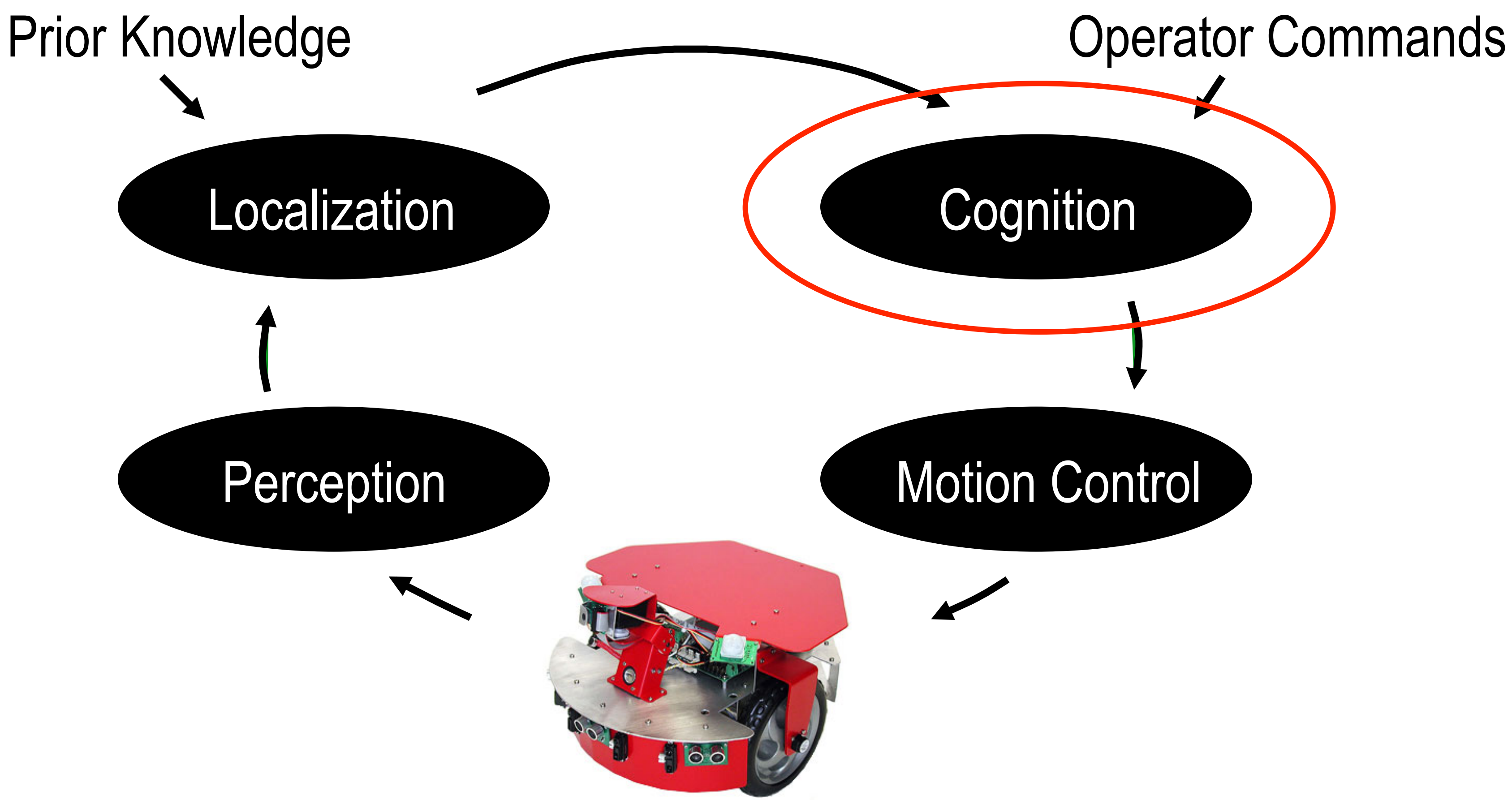
COS 495 – Lecture 22
Autonomous Robot Navigation

