CSE-571

Deterministic Path Planning in Robotics

Courtesy of Maxim Likhachev

University of Pennsylvania

Motion/Path Planning

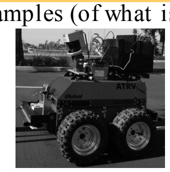
• Task:
find a feasible (and cost-minimal) path/motion from the current configuration of the robot to its goal configuration (or one of its goal configurations)

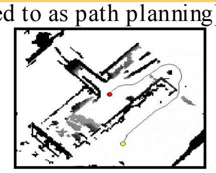
• Two types of constraints:
environmental constraints (e.g., obstacles)
dynamics/kinematics constraints of the robot

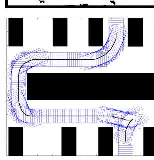
• Generated motion/path should (objective):
be any feasible path
minimize cost such as distance, time, energy, risk, …

Motion/Path Planning

Examples (of what is usually referred to as path planning):

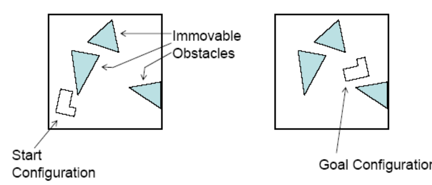



Motion/Path Planning

Examples (of what is usually referred to as motion planning):

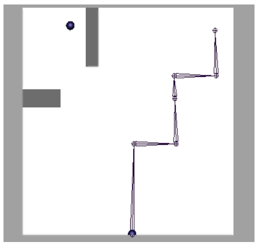


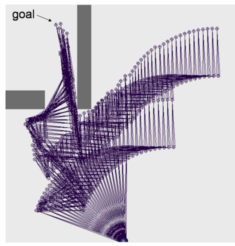
Piano Movers' problem

the example above is borrowed from www.cs.cmu.edu/~awm/tutorials

Motion/Path Planning

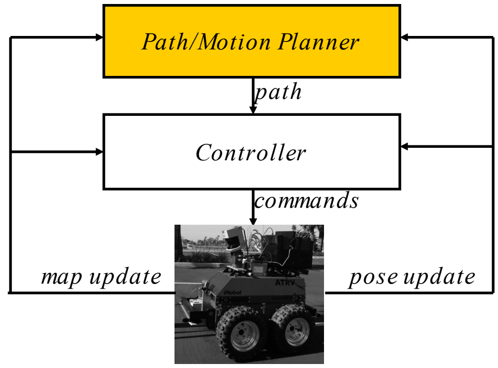
Examples (of what is usually referred to as motion planning)



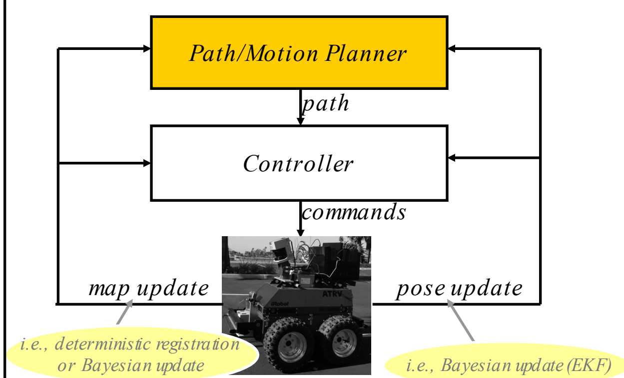


Planned motion for a 6DOF robot arm

Motion/Path Planning



Motion/Path Planning



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

Uncertainty and Planning

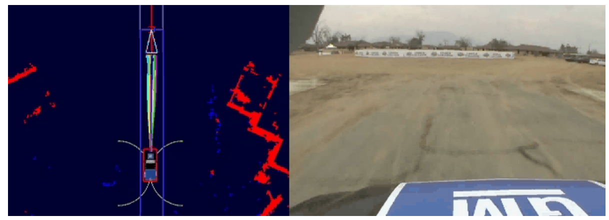
- Uncertainty can be in:
  - prior environment (i.e., door is open or closed)
  - execution (i.e., robot may slip)
  - sensing environment (i.e., seems like an obstacle but not sure)
  - pose
- Planning approaches:
  - deterministic planning:
    - assume some (i.e., most likely) environm
    - plan a single least-cost trajectory under th
    - re-plan as new information arrives

