15-381: AI – Fall 2009
Probabilistic Path Planning
(First review planning as search)

Manuela Veloso

See J.Bruce, M. Veloso, "Real-Time Randomized Path Planning for Robot Navigation", IROS'02
www.cs.cmu.edu/~mmv/papers/02iros.pdf

Carnegie Mellon University
Computer Science Department

Outline

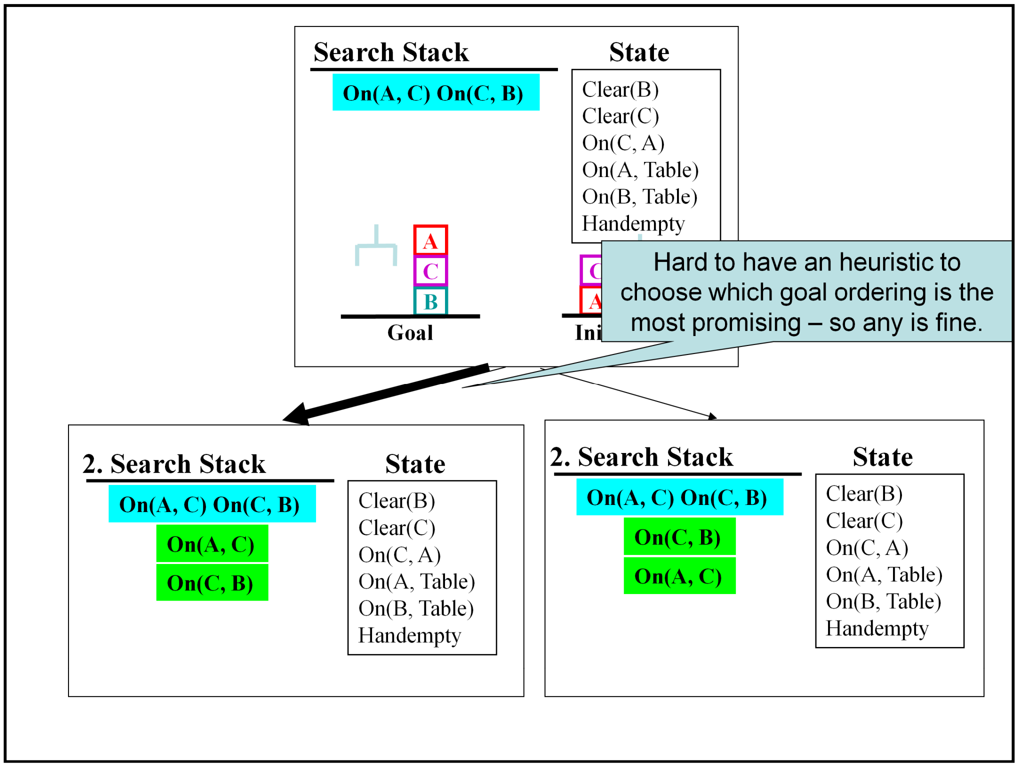
- Review: Classical planning as search
  - Choice points
- Probabilistic robot path planning

GPS - Classical Planning
Means-Ends Analysis – "reduce differences"

- **Choose** a goal (stack, set…)
  - one that is not true in state

3

Focus on a single goal that you want to reach first.



Here you have a conjunction of things that need to be true

Often we just flip a coin to decide which goal we will go for first.

GPS - Classical Planning
Means-Ends Analysis – "reduce differences"

- **Choose** a goal (stack, set…)
  - one that is not true in state
- **Choose** an action that **adds** the goal

Search Stack

On(A, C) On(C, B)

On(A, C)

Put_Block(C, B)

Holding(C) Clear(B)

