16-350
Planning Techniques for Robotics

Multi-Robot Planning

Maxim Likhachev

Robotics Institute

Carnegie Mellon University

Different Categorizations of Multi-Robot Planning

- Centralized vs. Decentralized
  - **Centralized**: one central control of (planning for) all the robots
  - **Decentralized**: each robot decides/plans what to do on its own

Different Categorizations of Multi-Robot Planning

- Centralized vs. Decentralized
  - **Centralized**: one central control of (planning for) all the robots
  - **Decentralized**: each robot decides/plans what to do on its own

Why do we need decentralized planning approaches?

Different Categorizations of Multi-Robot Planning

- Centralized vs. Decentralized
  - **Centralized**: one central control of (planning for) all the robots
  - **Decentralized**: each robot decides/plans what to do on its own

Why do we need decentralized planning approaches?

Robust to limits on or loss of communication
Robust to loosing some robots in the team
Computationally more scalable

Different Categorizations of Multi-Robot Planning

- Centralized vs. Decentralized
  - **Centralized**: one central control of (planning for) all the robots
  - **Decentralized**: each robot decides/plans what to do on its own

Why do we need decentralized planning approaches?

Robust to limits on or loss of communication
Robust to loosing some robots in the team
Computationally more scalable

Challenges with decentralized planning?

Different Categorizations of Multi-Robot Planning

- Centralized vs. Decentralized
  - **Centralized**: one central control of (planning for) all the robots
  - **Decentralized**: each robot decides/plans what to do on its own

Why do we need decentralized planning approaches?

Robust to limits on or loss of communication
Robust to loosing some robots in the team
Computationally more scalable

Challenges with decentralized planning?

How to guarantee that the overall team accomplishes its goal?

Different Categorizations of Multi-Robot Planning

- Multi-robot Path Planning vs. Multi-robot Cooperative Task Planning

  – **Multi-robot Path Planning**: how to plan paths for N robots so that they don't collide with each other during execution

  – **Multi-robot Cooperative Task Planning**: how to compute plans for N robots so that they achieve the overall goal that may require cooperation

Different Categorizations of Multi-Robot Planning

- Small teams vs. large teams (swarms) of robots
  - **Planning for small teams**: Compute plans for N (potentially heterogeneous) robots, where N is typically 2-10
  - **Planning for (control of) swarms of robots**: how to control a swarm of N (usually homogeneous) robots, where N is typically 10-1000

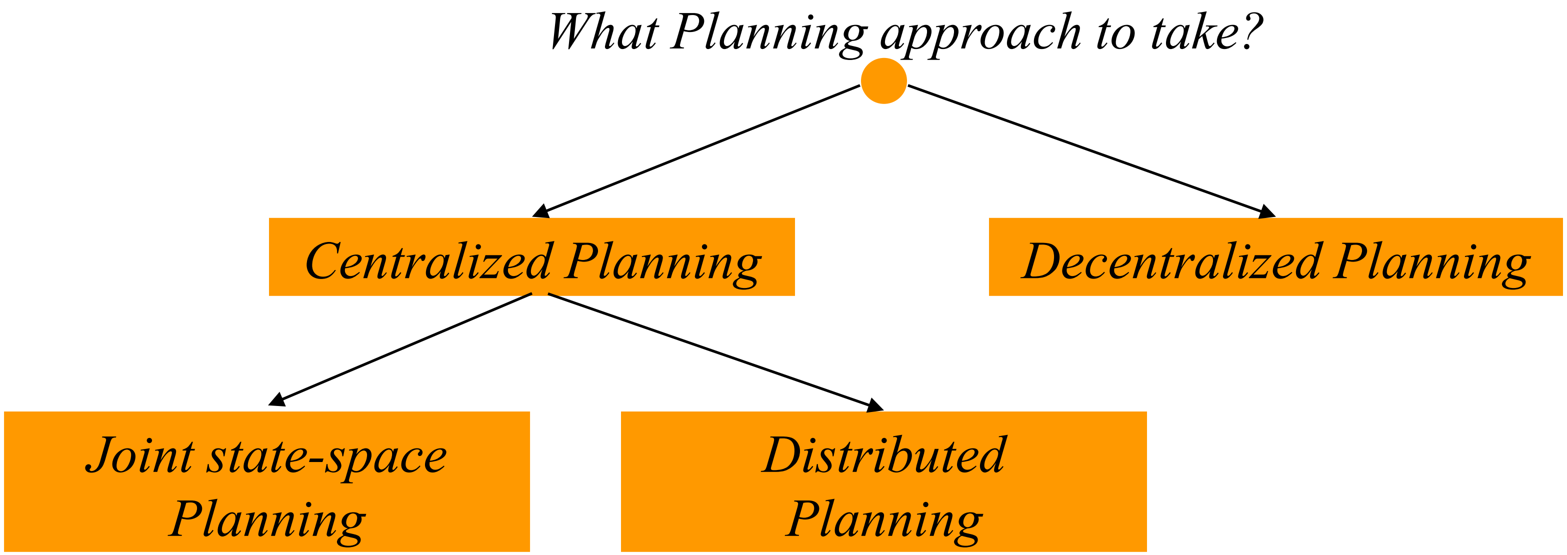
Different Categorizations of Multi-Robot Planning

- Small teams vs. large teams (swarms) of robots
  - **Planning for small teams**: Compute plans for N (potentially heterogeneous) robots, where N is typically 2-10
  - **Planning for (control of) swarms of robots**: how to control a swarm of N (usually homogeneous) robots, where N is typically 10-1000

Control of swarms is typically decentralized

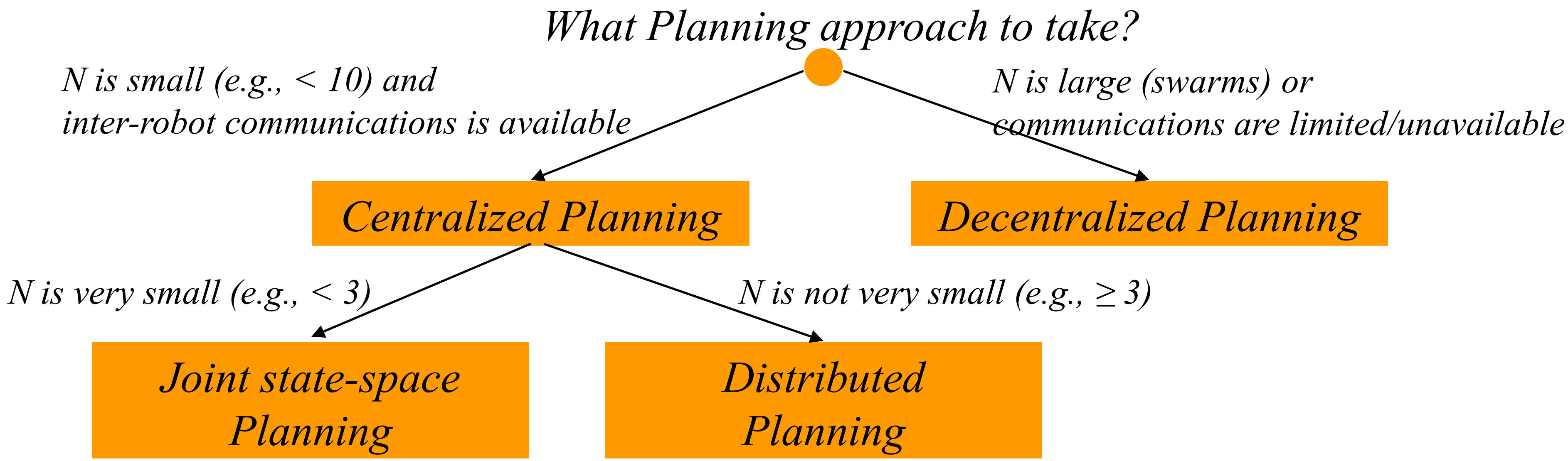
Different Categorizations of Multi-Robot Planning

