MULTI-AGENT NAVIGATION

BACK TO THE BEGINNING

A* ALGORITHM - REVISITED

- Nodes are in one of three states
  - Visited
    - Popped from the queue
  - Queued
    - Placed in the queue because a neighbor was visited
  - Unexplored
    - Hasn't been considered in any way

A* ALGORITHM - REVISITED

- Queued
  - They are placed in the queue with a value for f
  - NODES in the queue can have their f-value change
    - Changed f-value → changed path

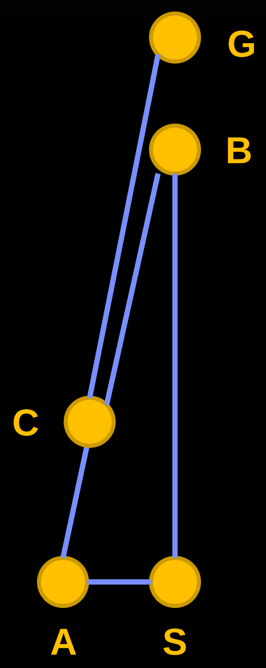
A* ALGORITHM - REVISITED

```
minDistance( start, end, nodes )
    closed = {}
    open = {start}
    g[ start ] = 0
    f[ start ] = g[ start ] + h( start, end )
    while ( ! open.isEmpty() )
        c = minF( open )
        if ( c == end ) return g[ c ]
        open = open \ {c}; closed = closed U {c}
        for each neighbor, n, of c
            if ( n in closed ) continue
            gTest = g[ c ] + E( n, c )
            if ( gTest < g[ n ] )
                g[ n ] = gTest; f[ n ] = gTest + h(n, end)
            open = open U {n}
```

Sean's A*

A* ALGORITHM - REVISITED

- Find the path from S → G

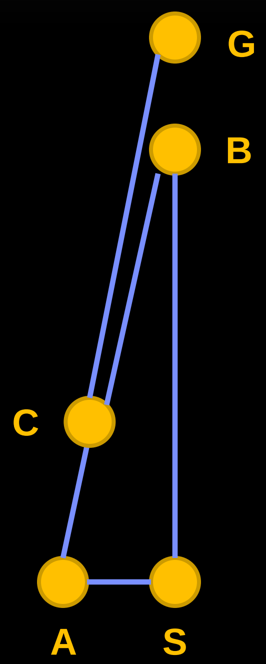



```
A*( S, G )
    Q = {S} // f(S)=||G-S||, prev(S)=NULL
    curr = S // Q = {}
    Q = {A} // f(A) = x, prev(A)=S
    Q = {A,B} // f(B) < f(A), prev(B)=S
    curr = B   // f(B) < f(A), Q = {A}
    Q = {A,C} // f(C) > f(B) > f(A)
              // prev(C) = B
    curr = A   // f(B) < f(C), Q={C}
    // C is already queued – don't change
    //   its value
    curr = C   // Q={}
    Q = {G} // prev(G) = C
    curr = G
    DONE!

    Build Path
    G
    prev(G) = C
    prev(C) = B
    prev(B) = S
    PATH: S → B → C → G
```

A* ALGORITHM - REVISITED

- Find the path from S → G




```
A*( S, G )
    Q = {S} // f(S)=||G-S||, prev(S)=NULL
    curr = S // Q = {}
    Q = {A} // f(A) = x, prev(A)=S
    Q = {A,B} // f(B) < f(A), prev(B)=S
    curr = B   // f(B) < f(A), Q = {A}
    Q = {A,C} // f(C) > f(B) > f(A)
              // prev(C) = B
    curr = A   // f(B) < f(C), Q={C}
    // C is already queued
    //   test if this is cheaper
    fA(C) < fB(C) → f(C) = fA(C) and prev(C) = A
    curr = C   // Q={}
    Q = {G} // prev(G) = C
    curr = G
    DONE!

    Build Path
    G
    prev(G) = C
    prev(C) = A
    prev(B) = S
    PATH: S → A → C → G
```

A* ALGORITHM - REVISITED

- How do you find the minimum value?
- Do you account for changing values?
- Typical min-heap implementations don’t allow this
  - (STL certainly doesn’t)
- I’ll send out a scenario in which this matters

NEXT HOMEWORK

- Implement pedestrian model
  - Force-based
    - Zanlungo 2011
    - Johansson 2007
- Much simpler than the roadmap planner
  - Algorithmically simpler
  - Simpler engineering as well
- Write-up will go out later this week

AGENT AI

- Temporally-dependent agent goals
  - How do you model an agent's changing goals?
- Menge uses an FSM
  - Why use an FSM?

AGENT AI - FSM

- States can encode:
  - Goal
  - Strategy technique
  - Unique agent state
- States can change w.r.t. time
  - Explicitly based on elapsed time
  - Implicitly based on achieved goals or change of simulation state
- What else is there?