re-plan every time sensory data arrives or robot deviates off its path

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

Example



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

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

Outline

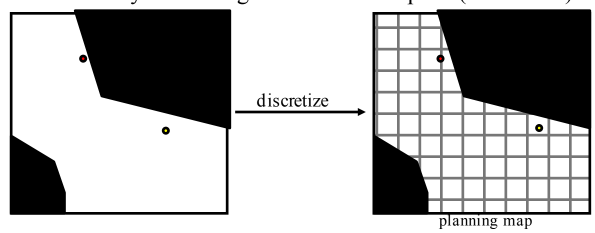
- Deterministic planning
  - constructing a graph
  - search with A*
  - search with D*

Outline

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

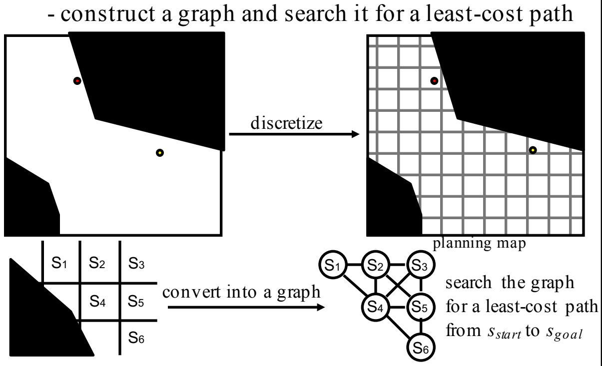
Planning via Cell Decomposition

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



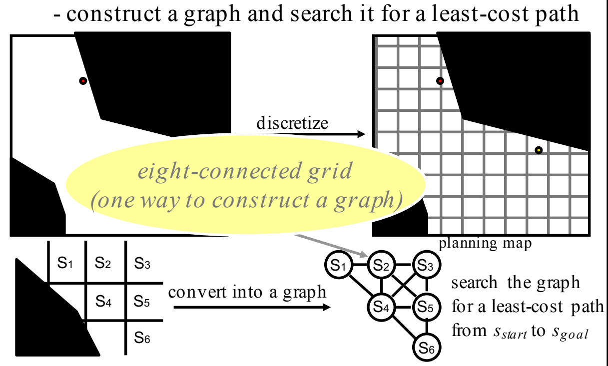
Planning via Cell Decomposition

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



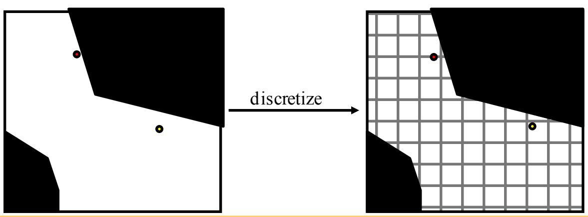
Planning via Cell Decomposition

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



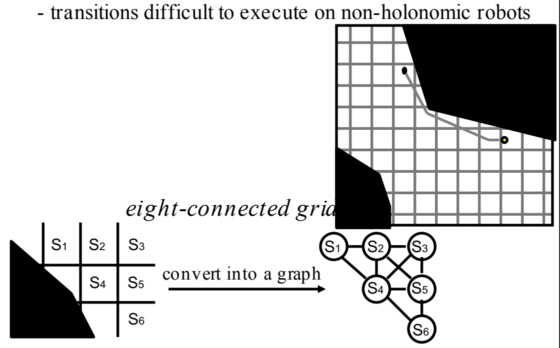
Planning via Cell Decomposition

- 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



Planning via Cell Decomposition

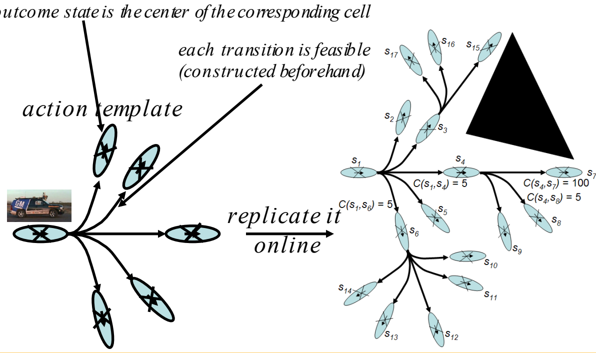
- Graph construction:
  - major problem with paths on the grid:
    - transitions difficult to execute on non-holonomic robots