- Joint state-space vs. distributed planning (within centralized)
  - **Joint state-space planning**: Planning for *N* robots in a state-space that represents joint configurations of robots
  - **Distributed planning**: Planning is split into *N* individual planners that share their results (and potentially re-plan) to obtain a final plan for all *N* robots



Different Categorizations of Multi-Robot Planning

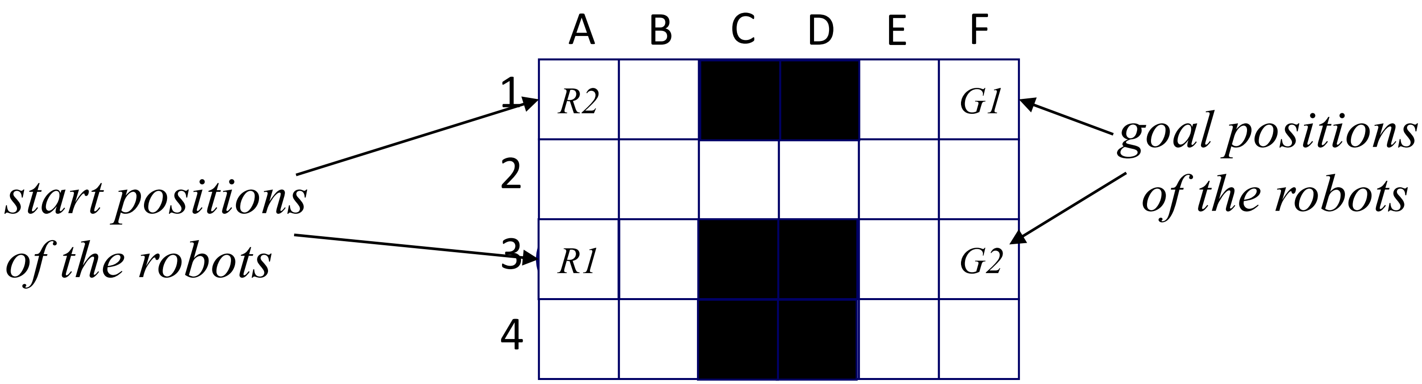
- Joint state-space vs. distributed planning (within centralized)
  - **Joint state-space planning**: Planning for *N* robots in a state-space that represents joint configurations of robots
  - **Distributed planning**: Planning is split into *N* individual planners that share their results (and potentially re-plan) to obtain a final plan for all *N* robots



Multi-Robot Path Planning

- Path planning for *N* robots to get to their goals w/o collisions

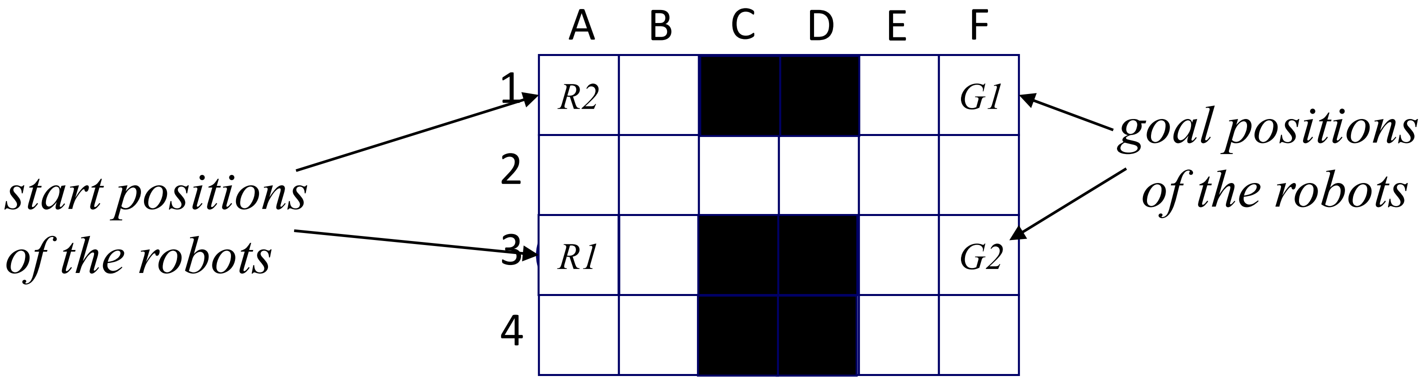
simple example for two omnidirectional point-size robots



Multi-Robot Path Planning

- Path planning for *N* robots to get to their goals w/o collisions

simple example for two omnidirectional point-size robots

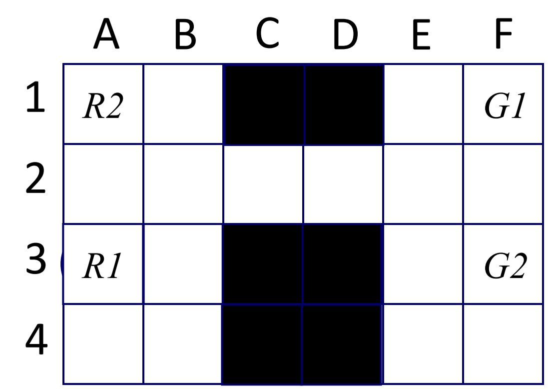


Any examples of this in industry?

Multi-Robot Path Planning

- Path planning for *N* robots to get to their goals w/o collisions

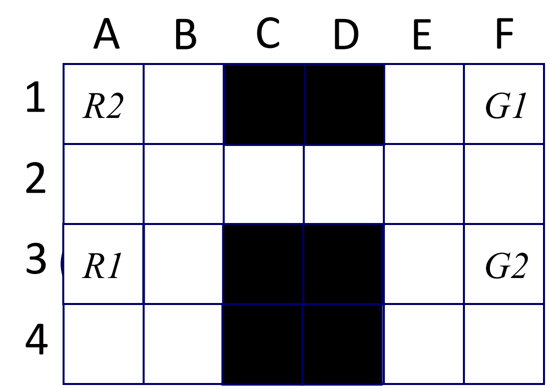
Joint state-space planning



Multi-Robot Path Planning

- Path planning for *N* robots to get to their goals w/o collisions

Joint state-space planning

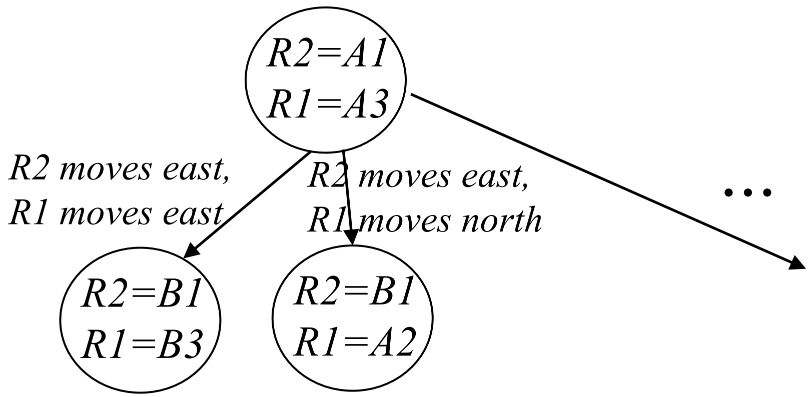


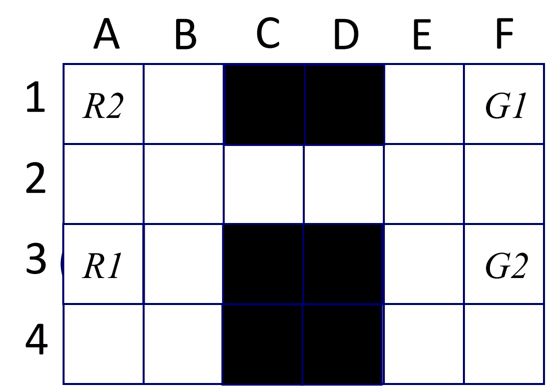
The simplest approach: construct and search a graph, where each state encodes positions of all the robots and each action encodes all possible movements