AGENT AI – BEHAVIOR TREE

- Currently en vogue in game AI
- http://www.altdevblogaday.com/2011/02/24/introduction-to-behavior-trees/
- Misnomer – they are not trees
  - They are directed, acyclic graphs (DAGs)
  - One node can have multiple parents
    - i.e. there are multiple ways to a particular behavior

AGENT AI – BEHAVIOR TREE

- Evaluating a BT
  - Start at the root and traverse the "whole" tree from the root at each time step
    - Evaluation of individual nodes affect traversal
      - Node evaluation produces signals
        - Ready – ready to evaluate
        - Success – evaluated and it worked
        - Running – Not finished, run again next time
        - Failed – failed, but unimportant
        - Error – failed, but important

AGENT AI – BEHAVIOR TREE

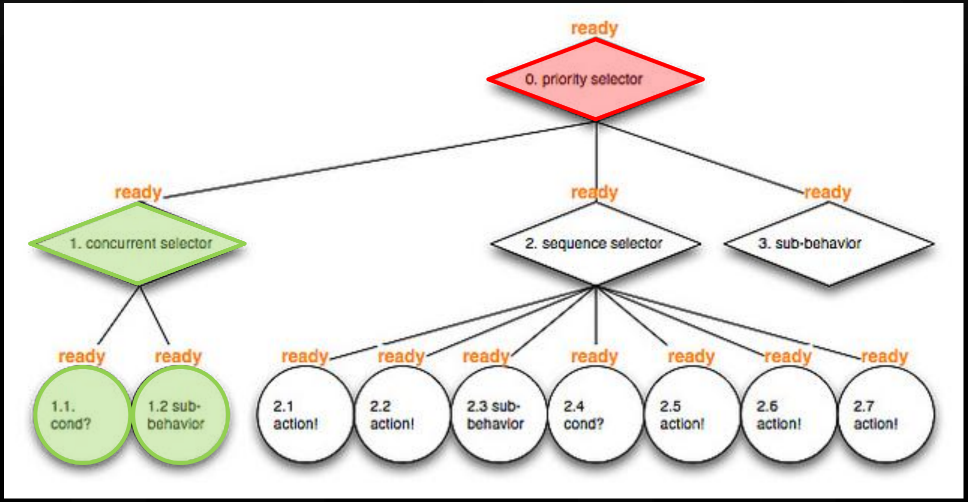
- Inner nodes dictate traversal
  - Priority nodes
    - evaluate in priority order, stop on success
  - Sequence nodes
    - Run children in sequence
  - Loop nodes
    - Run children in continuous sequence
  - Random
    - Select child
  - Concurrent
    - Run all children (success dependent on child success rate)
  - Decorator
    - Apply evaluation constraints on children (temporal, pauses, etc.)

AGENT AI – BEHAVIOR TREE

- Leaf nodes
  - Actions
    - Agent behavior
    - Game state changes
  - Conditions
    - Typically siblings of actions
    - Used in sequence and concurrent nodes to enforce invariants

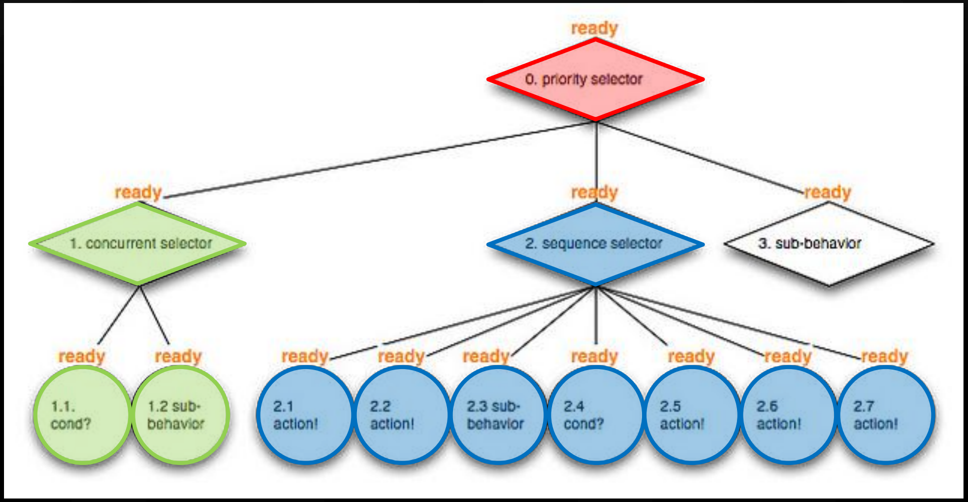
AGENT AI – BEHAVIOR TREE

- Dragon behavior
  - Priority selector
  - Concurrent - Guard treasure
    - Condition – is thief near?
    - Sub-tree - Chase thief



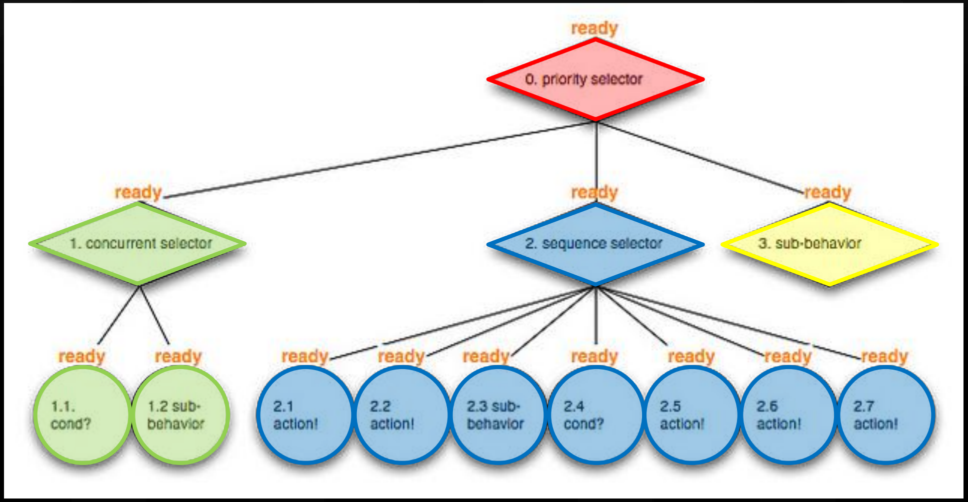
AGENT AI – BEHAVIOR TREE

- Dragon behavior
  - Sequence – get more treasure
    - Action – choose castle
    - Action – fly to castle
    - Sub-tree – fight guards
    - Condition – Can carry gold?
    - Action – take gold
    - Action – Fly home
    - Action – store gold



AGENT AI – BEHAVIOR TREE

- Dragon behavior
  - Sub-tree – post pictures on facebook



AGENT AI

- What is the difference between FSM and BT?
  - What can you do with one that you can't do with the other?
  - What can you do easily with one that you can't do easily with the other?

