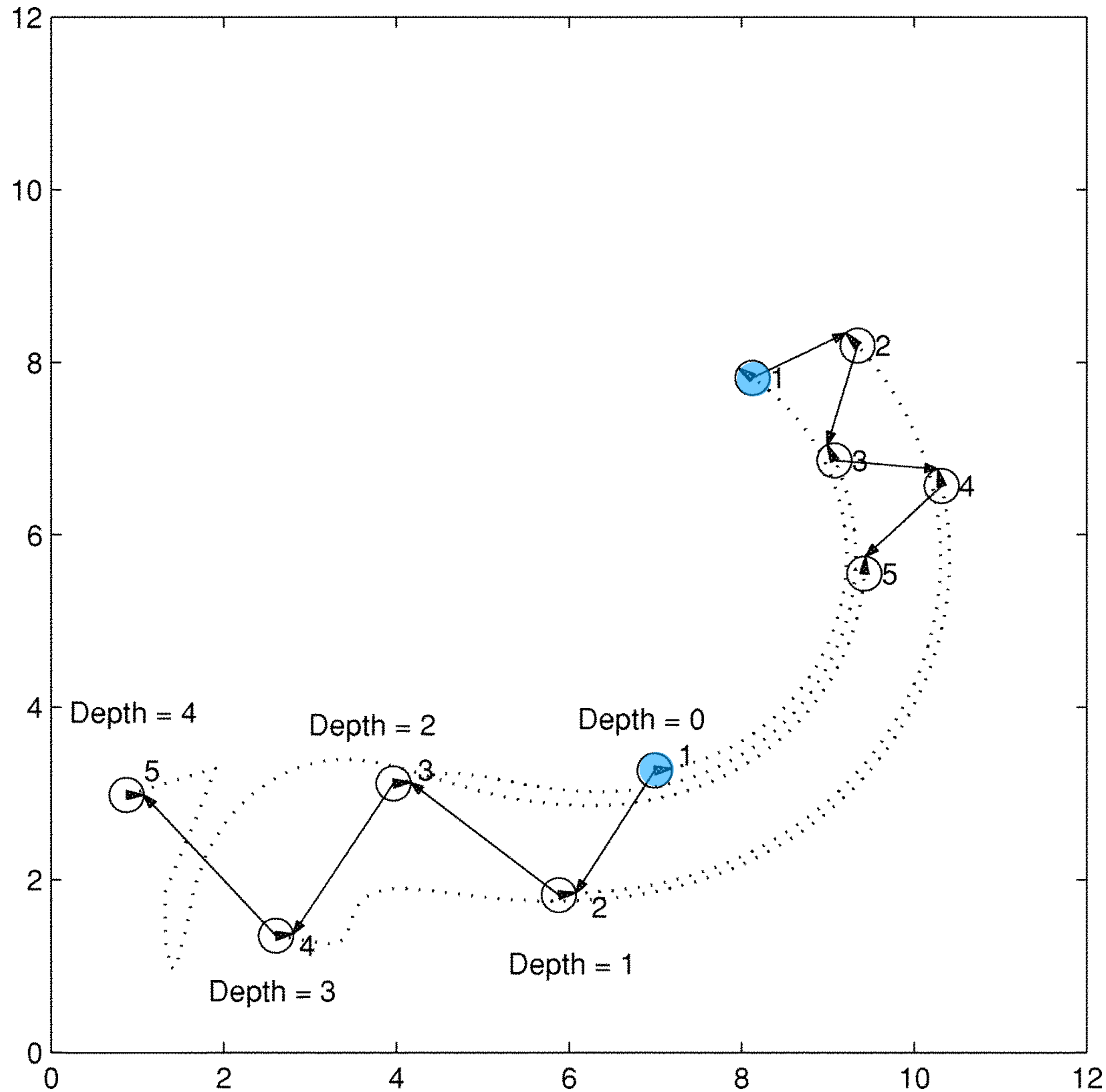
Proportional control law:

$$\underbrace{\dot{\mathbf{z}}_{ij} = \mathbf{k}(\mathbf{z}_{ij}^d - \mathbf{z}_{ij})}_{\text{closed-loop linearized system}}$$

This guarantees convergence to desired relative state $\mathbf{z}_{ij}^d$ (Stability is proven in paper.)

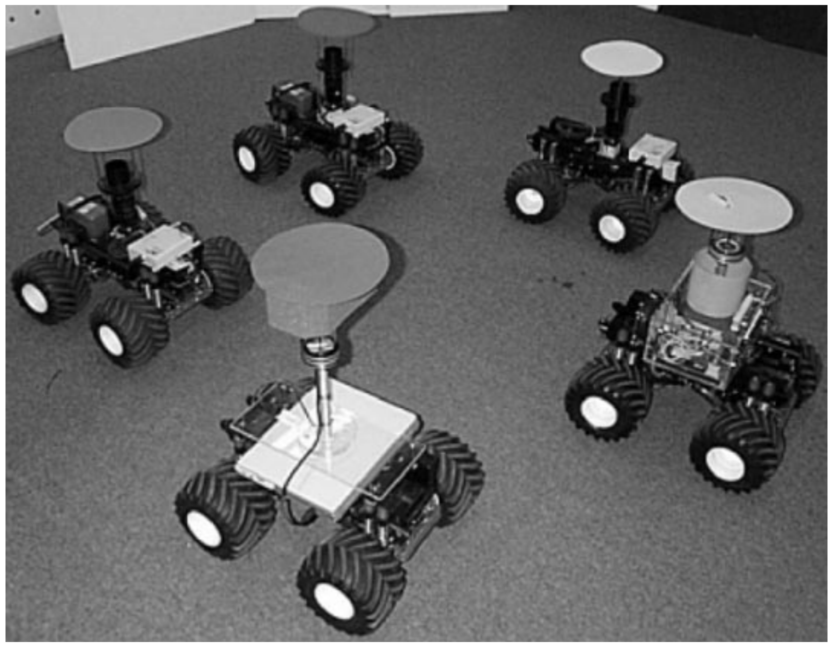
Control: $\mathbf{u}_j = G^{-1}\left(\mathbf{k}(\mathbf{z}_{ij}^d - \mathbf{z}_{ij}) - F\mathbf{u}_i\right)$

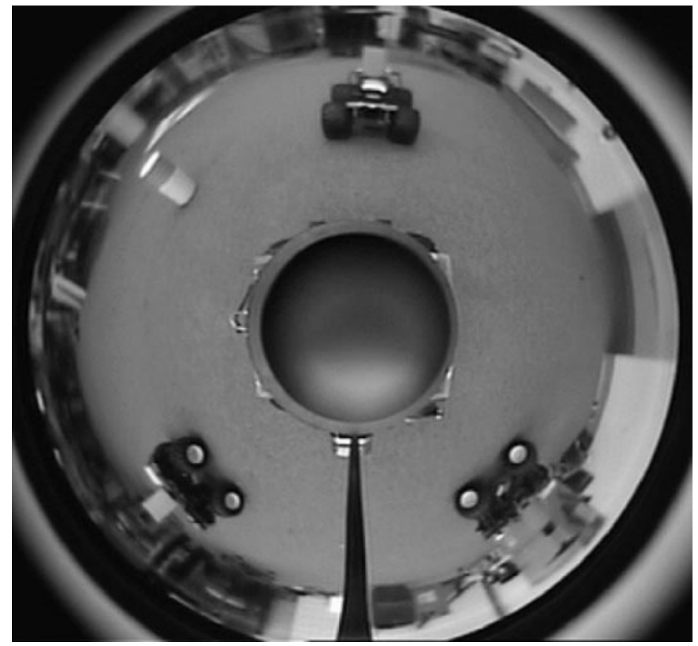
Closed-Loop Control for Formations



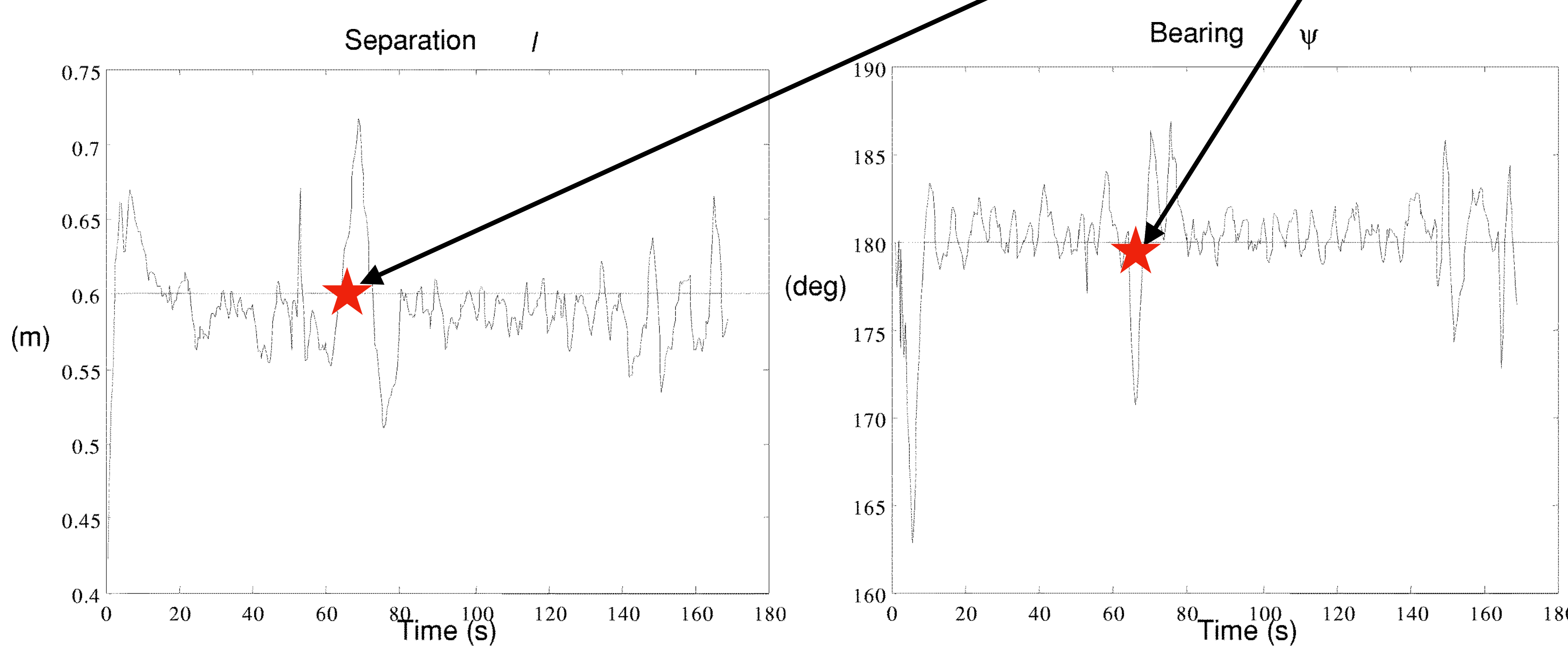
Closed-Loop Control for Formations

Four robots with omnidirectional cameras:



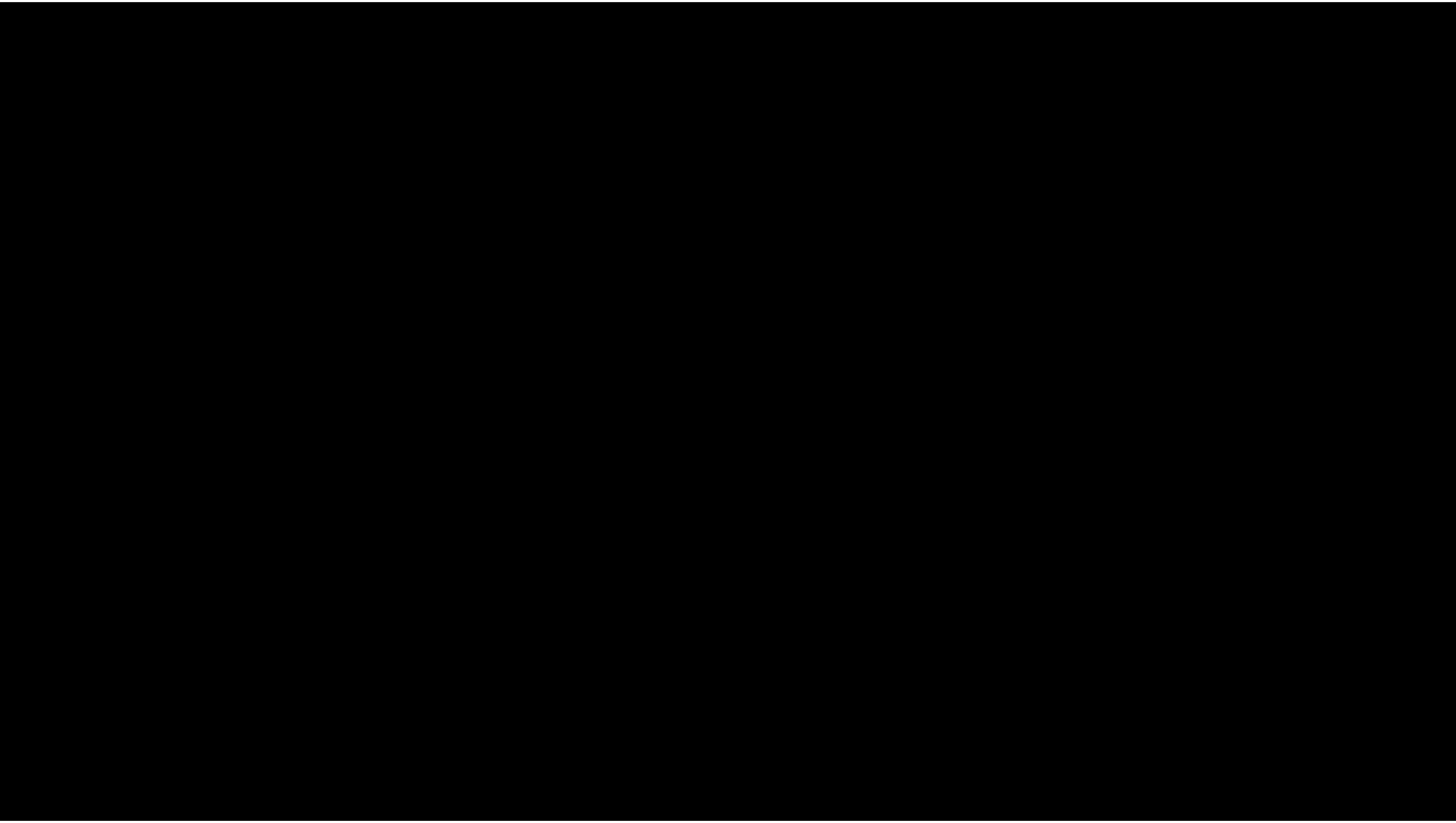


manually restrained follower



[Das et al., 2002]

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

2019 IEEE RAS Summer School on Multi-Robot Systems

Task Assignment in Multi-Robot Systems

Lecture 2

Dr. Amanda Prorok

Assistant Professor, University of Cambridge

asp45@cam.ac.uk

www.proroklab.org

UNIVERSITY OF CAMBRIDGE

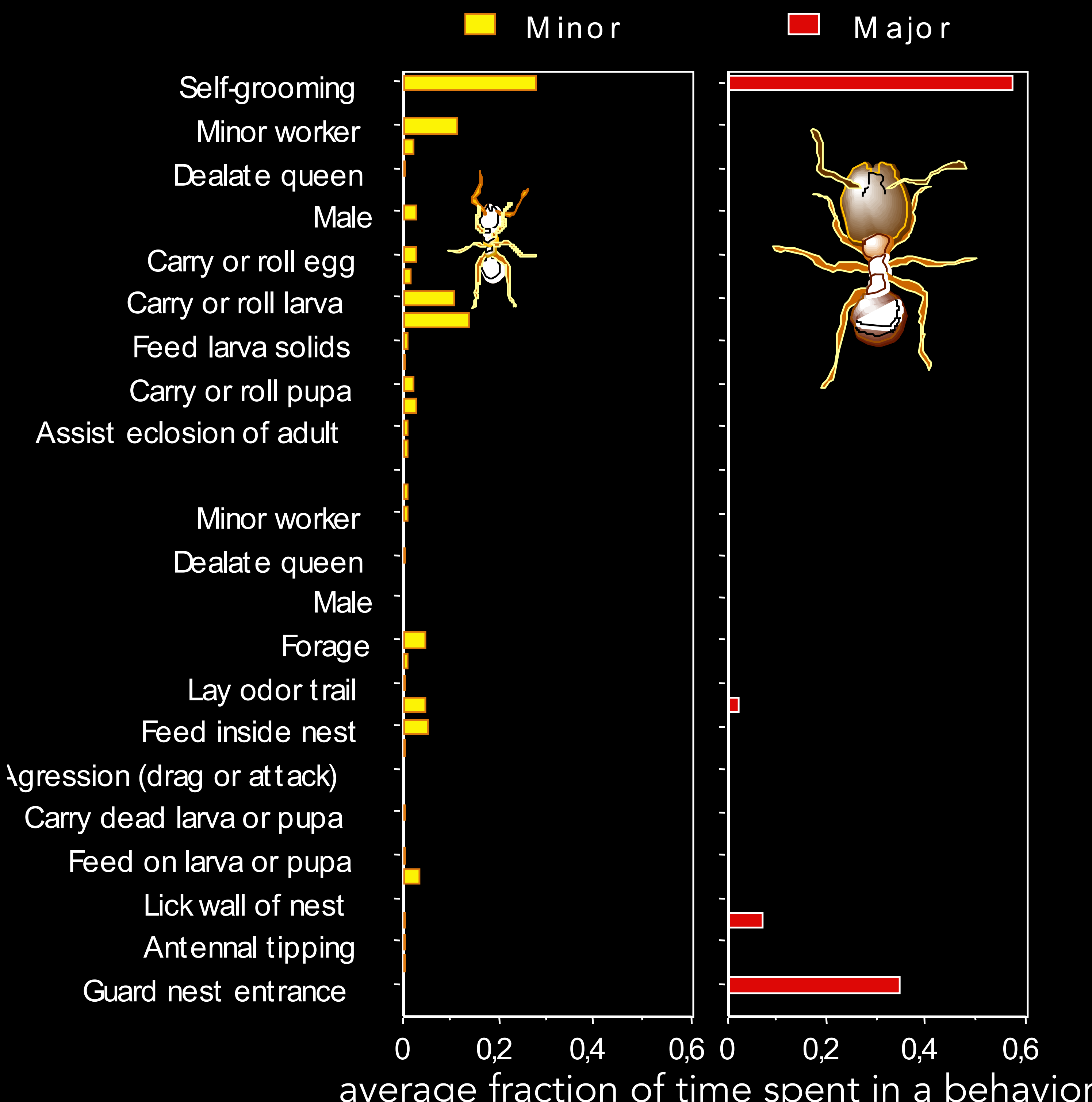
In this Lecture

- 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

Task Allocation vs. Division of Labor

In **nature**: physical castes

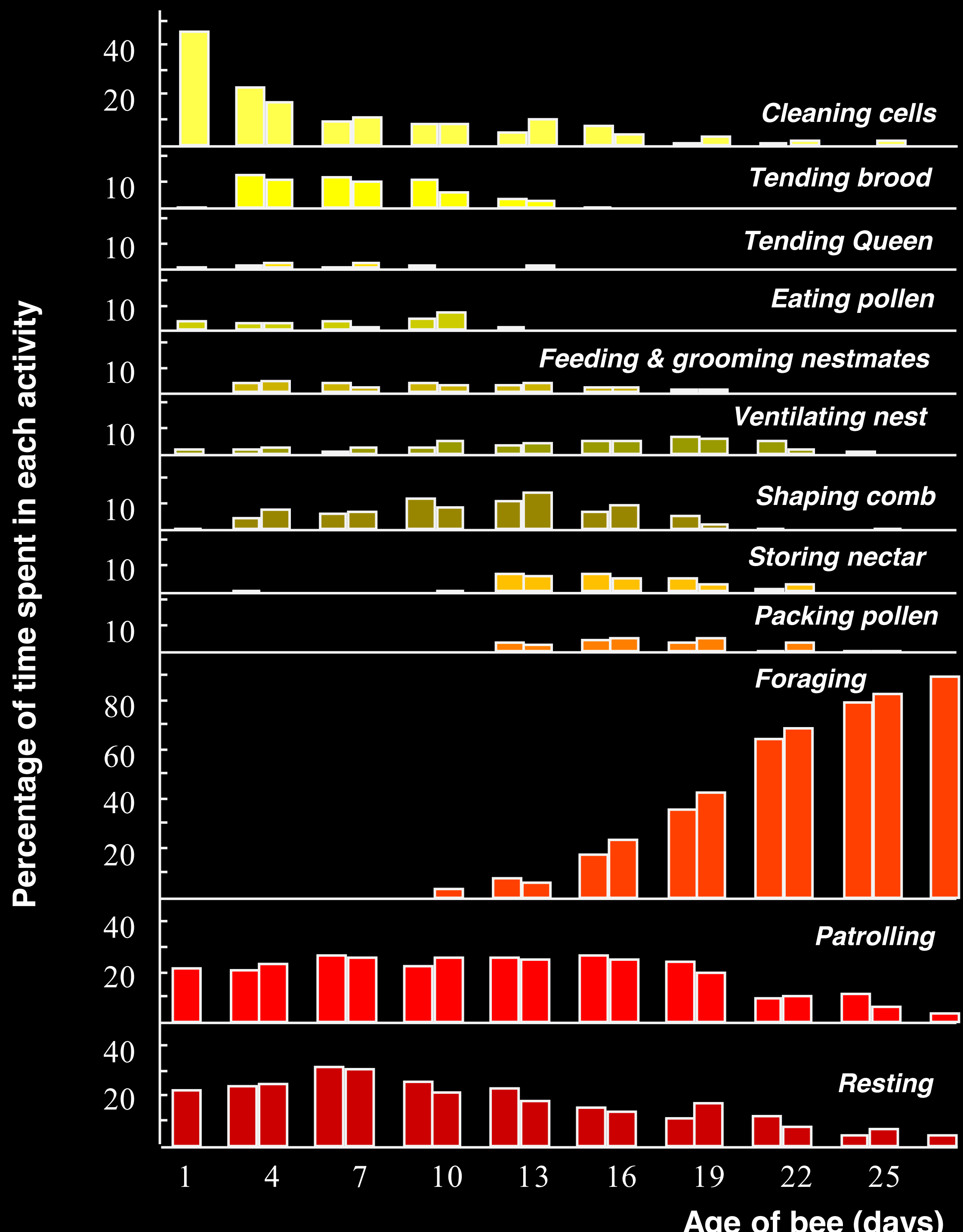


Behavioral repertoire of majors and minors: In *Pheidole guilelmimuelleri* the minors show ten times as many different basic behaviors as the majors.

*image credit: Alcherio Martinoli

Task Allocation vs. Division of Labor

In **nature**: temporal polyethism



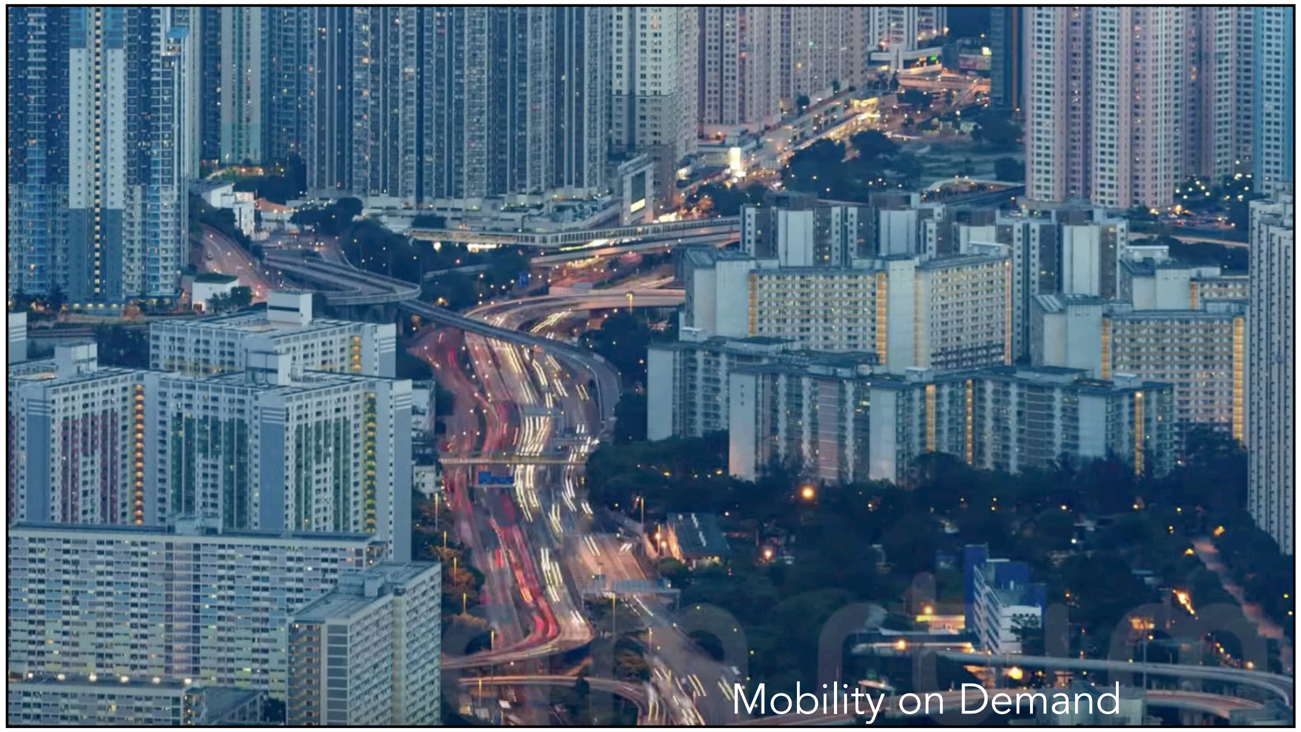
Behavioral change in worker bees as a function of age; young individuals work on internal tasks (brood care and nest maintenance), older workers forage for food and defend the nest.

