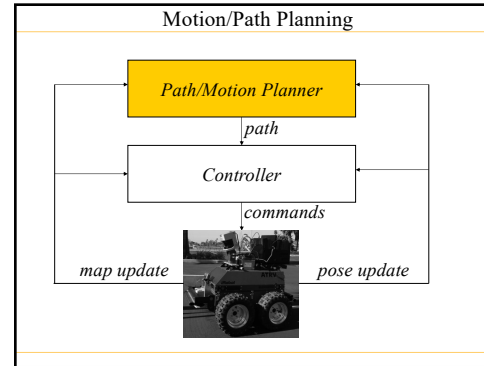


**CSE-571**

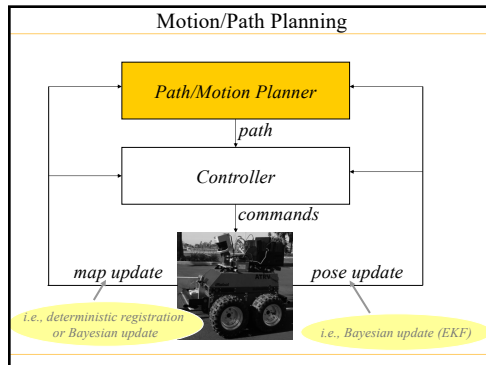
**Deterministic Path Planning**

Courtesy of Maxim Likhachev  
Carnegie Mellon University

1



2



3

Uncertainty and Planning

- Uncertainty can be in:
  - prior environment (i.e., door is open or closed)
  - execution (i.e., robot may slip)
  - sensing environment (i.e., seems like an obstacle but not sure)
  - pose
- Planning approaches:
  - deterministic planning:
    - assume some (i.e., most likely) environment, execution, pose
    - plan a single least-cost trajectory under this assumption
    - re-plan as new information arrives
  - planning under uncertainty:
    - associate probabilities with some elements or everything
    - plan a policy that dictates what to do for each outcome of sensing/action and minimizes expected cost-to-goal
    - re-plan if unaccounted events happen

4

### Uncertainty and Planning

- Uncertainty can be in:
  - prior environment (i.e., door is open or closed)
  - execution (i.e., robot may slip)
  - sensing environment (i.e., seems like an obstacle but not sure)
  - pose
- Planning approaches:
  - deterministic planning:
    - assume some (i.e., most likely) environment, *re-plan every time sensory data arrives or robot deviates off its path*
    - plan a single least-cost trajectory under this assumption
    - re-plan as new information arrives *re-planning needs to be FAST*
  - planning under uncertainty:
    - associate probabilities with some elements or everything
    - plan a policy that dictates what to do for each outcome of sensing/action and minimizes expected cost-to-goal
    - re-plan if unaccounted events happen

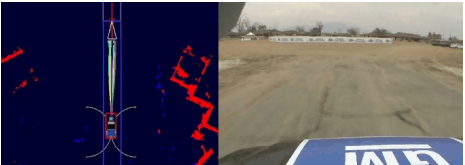
5

### Uncertainty and Planning

- Uncertainty can be in:
  - prior environment (i.e., door is open or closed)
  - execution (i.e., robot may slip)
  - sensing environment (i.e., seems like an obstacle but not sure)
  - pose
- Planning approaches:
  - deterministic planning:
    - assume some (i.e., most likely) environment, execution, pose
    - plan a single least-cost trajectory under this assumption
    - re-plan as new information arrives
  - planning under uncertainty:
    - associate probabilities with some elements or everything
    - plan a policy that dictates what to do for each outcome of sensing/action and minimizes expected cost-to-goal
    - re-plan if unaccounted events happen *computationally MUCH harder*

6

### Example



*Urban Challenge Race, CMU team, planning with Anytime D\**

7

### Outline

- Deterministic planning
  - constructing a graph
  - search with A\*
  - search with D\*

8

### Outline

- Deterministic planning
  - constructing a graph
  - search with A\*
  - search with D\*

9

### Planning via Cell Decomposition

- Approximate Cell Decomposition:
  - overlay uniform grid over the C-space (discretize)

10

### Planning via Cell Decomposition

- Approximate Cell Decomposition:
  - construct a graph and search it for a least-cost path

11

### Planning via Cell Decomposition

- Approximate Cell Decomposition:
  - construct a graph and search it for a least-cost path