MOTION PLANNING

- Return to classic motion planning

COUPLED PLANNING

- Crowd simulation
  - Decoupled/decentralized/distributed planning
  - Limited coordination
    - In principle, no coordination
    - However, coordination can be added
  - **No** guarantees on convergence
    - If there is a solution, can you promise you'll get it?

MULTI-ROBOT MOTION PLANNING

Jur van den Berg

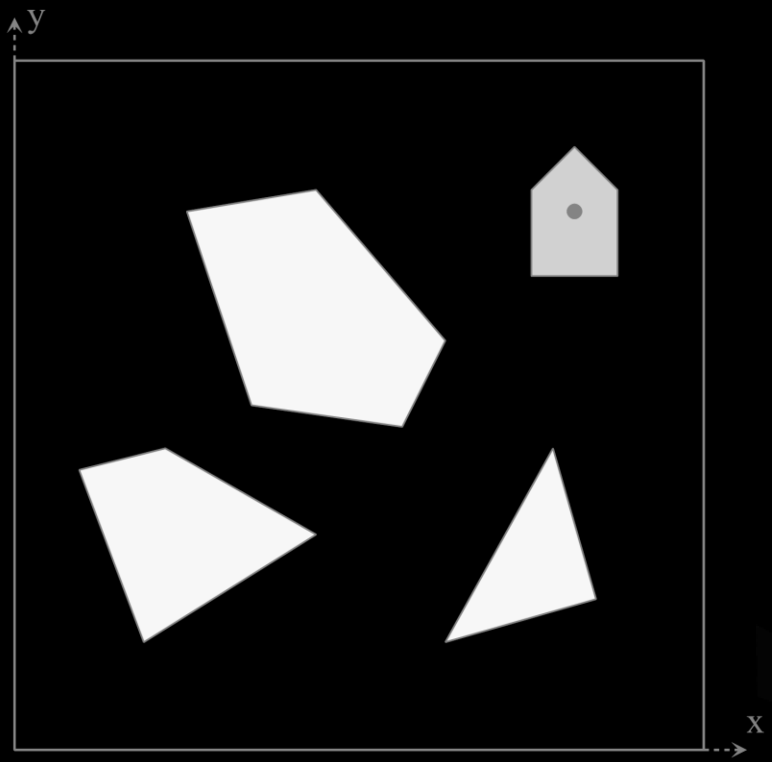
OUTLINE

- Recap: Configuration Space for Single Robot
- Multiple Robots: Problem Definition
- Multiple Robots: Composite Configuration Space
- Centralized Planning
- Decoupled Planning
- Optimization Criteria