*image credit: Alcherio Martinoli

Task Allocation vs Division of Labor

In robotics:

Monitoring

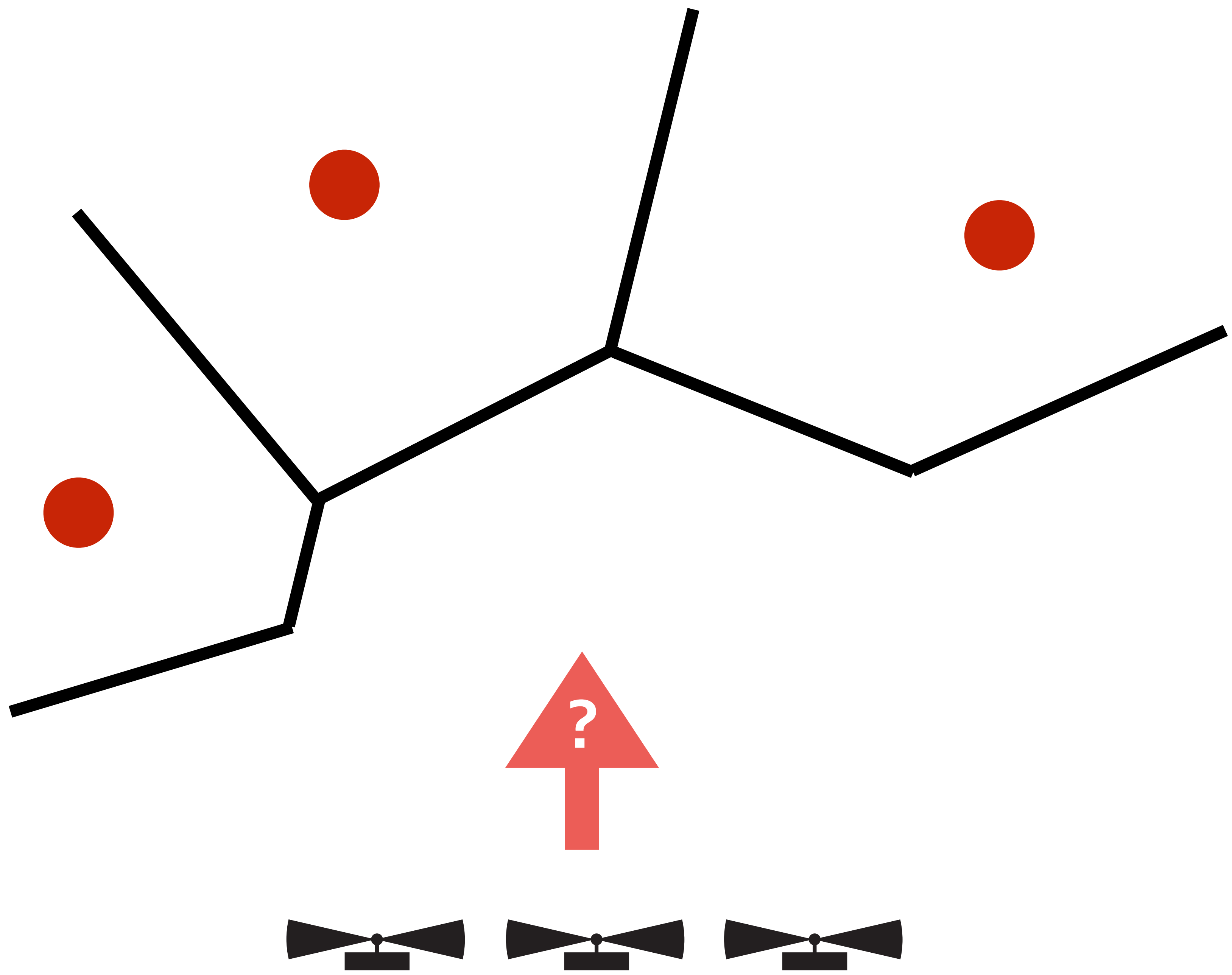


Mobility on Demand

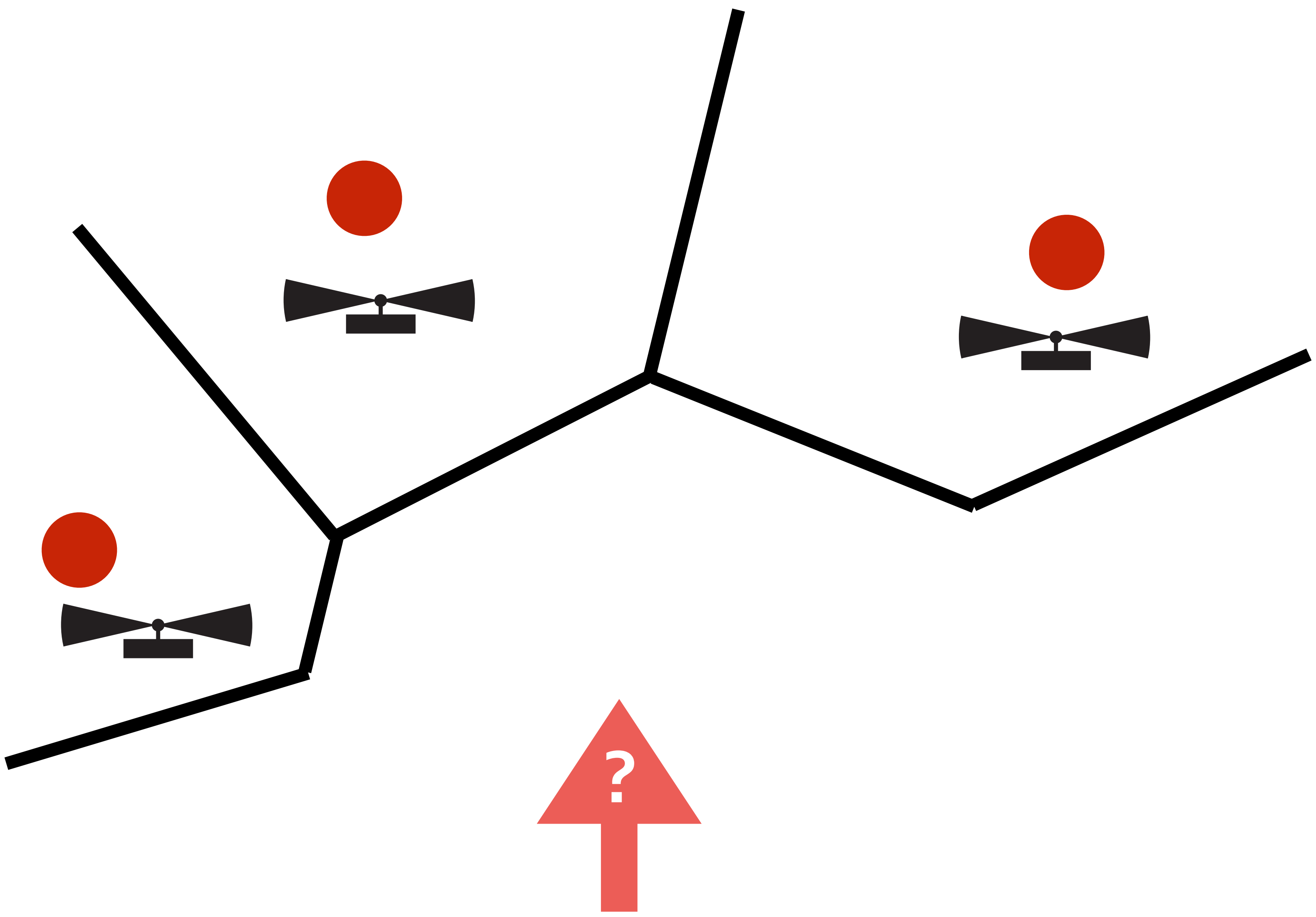
Warehousing and Product Delivery

Situational Awareness

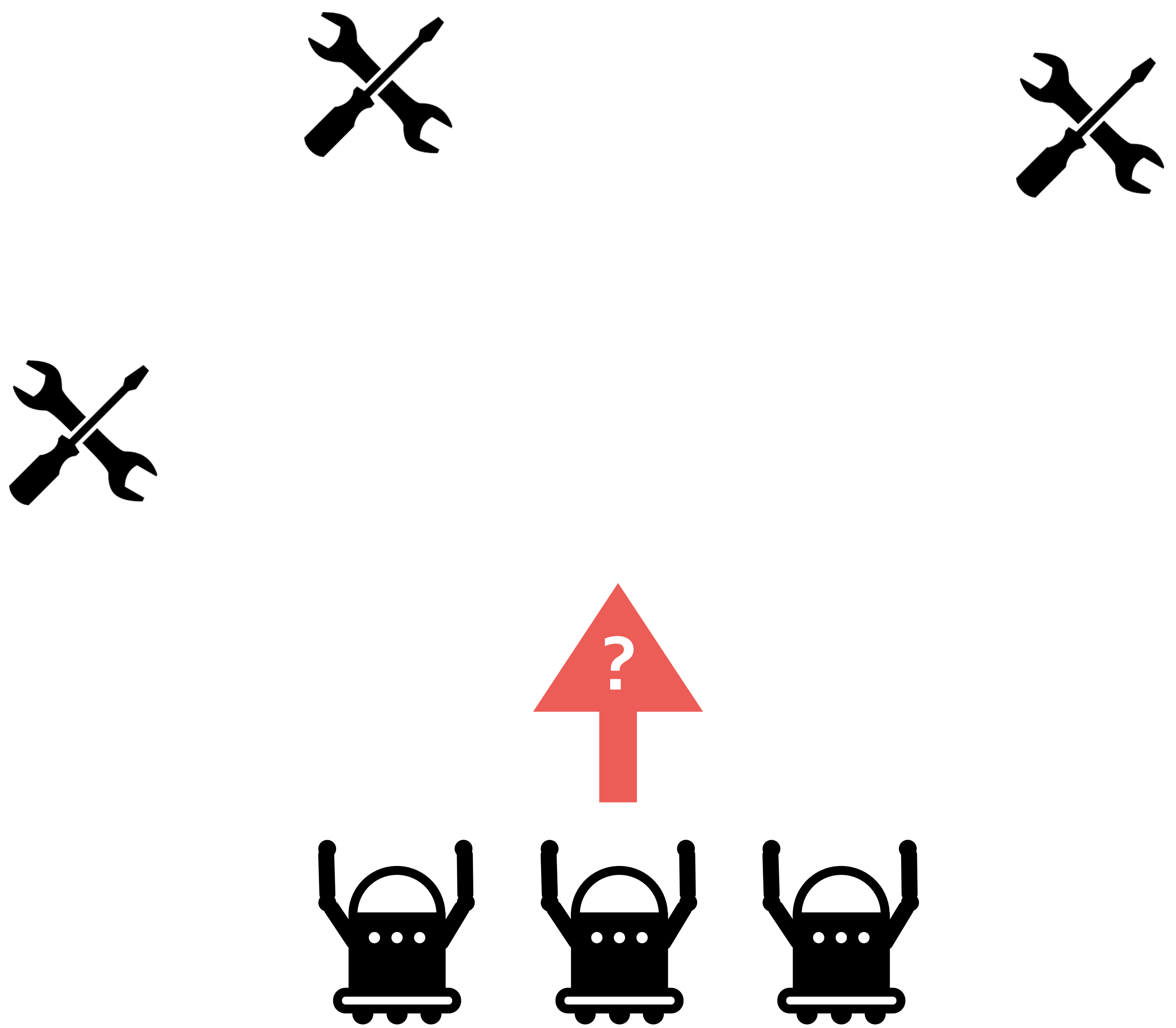
Assignment Problems



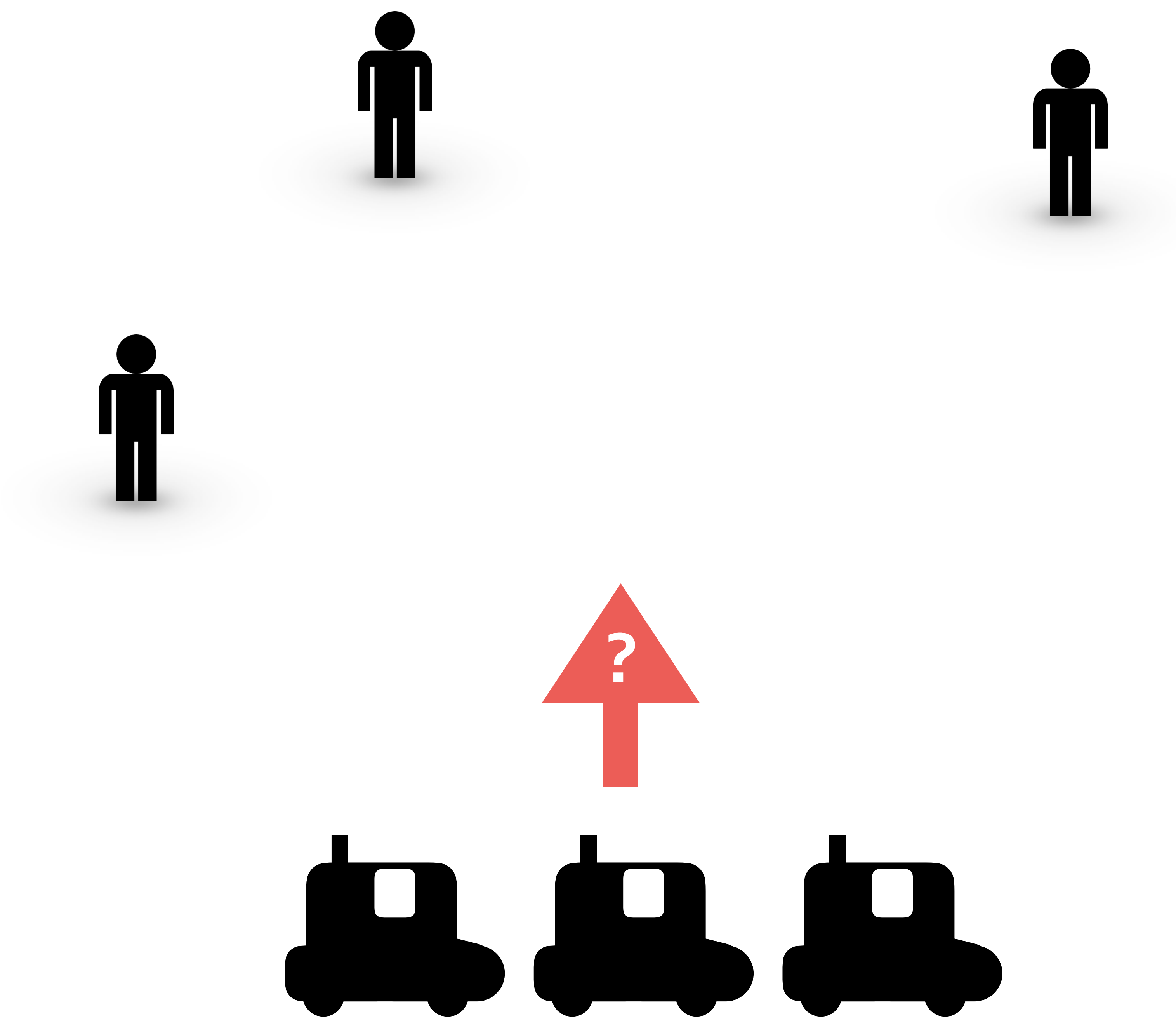
Assignment Problems



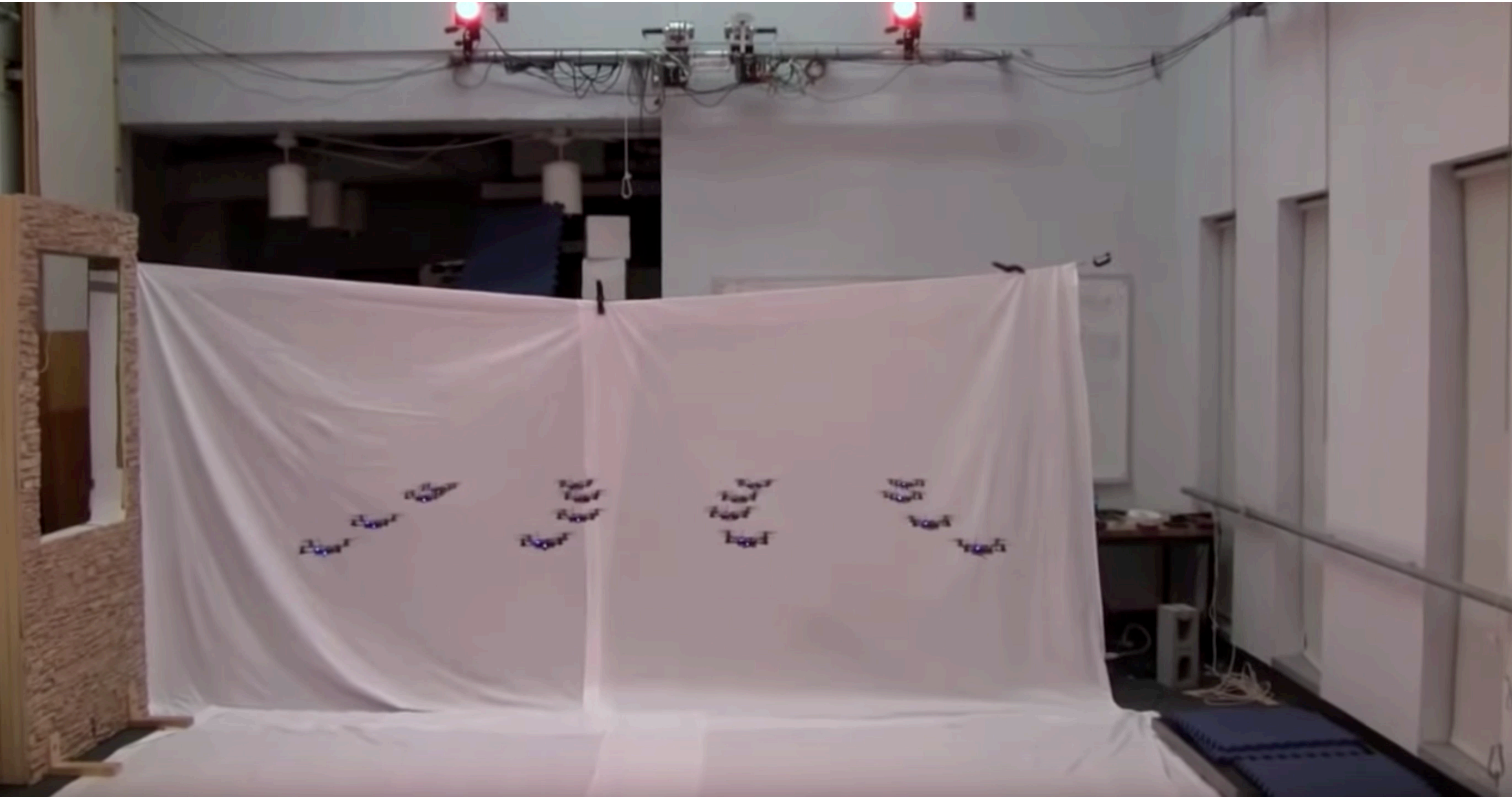
Assignment Problems



Assignment Problems



Assignment Problems



[Kumar et al.; UPenn]

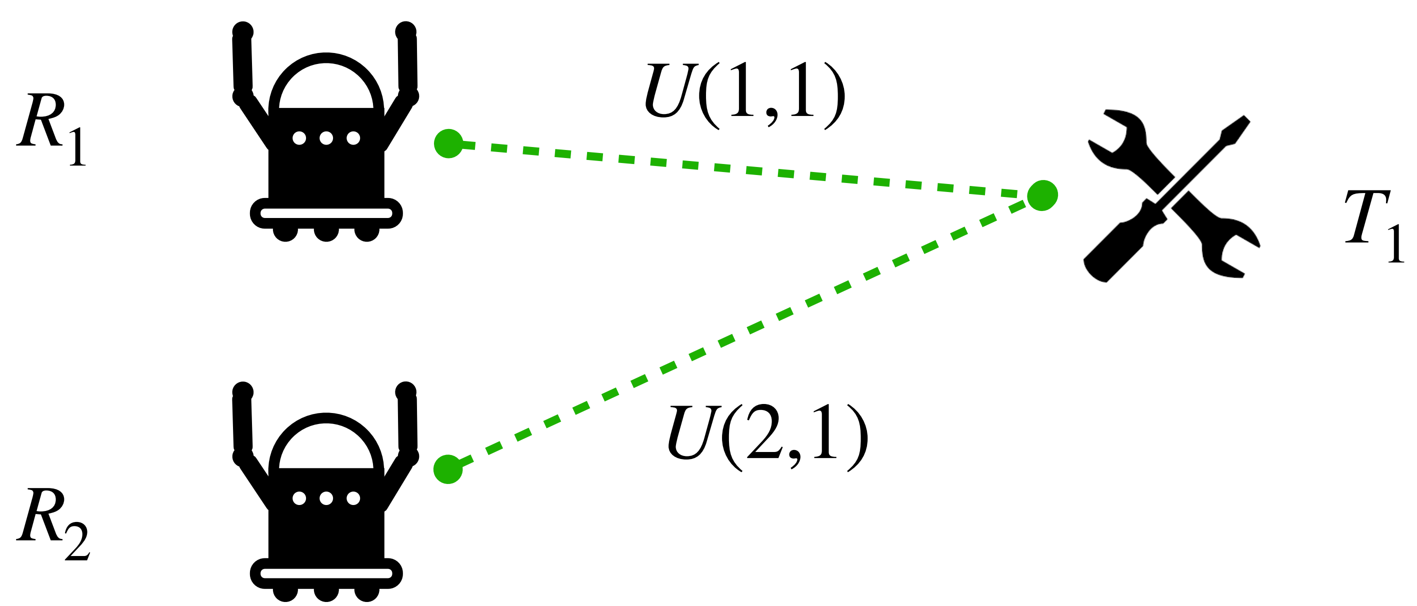
The Assignment Problem

- 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 $\longrightarrow$ *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:

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



The Linear Assignment Problem

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

find x_{ij} that maximize:

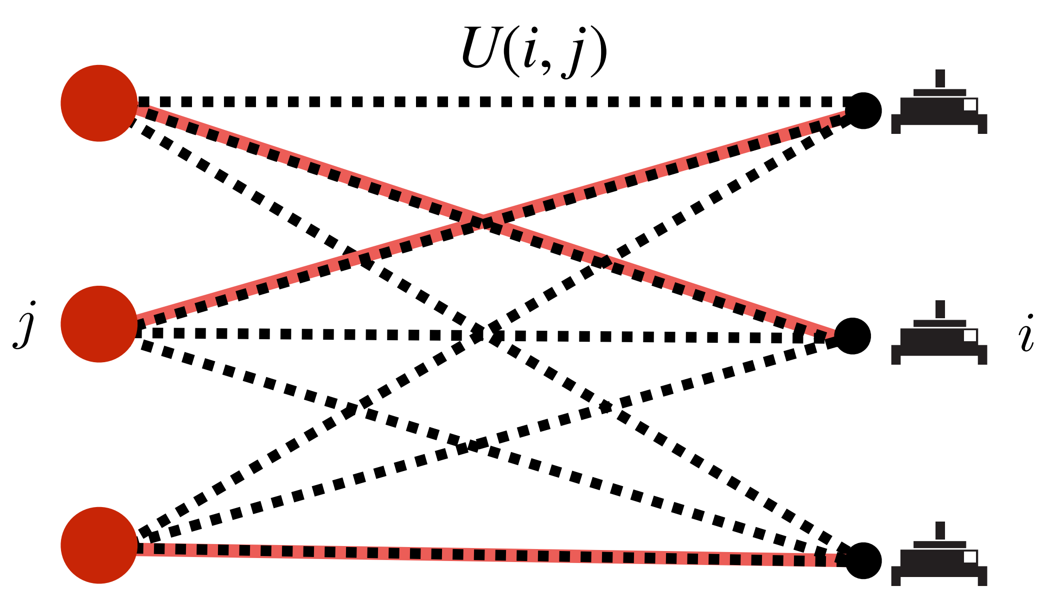
$$\mathcal{U} = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}\, U(i, j)$$

subject to

$$\sum_{i=1}^{m} x_{ij} = 1, \quad 1 \le j \le n$$

$$\sum_{j=1}^{m} x_{ij} = 1, \quad 1 \le i \le m$$

tasks



robots

bipartite perfect matching (complete graph)

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 x 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

	T1	T2	T3	T4
R1	82	83	69	92
R2	77	37	49	92
R3	11	69	5	86
R4	8	9	98	23

Step 1: subtract row minima

	T1	T2	T3	T4	
R1	13	14	0	23	-69
R2	40	0	12	55	-37
R3	6	64	0	81	-5
R4	0	1	90	15	-8

Step 2: subtract column minima

	T1	T2	T3	T4
R1	13	14	0	8
R2	40	0	12	40
R3	6	64	0	66
R4	0	1	90	0
	-0	-0	-0	-15

Step 3: cover all zeros with a minimum of lines

	T1	T2	T3	T4
R1	13	14	0	8
R2	40	0	12	40
R3	6	64	0	66
R4	0	1	90	0

3 lines found

Step 4: create additional zeros

	T1	T2	T3	T4
R1	13	14	0	8
R2	40	0	12	40
R3	6	64	0	66
R4	0	1	90	0

-6: unmarked elements
+6: twice marked elements

(find smallest uncovered element)

	T1	T2	T3	T4
R1	7	8	0	2
R2	40	0	18	40
R3	0	58	0	60
R4	0	1	96	0

Step 3: cover all zeros with a minimum of lines

	T1	T2	T3	T4
R1	7	8	0	2
R2	40	0	18	40
R3	0	58	0	60
R4	0	1	96	0

4 lines found

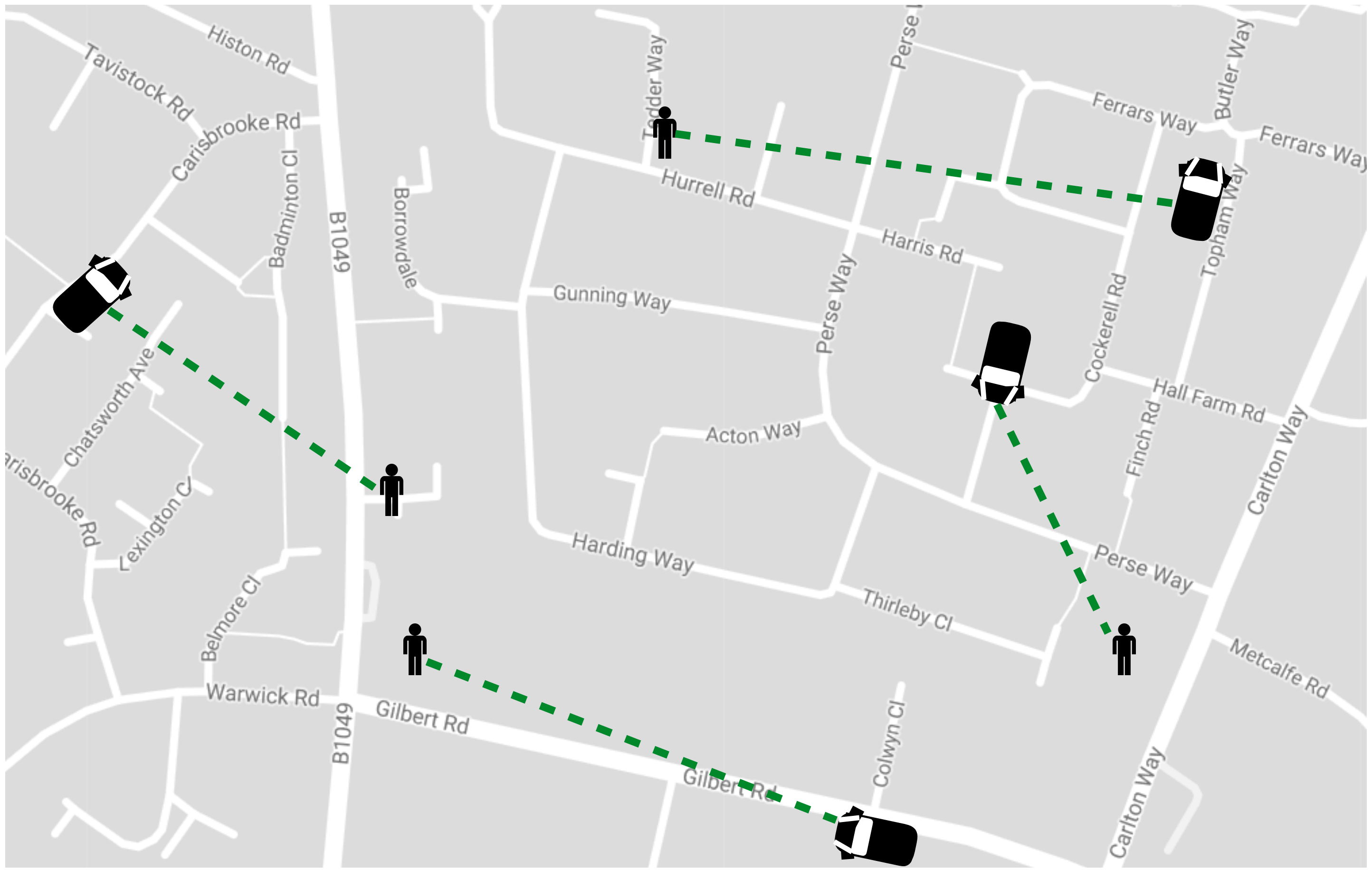
Stop: An optimal assignment exists.