12

### Planning via Cell Decomposition

- Approximate Cell Decomposition:
  - construct a graph and search it for a least-cost path
  - VERY popular due to its simplicity and representation of arbitrary obstacles
  - Problem: transitions difficult to execute on non-holonomic robots

discretize

13

### Planning via Cell Decomposition

- Graph construction:
  - lattice graph

outcome state is the center of the corresponding cell

each transition is feasible (constructed beforehand)

action template

replicate it online

$C(s_1, s_2) = 5$   
 $C(s_2, s_3) = 5$   
 $C(s_3, s_4) = 5$   
 $C(s_4, s_5) = 5$   
 $C(s_5, s_6) = 5$   
 $C(s_6, s_7) = 5$   
 $C(s_7, s_8) = 5$   
 $C(s_8, s_9) = 5$   
 $C(s_9, s_{10}) = 5$   
 $C(s_{10}, s_{11}) = 5$   
 $C(s_{11}, s_{12}) = 5$   
 $C(s_{12}, s_{13}) = 5$   
 $C(s_1, s_3) = 0$   
 $C(s_1, s_4) = 5$   
 $C(s_1, s_5) = 100$   
 $C(s_1, s_6) = 5$

14

### Planning via Cell Decomposition

- Graph construction:
  - lattice graph
  - pros: sparse graph, feasible paths
  - cons: possible incompleteness

action template

replicate it online

$C(s_1, s_2) = 5$   
 $C(s_2, s_3) = 5$   
 $C(s_3, s_4) = 5$   
 $C(s_4, s_5) = 5$   
 $C(s_5, s_6) = 5$   
 $C(s_6, s_7) = 5$   
 $C(s_7, s_8) = 5$   
 $C(s_8, s_9) = 5$   
 $C(s_9, s_{10}) = 5$   
 $C(s_{10}, s_{11}) = 5$   
 $C(s_{11}, s_{12}) = 5$   
 $C(s_{12}, s_{13}) = 5$   
 $C(s_1, s_3) = 0$   
 $C(s_1, s_4) = 5$   
 $C(s_1, s_5) = 100$   
 $C(s_1, s_6) = 5$

15

### Outline

- Deterministic planning
  - constructing a graph
  - search with A\*
  - search with D\*
- Planning under uncertainty
  - Markov Decision Processes (MDP)
  - Partially Observable Decision Processes (POMDP)

16





### A\* Search

- Computes optimal g-values for relevant states

**ComputePath function**  
while( $s_{goal}$  is not expanded)  
remove  $s$  with the smallest  $[f(s) = g(s)+h(s)]$  from *OPEN*;  
insert  $s$  into *CLOSED*;  
for every successor  $s'$  of  $s$  such that  $s'$  not in *CLOSED*  
if  $g(s') > g(s) + c(s,s')$   
 $g(s') = g(s) + c(s,s')$ ;  
insert  $s'$  into *OPEN*;

$CLOSED = \{s_{start}, s_2, s_1, s_4, s_{goal}\}$   
 $OPEN = \{s_3\}$   
done

25

### A\* Search

- Computes optimal g-values for relevant states

**ComputePath function**  
while( $s_{goal}$  is not expanded)  
remove  $s$  with the smallest  $[f(s) = g(s)+h(s)]$  from *OPEN*;  
insert  $s$  into *CLOSED*;  
for every successor  $s'$  of  $s$  such that  $s'$  not in *CLOSED*  
if  $g(s') > g(s) + c(s,s')$   
 $g(s') = g(s) + c(s,s')$ ;  
insert  $s'$  into *OPEN*;

*for every expanded state  $g(s)$  is optimal  
for every other state  $g(s)$  is an upper bound  
we can now compute a least-cost path*

26

### A\* Search

- Computes optimal g-values for relevant states

**ComputePath function**  
while( $s_{goal}$  is not expanded)  
remove  $s$  with the smallest  $[f(s) = g(s)+h(s)]$  from *OPEN*;  
insert  $s$  into *CLOSED*;  
for every successor  $s'$  of  $s$  such that  $s'$  not in *CLOSED*  
if  $g(s') > g(s) + c(s,s')$   
 $g(s') = g(s) + c(s,s')$ ;  
insert  $s'$  into *OPEN*;

*for every expanded state  $g(s)$  is optimal  
for every other state  $g(s)$  is an upper bound  
we can now compute a least-cost path*

27

### A\* Search

- Is guaranteed to return an optimal path (in fact, for every expanded state) – optimal in terms of the solution
- Performs provably minimal number of state expansions required to guarantee optimality – optimal in terms of the computations

28

### A\* Search

- Is guaranteed to return an optimal path (in fact, for every expanded state) – *helps with robot deviating off its path if we search with A\* backwards (from goal to start)*
- Performs provably minimal number of state expansions required to guarantee optimality – optimal in terms of the computations

29

### Effect of the Heuristic Function

- A\* Search: expands states in the order of  $f = g+h$  values

30

### Effect of the Heuristic Function

- A\* Search: expands states in the order of  $f = g+h$  values

*for large problems this results in A\* quickly running out of memory (memory:  $O(n)$ )*

31

### Effect of the Heuristic Function

- Weighted A\* Search: expands states in the order of  $f = g + \epsilon h$  values,  $\epsilon > 1$  = bias towards states that are closer to goal

*solution is always  $\epsilon$ -suboptimal:  $cost(solution) \leq \epsilon \cdot cost(optimal solution)$*