Multi-Robot Path Planning

- Path planning for *N* robots to get to their goals w/o collisions

Joint state-space planning



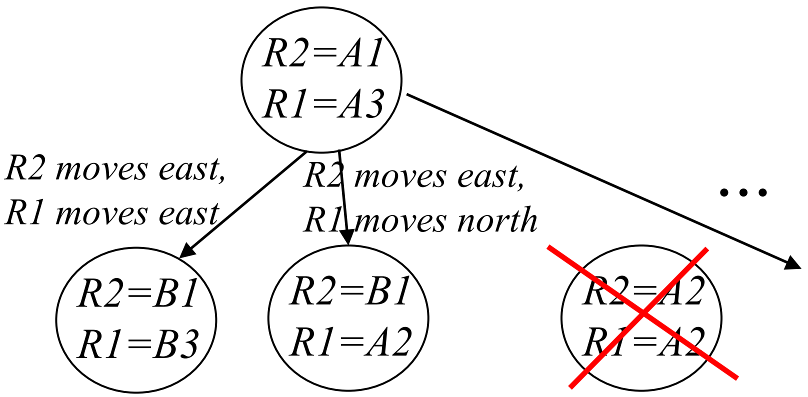


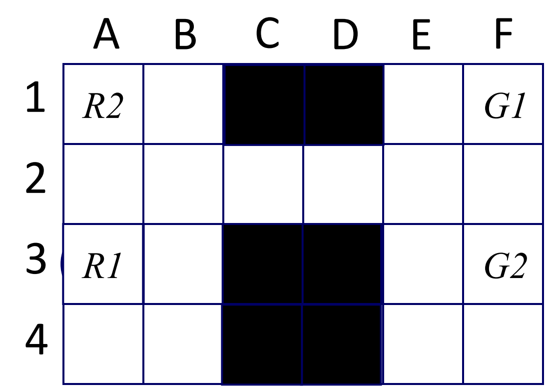
The simplest approach: construct and search a graph, where each state encodes positions of all the robots and each action encodes all possible movements

Multi-Robot Path Planning

- Path planning for *N* robots to get to their goals w/o collisions

Joint state-space planning



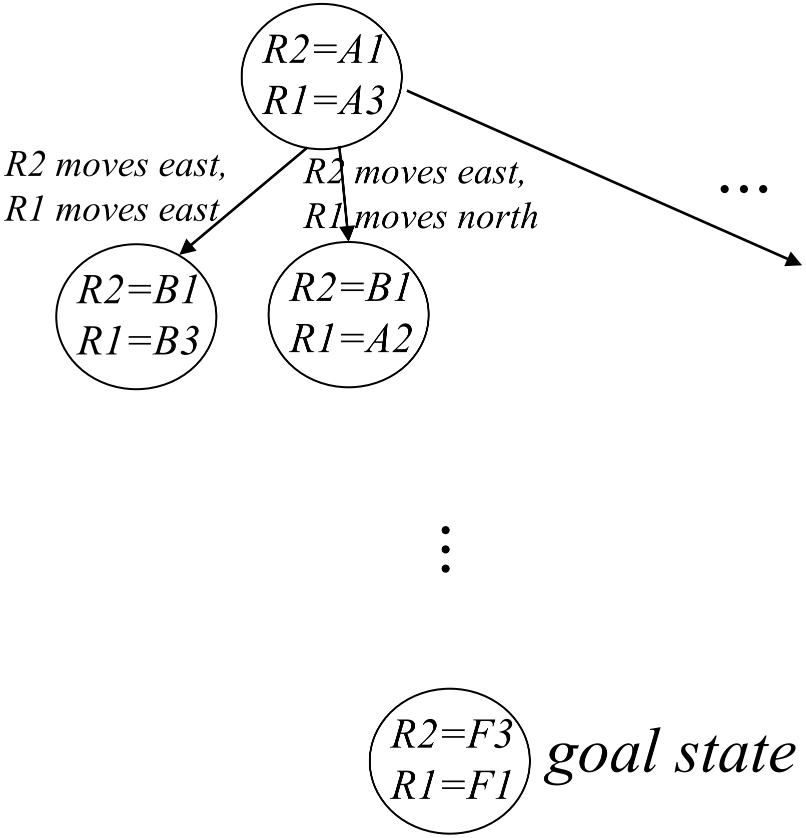


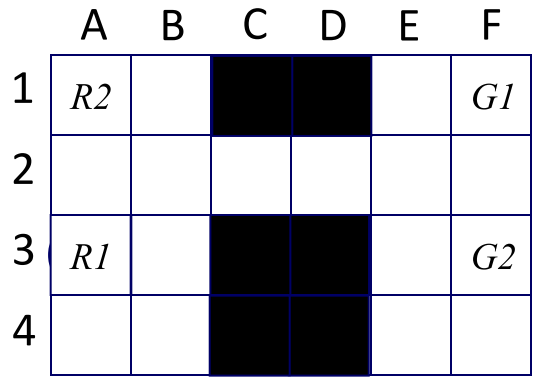
The simplest approach: construct and search a graph, where each state encodes positions of all the robots and each action encodes all possible movements