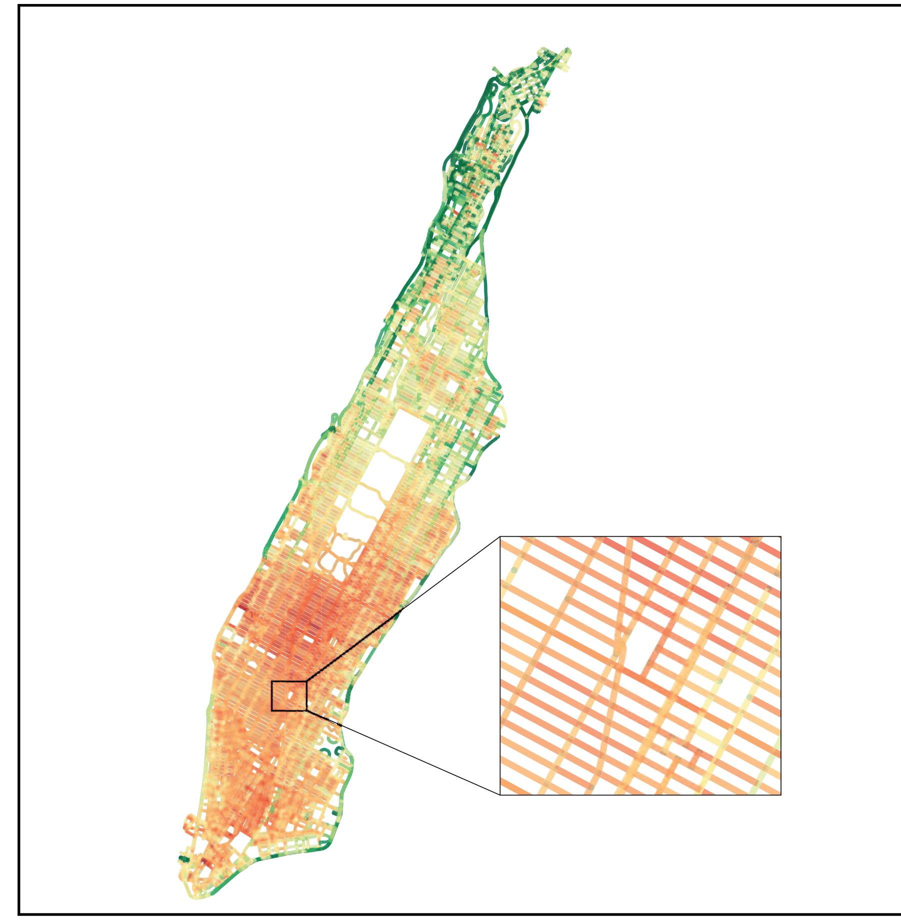
	T1	T2	T3	T4
R1	7	8	0	2
R2	40	0	18	40
R3	0	58	0	60
R4	0	1	96	0

unique, optimal assignment found

*Example from www.hungarianalgorithm.com

Application: Vehicle-to-Passenger Assignment





Goal: find optimal assignment matrix **A***

$$\mathbf{A}^\star = \underset{\mathbf{A}}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} a_{ij}$$

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

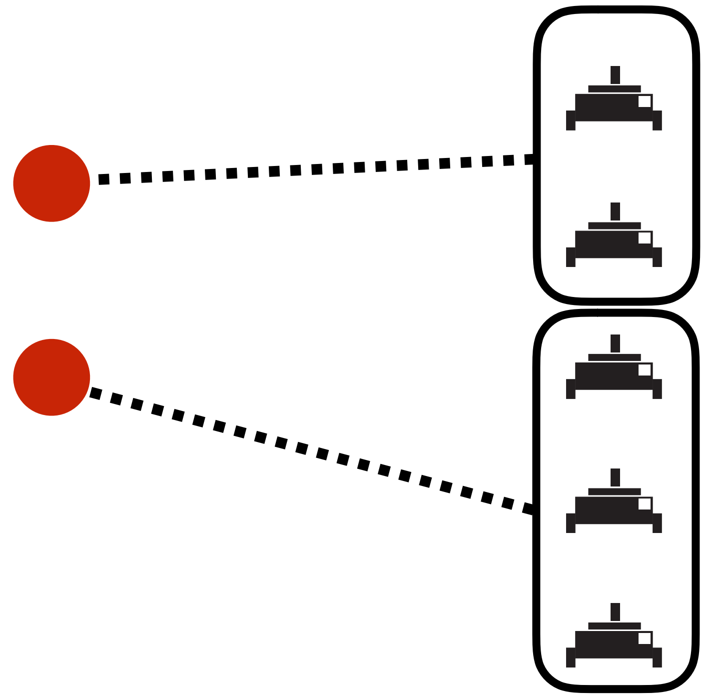
- Assumptions when using an assignment algorithm such as the Hungarian method:
  - Costs (utilities) are known at a **centralized** computation unit.
  - Costs (utilities) are **deterministic** (no noise).
  - Costs (utilities) do not change (**constant**).
  - **1-to-1** assignment (one robot per task,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

all of these issues are very common in robotics!!

**see Prorok, DARS 2018 for a solution

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**:

$\boldsymbol{E}$ is the ground set (all robots) and $\boldsymbol{X}$ is a family of subsets.

$y \bigcap z = \varnothing \quad \forall y, z \in X, y \neq z$ robot subsets are mutually disjoint

$\bigcup_{x \in X} = E$ the union of subsets is equivalent to the ground set.

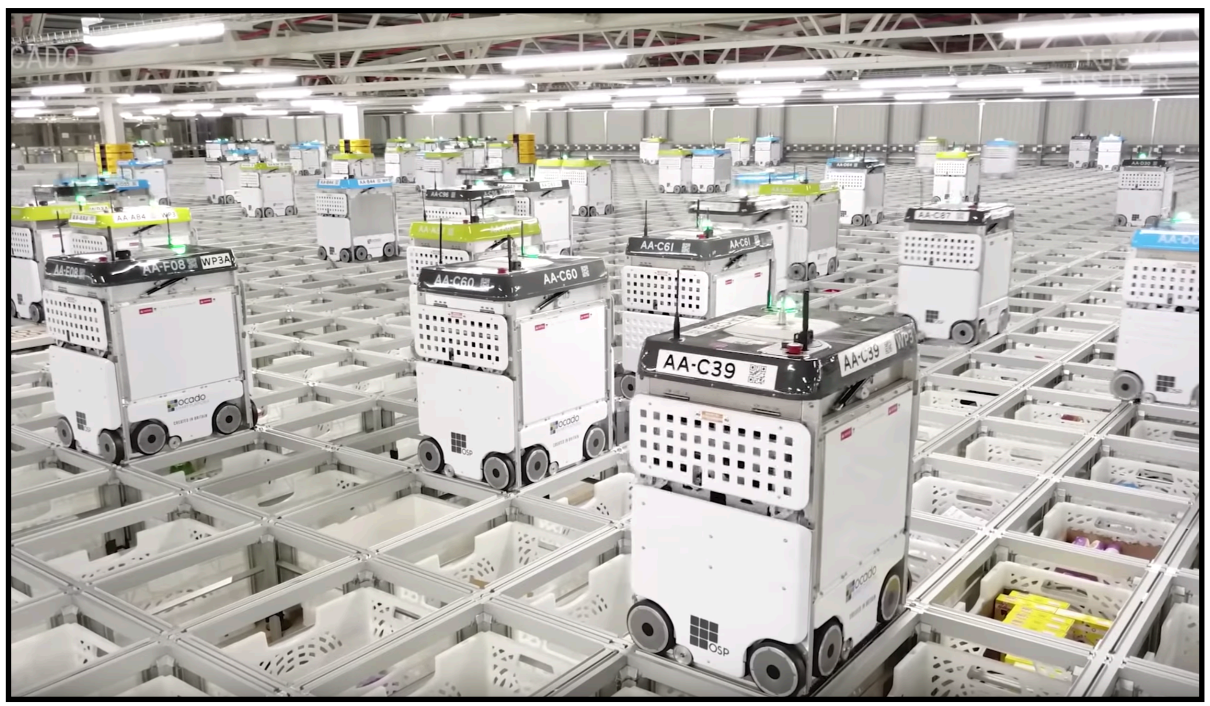
Set Partitioning Problem: Given a finite set E, a family F of acceptable subsets of E, and a utility function $u : F \mapsto \mathbb{R}_+$, find a maximum-utility family X of elements in F such that X is a partition of E.

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

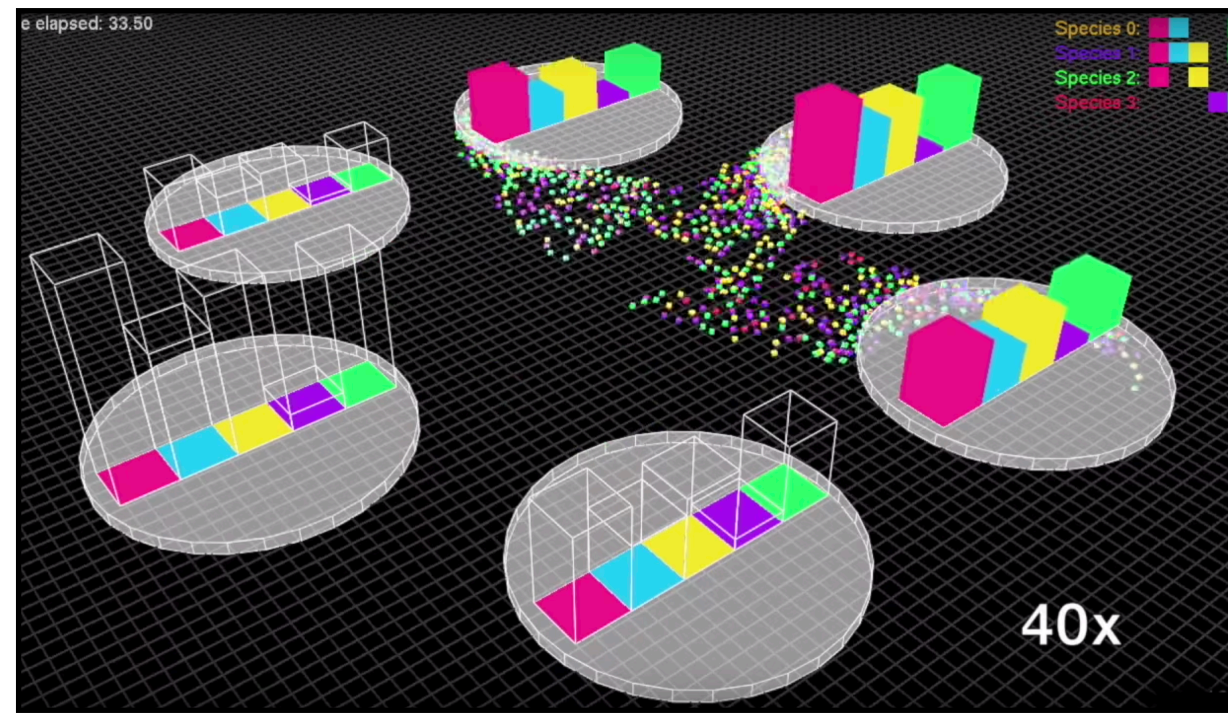
… One potential solution: *relaxation of the problem to the continuous domain.*

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!



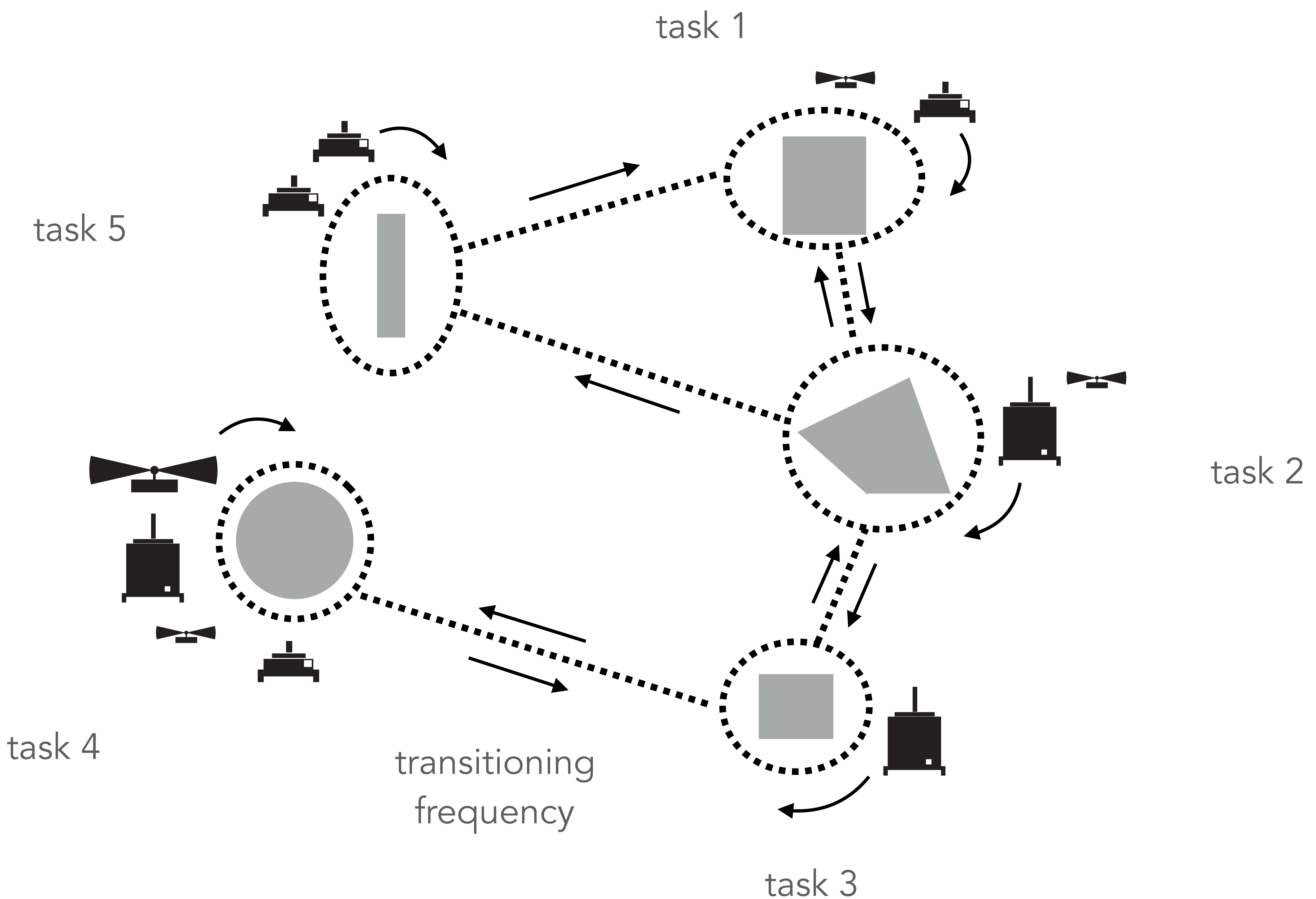
- robot-to-task allocation
- method: **combinatorial approach**
- exact, but computationally demanding



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

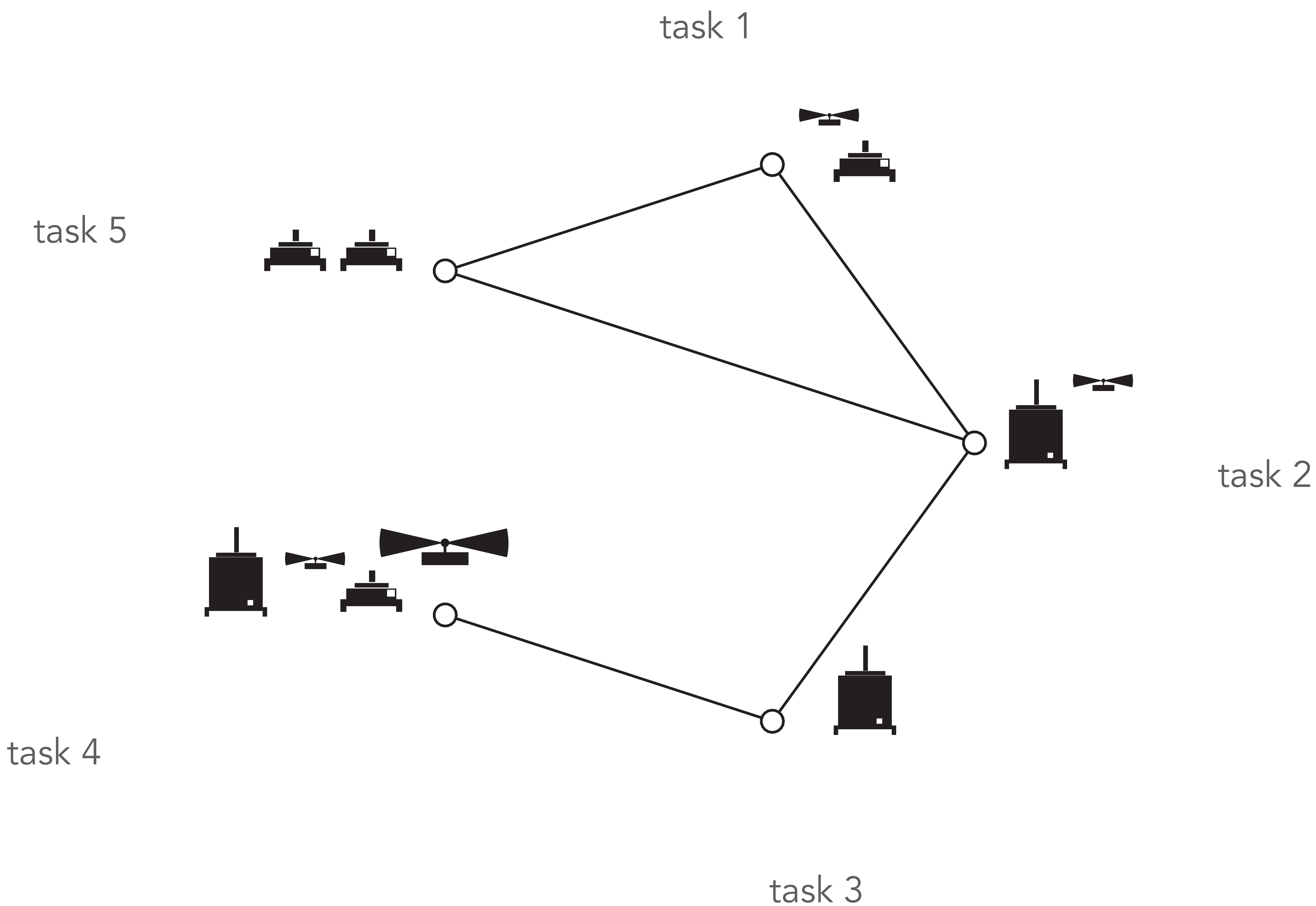
Redistribution of a Swarm of Robots

Example: monitor geographical sites



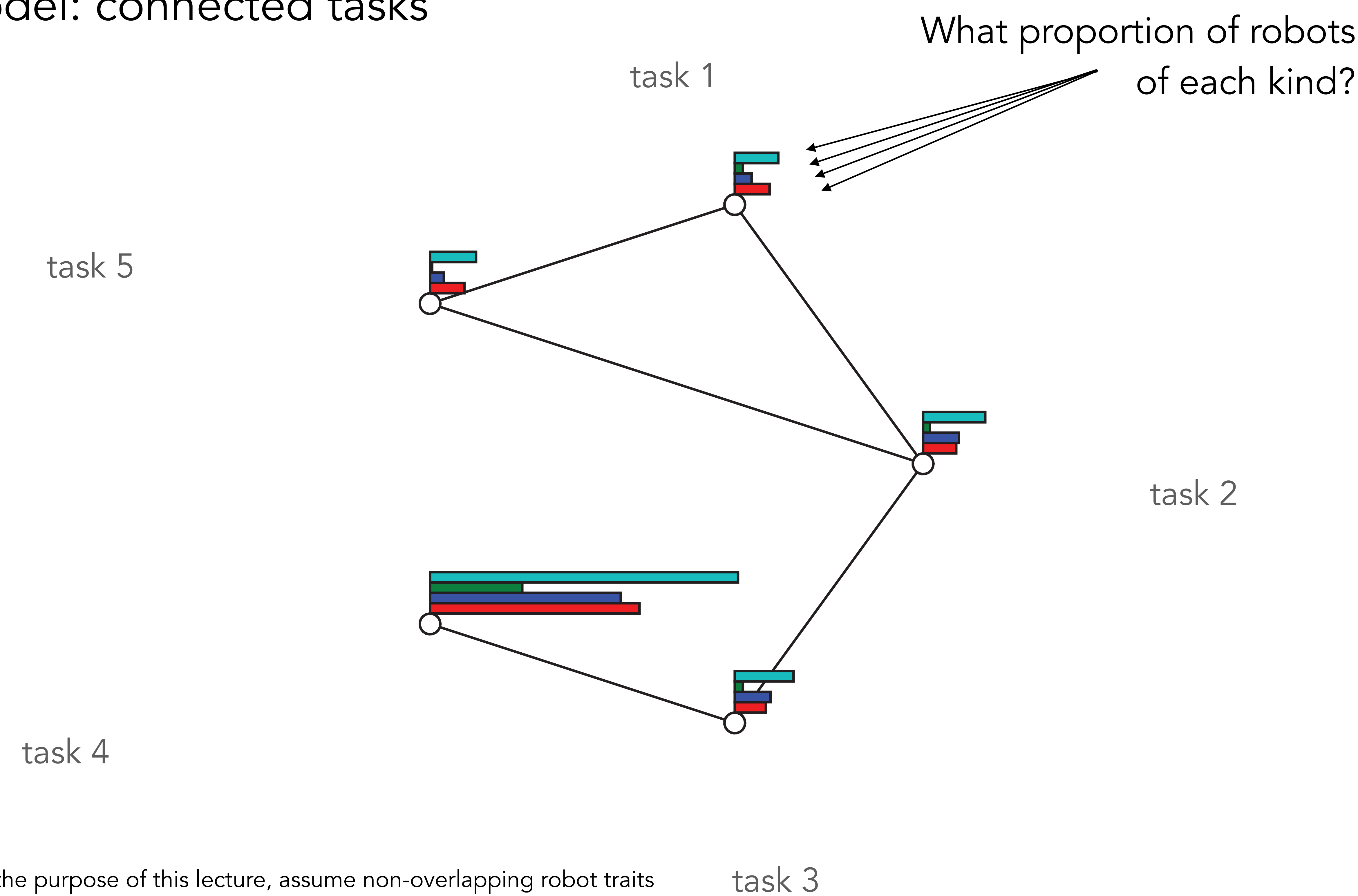
Redistribution of a Swarm of Robots

Model: connected tasks



Redistribution of a Swarm of Robots

Model: connected tasks



*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.: $\mathbf{x} = [0.3, 0.2, 0.1, 0.1, 0.3]^{\top}$

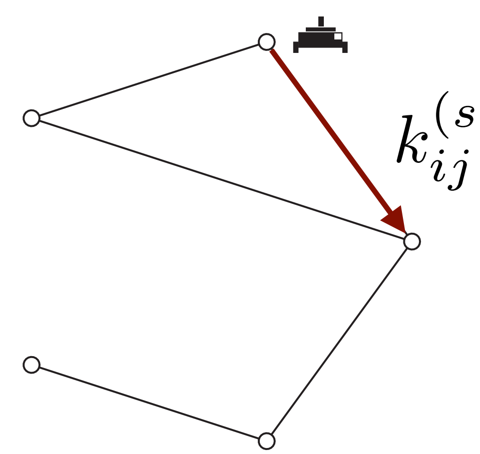
proportion of swarm at task 1

transition rate matrix

distribution of robots over tasks

Distribution dynamics:

$$\underbrace{\dot{\mathbf{x}}^{(s)}}_{\substack{\text{change in distribution of} \\ \text{robots of type } (s) \text{ over tasks}}} = \underbrace{\mathbf{K}^{(s)}}_{\substack{\text{rates} \\ \text{M x M}}} \underbrace{\mathbf{x}^{(s)}}_{\substack{\text{robots} \\ \text{M x 1}}}$$



(s): robot species

Note: if matrix $\mathbf{K}$ has certain properties, this system is stable.