Planning via Cell Decomposition

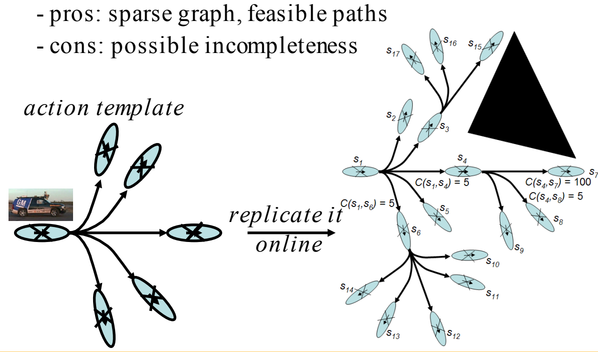
- Graph construction:
  - lattice graph

outcome state is the center of the corresponding cell



Planning via Cell Decomposition

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



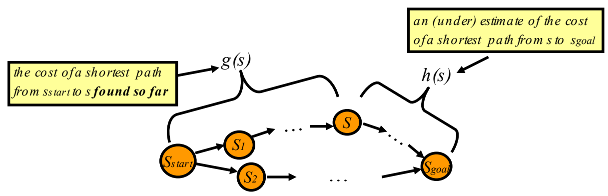
Outline

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

A* Search

- Computes optimal g-values for relevant states

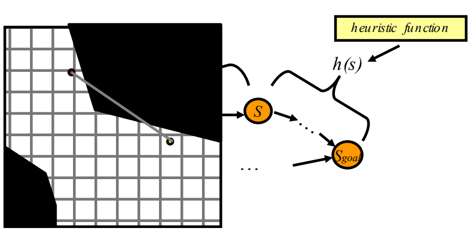
at any point of time:



A* Search

- Computes optimal g-values for relevant states

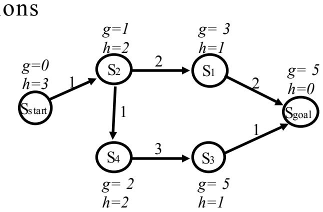
at any point of time:



one popular heuristic function – Euclidean distance

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

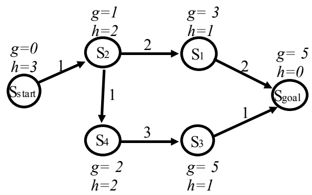


A* Search

- Is guaranteed to return an optimal path (in fact, for every expanded state) – ion

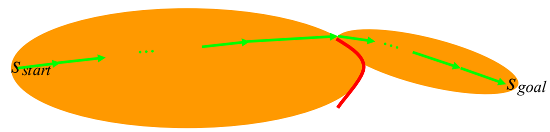
helps with robot deviating off its path if we search with A backwards (from goal to start)*

- Performs provably minimal number of state expansions required to guarantee optimality – optimal in terms of the computations



Effect of the Heuristic Function

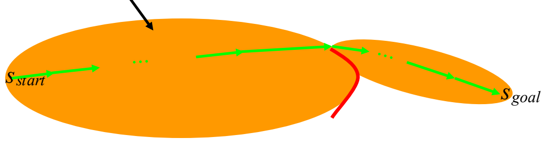
- A* Search: expands states in the order of $f = g+h$ values



Effect of the Heuristic Function

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

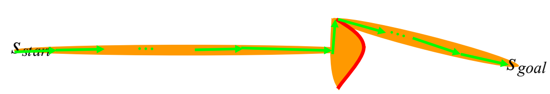
for large problems this results in A quickly running out of memory (memory: O(n))*



Effect of the Heuristic Function

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

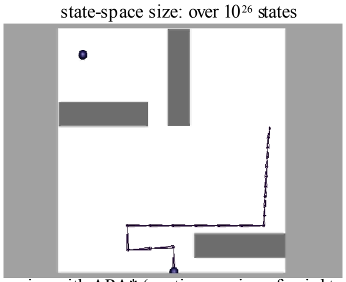
solution is always ε-suboptimal: cost(solution) ≤ ε·cost(optimal solution)