Multi-Robot Path Planning

- Path planning for *N* robots to get to their goals w/o collisions

Joint state-space planning



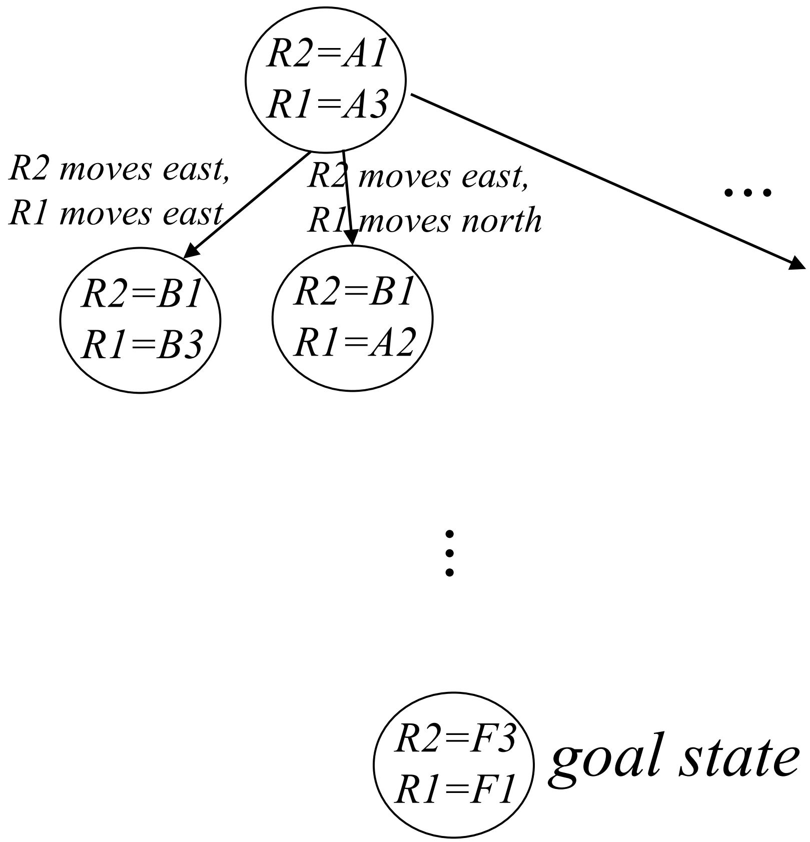


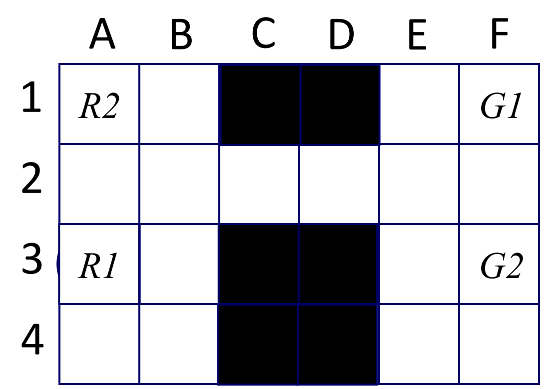
Multi-Robot Path Planning

- Path planning for *N* robots to get to their goals w/o collisions

Assuming 4-connected grid,
what is the maximum branching factor
(how many actions/successors)?

Joint state-space planning



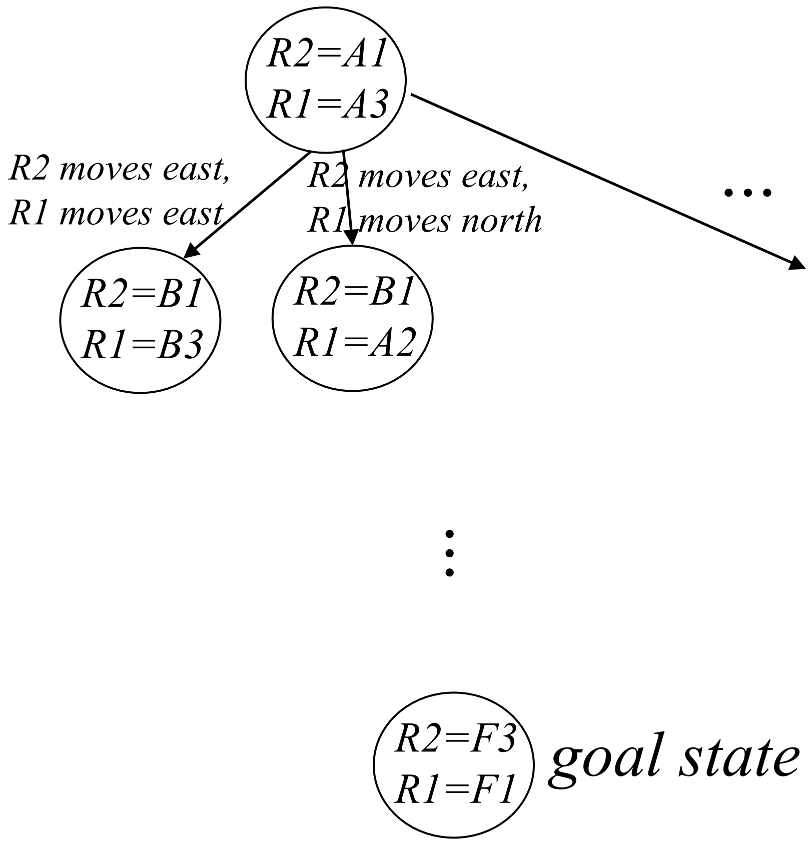


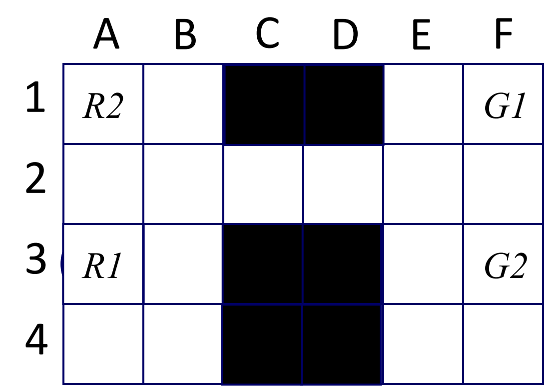
Multi-Robot Path Planning

- Path planning for *N* robots to get to their goals w/o collisions

What is the size of the graph?

Joint state-space planning



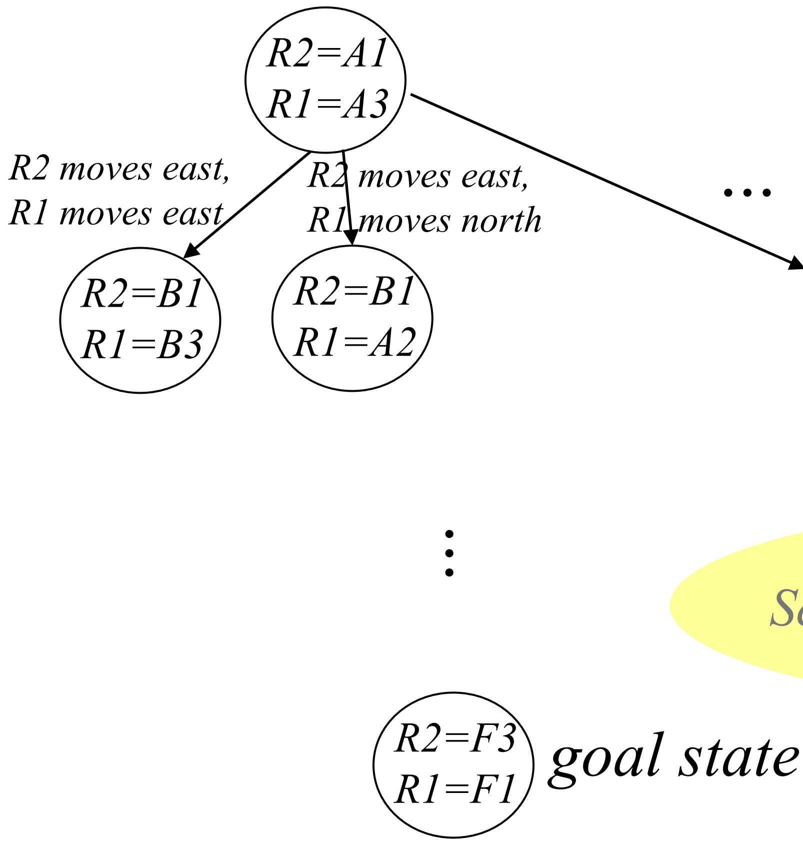


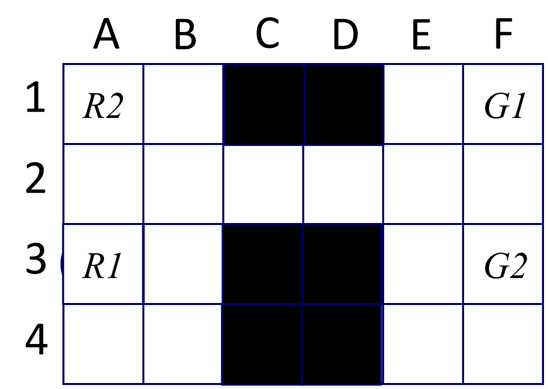
Multi-Robot Path Planning

- Path planning for *N* robots to get to their goals w/o collisions

What is the size of the graph?