Redistribution of a Swarm of Robots

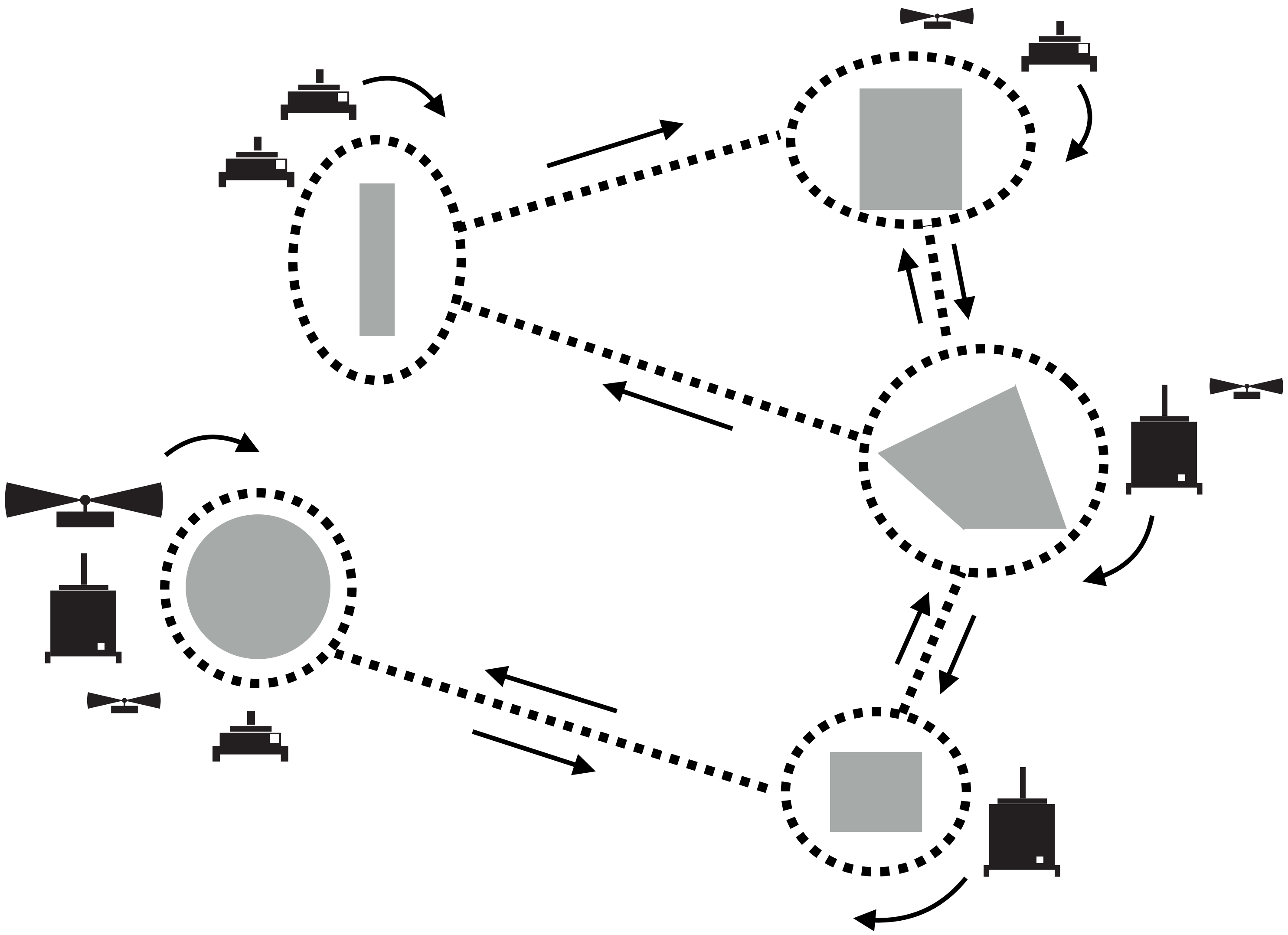
Robot distribution dynamics: $$\dot{\mathbf{x}}^{(s)} = \underbrace{\mathbf{K}^{(s)}}_{\substack{\text{rates} \\ \text{M x M}}} \underbrace{\mathbf{x}^{(s)}}_{\substack{\text{robots} \\ \text{M x 1}}}$$

Solution: $$\mathbf{x}^{(s)}(t) = e^{\mathbf{K}^{(s)}t}\,\mathbf{x}_0^{(s)}$$

Given a desired robot distribution $\mathbf{x}^{(s)\star}$
Find transition rates $\mathbf{K}^{(s)\star}$ that are fastest to satisfy $\mathbf{x}^{(s)\star}$

Methods:
1. Explicit optimization; [Prorok 2016]
2. Approximation of **K**; semi-definite programming [Berman 2009]
3. Stochastic optimization [Matthey 2009, Hsieh 2008]

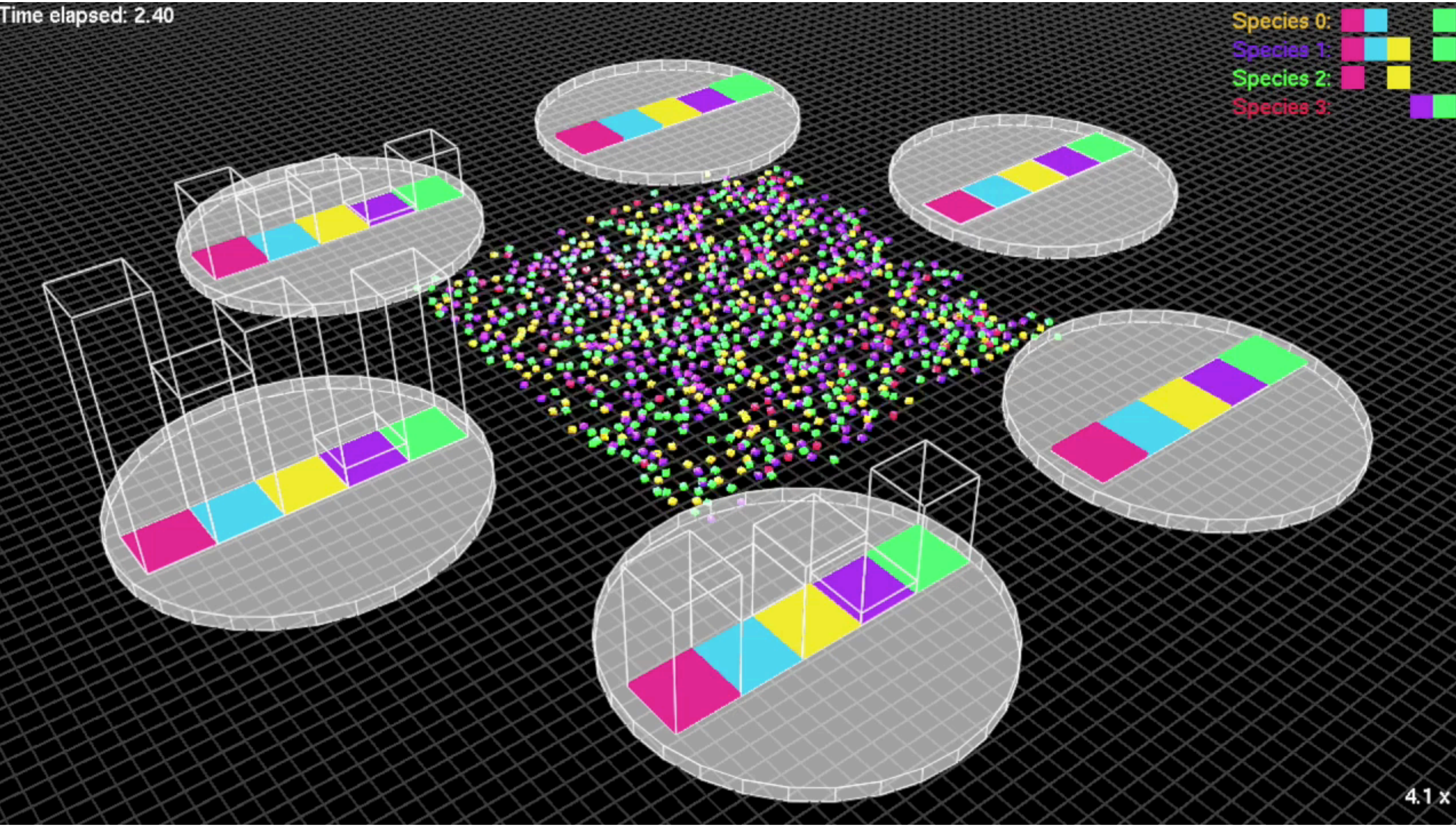
Controller Synthesis



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

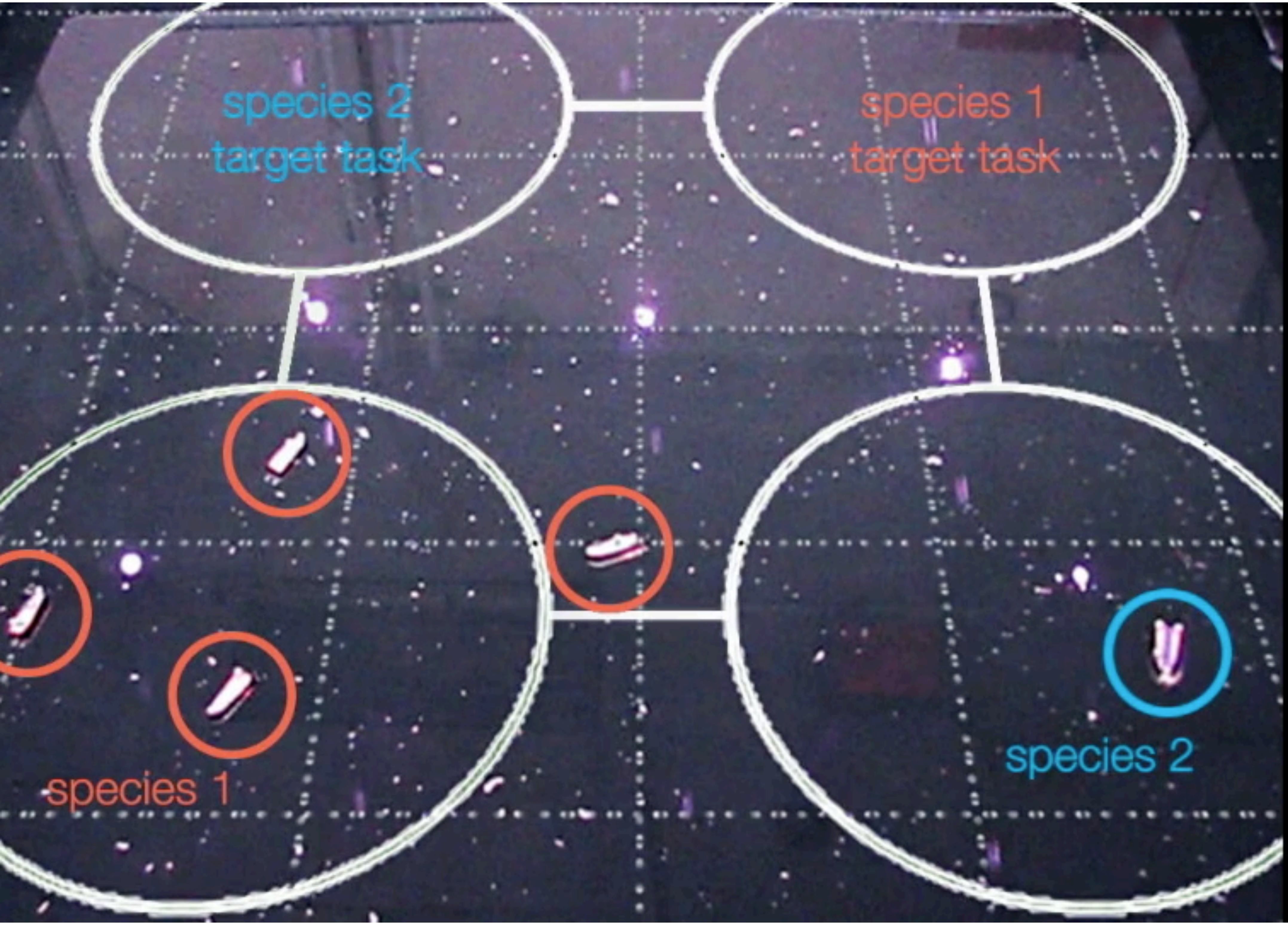
- Probabilistic controller is immediate
- Deterministic controller can also be derived
- Architecture: both open-loop and closed-loop possible

Redistribution of a Heterogeneous Swarm



[Prorok et al.; ICRA 2016; T-RO 2017]

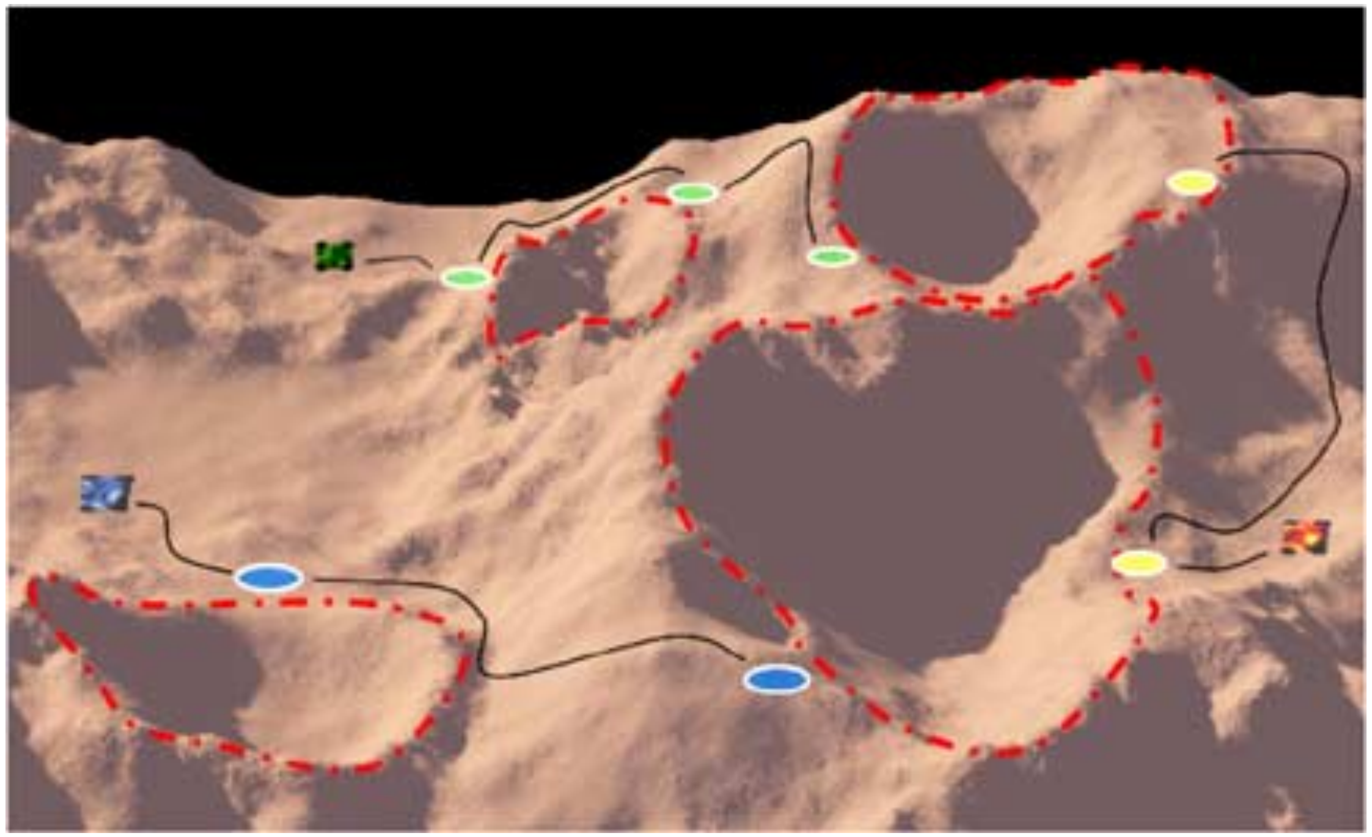
Redistribution of a Heterogeneous Swarm



[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?

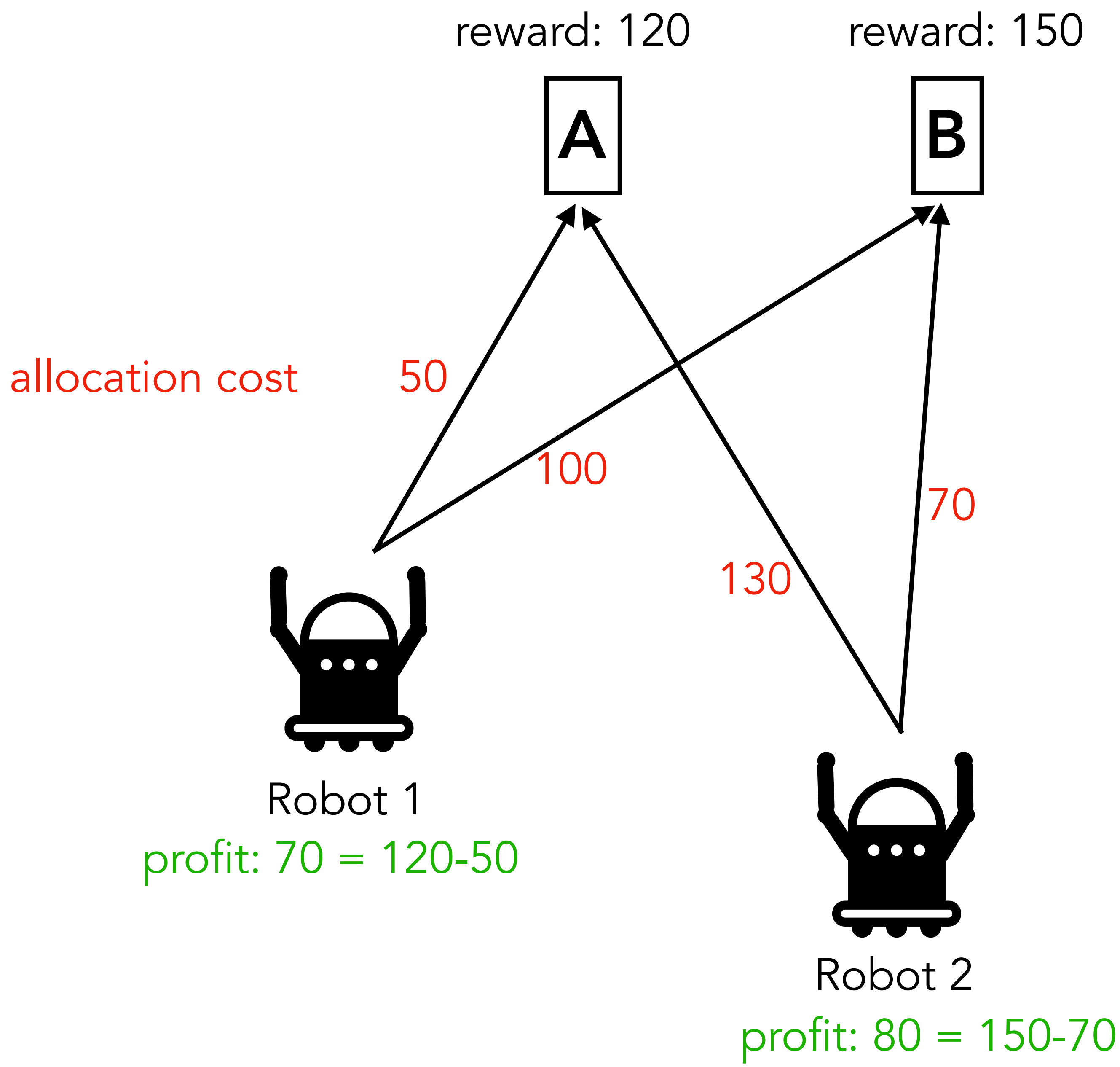
*image credit: Dias et al.

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

Market-Based Coordination

A simple example (multi-item auction)



bids placed for tasks

	A	B
Robot 1	50	100
Robot 2	-	70

reserve price not met

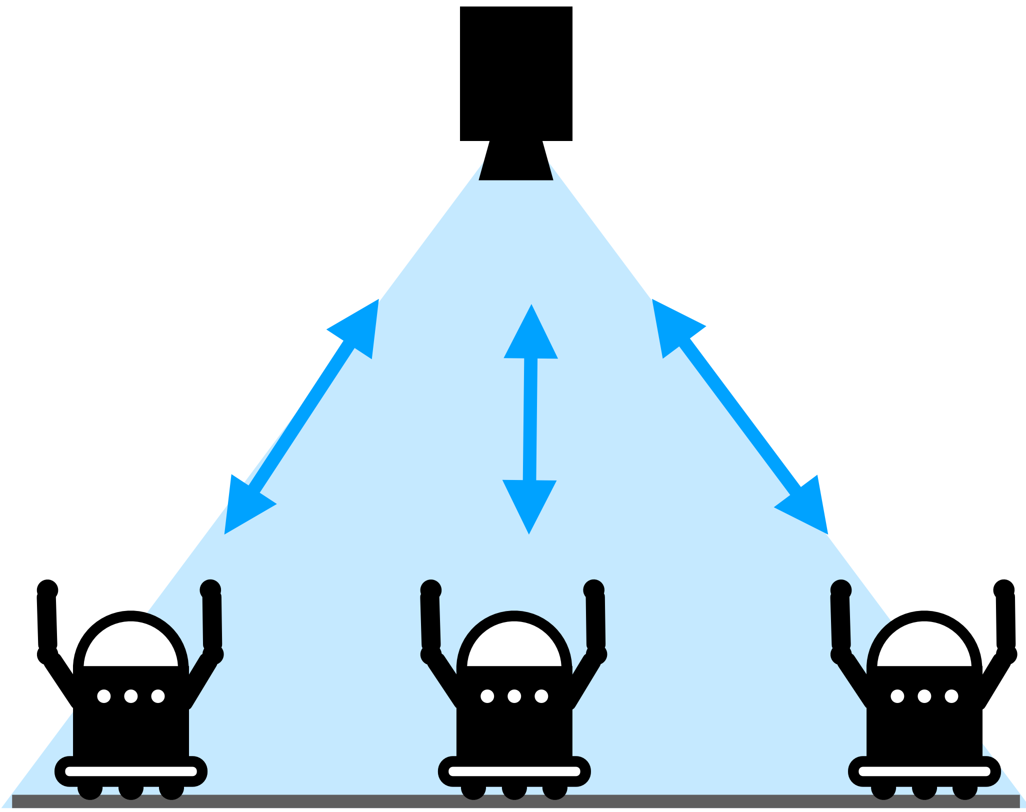
system cost: 50+70 = 120

Running time: *O(NRM)* (greedy) or *O(N^2R)* (optimal) [T. Sandholm; 2002]

Market-Based Allocation Frameworks

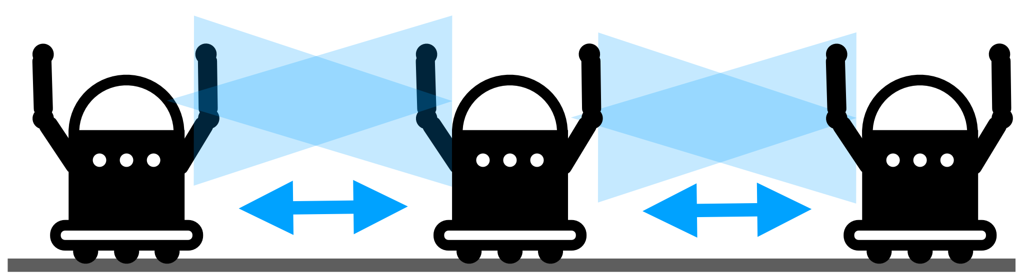
- Murdoch [Gerkey, Mataric; 2002]
  - loosely coordinated tasks
  - demonstrated on box pushing
  - demonstrated robustness, fast auctioning
- TraderBots [Dias et al.; 2004]
  - loosely coordinated tasks
  - demonstrated on exploration tasks
  - demonstrated robustness, scalability, auction types, task trees
- Hoplites [Kalra, Stentz; 2005]
  - tightly coordinated spatial tasks
  - robots auction plans not tasks
  - demonstrated on perimeter sweeping, constrained exploration

Centralized vs Decentralized Assignment



centralized

- Centralized assignment. Cost estimates are known at a central point (computational unit). The unit performs the assignment and communicates with all robots.



decentralized

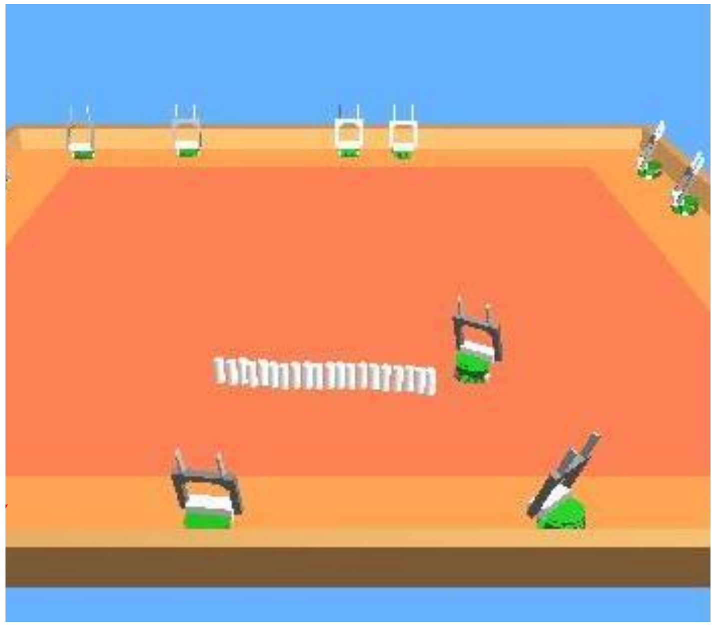
- Decentralized assignment. Robots do not have global knowledge of each other's costs. They locally negotiate assignments.