Effect of the Heuristic Function

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

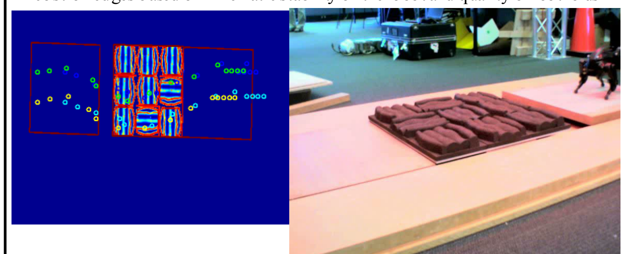
20DOF simulated robotic arm
state-space size: over 10^{26} states



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

Effect of the Heuristic Function

- planning in 8D (<x,y> for each foothold)
- heuristic is Euclidean distance from the center of the body to the goal location
- cost of edges based on kinematic stability of the robot and quality of footholds



planning with R* (randomized version of weighted A*)

joint work with Subhrajit Bhattacharya, Jon Bohren, Sachin Chitta, Daniel D. Lee, Aleksandr Kushleyev, Paul Vernaza

Outline

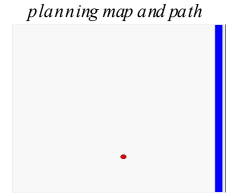
- Deterministic planning
  - constructing a graph
  - search with A*
  - search with D*

Incremental version of A* (D*/D* Lite)

- Robot needs to re-plan whenever
  - new information arrives (partially-known environments or/and dynamic environments)
  - robot deviates off its path

ATRV navigating initially-unknown environment

planning map and path

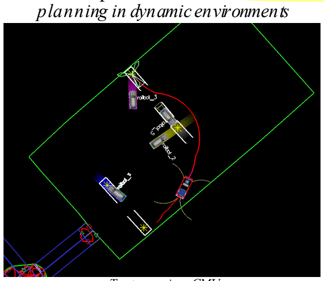


Incremental version of A* (D*/D* Lite)

- Robot needs to re-plan whenever
  - new information arrives (partially-known environments or/and dynamic environments)
  - robot deviates off its path

incremental planning (re-planning): reuse of previous planning efforts

planning in dynamic environments

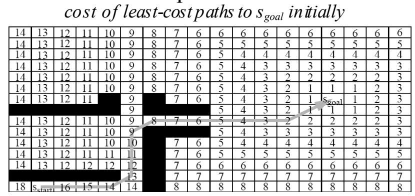


Tartanracing, CMU

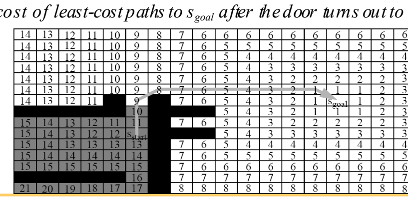
Motivation for Incremental Version of A*

- Reuse state values from previous searches

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



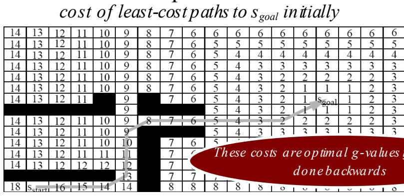
cost of least-cost paths to s_{goal} after the door turns out to be closed



Motivation for Incremental Version of A*

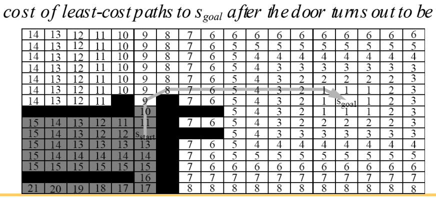
- Reuse state values from previous searches

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



These costs are optimal g-values if search is done backwards

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



Motivation for Incremental Version of A*

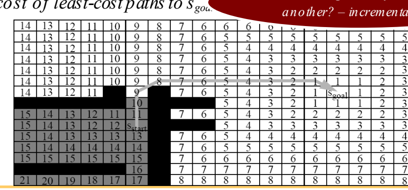
- Reuse state values from previous searches