Joint state-space planning



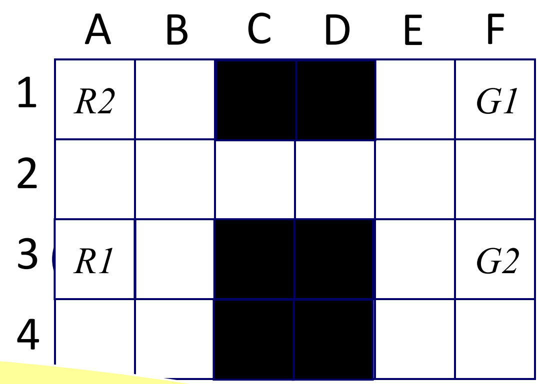


Scalability w.r.t. N is clearly an issue!

Multi-Robot Path Planning

- Path planning for *N* robots to get to their goals w/o collisions

Distributed planning



One popular approach: Prioritized Planning

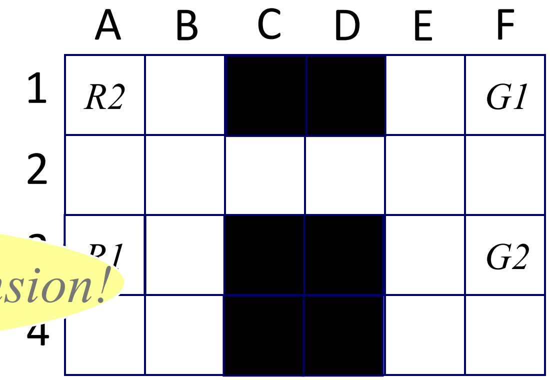
For i = 1:N

Compute path for robot R_i that avoids collisions with paths for robots $R_1..R_{i-1}$

Multi-Robot Path Planning

- Path planning for *N* robots to get to their goals w/o collisions

Distributed planning



Each planning needs to include time as a dimension!

One popular approach: Prioritized Planning

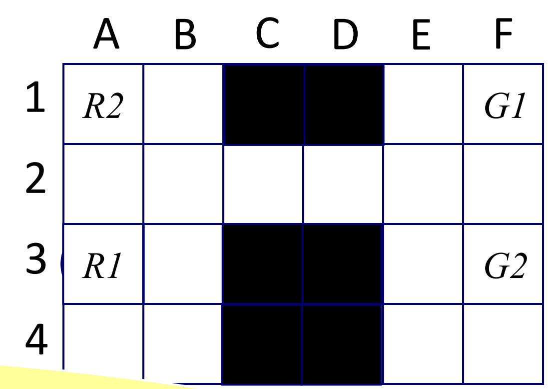
For i = 1:N

Compute path for robot R_i that avoids collisions with paths for robots $R_1..R_{i-1}$

Multi-Robot Path Planning

- Path planning for *N* robots to get to their goals w/o collisions

Distributed planning



What will be the plan returned by Prioritized Planning?

One popular approach: Prioritized Planning

For i = 1:N

Compute path for robot R_i that avoids collisions with paths for robots $R_1..R_{i-1}$

Multi-Robot Path Planning

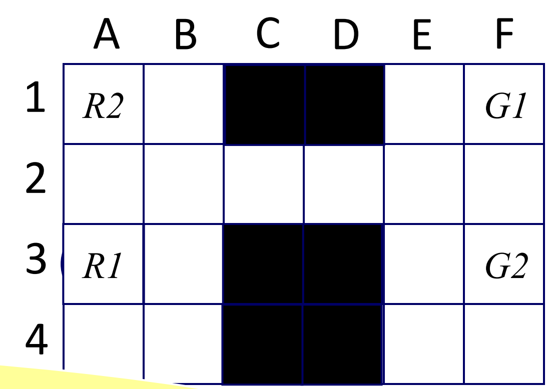
- Path planning for *N* robots to get to their goals w/o collisions

Distributed planning

Is it complete?

Is it optimal?

What is the complexity of Prioritized Planning?



One popular approach: Prioritized Planning

For i = 1:N

Compute path for robot R_i that avoids collisions with paths for robots $R_1..R_{i-1}$

Multi-Robot Path Planning

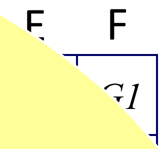
- Path planning for *N* robots to get to their goals w/o collisions

Distributed planning

***complete & optimal** approach*

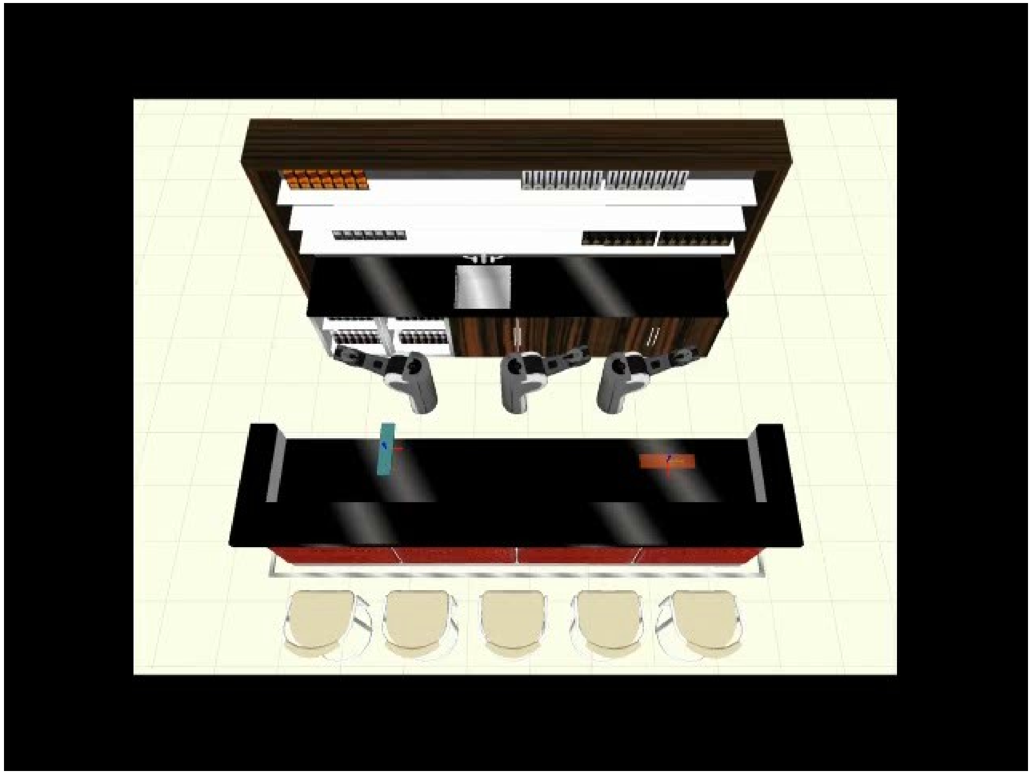
Conflict-based Search (CBS)

```
C(Sstart) = {}; //no constraints on paths at the start state
OPEN = {Sstart}; g(Sstart) = 0;
while (OPEN ≠ 0)
    remove S with the smallest g-value from OPEN
    if no collisions between paths associated with S, then return them as the overall plan
    for each (or at least one) collision of robots Ri and Rj at vertex v at time t
        C1 = C(S) U {Ri(t) ≠ v} //Ri can't be at vertex v at time t
        compute N paths that satisfy C1 constraints (w/o collision-checking
                between paths); let the overall cost be PLANCOST1
        C2 = C(S) U {Rj(t) ≠ v} //Rj can't be at vertex v at time t
        compute N paths that satisfy C2 constraints (w/o collision-checking
                between paths); let the overall cost be PLANCOST2
        insert state S1 with C(S1) = C1 and g(S1) = PLANCOST1
        insert state S2 with C(S2) = C2 and g(S2) = PLANCOST2
```