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

initial situation



final situation

Threshold-Based Assignment

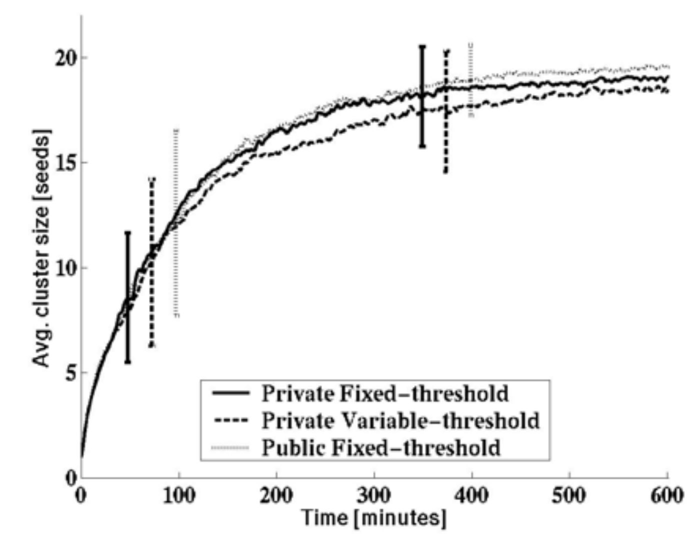
- **Stimulus**: time needed to find a stick to manipulate (the longer the time, the lower the stimulus associated with the task).
- **Threshold** is self-calibrated (fully decentralized).
- Key: The number of manipulation sites (either end of line of sticks) decreases as global task nears completion.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

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

threshold

number of sticks successfully collected so far

time taken to find k^{th} stick



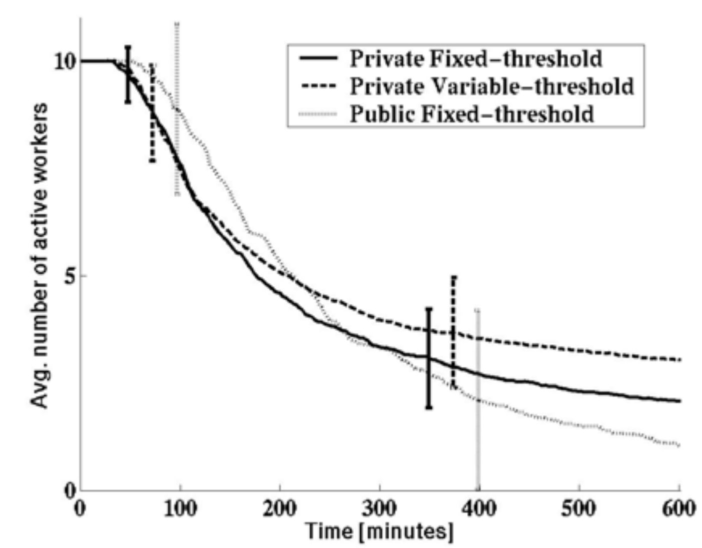


*image credit: Agassounon et al.

Overview of Allocation Methods

	centralized vs decentralized	**optimality**	**completeness**
Hungarian method	centralized	optimal	guaranteed
Mean-field approach	centralized or decentralized	approximative	The system converges. With high probability, completeness is guaranteed
Market-based approach	centralized or decentralized	greedy (sub-optimal) or optimal	depends on reserve price
Threshold-based approach	decentralized	suboptimal	not guaranteed

Further Reading

Nice overview of the classical problem:

http://www.assignmentproblems.com/

Seminal papers:

- B. Gerkey and M. Mataric, "A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems". Int. Journal of Robotics Research, 2004.
- M. B. Dias et al; "Market-Based Multirobot Coordination: A Survey and Analysis"; 2006
- D.P. Bertsekas, "The Auction Algorithm: A Distributed Relaxation Method for the Assignment Problem"; 1988.
- 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 in heterogeneous robot swarms: A. Prorok, M. A. Hsieh, and V. Kumar. The Impact of Diversity on Optimal Control Policies for Heterogeneous Robot Swarms. IEEE Transactions on Robotics (T-RO); 2017.
- 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

2019 IEEE RAS Summer School on Multi-Robot Systems

Multi-Robot Navigation and Path Planning

Lecture 3

Dr. Amanda Prorok

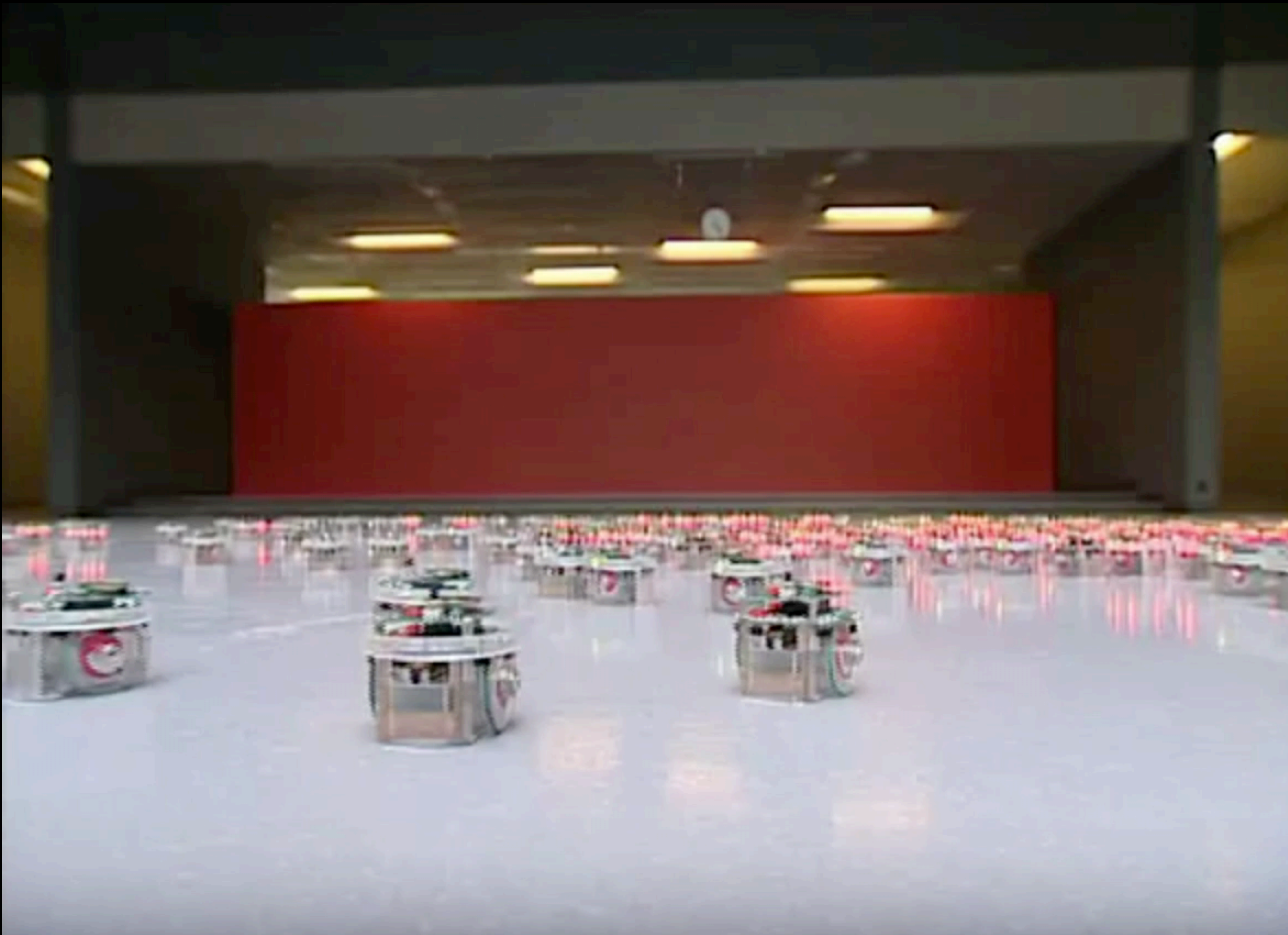
Assistant Professor, University of Cambridge

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

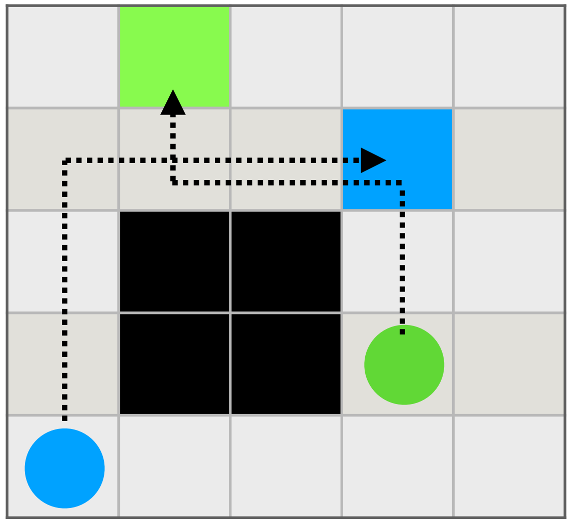


Taxonomy of Multi-Robot Path Planning Problems

- 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

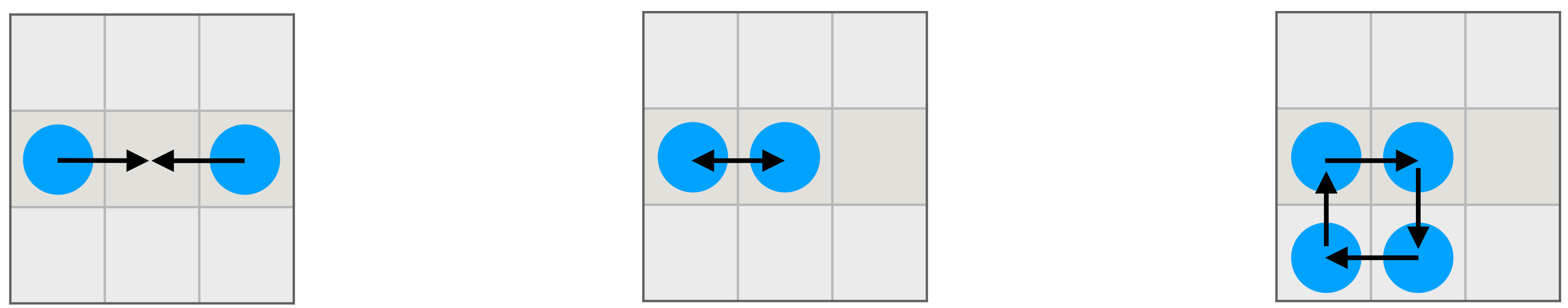
- Multi-robot path planning $\longrightarrow$ multi-agent path planning:
  - discretized environment (grids or planar graphs)
  - point robots (holonomic, no motion constraints)



- 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

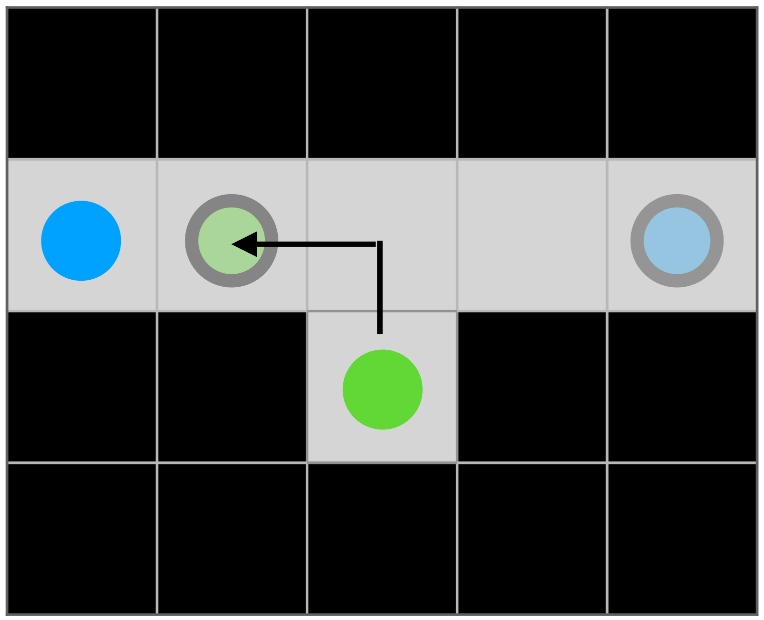
- Allowed motion: North, East, South, West
- Collisions:



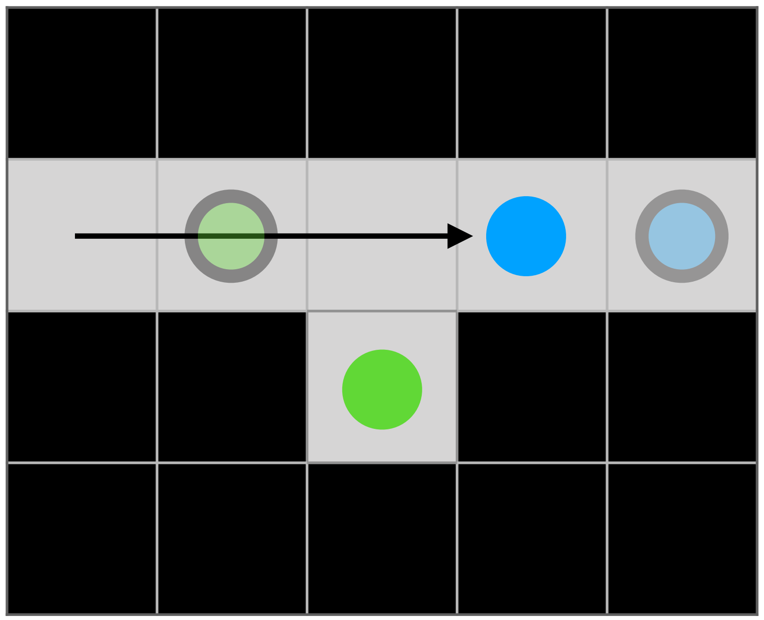
vertex-collision   edge-collision   no collision

- Performance metrics
  - **Makespan**: time of last robot's arrival time
  - **Flowtime**: sum of arrival times, over all robots

Coupled vs Decoupled Path Planning



Potential deadlock



Completeness achieved.

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

Coupled Path Planning

Coupled formulation:

Robot i has configuration space: $\mathscr{C}_i$

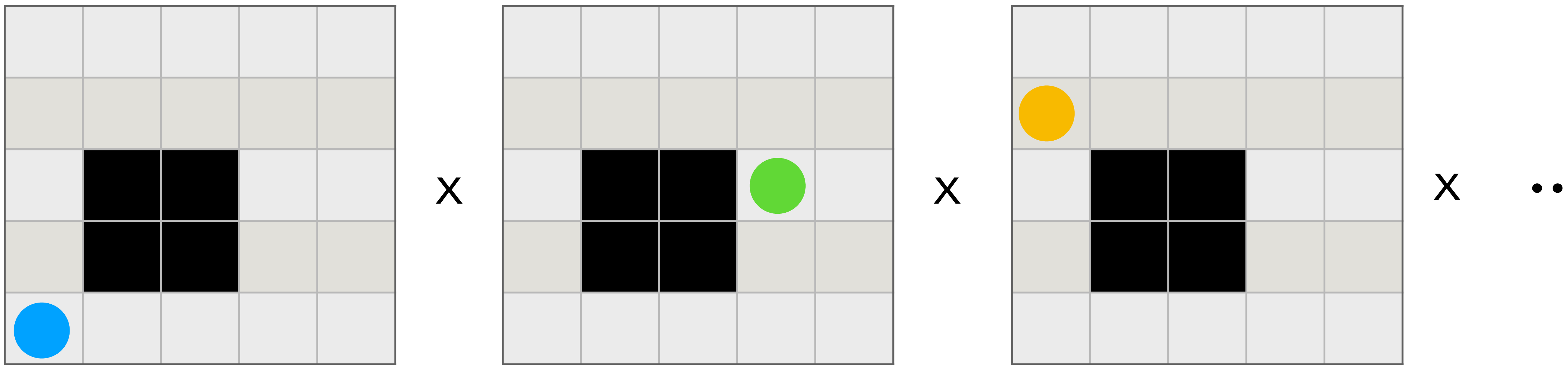
The joint state space is given by the Cartesian product:

$$X = \mathscr{C}_1 \times \mathscr{C}_2 \times \ldots \times \mathscr{C}_n$$

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:



$\mathscr{C}_1$ x $\mathscr{C}_2$ x $\mathscr{C}_3$ x …

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

E.g., worst case complexity for A*: $O(|E|) \approx O(|V|) = O(M^N)$

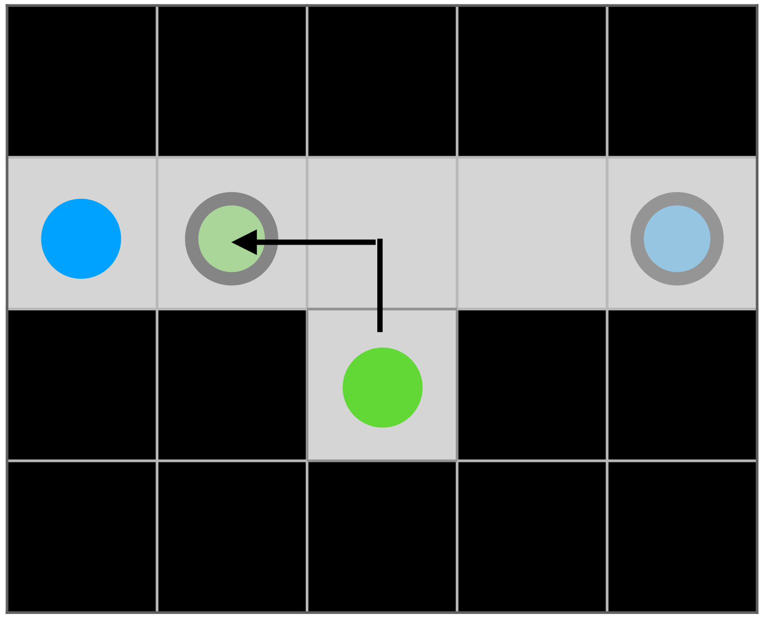
Exponential complexity in the number of robots!

* if graph is sparse

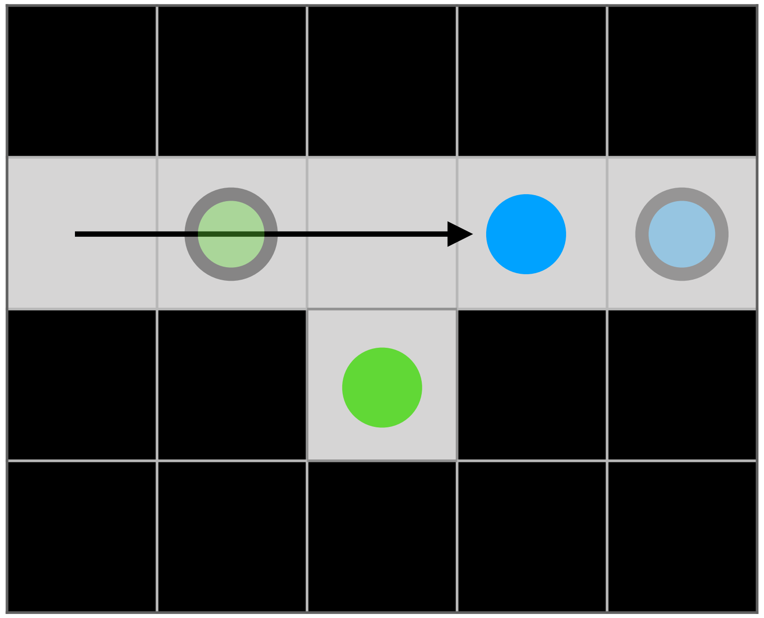
Coupled Path Planning

- 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)

Coupled vs Decoupled Path Planning



Potential deadlock

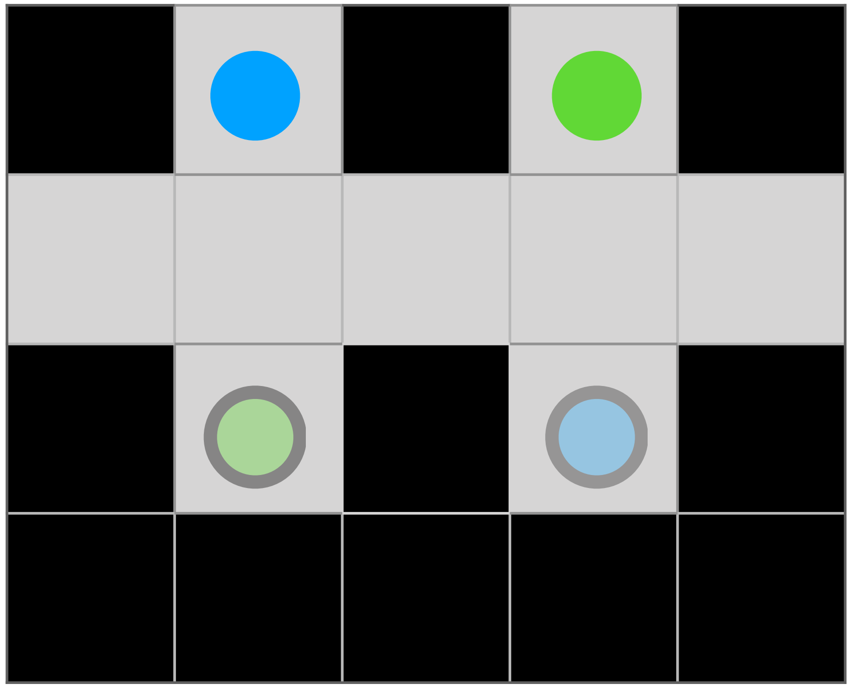


Completeness achieved.