Instructor: Chris Clark

Semester: Fall 2011

Figures courtesy of Siegwart & Nourbakhsh

Control Structure

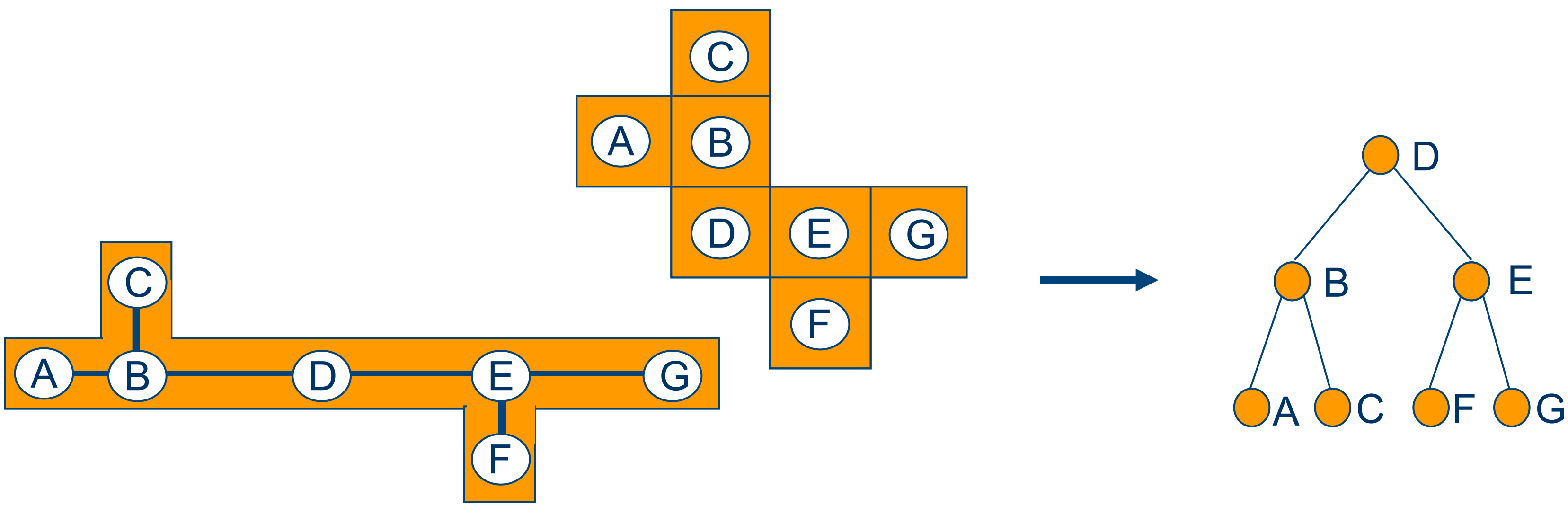


Graph Search: Outline

- Search Algorithms
  1. Breadth First Search
  2. Depth First Search
  3. A*

Motion Planning: Tree Search

- Once the configuration space is discretized, we can perform a tree search
  - Note: we know the connections, not the whole tree!
  - Example: How do we get from D to G?



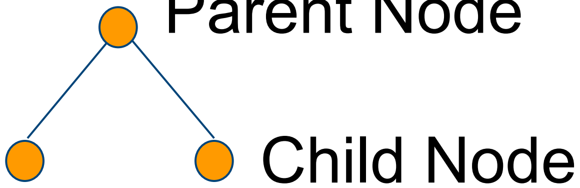
Motion Planning: Search Algorithms

- There are many tree searches available, but how are they different?

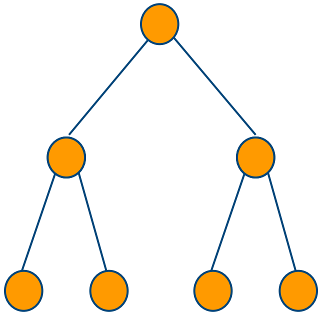
1. Breadth First Search
2. Depth First Search
3. Depth limited search
4. A*
5. …

Motion Planning: Breadth First Search

- Tree nomenclature:



- Algorithms differ in the order in which they search the branches (edges) of the tree