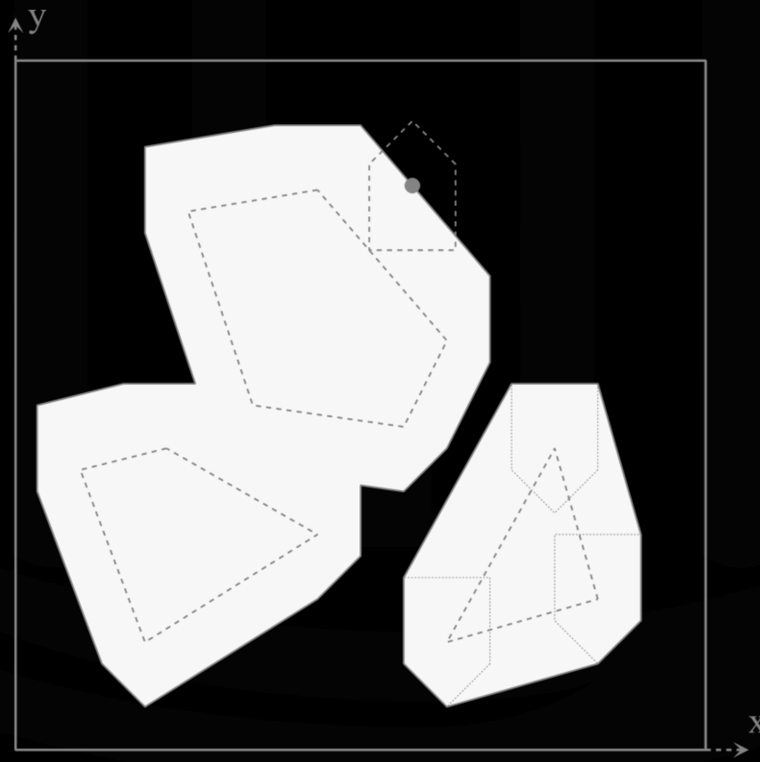
CONFIGURATION SPACE

- Single Robot
- Dimension = #DOF

- Translating in 2D
- Minkowski Sums



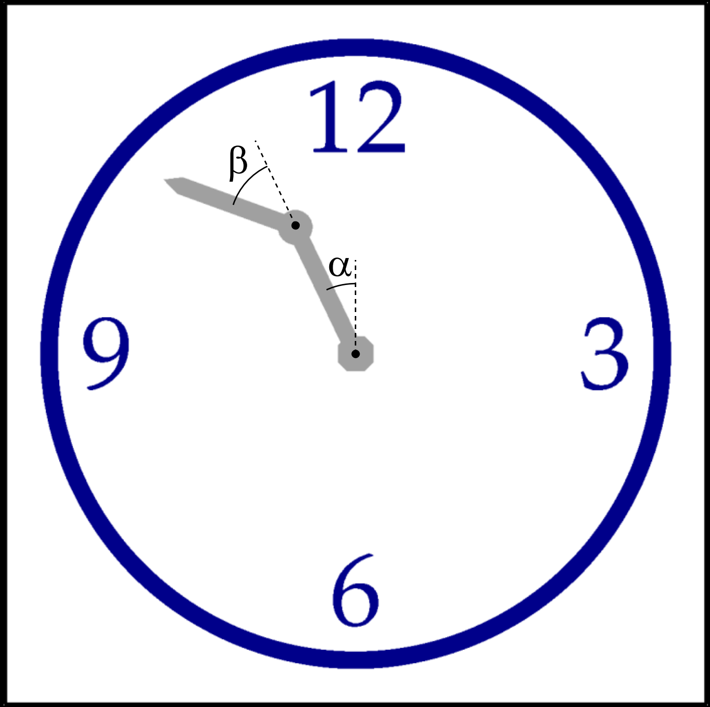
Workspace



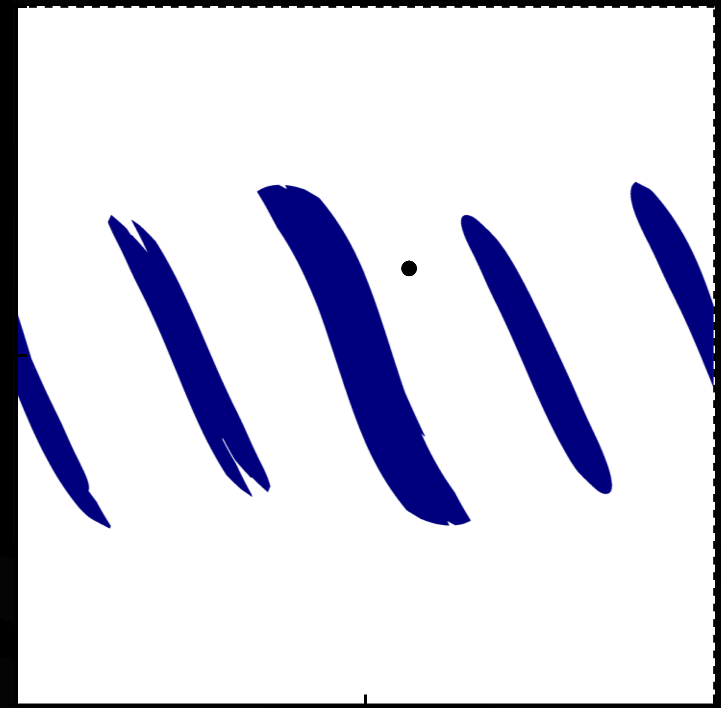
Configuration Space

CONFIGURATION SPACE

- A Single Articulated Robot (2 Rotating DOF)
- Hard to compute explicitly



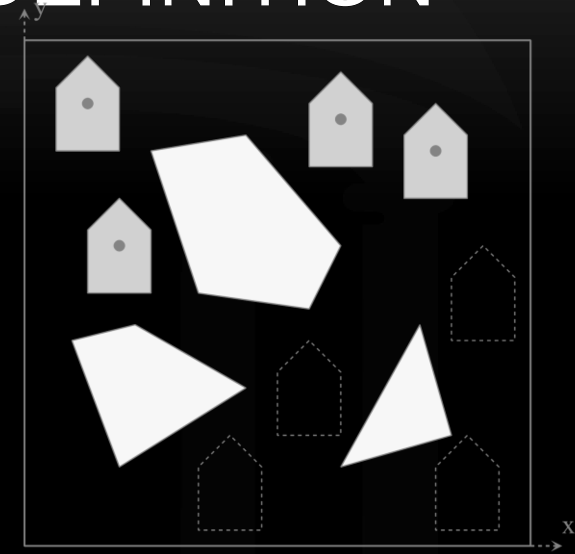
Workspace

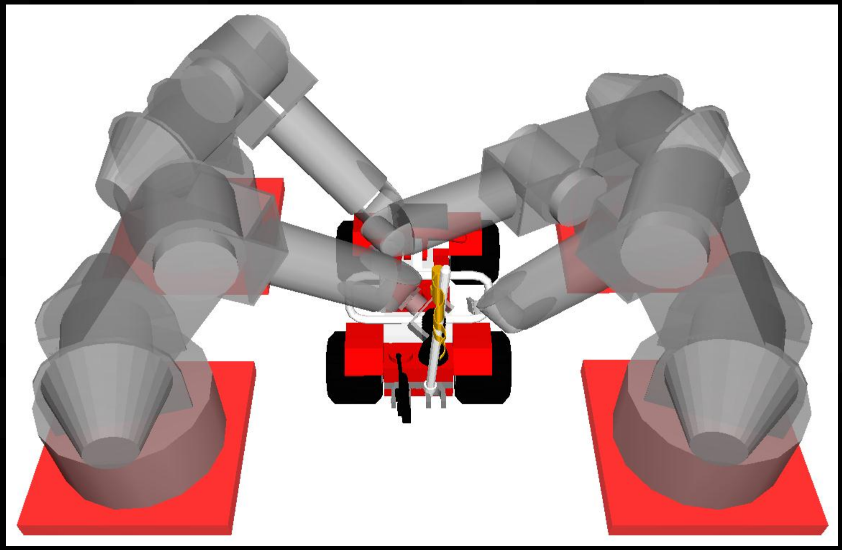


Configuration Space

MULTIPLE ROBOTS: PROBLEM DEFINITION

- N robots $R_1, R_2, \ldots, R_N$ in same workspace
- Start configurations $(s_1, s_2, \ldots, s_N)$
- Goal configurations $(g_1, g_2, \ldots, g_N)$
- Find trajectory for all robots without collisions with obstacles and *mutual collisions*
- Robots may be of different type