Multi-Robot Cooperative Planning/Task Allocation

- Example: planning for N robotic arms to move an object



[Cohen et al., '14]
(performs joint state-space planning)

Multi-Robot Cooperative Planning/Task Allocation

- Example: planning for N robotic arms to move an object

(planning is distributed: plan on Roman platform first, then on PR2)

Multi-Robot Cooperative Planning/Task Allocation

- Example: planning for multi-robot exploration/mapping

N robots need to explore and build a map of unknown environment

One approach: Distributed Greedy Mapping

For i = 1:N

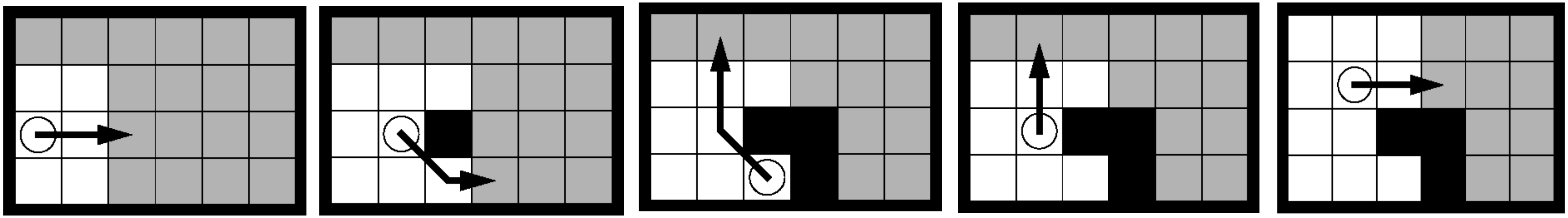
Compute a path using Greedy Mapping approach for robot R_i taking into account what paths were computed for $R_1..R_{i-1}$ (and what cells they would see)

Multi-Robot Cooperative Planning/Task Allocation

- Example: planning for multi-robot exploration/mapping

N robots need to explore and build a map of unknown environment

- Greedy Mapping for a single robot:
  - always move the robot on a shortest path to the closest unobserved (or unvisited) cell
  - it always achieves a gain in information.
  - thus, it is guaranteed to map the environment that is reachable (assuming all moves are reversible)



One approach: Distributed Greedy Mapping

For i = 1:N

Compute a path using Greedy Mapping approach for robot R_i taking into account what paths were computed for $R_1..R_{i-1}$ (and what cells they would see)

Multi-Robot Cooperative Planning/Task Allocation

- Example: planning for multi-robot exploration/mapping

N robots need to explore and build a map of unknown environment

[Butzke et al., '11]

One approach: Distributed Greedy Mapping

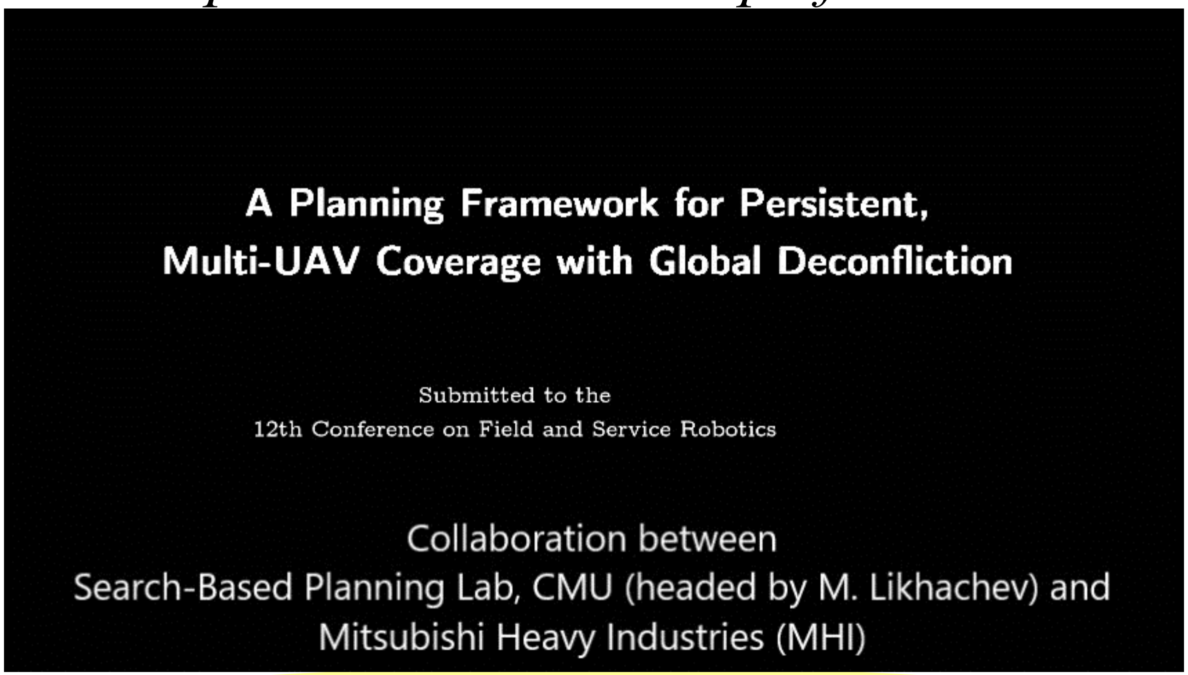
For i = 1:N

Compute a path using Greedy Mapping approach for robot R_i taking into account what paths were computed for $R_1..R_{i-1}$ (and what cells they would see)

Multi-Robot Cooperative Planning/Task Allocation

- Example: planning for multi-robot exploration/mapping

N robots need to explore and build a map of unknown environment



[Kusner et al., '19]