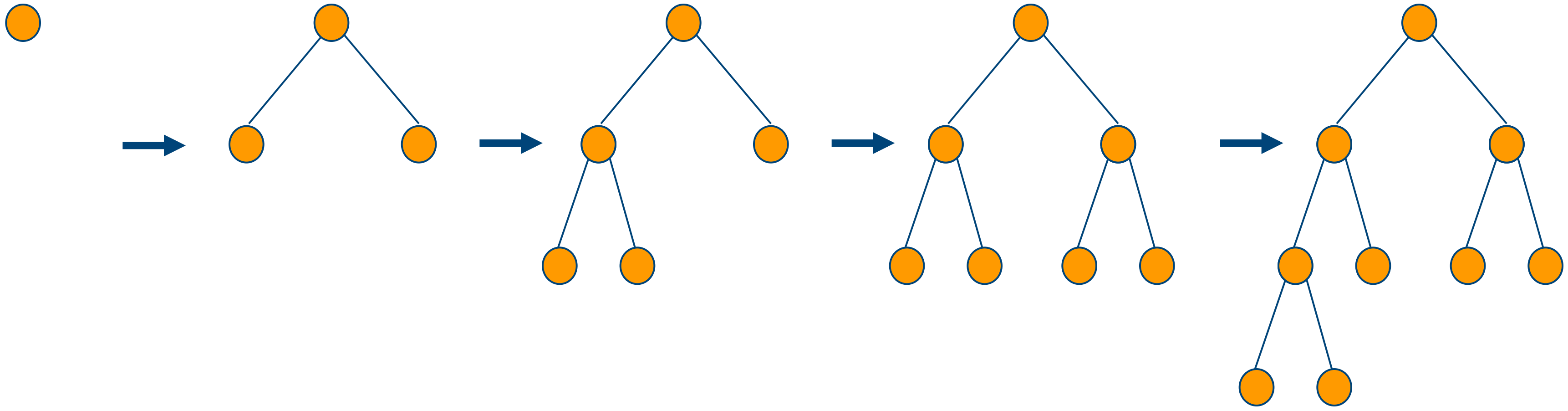


Graph Search: Outline

- Search Algorithms
  1. Breadth First Search
  2. Depth First Search
  3. A*

Motion Planning: Breadth First Search

- Search a tree, one level at a time.



Motion Planning: Breadth First Search

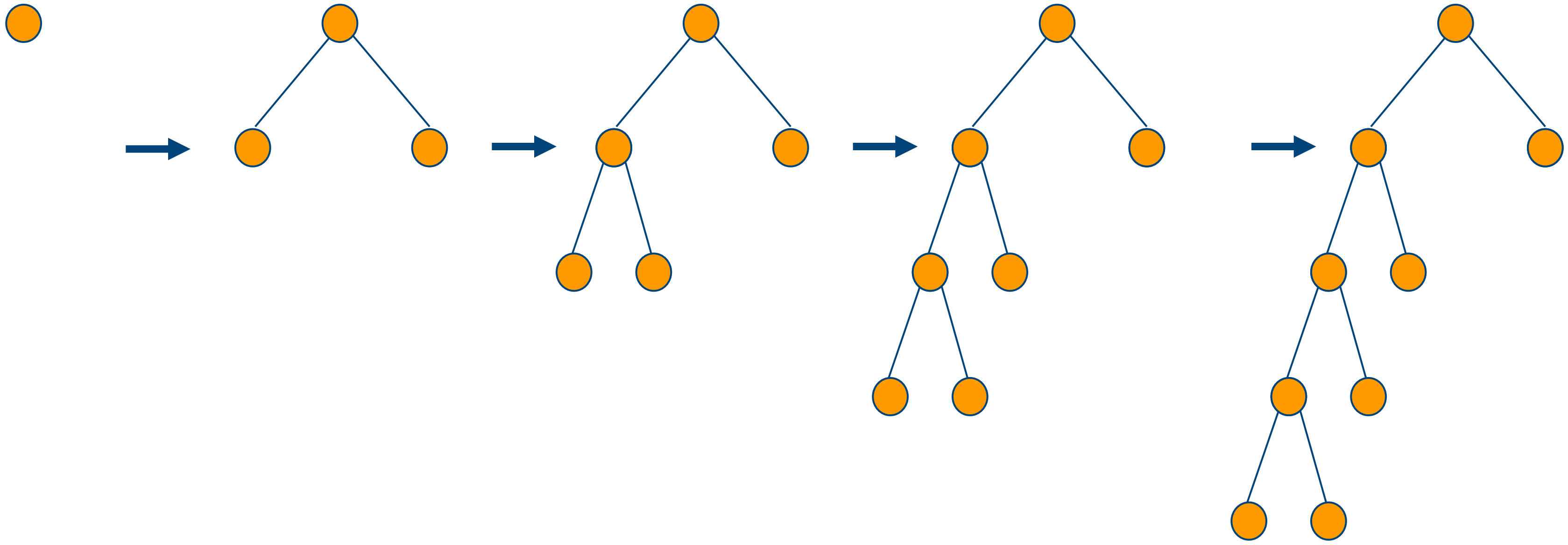
- Complete
- Optimal if cost is increasing with path depth.
- Computational complexity $O(b^d)$, where b is the branching factor and d is the depth
- Space (memory) complexity $O(b^d)$