PROBLEM CHARACTERIZATION

- Each of N robots has its own configuration space: (C_1, C_2, …, C_N)
- Example with two robots: one translating robot in 3D, and one articulated robot with two joints:
  - $C_1 = \mathbf{R}^3$
  - $C_2 = [0, 2\pi)^2$

COMPOSITE CONFIGURATION SPACE

- Treat multiple robots as one robot
- Composite Configuration Space C
  - $C = C_1 \times C_2 \times \ldots \times C_N$
- Example: $C = \mathbf{R}^3 \times [0, 2\pi)^2$
  - Configuration $c \in C$: $c = (x, y, z, \alpha, \beta)$
- Dimension of Composite Configuration Space
  - *Sum* of dimensions of individual configuration spaces (number of degrees of freedom)

OBSTACLES IN COMPOSITE C-SPACE

- Composite configurations are in forbidden region when:
  - One of the robots collides with an obstacle
  - A pair of robots collide with each other
- $CO = \{c_1 \times c_2 \times \ldots \times c_N \in C \mid \exists i \in 1\ldots N :: c_i \in CO_i \vee \exists i, j \in 1\ldots N :: R_i(c_i) \cap R_j(c_j) \neq \emptyset\}$
- Planning in Composite C-Space?

PLANNING FOR MULTIPLE ROBOTS

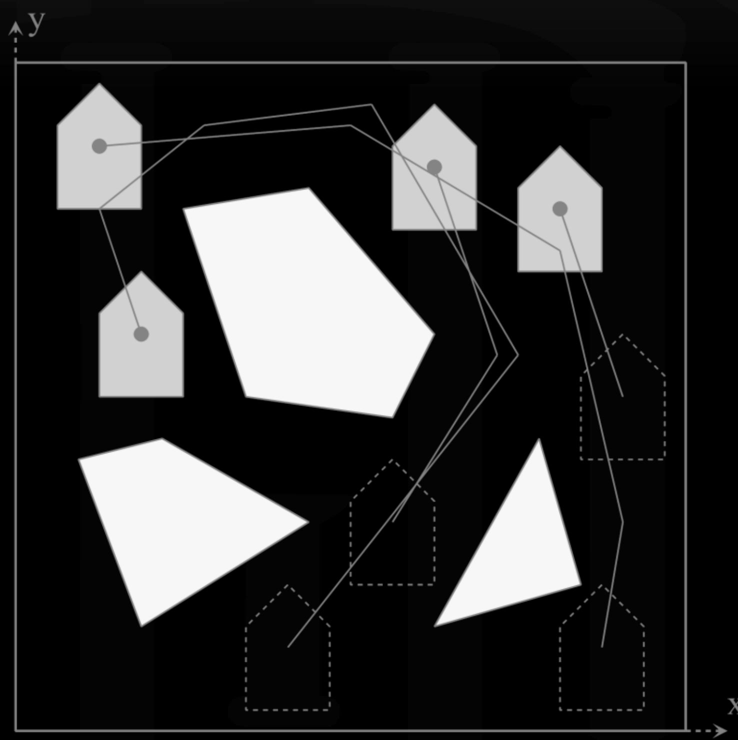
- Any single robot planning algorithm can be used in the Composite configuration space.
- Grid
- Cell Decomposition
- Probabilistic Roadmap Planner

PROBLEM

- The running time of Motion Planning Algorithms is exponential in the dimension of the configuration space
- Thus, the running time is exponential in the number of robots
- Algorithms not practical for 4 or more robots
- Solution?

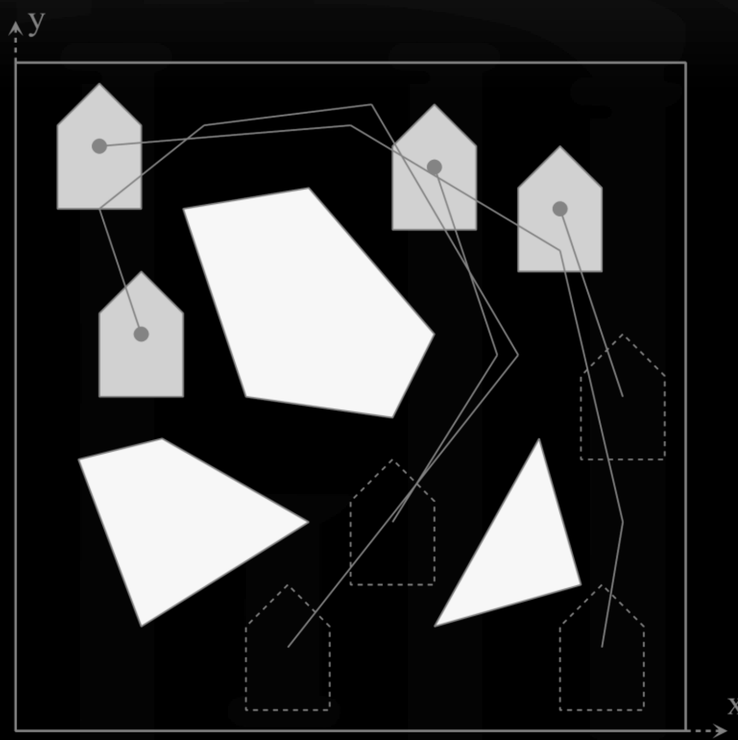
DECOUPLED PLANNING

- First, plan a path for each robot in its own configuration space
- Then, tune velocities of robots along their path so that they avoid each other
- Advantages?
- Disadvantages?