- 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]

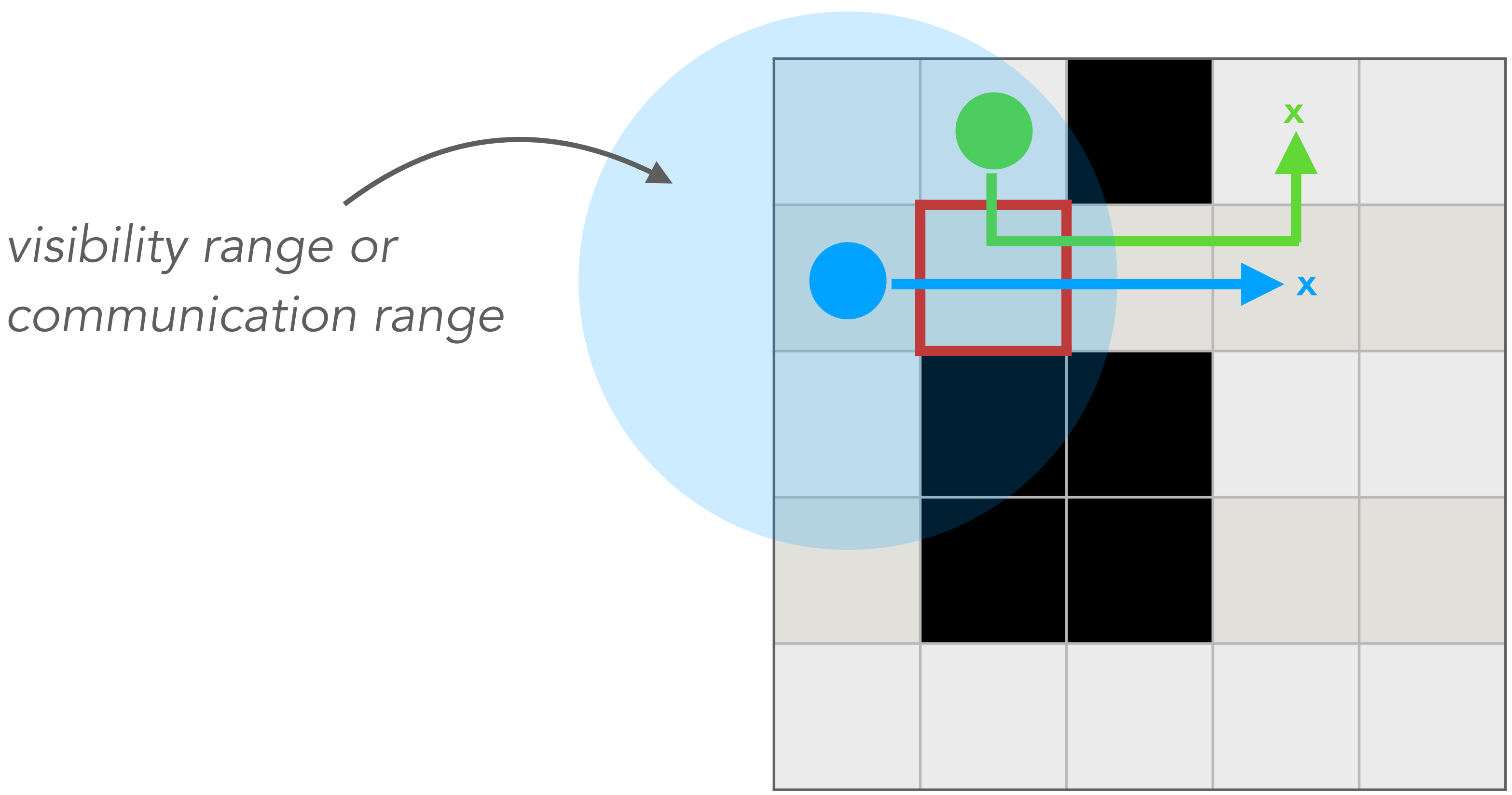
Decoupled Path Planning



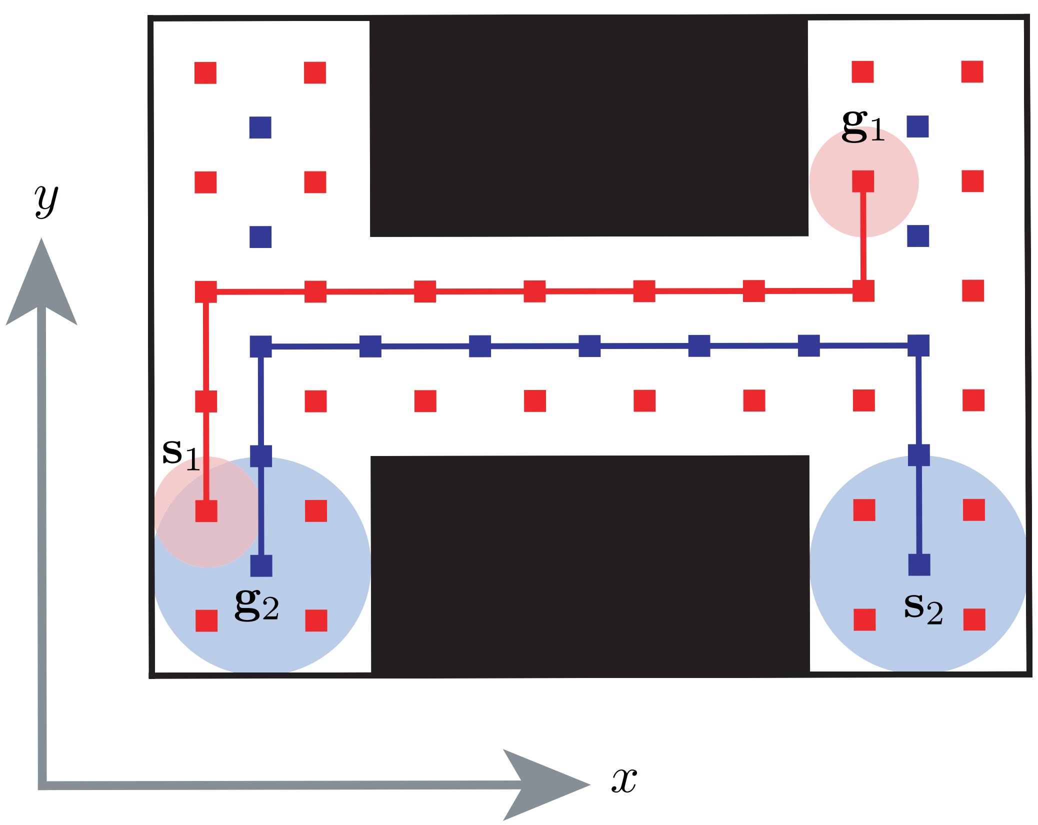
- Well-formed environment:
  - There must exist a path between any two endpoints.
  - That path must have with at least *R*-clearance with respect to static obstacles and at least *2R*-clearance to any other endpoint.
  - A robot is always able to find a collision-free trajectory to its goal by waiting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

Decoupled Path Planning

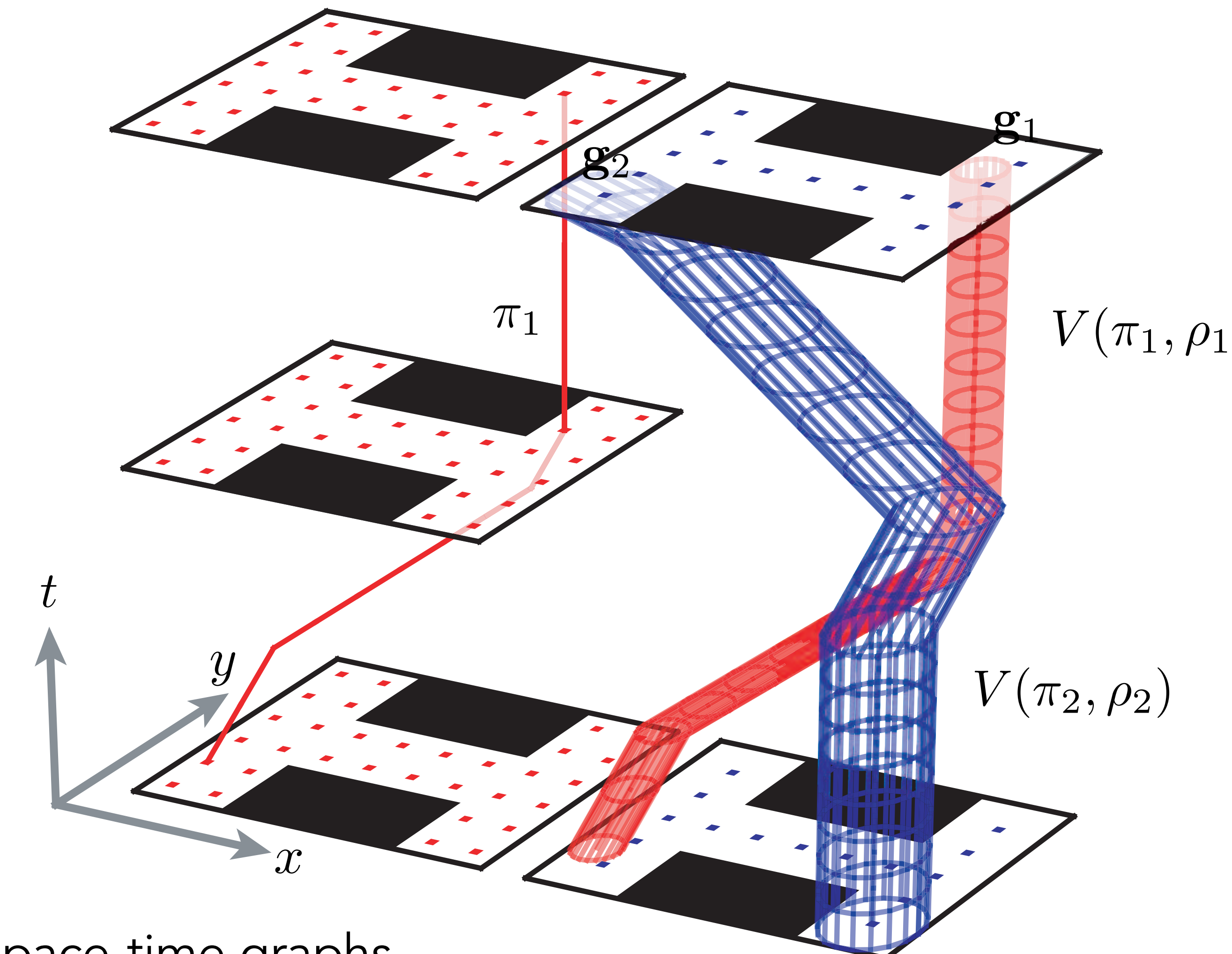
- De-coupling the problem:
  - Each robot plans in its own space-time
  - Robots negotiate path plans as conflicts arise
  - De-confliction can be online (dynamic) or offline (a-priori)



Decoupled, Prioritized Path Planning



Ideal trajectories for 2 robots



Space-time graphs

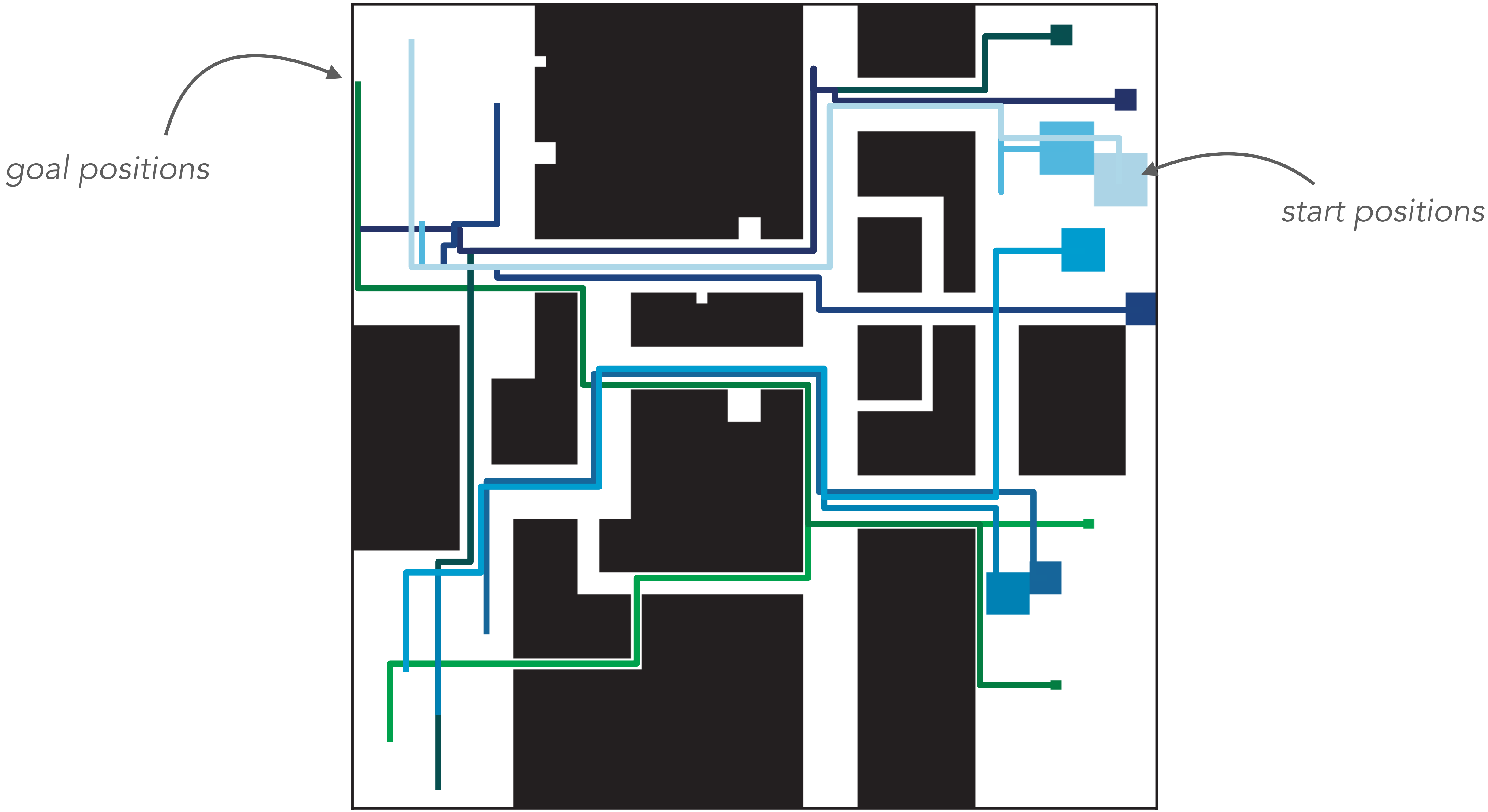
The red robot is prioritized and plans a space-time path that is optimal.
The blue robot plans a path that does not collide with the red robot's path.

[Wu, Bhattacharya, Prorok; arxiv 2019]

Decoupled, Prioritized Path Planning

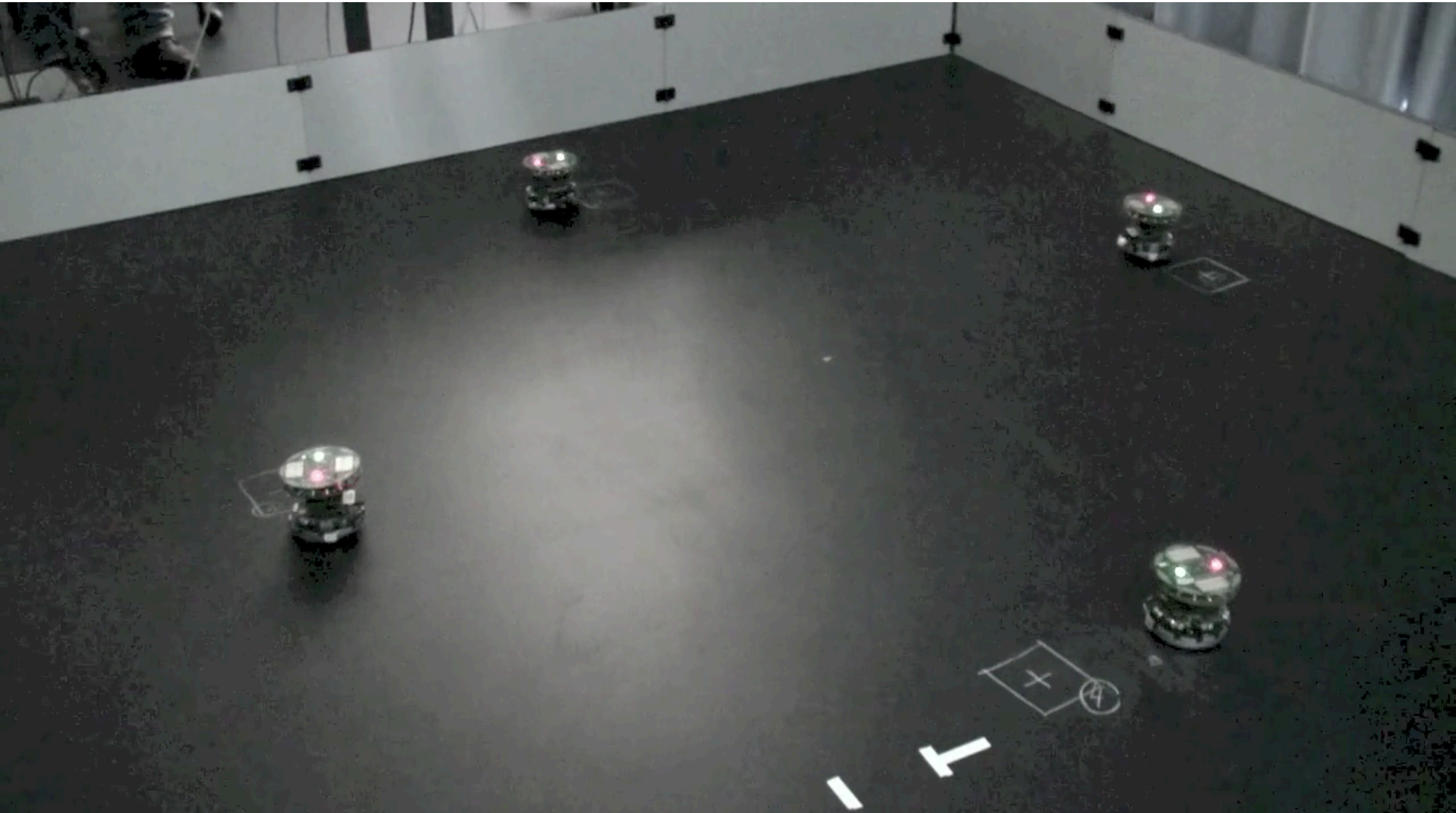
- Key question: How to prioritize robots?
- Online, exhaustive method:
  - Evaluate all $N!$ options (where N is robots within communication or visibility neighborhood) [Azarm, Schmidt; 1997]
- Existing **prioritization heuristics** (online and offline):
  - Ideal path length: Robots with longer ideal 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

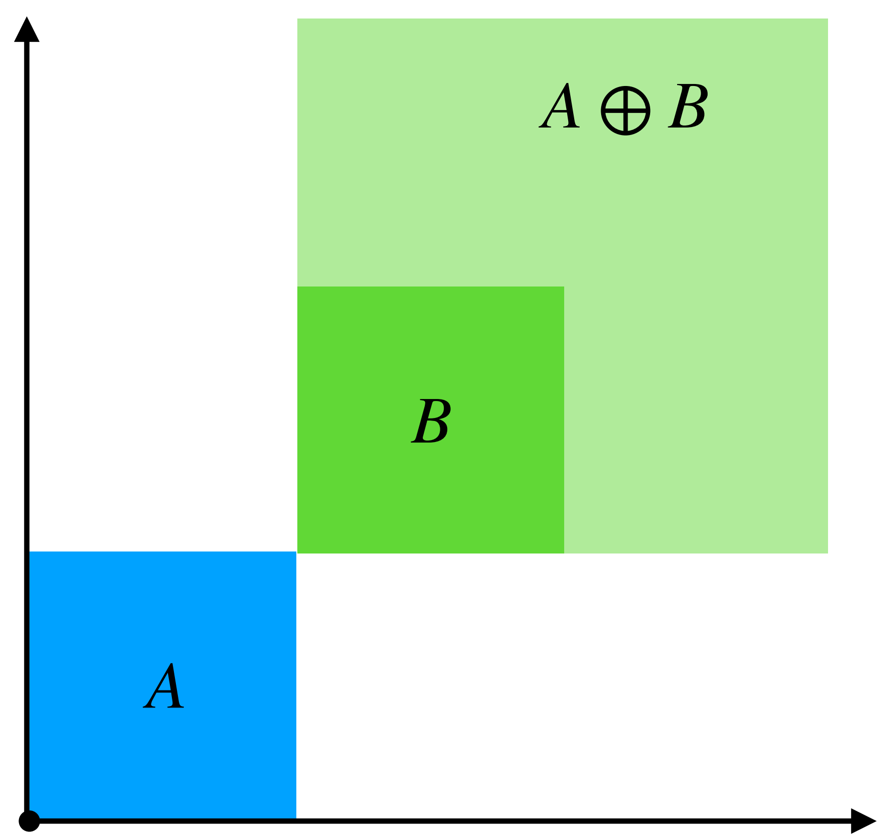


*movie credit: Gowal, Martinoli

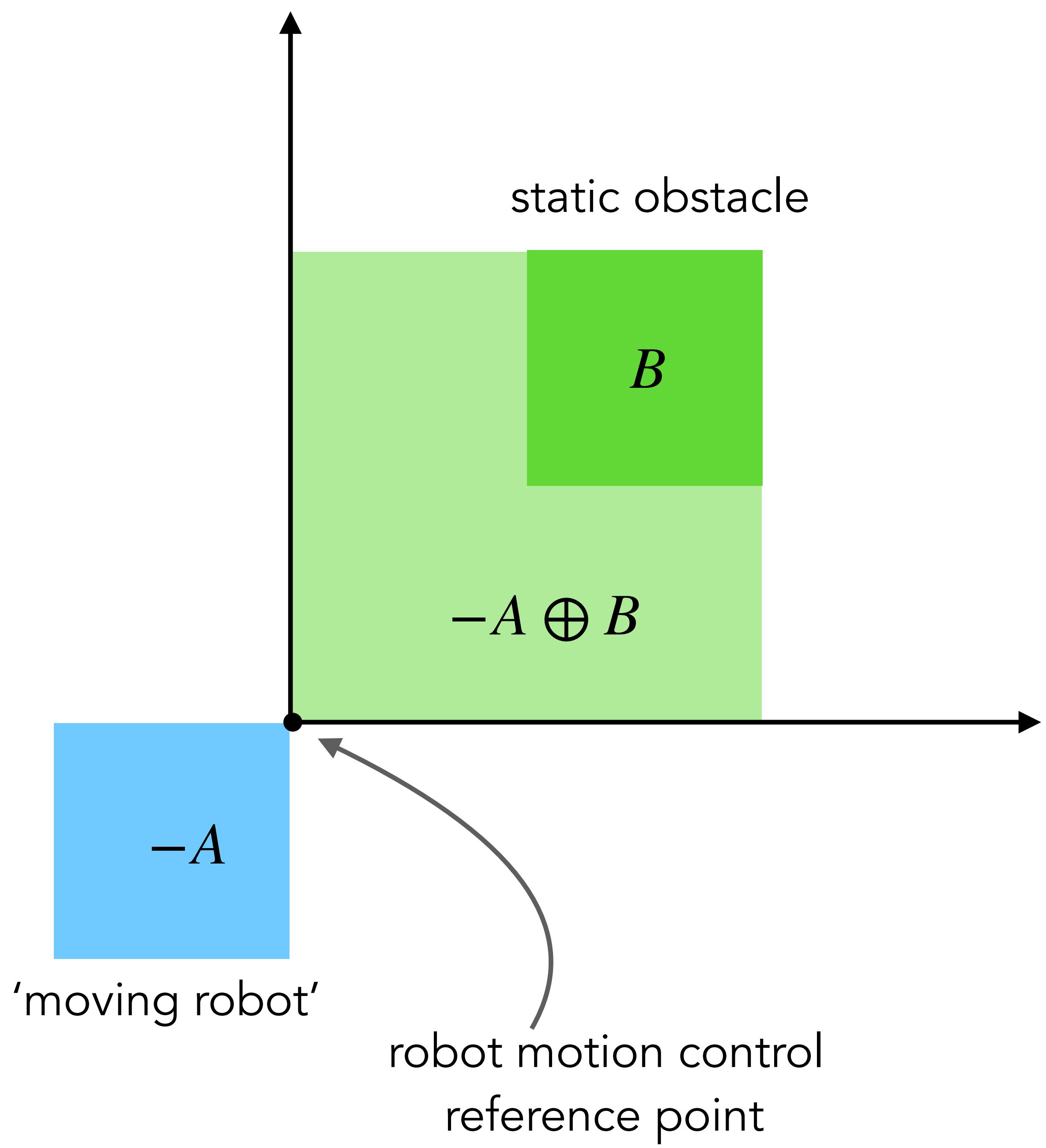
Minkowski Sum

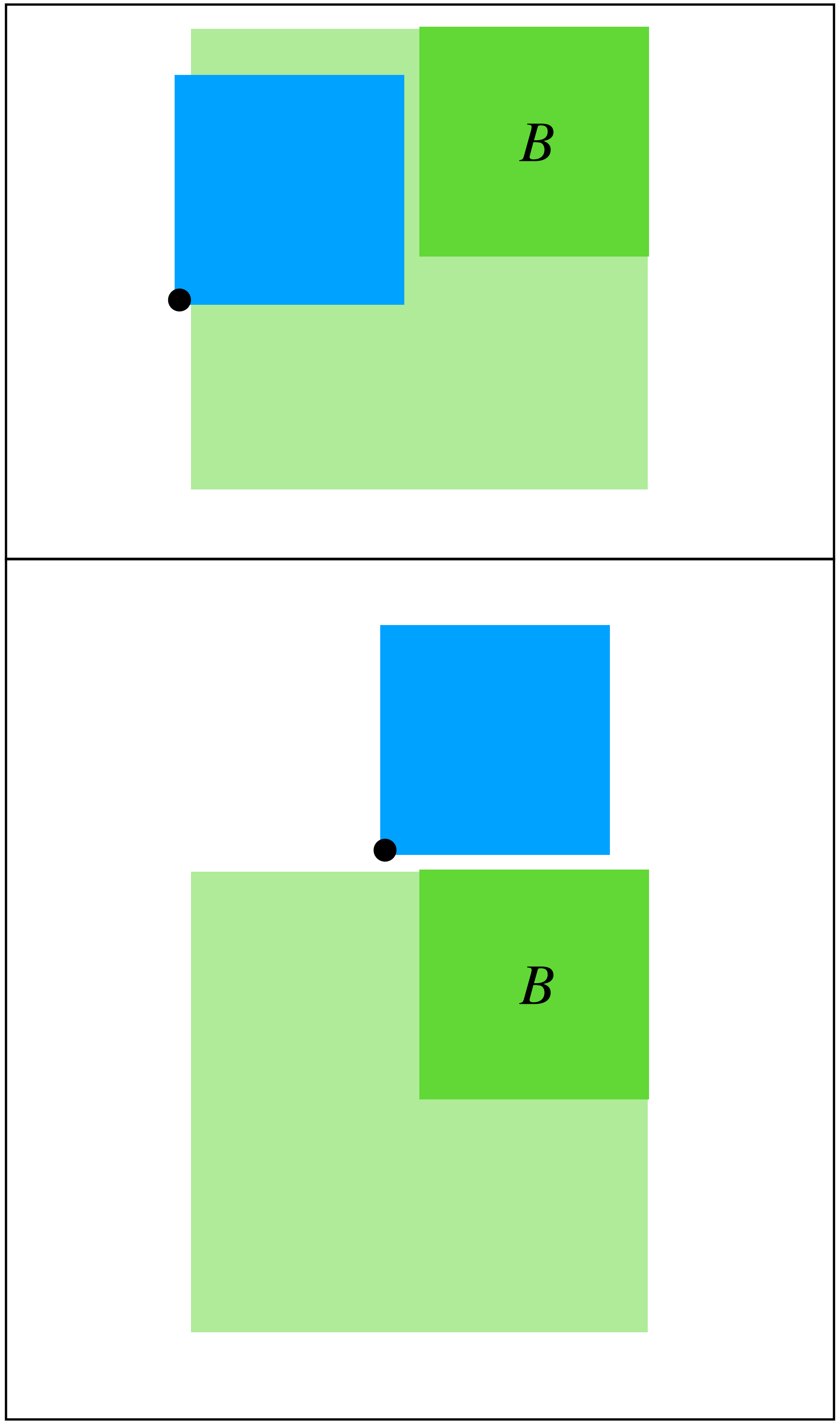
- In geometry, the Minkowski sum (also known as dilation) of two sets of position vectors ***A*** and ***B*** in Euclidean space is formed by adding each vector in ***A*** to each vector in ***B***, i.e., the set:

$$A \oplus B = \{\mathbf{a} + \mathbf{b} \,|\, \mathbf{a} \in A, \mathbf{b} \in B\}$$



Minkowski Sum

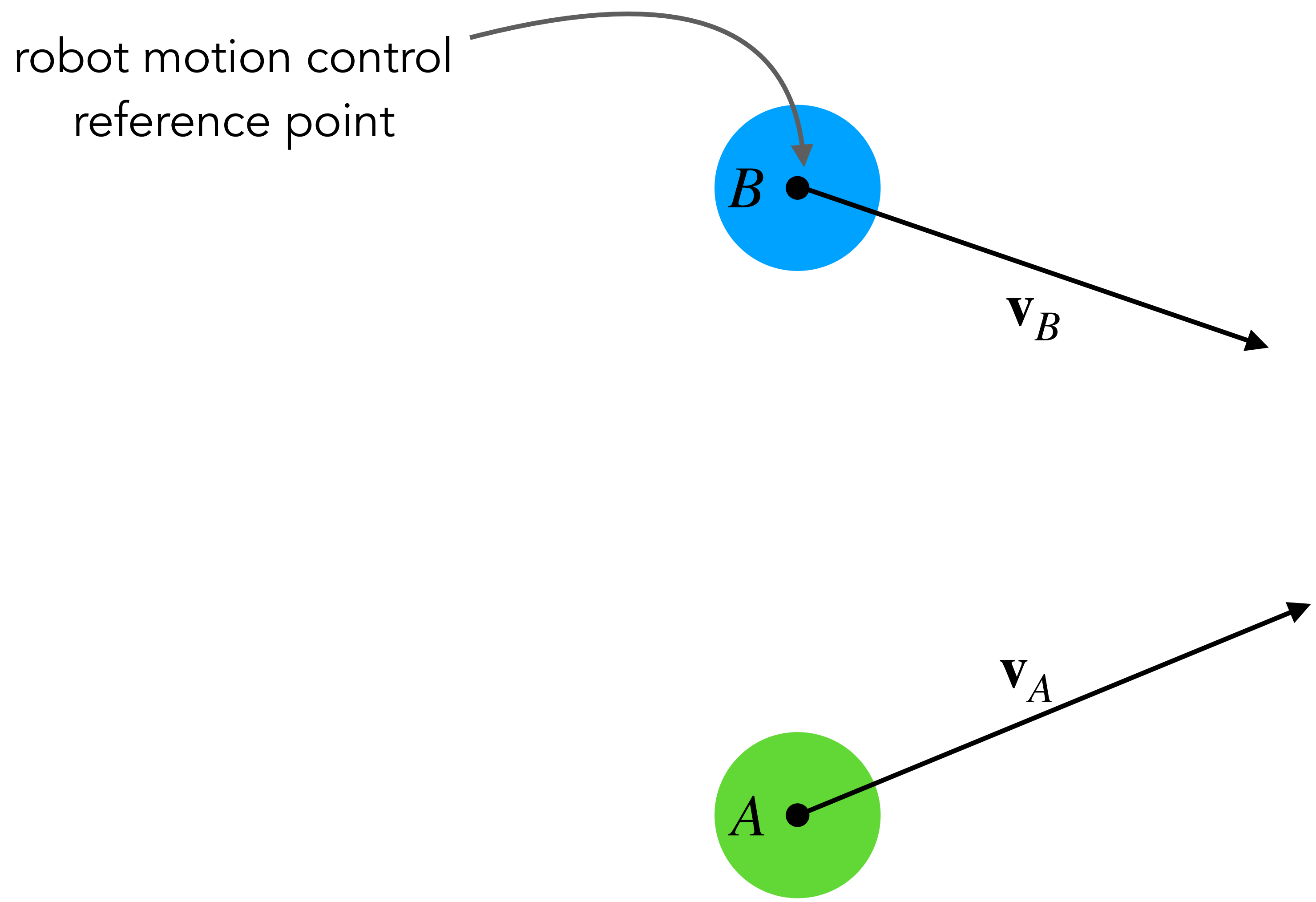




As long as reference point stays outside dilated area, there will be no collisions.