Graph Search: Outline

- Search Algorithms
  1. Breadth First Search
  2. Depth First Search
  3. A*

Motion Planning: Depth First Search

- Search a tree, always expand to deepest level until final depth is reached.



Motion Planning: Depth First Search

- NOT Complete if infinite depth
- NOT Optimal
- Computational complexity $O(b^m)$, where b is the branching factor and m is the depth
- Space (memory) complexity $O(bm)$
- Good if there are many GOOD solutions

Graph Search: Outline

- Search Algorithms
  1. Breadth First Search
  2. Depth First Search
  3. A*

Motion Planning:
A* Search

- There are a set of algorithms called "Best-First Search"
- They try to search the children of the "best" node to expand.
- A* has become incredibly popular because it attempts to make the best node the one that will find the optimal solution and do so in less time.

Motion Planning: A* Search

- A* is optimal and complete, but can take time…
- Its complexity depends on the heuristic, but is exponential with the size of the graph.

Motion Planning: A* Search

- We evaluate a node n for expansion based on the function:

$$f(n) = g(n) + h(n)$$

- Where

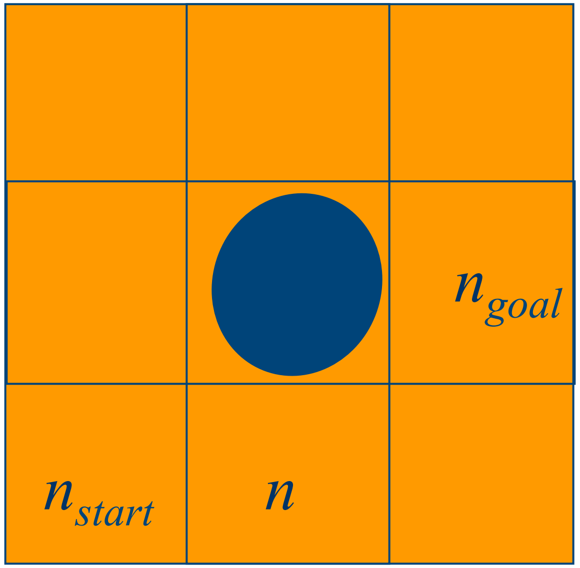
  $g(n)$ = path cost from the start node to n