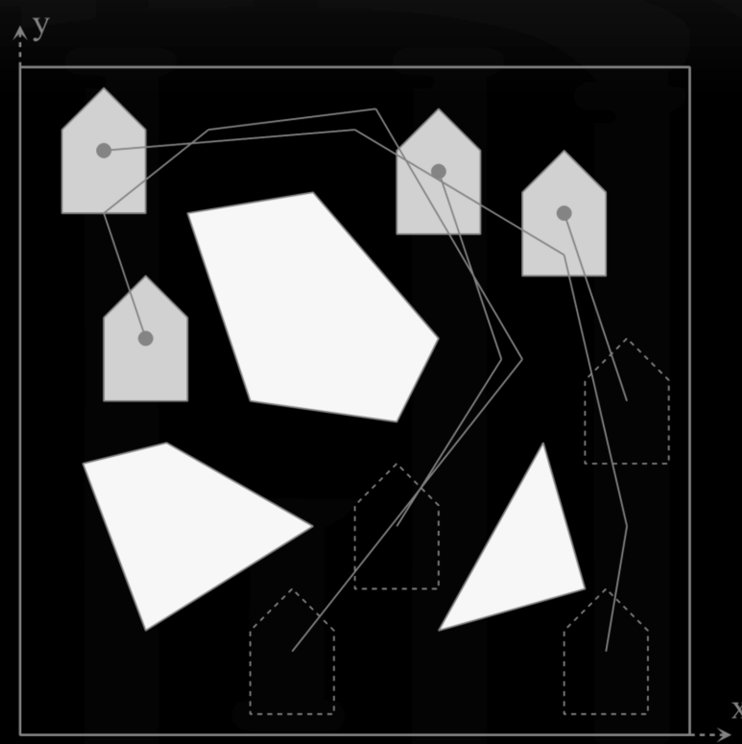
ADVANTAGES

- You don’t have to deal with collisions with obstacles anymore
- The number of degrees of freedom for each robot has been reduced to one



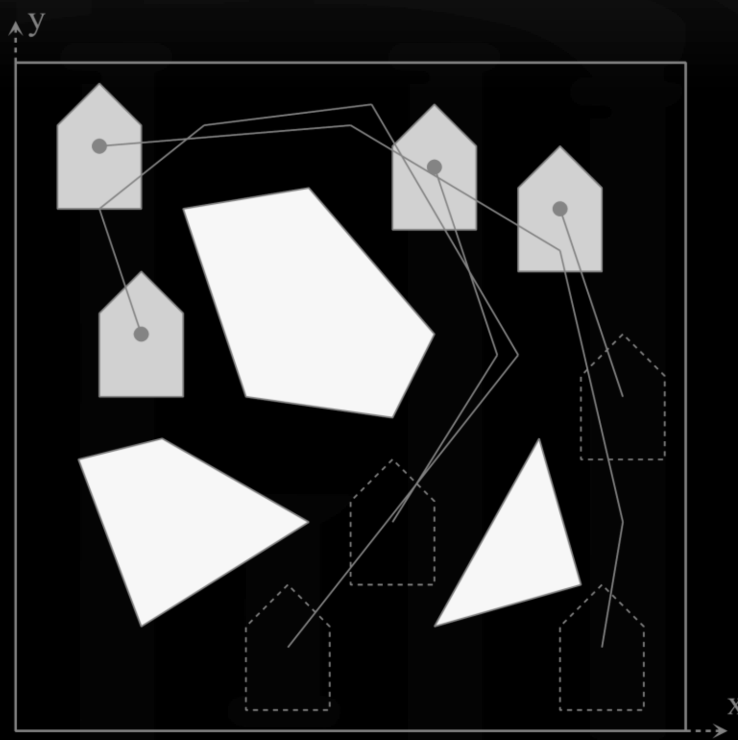
DISADVANTAGES

- The running time is still exponential in the number of robots
- A solution may no longer be found, even when one exists (incompleteness)
- Solution?