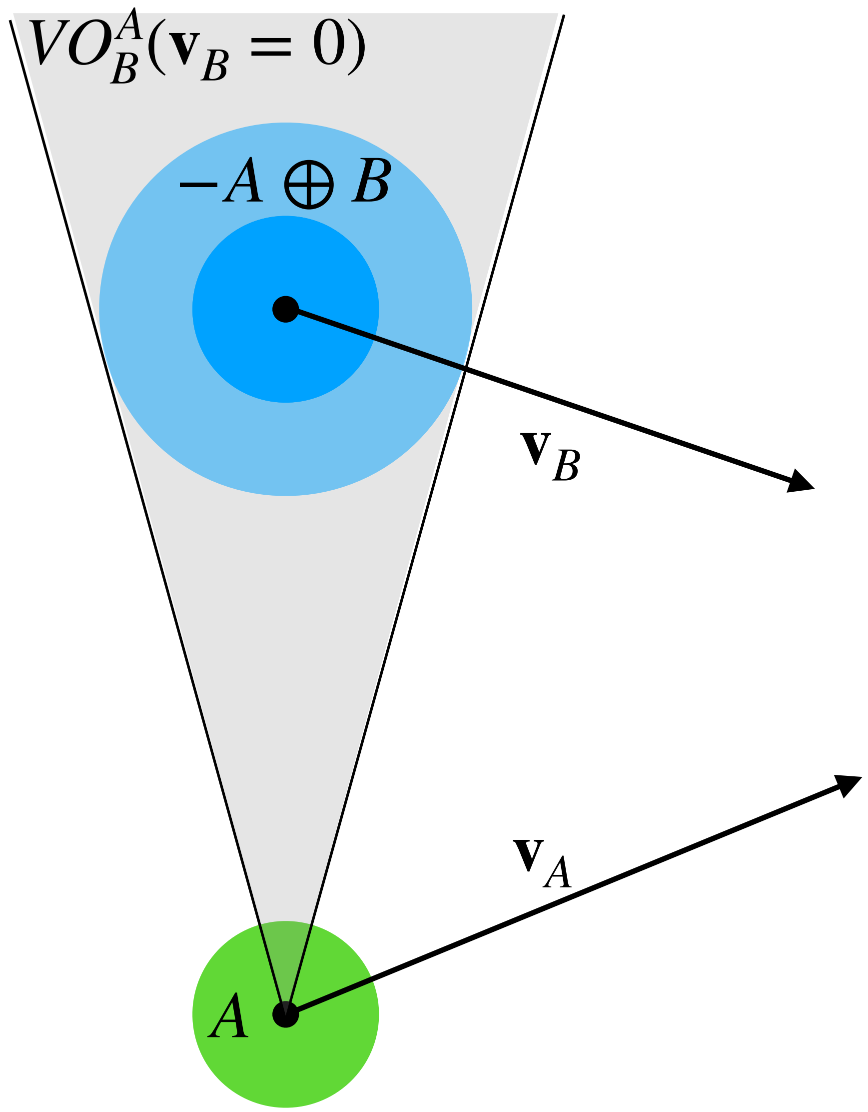
Velocity Obstacle Method

[Fiorini, Shiller; 1998]



Two robots, A and B, translating in space. Will they collide?

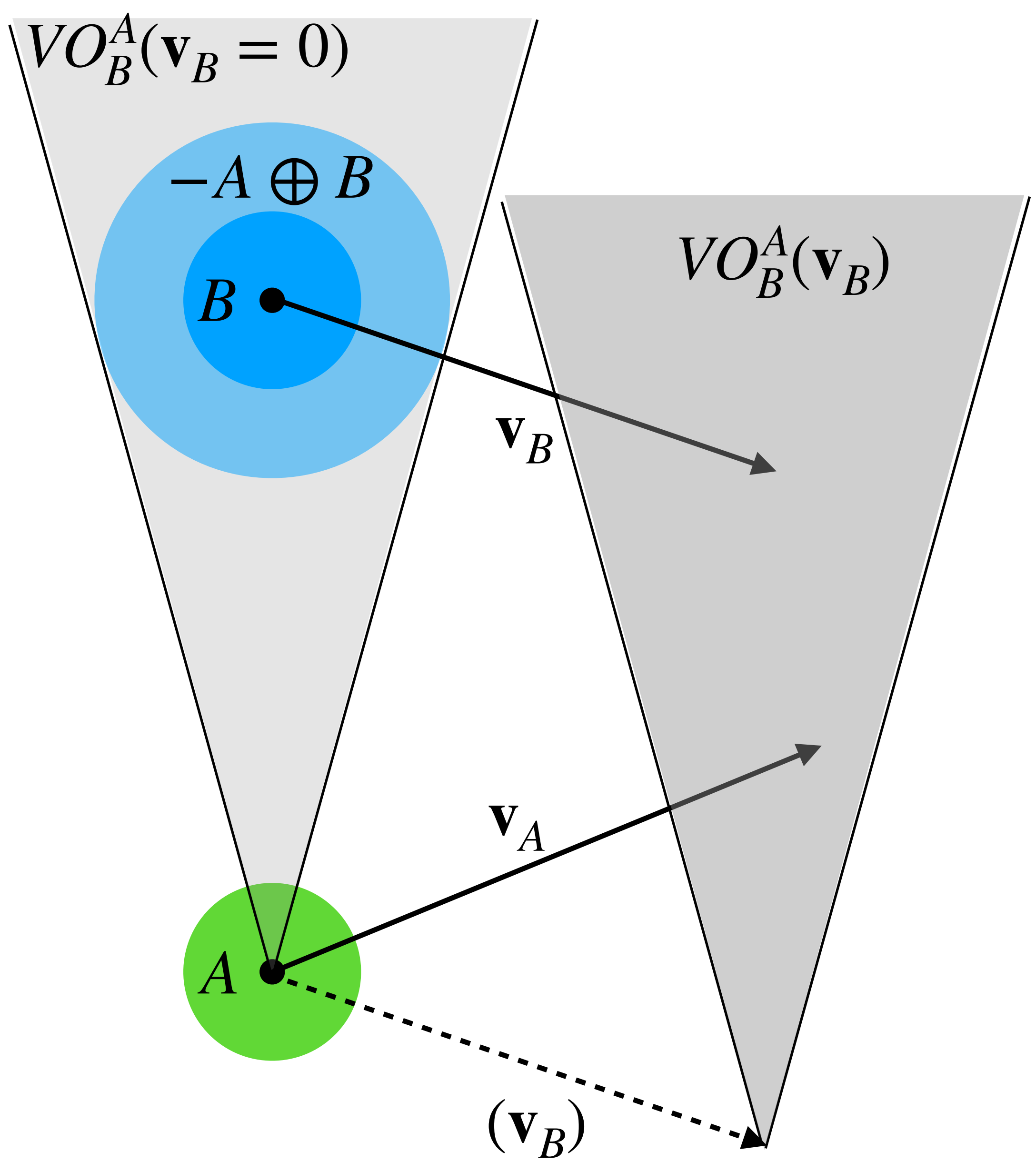
Velocity Obstacle Method



Two robots, A and B, translating in space. Will they collide?

Step 1: inflate robot B by area of robot A.

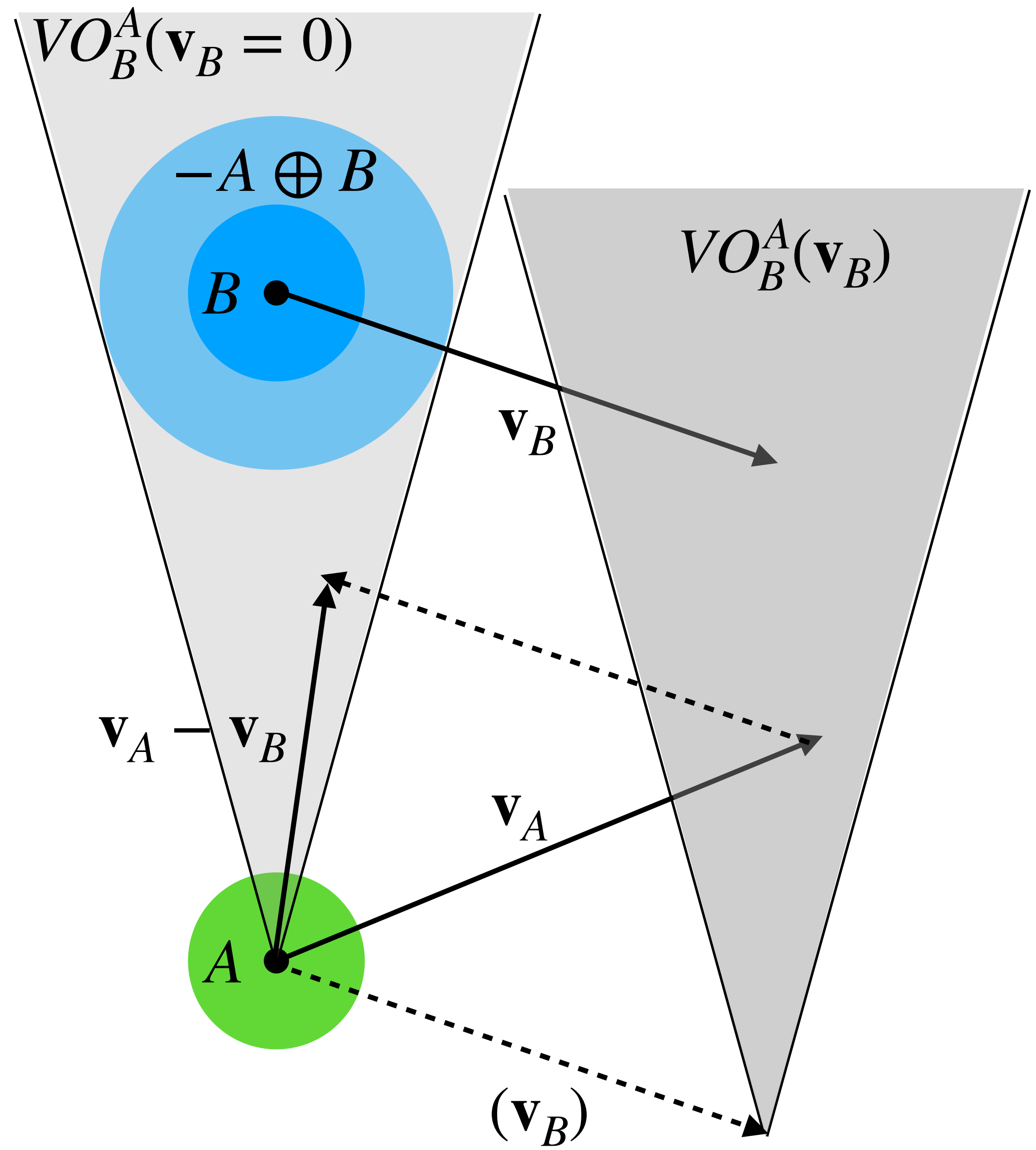
Velocity Obstacle Method



Step 2: determine whether $\boldsymbol{v}_A$ lies in the velocity obstacle of B to A

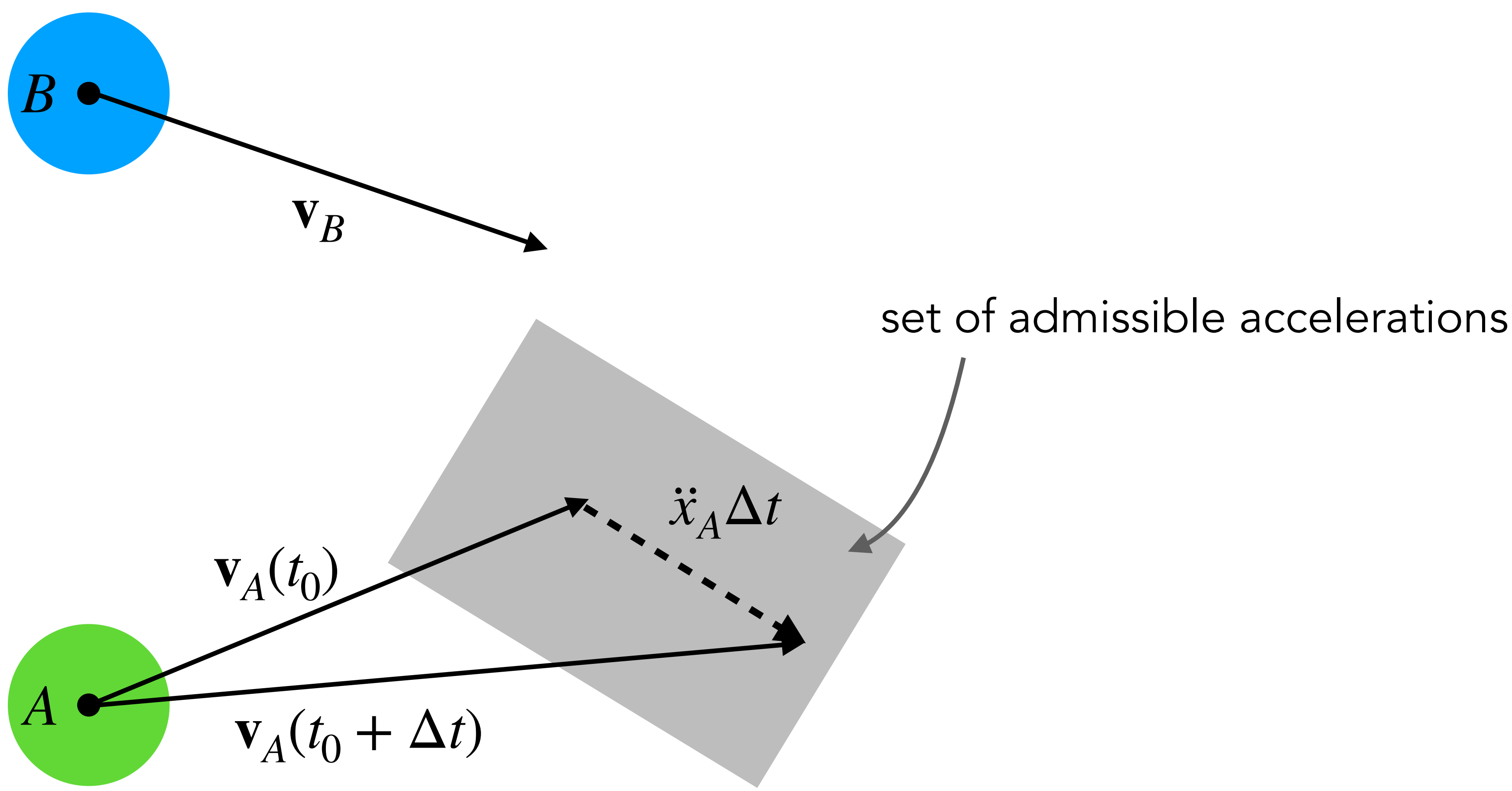
If $\boldsymbol{v}_A$ is outside the VO, then the robots will never collide.

Velocity Obstacle Method



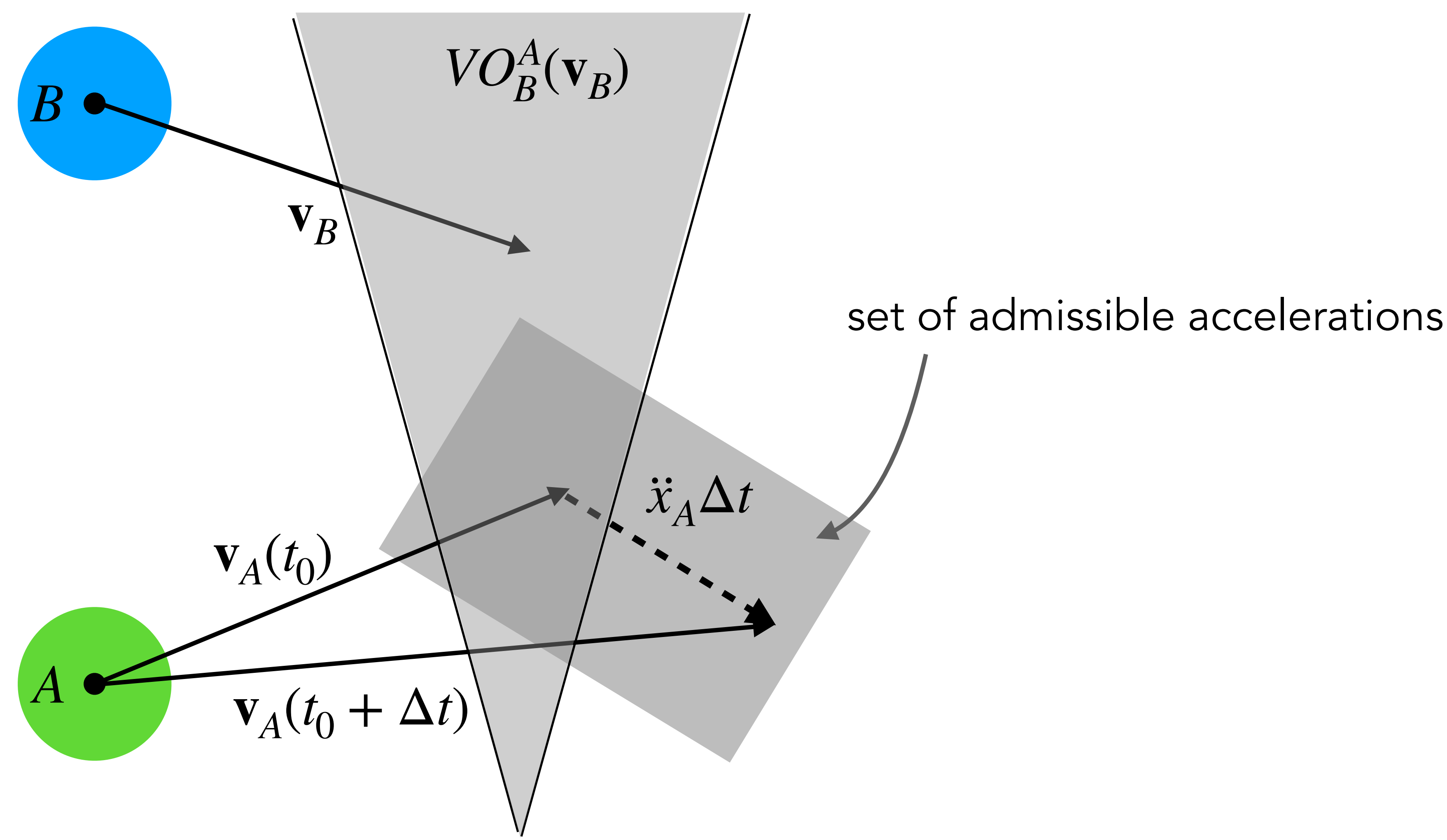
<u>Equivalence</u>: $\boldsymbol{v}_A$ lies in the velocity obstacle of B to A $\longrightarrow$ the relative velocity $\boldsymbol{v}_A$ - $\boldsymbol{v}_B$ lies in the velocity obstacle of B to A, assuming B does not move.

Velocity Obstacle Method



Compute set of admissible accelerations for robot A.

Velocity Obstacle Method



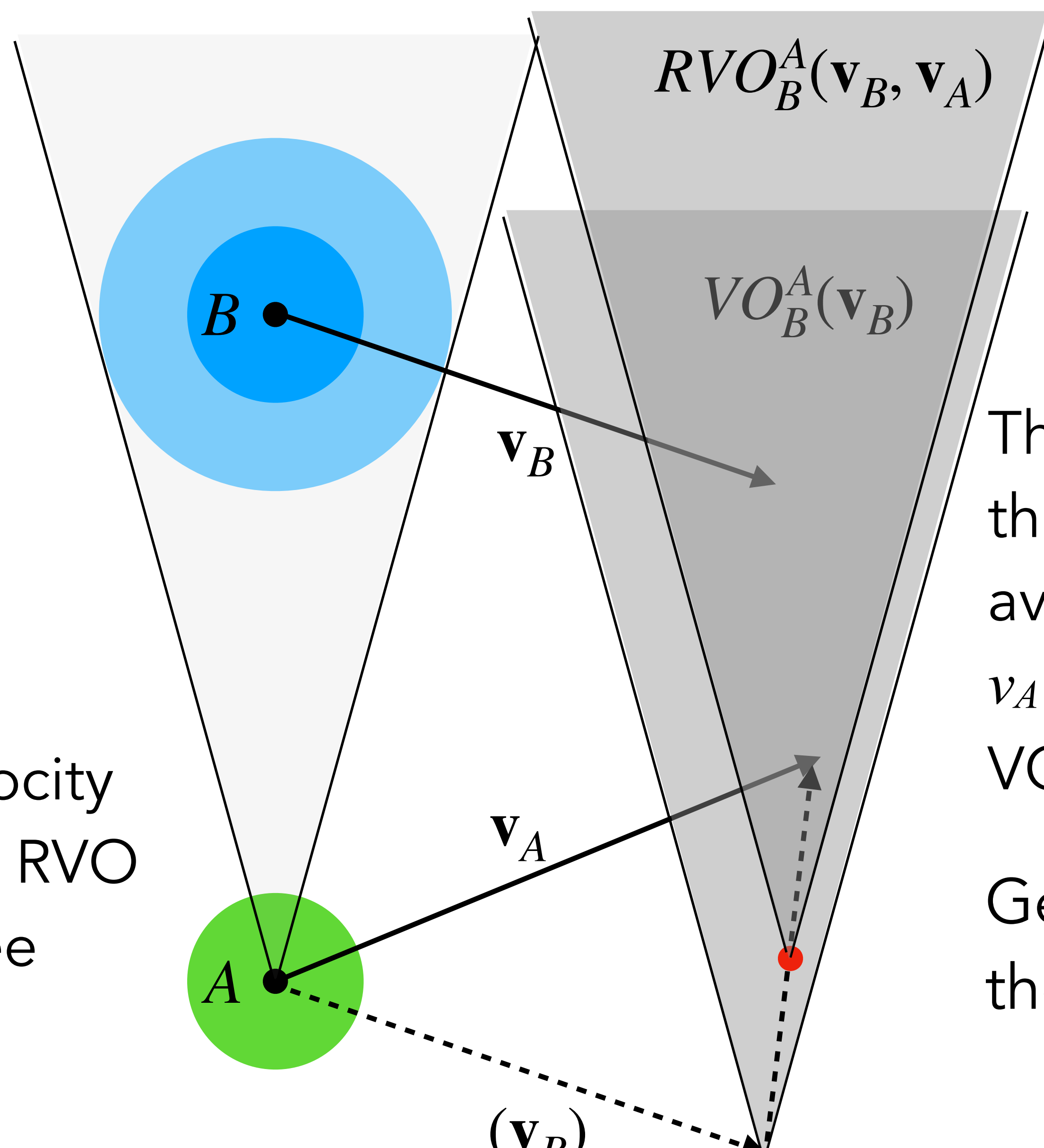
Check that new velocity is outside VO.