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

These costs are optimal g-values if search is done backwards

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

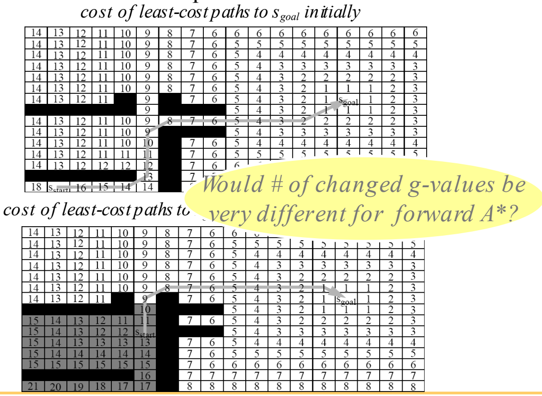
*How to reuse these g-values from one search to another? – incremental A**



Motivation for Incremental Version of A*

- Reuse state values from previous searches

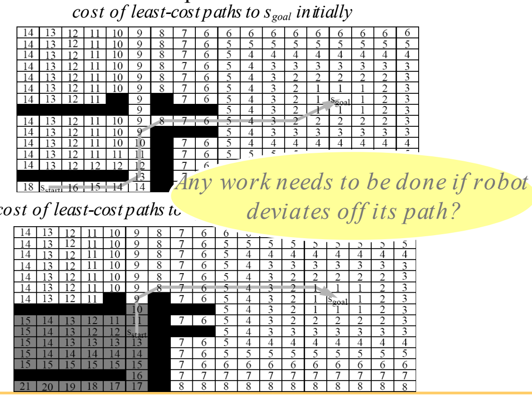
cost of least-cost paths to s_{goal} initially



Motivation for Incremental Version of A*

- Reuse state values from previous searches

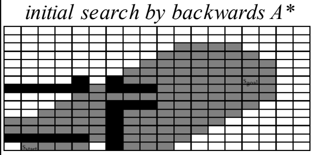
cost of least-cost paths to s_{goal} initially



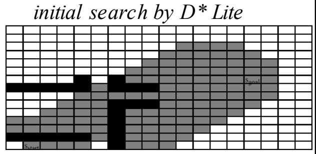
Incremental Version of A*

- Reuse state values from previous searches

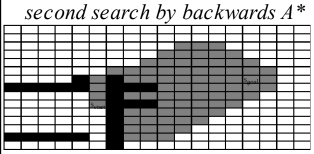
*initial search by backwards A**



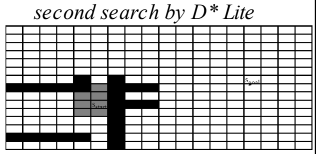
initial search by D Lite*



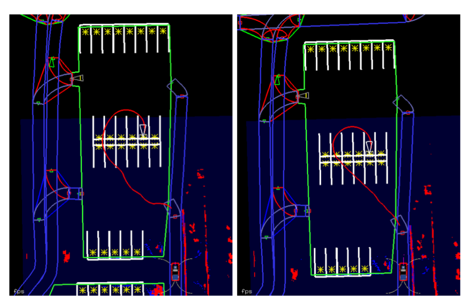
*second search by backwards A**



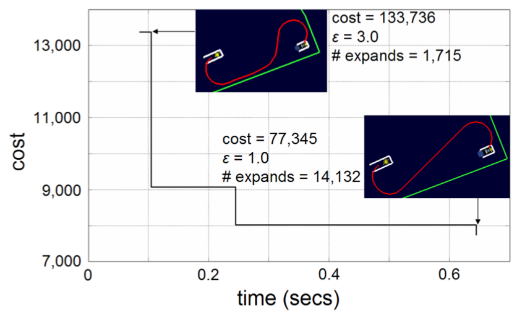
second search by D Lite*



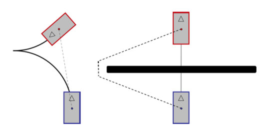
Anytime Aspects

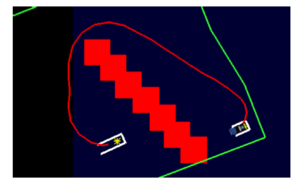


Anytime Aspects



Heuristics





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

Summary

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

used a lot in real-time

think twice before trying to use it in real-time

- Planning under uncertainty
  - Markov Decision Processes (MDP)
  - Partially Observable Decision Processes (POMDP)

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

Many useful approximate solvers for MDP/POMDP exist!!