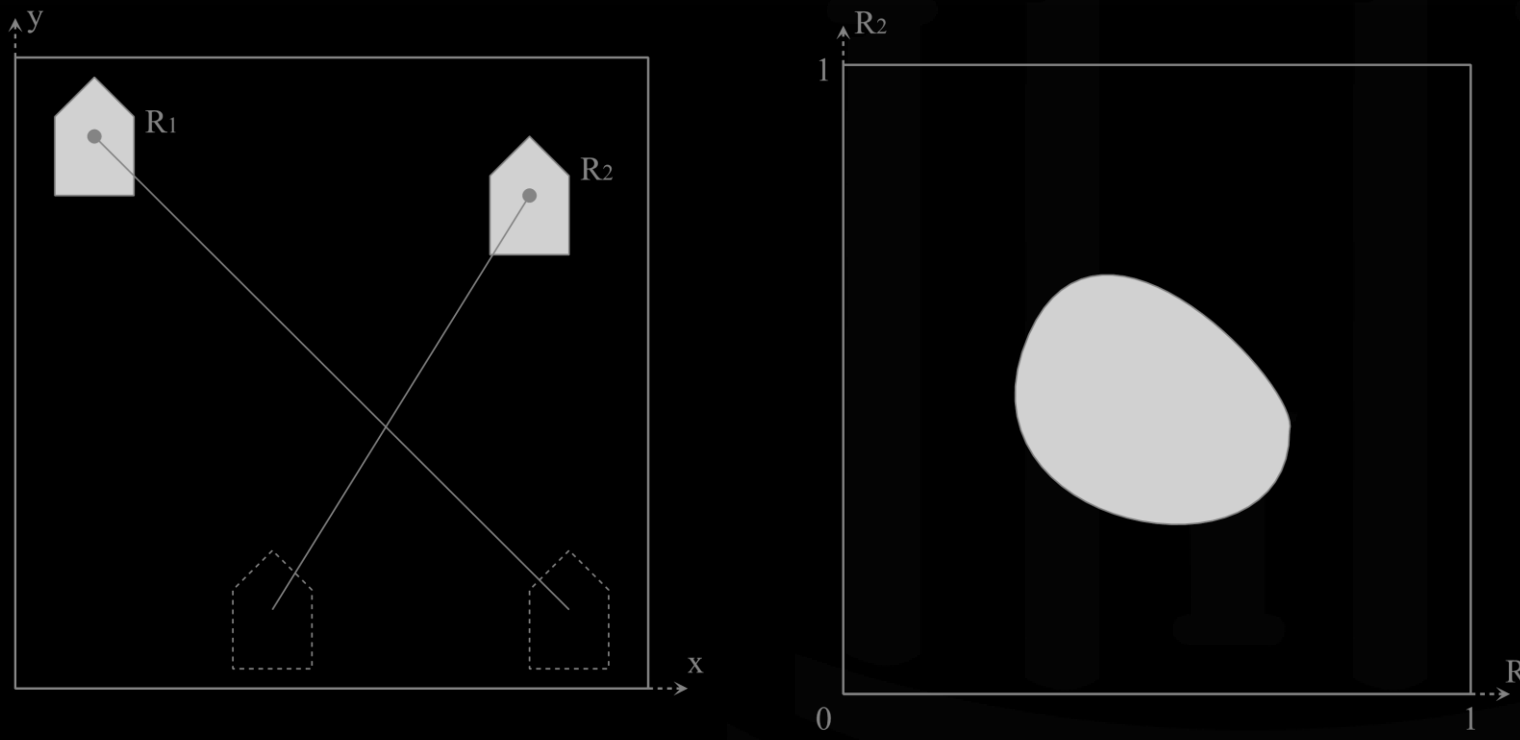
POSSIBLE SOLUTION

- Only plan paths that avoid the other robots at start and final position
- Why is that a solution?
- However, such paths may not exist, even if there is a solution



COORDINATION SPACE

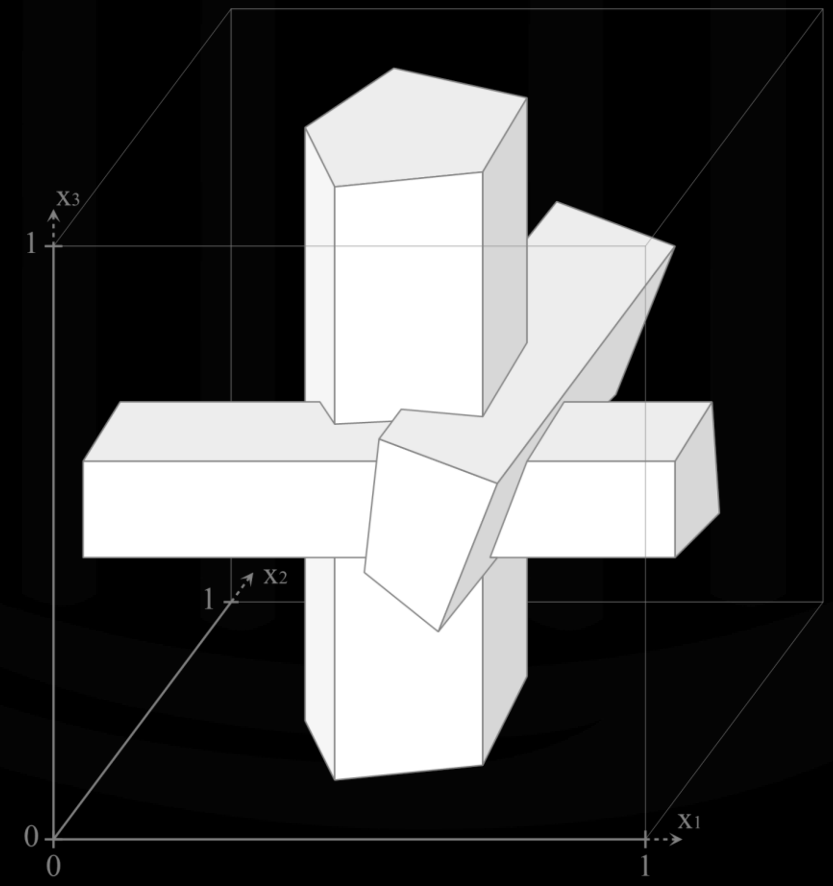
- Each axis corresponds to a robot



- How is the coordination-space obstacle computed?

CYLINDRICAL OBSTACLES

- Obstacles are cylindrical (also in Composite C-Space)
- Example: 3D-Coordination Space
- Why?
- How can this be exploited?