  $h(n)$ = estimated cost of the cheapest path from node n to the goal

Motion Planning: A* Search

- Example: Cost for one particular node

$$f(n) = g(n) + h(n)$$



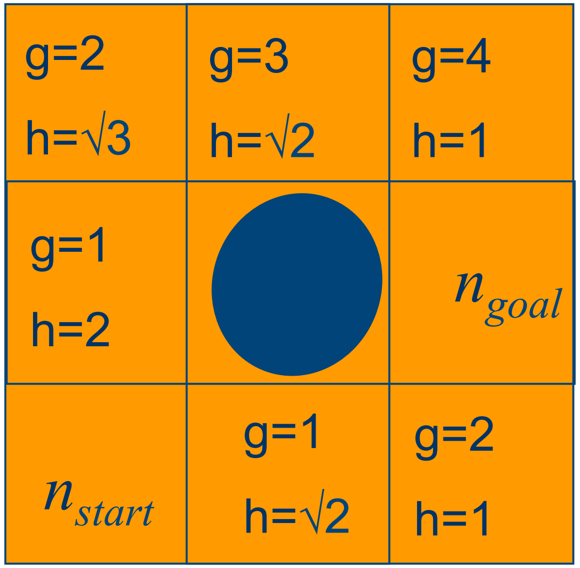
$$g(n) = 1$$

$$h(n) = \sqrt{2}$$

Motion Planning: A* Search

- Example: Cost for each node

$$f(n) = g(n) + h(n)$$



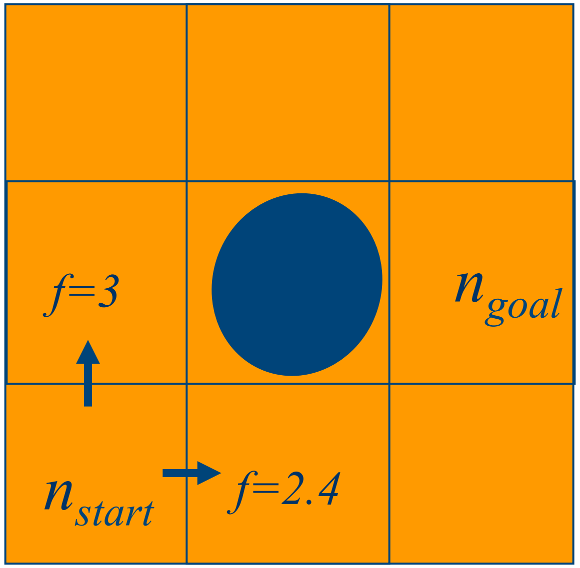
Motion Planning:
A* Search

- The strategy is to expand the node with the cheapest path (lowest f).
- This is proven to be complete and optimal, if $h(n)$ is an *admissible* heuristic.
- Here, an admissible heuristic is one that never *overestimates* the cost to the goal
  - Example: the Euclidean distance.

Motion Planning: A* Search

- Search example: Iteration 1

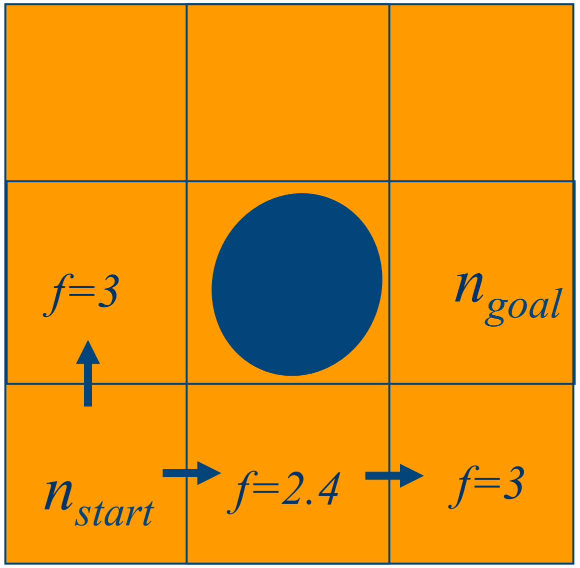
$$Fringe\ set = \{f_1 = 2.4, f_2 = 3\}$$