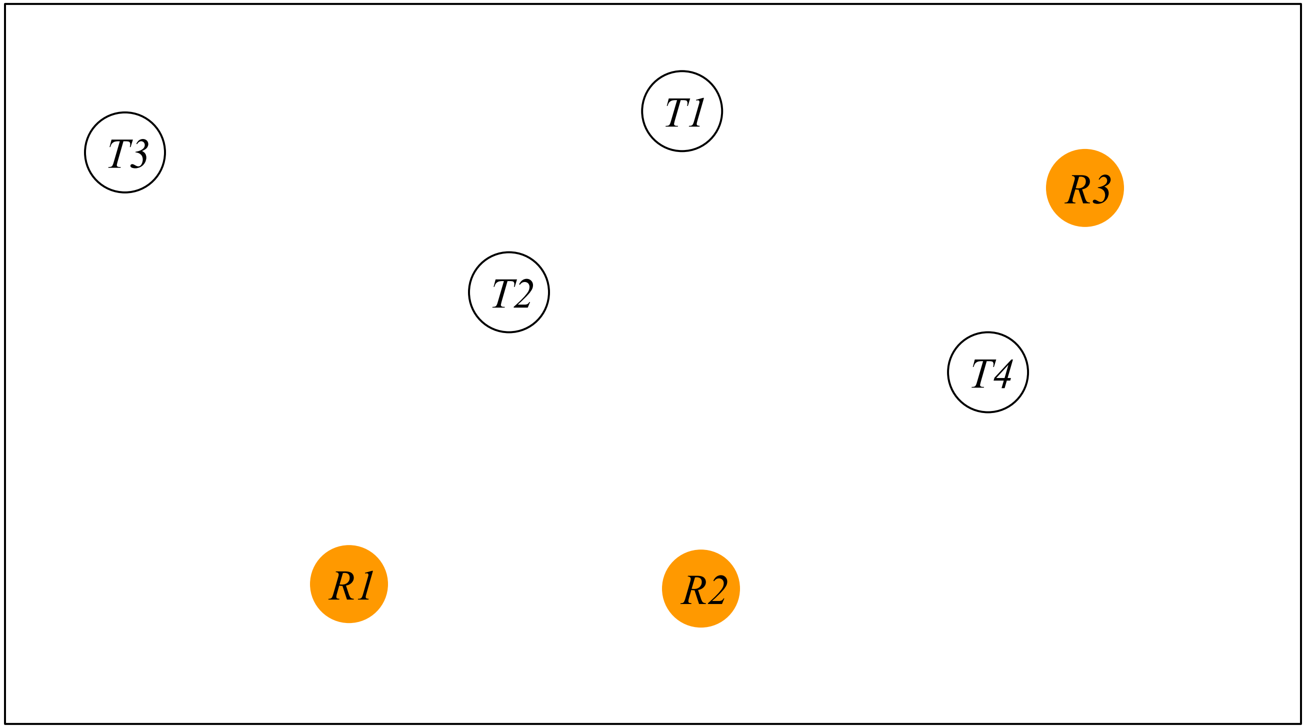
One approach: Distributed Greedy Mapping

For i = 1:N

Compute a path using Greedy Mapping approach for robot R_i taking into account what paths were computed for $R_1..R_{i-1}$ (and what cells they would see)

Multi-Robot Cooperative Planning/Task Allocation

- Market-based approach (very popular distributed approach)
  - Consider planning the allocation of tasks to *N* robots
  - General scheme: *robots auction out their tasks to their teammates with the goal of increasing their own revenue*

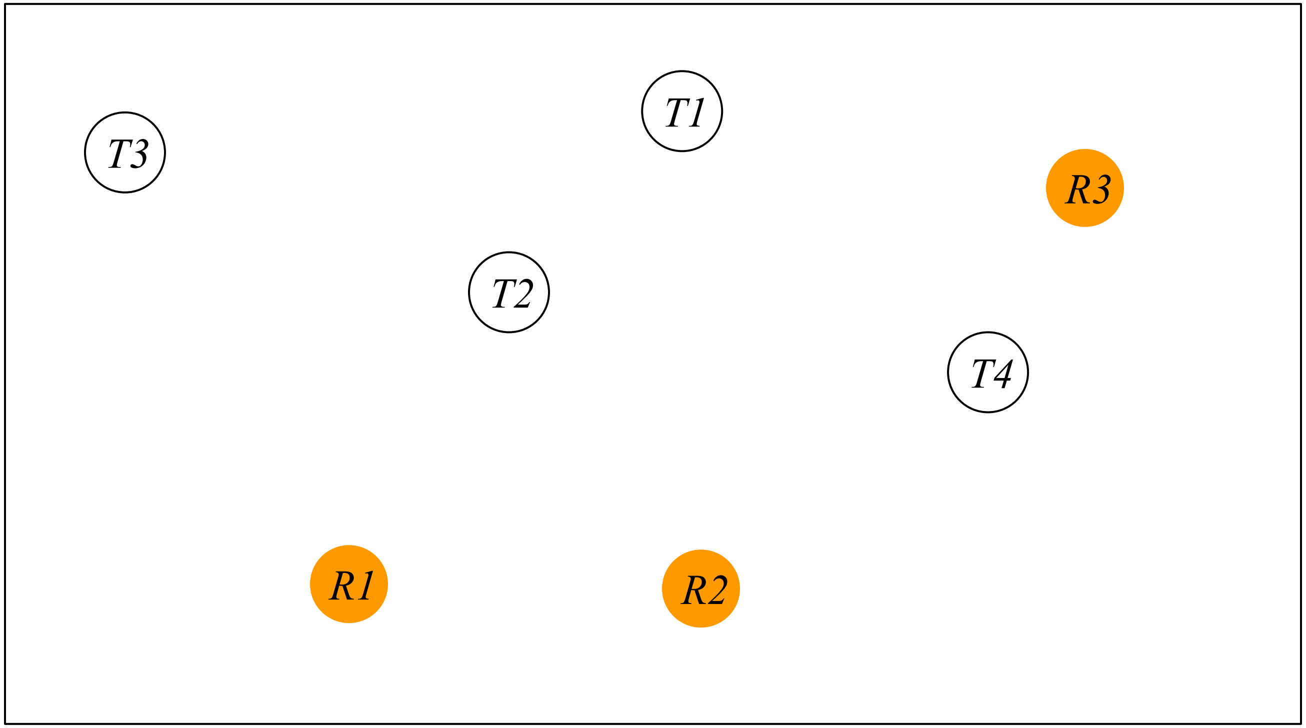


Market-based Approach

Given N robots $R_1 \ldots R_N$, M tasks $T_1 \ldots T_M$, and C_i^{Rj} – cost of executing task i by robot R_j (cost may depend on other tasks executed by this robot)

Planner needs to decide: Which task gets executed by which robot?

Find a plan(mapping) $\pi^*: T_i \rightarrow R_j$ *such that* $\pi^* = \text{argmin} \sum C_i^{\pi(Ti)}$



Market-based Approach

Given N robots $R_1 \ldots R_N$, M tasks $T_1 \ldots T_M$, and C_i^{Rj} – cost of executing task i by robot R_j (cost may depend on other tasks executed by this robot)

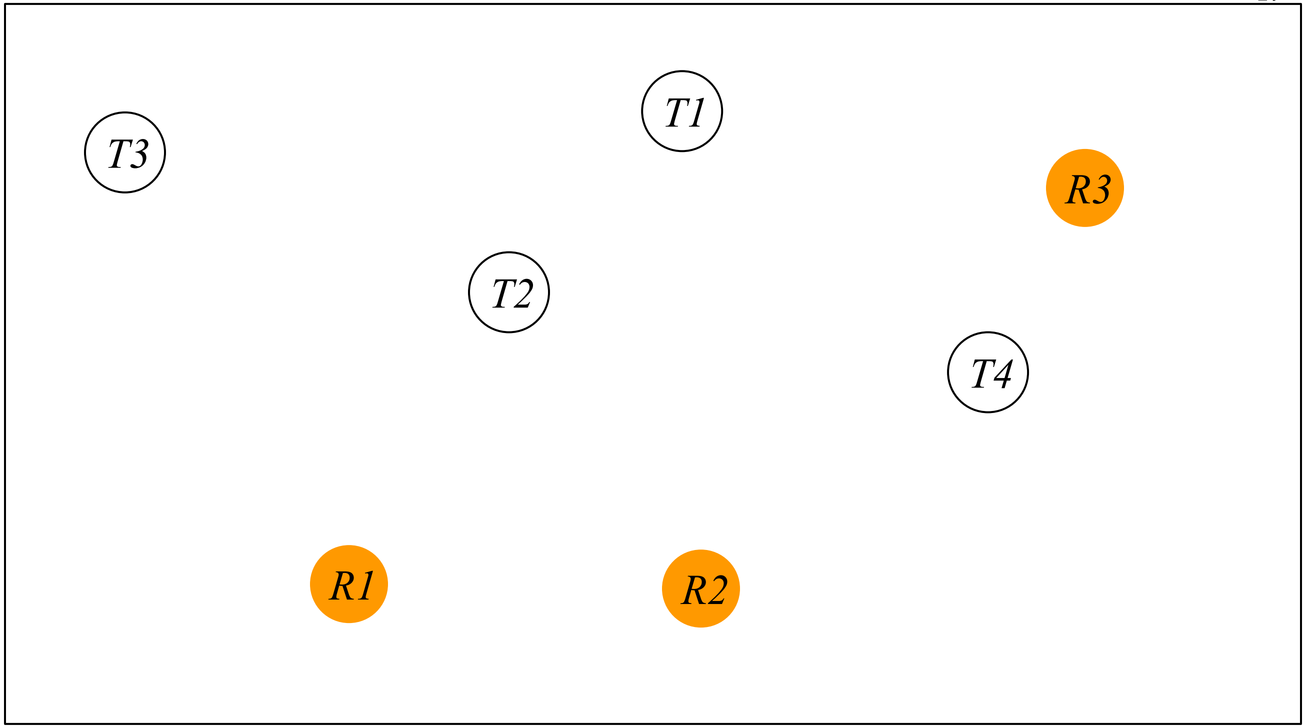
Iterate over steps 1-4 until convergence or planning time expires