OPTIMIZATION CRITERIA

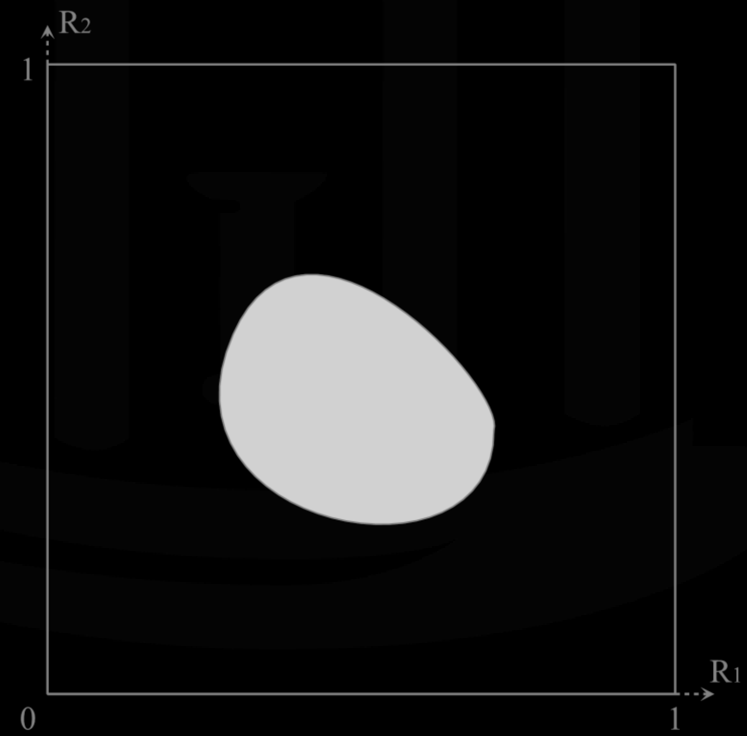
- There are (in most cases) multiple solutions to multi-robot planning problems.
- Each solution has an arrival time T_i for each of the robots: $(T_1, T_2, \ldots, T_N)$
- Select the "best" solution.
- What is best?

COST FUNCTION

- cost = $\max_i (T_i)$
- cost = $\sum_i (T_i)$
- Minimize cost

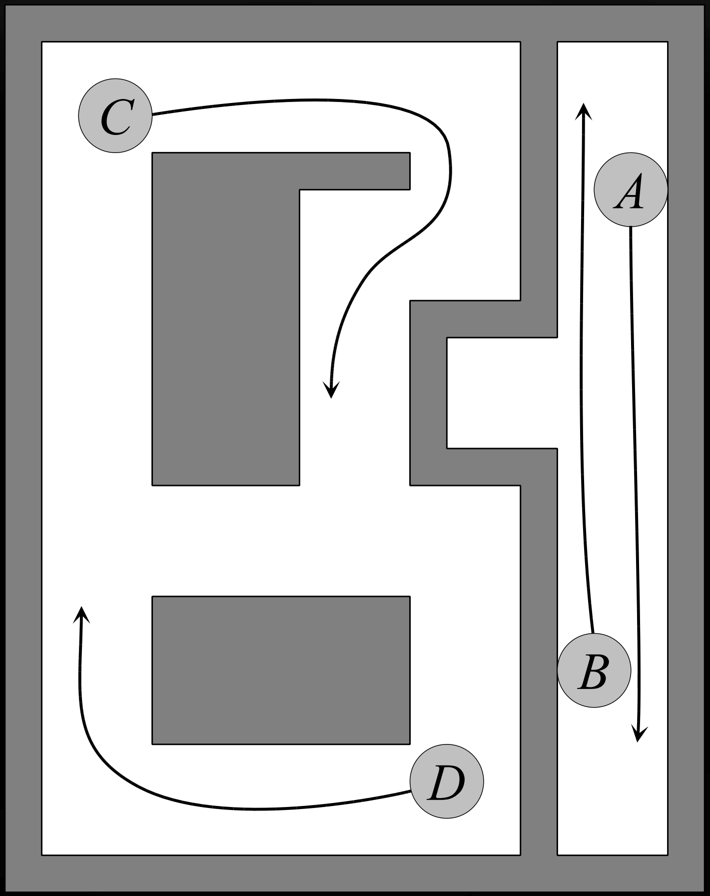
PARETO-OPTIMALITY

- Other approach: pareto-optimal solutions
- A solution $(T_1, \ldots, T_N)$ is *better* than $(T'_1, \ldots, T'_N)$ if $(\exists i \in 1\ldots N :: T_i < T'_i) \wedge (\forall j \in 1\ldots N :: T_j \leq T'_j)$
- A solution is pareto-optimal if there does not exist a better solution
- Multiple solutions can be pareto-optimal
- Which ones? How many?



CHALLENGE / OPEN PROBLEM

- Distribute computation
- Composite Configuration Space in worst case
- But not always necessary
- Complete planner
- Any ideas?



REFERENCES

- Latombe. Robot Motion Planning. (book)
- Kant, Zucker. Toward Efficient Trajectory Planning: The Path-Velocity Decomposition
- Leroy, Laumond, Simeon. Multiple Path Coordination for Mobile Robots: a Geometric Approach
- Svestka, Overmars. Coordinated Path Planning for Multiple Robots.
- Lavalle, Hutchinson. Optimal Motion Planning for Multiple Robots Having Independent Goals
- Sanchez, Latombe. Using a PRM Planner to Compare Centralized and Decoupled Planning for Multi-Robot Systems
- Ghrist, O’Kane, Lavalle. Computing Pareto Optimal Coordinations on Roadmaps

QUESTIONS?