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 $\boldsymbol{v}_A$ and $\boldsymbol{v}_B$ currently lie in each others VOs. **Both** robots select new $\boldsymbol{v}'_A$ and $\boldsymbol{v}'_B$ such that new velocities lie outside respective VOs. In new situation, the old velocities $\boldsymbol{v}_A$ and $\boldsymbol{v}_B$ lie outside VOs. If $\boldsymbol{v}_A$ and $\boldsymbol{v}_B$ 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]



The RVO of B to A contains all the velocities of A that are the average of the current velocity v_A and a velocity inside the VO of B to A.

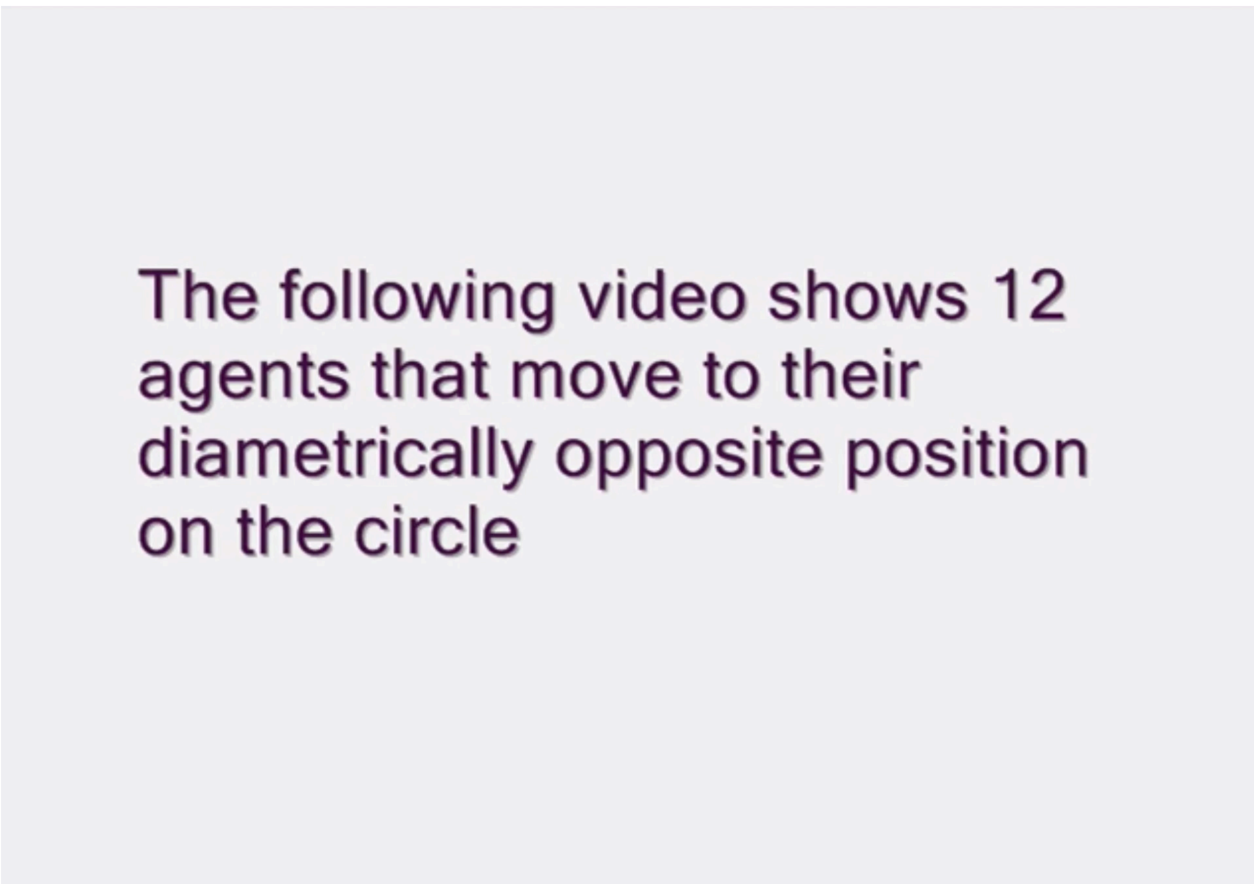
Choosing the closest velocity outside the other agent's RVO guarantees oscillation-free navigation.

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.

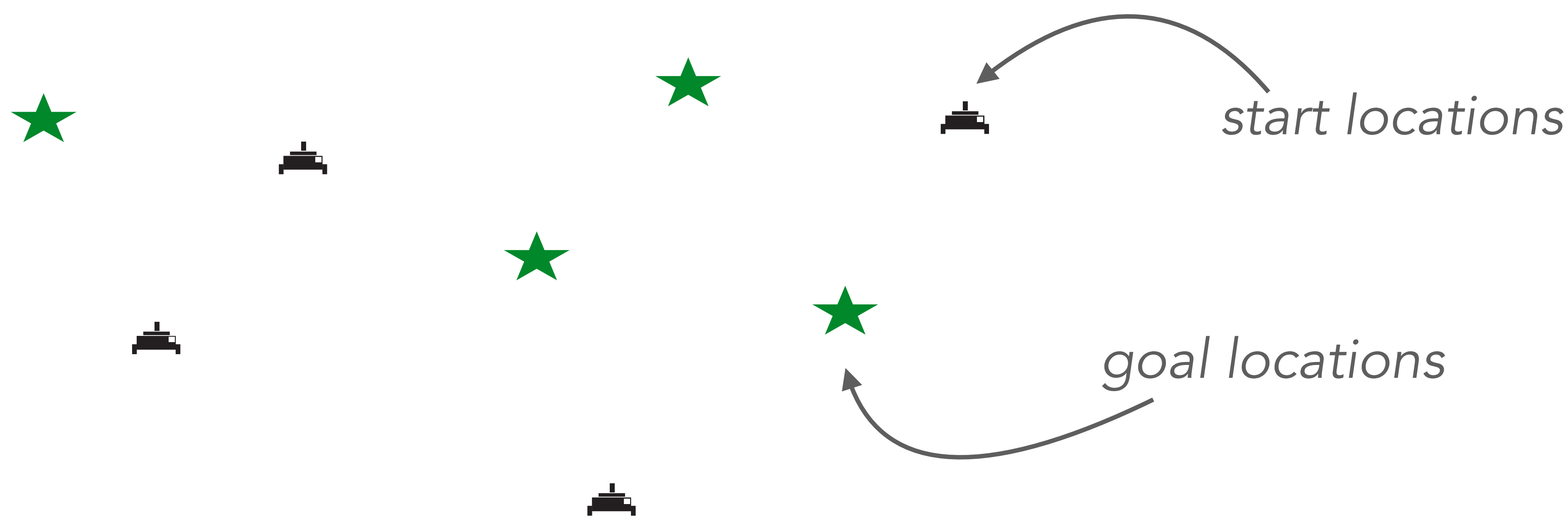
Reciprocal Velocity Obstacle Method



[D. Manocha et al.]

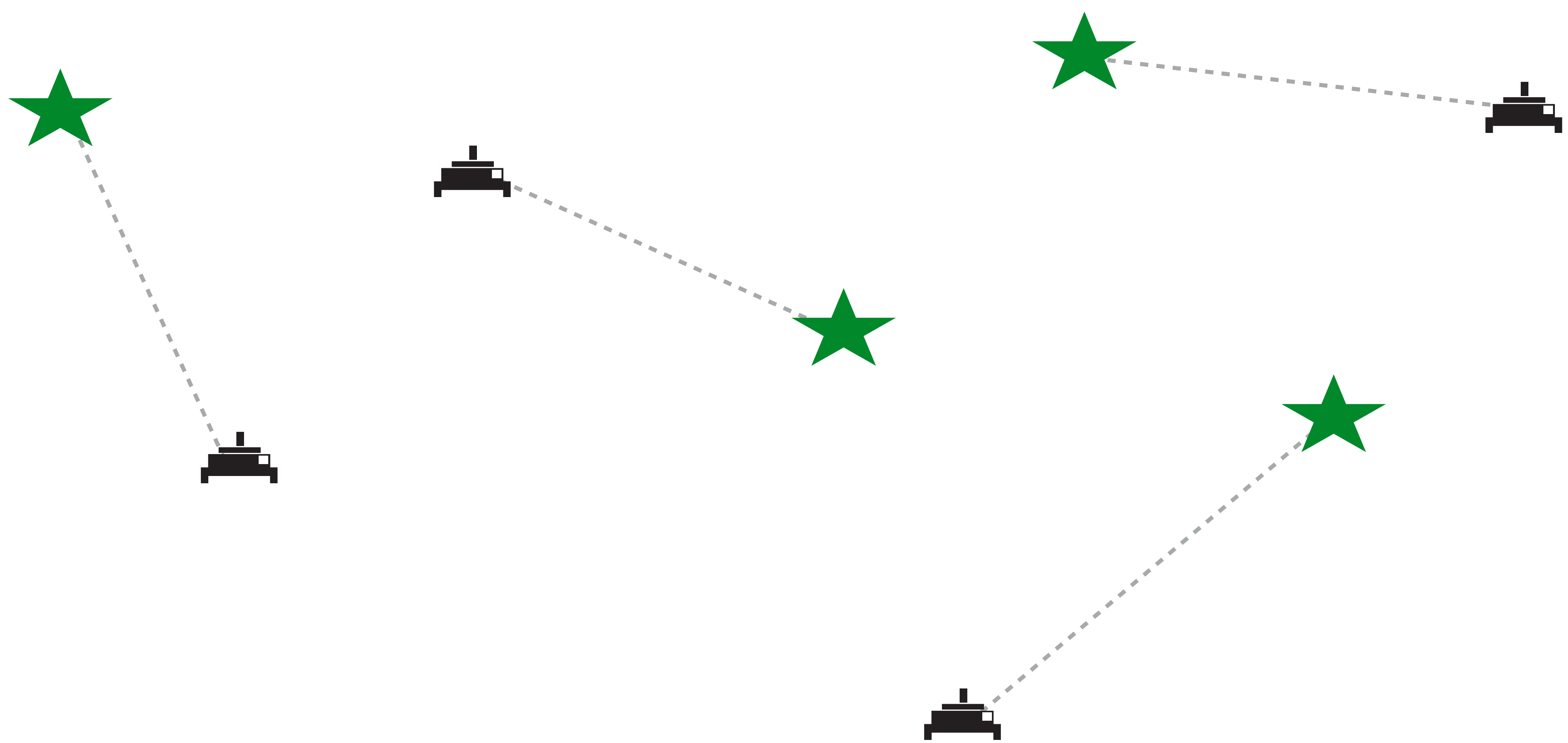
Concurrent Assignment and Planning of Trajectories

- 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



Concurrent Assignment and Planning of Trajectories

Given start and goal locations, find assignments **AND** trajectories that are optimal and collision-free



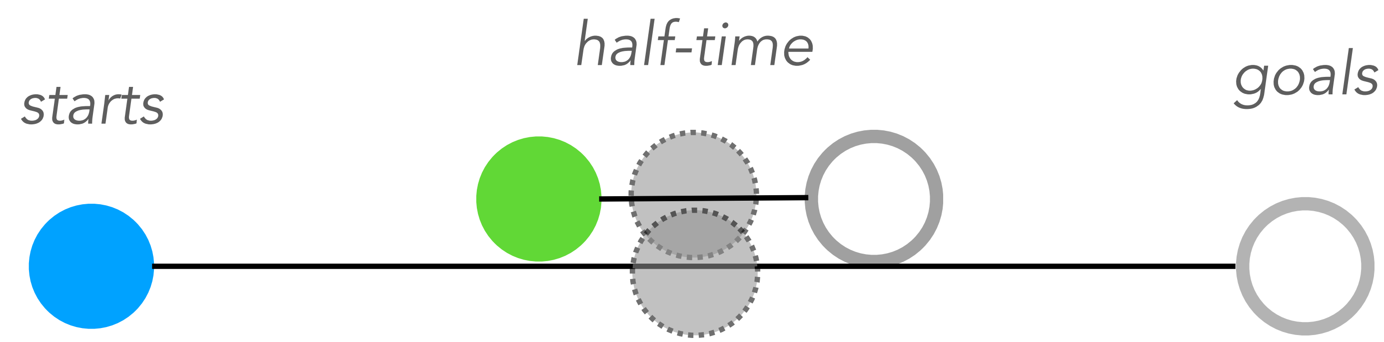
Concurrent Assignment and Planning of Trajectories

Given start and goal locations, find assignments **AND** trajectories that are optimal and collision-free

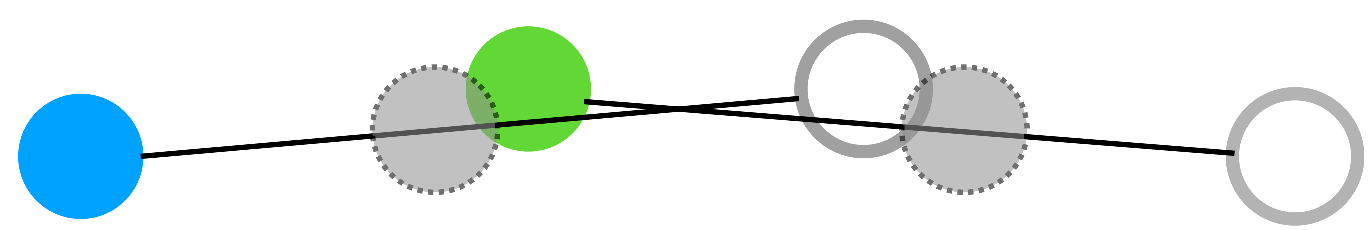
Concurrent Assignment and Planning of Trajectories

What is the **optimization objective**?

Sum of distances:



Sum of distances squared:



[Turpin et al.; IJRR 2013]

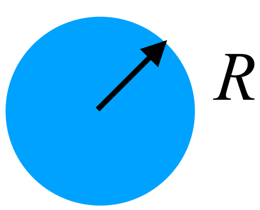
Concurrent Assignment and Planning of Trajectories

Objective:

$$\underset{\phi,\gamma(t)}{\text{minimize}} \quad \sum_{i=1}^{N} \int_{t_0}^{t_f} \dot{\mathbf{x}}_i(t)^{\mathrm{T}} \dot{\mathbf{x}}_i(t) dt$$

Key result:

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

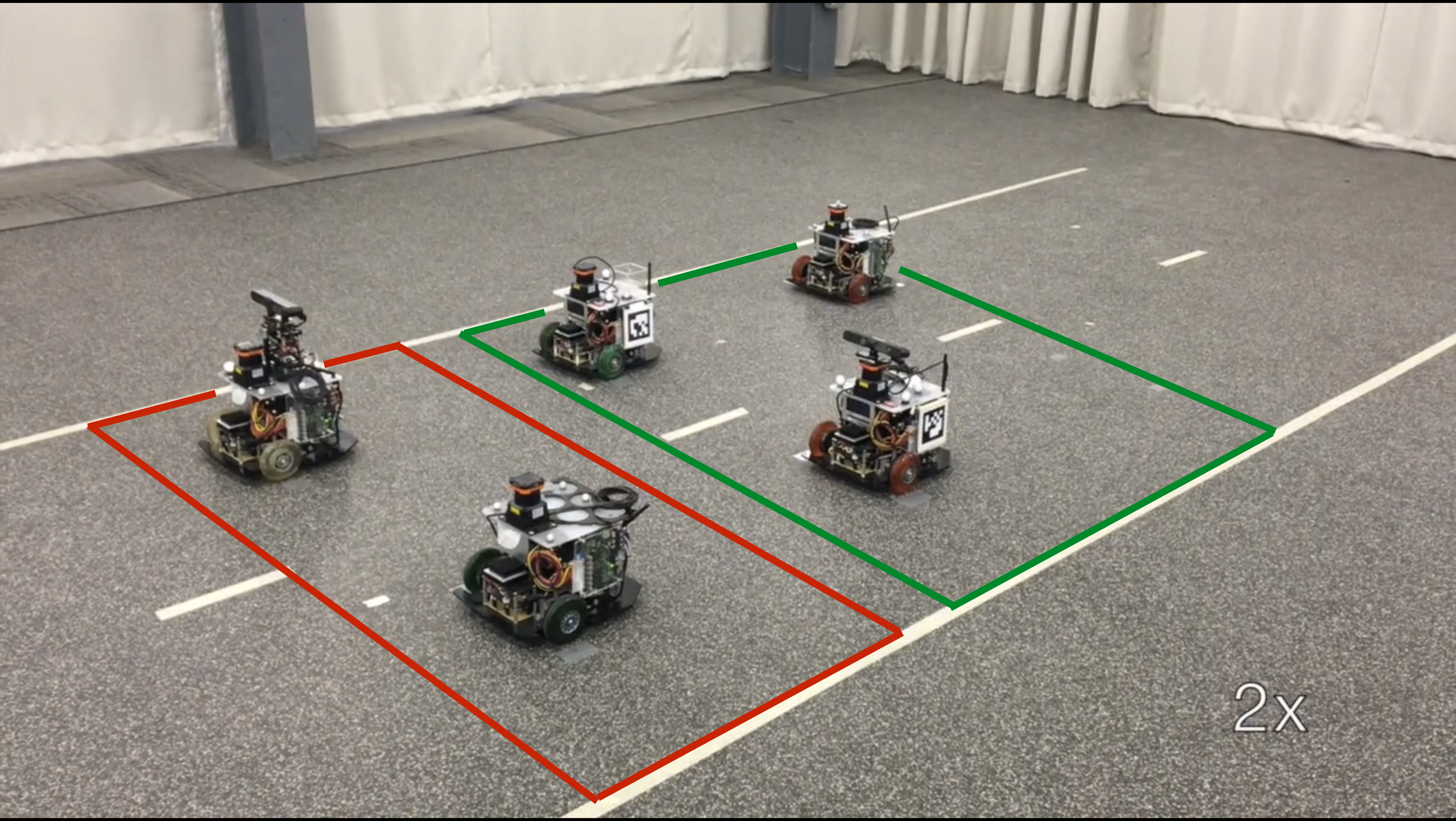


Solve assignment:

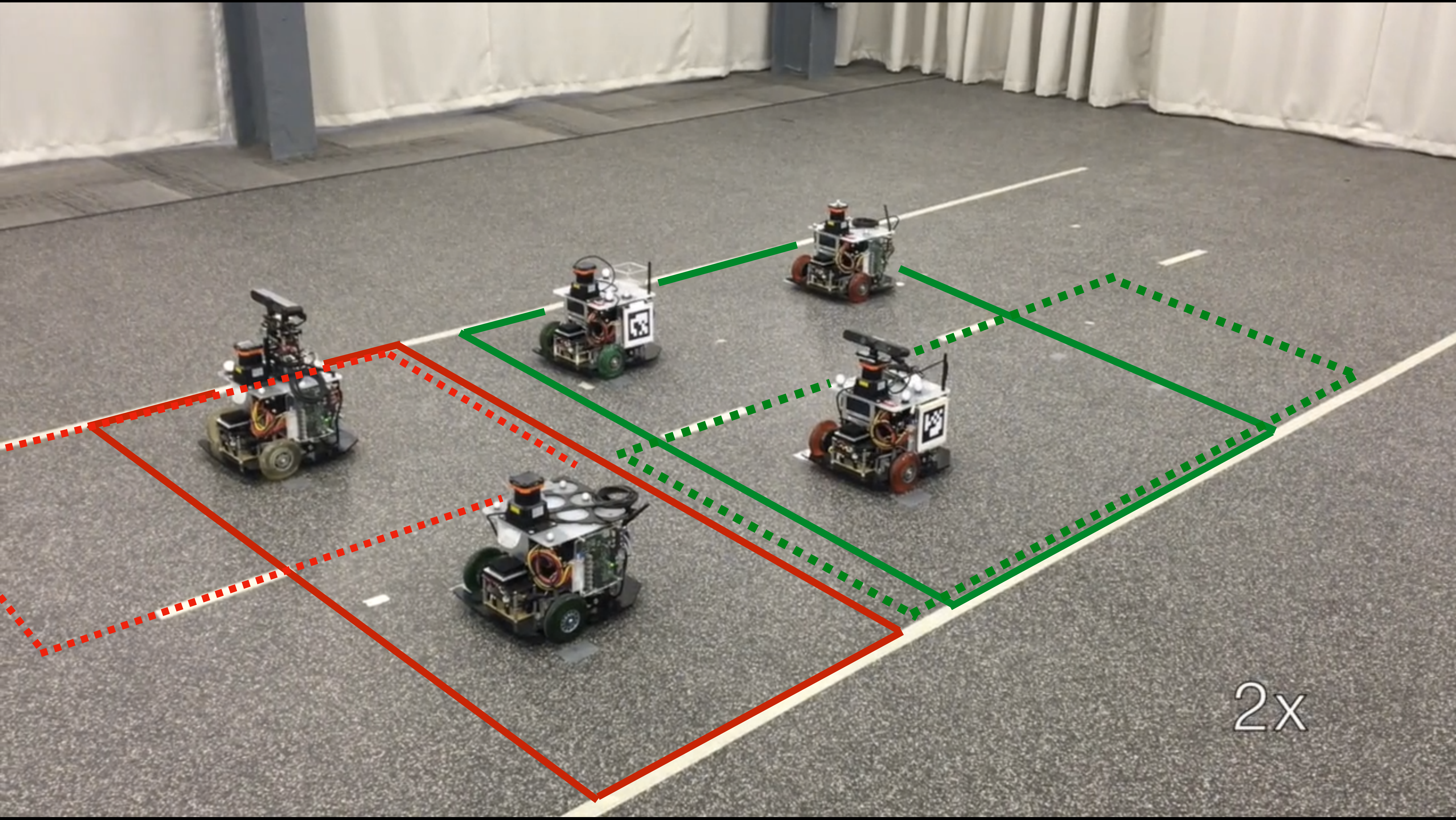
$$\phi^\star = \underset{\phi}{\text{argmin}} \quad \sum_{i=1}^{N} \sum_{j=1}^{N} \phi_{i,j} D_{i,j}$$

cost: distance squared

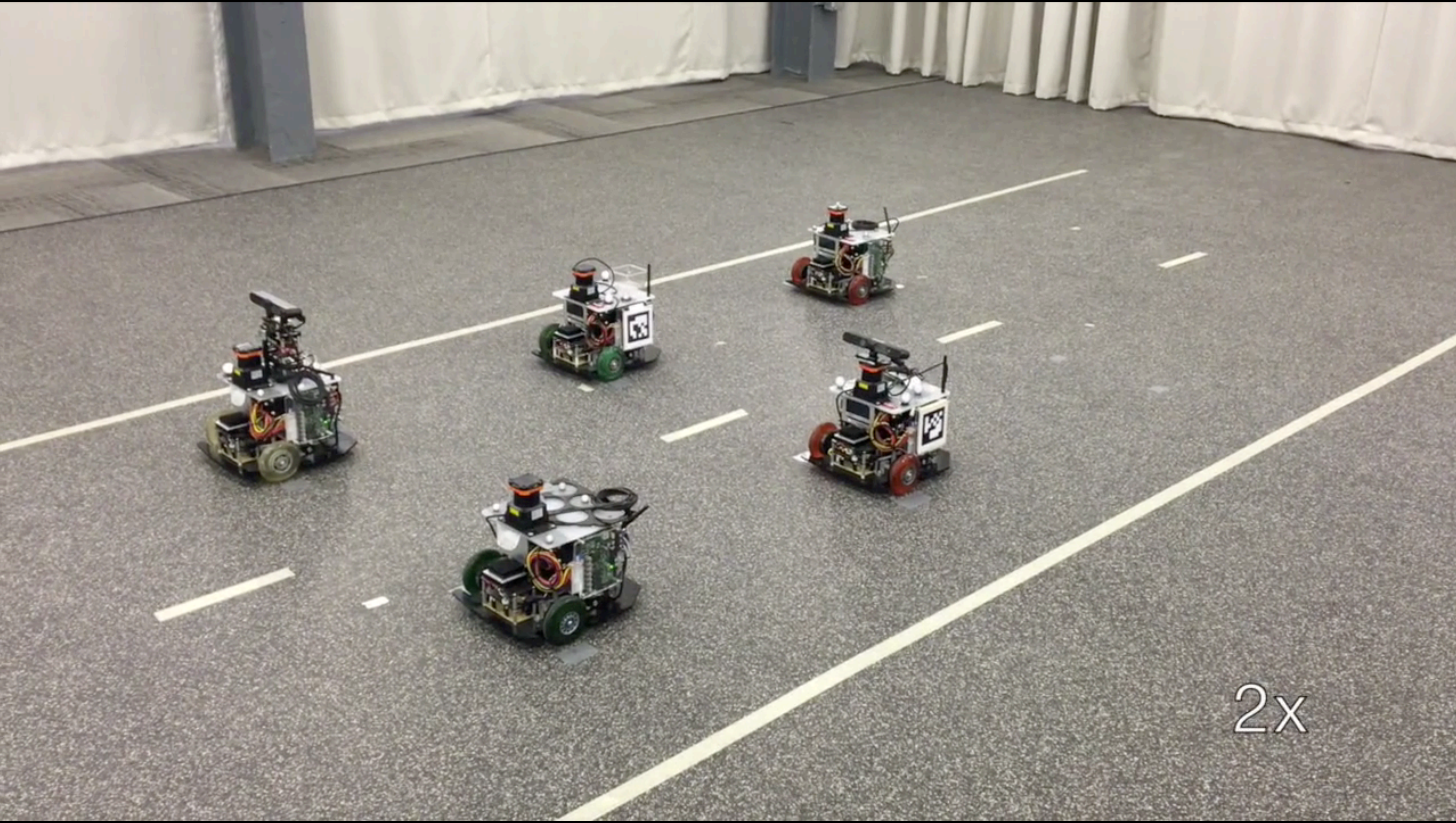
[Turpin et al.; IJRR 2013]



[Whitzer, Kennedy, Prorok, Kumar; 2016]



[Whitzer, Kennedy, Prorok, Kumar; 2016]



[Whitzer, Kennedy, Prorok, Kumar; 2016]

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
- M. Čáp, P. Novák, A. Kleiner, M. Selecký; "Prioritized Planning Algorithms for Trajectory; "Coordination of Multiple Mobile Robots"; 2015