Step 1: start with an arbitrary plan π

Step 2: all robots offer their tasks T_i *at auction at the max. price of* $C_{Ti}^{\pi(Ti)} - \epsilon$

Step 3: all robots R_j *bid on the offered tasks* T_i *with the bid* $= C_i^{Rj} + \epsilon$

Step 4: robots sell to the lowest bidders if they are below max. price and get profit: $C_{Ti}^{\pi(Ti)} - C_i^{Rj} - \epsilon$



Market-based Approach

Given N robots $R_1 \ldots R_N$, M tasks $T_1 \ldots T_M$, and C_i^{Rj} – cost of executing task i by robot R_j (cost may depend on other tasks executed by this robot)

When does it converge in one iteration?

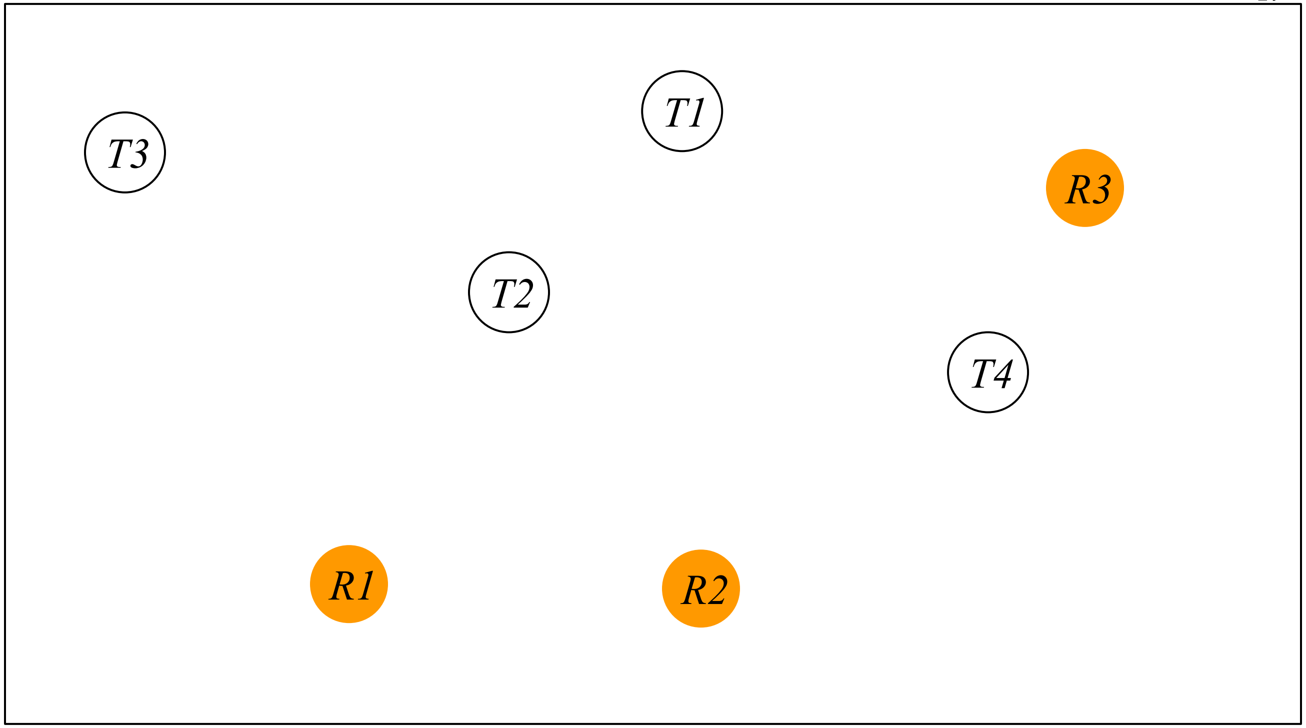
Iterate over steps 1-4 until convergence or planning time expires

Step 1: start with an arbitrary plan π

Step 2: all robots offer their tasks T_i *at auction at the max. price of* $C_{Ti}^{\pi(Ti)} - \epsilon$

Step 3: all robots R_j *bid on the offered tasks* T_i *with the bid* $= C_i^{Rj} + \epsilon$

Step 4: robots sell to the lowest bidders if they are below max. price and get profit: $C_{Ti}^{\pi(Ti)} - C_i^{Rj} - \epsilon$

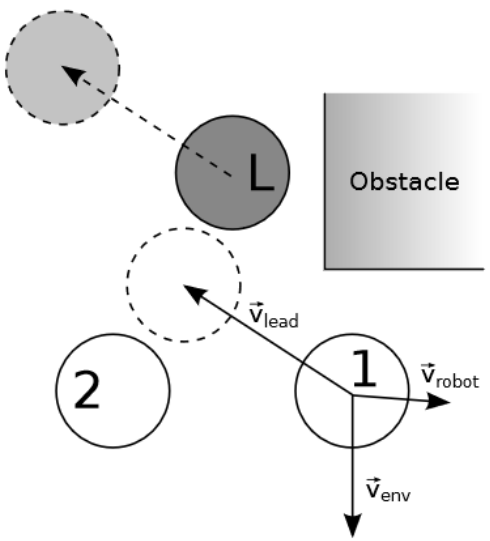


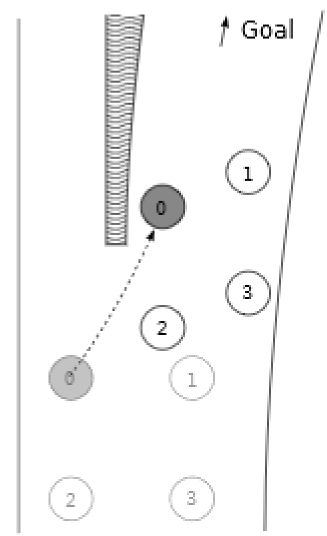
Planning for Leader-based Coordination

• **Fully decentralized approach** (doesn't rely on the presence of communication between robots)

• Plan for the "leader" robot (sometimes leader can be just a centroid of the team or some other reference point)

• All other robots execute either "follow the leader" or "follow neighbors within field-of-view" behaviors while avoiding collisions





What You Should Know…

- Different styles of multi-robot planning
  - Centralized vs. decentralized
  - Joint state-space planning vs. distributed planning
  - Multi-robot path planning vs. cooperative task planning

- Prioritized Multi-robot Path Planning

- Market-based Approach to multi-robot planning