32



### Effect of the Heuristic Function

- Weighted A\* Search: expands states in the order of  $f = g + \epsilon h$  values,  $\epsilon > 1$  = bias towards states that are closer to goal

20DOF simulated robotic arm  
state-space size: over  $10^{26}$  states

planning with ARA\* (anytime version of weighted A\*)

33

### Adaptive Real-Time A\*

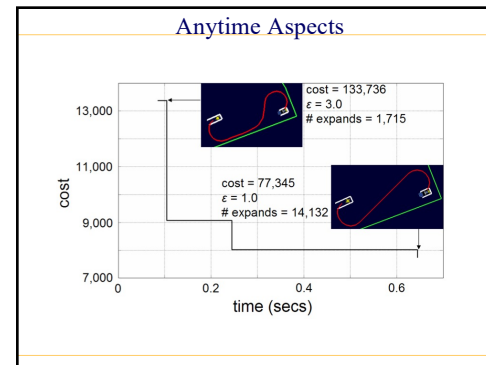
$\epsilon = 2.5$        $\epsilon = 1.5$        $\epsilon = 1.0$  (optimal search)

initial search ( $\epsilon = 2.5$ )    second search ( $\epsilon = 1.5$ )    third search ( $\epsilon = 1.0$ )

34

### Anytime Aspects

35



36



### Motivation for Incremental Version of A\*

- Reuse state values from previous searches
  - cost of least-cost paths to  $s_{goal}$  initially

cost of least-cost paths to  $s_{goal}$  after the door turns out to be closed

41

### Motivation for Incremental Version of A\*

- Reuse state values from previous searches
  - cost of least-cost paths to  $s_{goal}$  initially

cost of least-cost paths to  $s_{goal}$

42

### Motivation for Incremental Version of A\*

- Reuse state values from previous searches
  - cost of least-cost paths to  $s_{goal}$  initially

cost of least-cost paths to  $s_{s}$

43

### Motivation for Incremental Version of A\*

- Reuse state values from previous searches
  - cost of least-cost paths to  $s_{goal}$  initially

cost of least-cost paths to  $s_{s}$

44

### Incremental Version of A\*

- Reuse state values from previous searches

*initial search by backwards A\**

*initial search by D\* Lite*

*second search by backwards A\**

*second search by D\* Lite*

45

### Searching the Graph

- Incremental behavior of Anytime D\*:

*initial path*

*a path after re-planning*

46

### Searching the Graph

- Performance of Anytime D\* depends strongly on heuristics  $h(s)$ : estimates of cost-to-goal

should be consistent and admissible (never overestimate cost-to-goal)

$s = (x, y, \theta, v)$

$h(s)$

$S_{goal}$

47

### Searching the Graph

- In our planner:  $h(s) = \max(h_{mech}(s), h_{env}(s))$ , where
  - $h_{mech}(s)$  – mechanism-constrained heuristic
  - $h_{env}(s)$  – environment-constrained heuristic

$h_{mech}(s)$  – considers only dynamics constraints and ignores environment

$h_{env}(s)$  – considers only environment constraints and ignores dynamics

48

### Searching the Graph

- In our planner:  $h(s) = \max(h_{mech}(s), h_{env}(s))$ , where
  - $h_{mech}(s)$  – mechanism-constrained heuristic
  - $h_{env}(s)$  – environment-constrained heuristic

$h_{mech}(s)$  – considers only dynamics constraints and ignores environment

$h_{env}(s)$  – considers only environment constraints and ignores dynamics

pre-computed as a table lookup for high-res. lattice

computed online by running a 2D A\* with late termination

49

49

### Heuristics

heuristic	states expanded	time (secs)
$h$	2,019	0.06
$h_{2D}$	26,108	1.30
$h_{fsh}$	124,794	3.49

50

50

### Example, again

Urban Challenge Race, CMU team, planning with Anytime D\*

51

51

### Summary

- Deterministic planning
  - constructing a graph
  - search with A\*
  - search with D\*
- Planning under uncertainty
  - Markov Decision Processes (MDP)
  - Partially Observable Decision Processes (POMDP)

used a lot in real-time

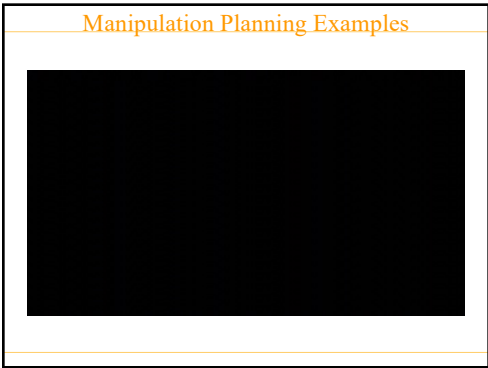
think twice before trying to use it in real-time

think three or four times before trying to use it in real-time

Many useful approximate solvers for MDP/POMDP exist!!

52

52



53