Motion Planning: A* Search

- Search example: Iteration 2

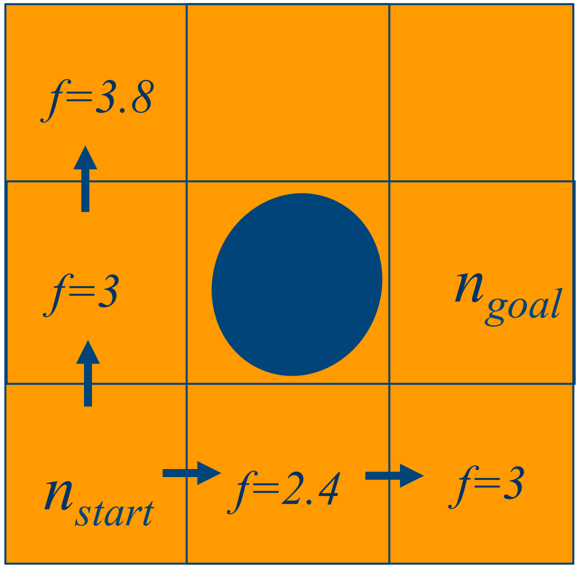
$$Fringe\ set = \{f_2 = 3, f_3 = 3\}$$



Motion Planning: A* Search

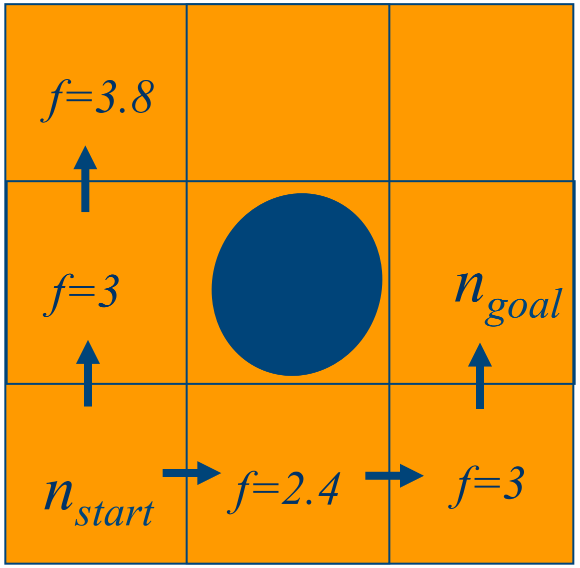
- Search example: Iteration 3

$$Fringe\ set = \{f_3 = 3, f_4 = 3.8\}$$



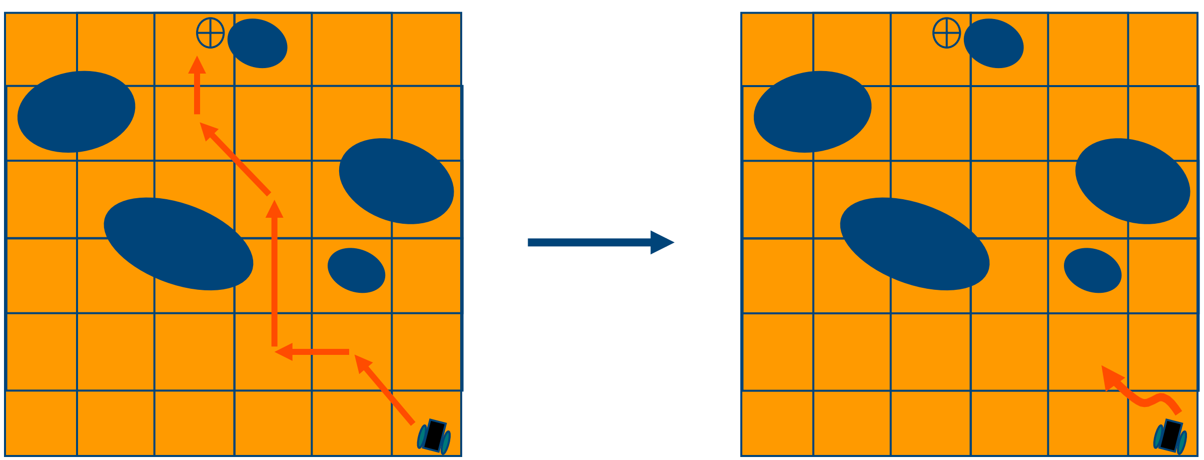
Motion Planning: A* Search

- Search example: Iteration 4



Motion Planning: Final Note

- A robot is often implemented with two planners:
  - Global Planner: A planner that plans an optimal plan with respect to some course discretization of a map.
  - Local Planner: A reactive planner for obstacle avoidance and kinematic consideratations.



Motion Planning: Final Note

- A * is often used as a global planner
- Planner that considers kinematic/dynamic constraints is used for local planning.