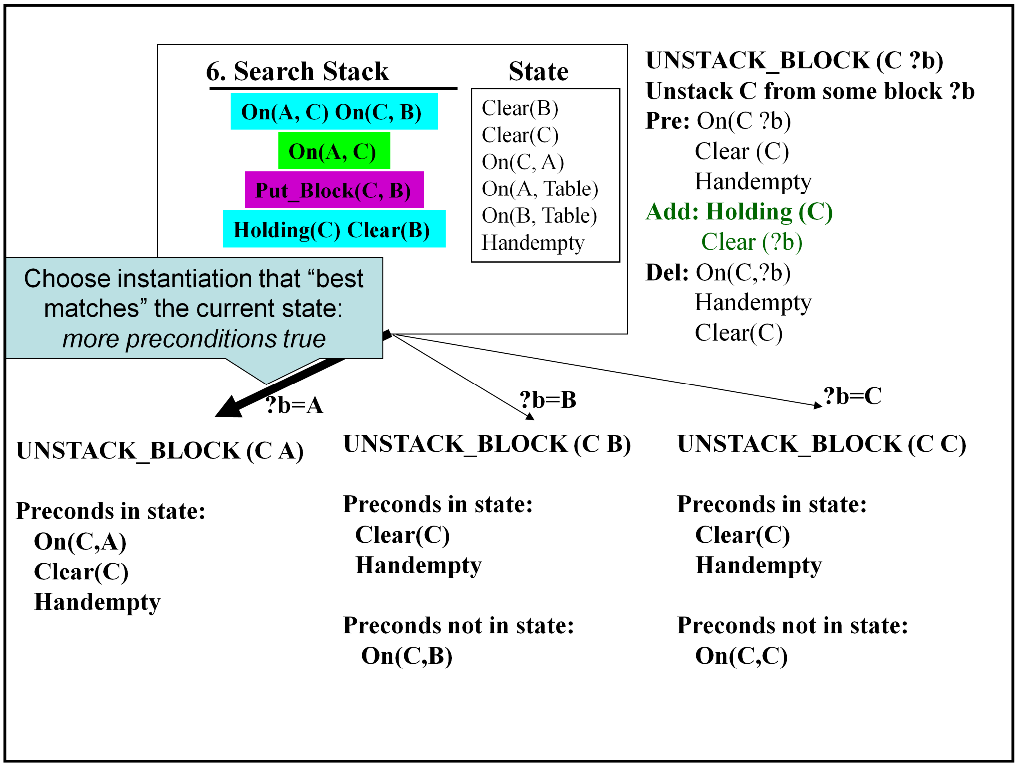
State

Clear(B)
Clear(C)
On(C, A)
On(A, Table)
On(B, Table)
Handempty

Choose action that "best matches" the current state: C is not on the table, so choose UNSTACK.

UNSTACK_BLOCK (C ?b)
Unstack C from some block ?b
Pre: On(C ?b)
Clear (C)
Handempty
Add: Holding (C)
Clear (?b)
Del: On(C,?b)
Handempty
Clear(C)

PICK_UP_BLOCK (C)
Pick up C from table
Pre: On(C, Table)
Clear (C)
Handempty
Add: Holding (C)
Del: On (C, Table)
Handempty
Clear(C)



3 choices of what can be ?b

Only one satisfies the pre-conditions. Still, there are 3 choices, even though 2 will immediately fail.

GPS - Classical Planning
Means-Ends Analysis – “reduce differences”

- Choose a goal (stack, set…)
  - one that is not true in state
- Choose an action that **adds** the goal
  - one that has more preconditions true in state
- If the action is applicable in state
  - apply action, change state, add action to solution plan
- If the action is not applicable, “subgoal,”i.e.:
  - Add preconditions of action as goals

Classical Planning

- Complete state
- Deterministic actions
  - Known preconds, adds, and deletes
- Many algorithms besides GPS
  - All have some advantages and some disadvantages
  - No universal dominance
  - Planning is NP-complete

9

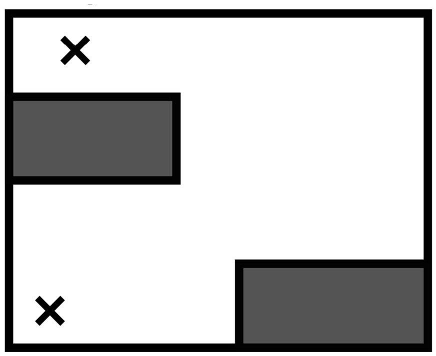
Complete state: you know everything about a state

Every action has deterministic results

Robot Motion Planning

- A mobile robot needs *to navigate:*
  - Navigation is carrying out locomotion primitives to move between points
  - Navigation includes avoiding obstacles.

Robot Path Planning Problem



Initial location
Goal location
Obstacles

11

At its most simple, you go from initial location to the goal location, avoiding obstacles.

Robot Motion Planning

- A mobile robot needs *to navigate:*
  - Navigation is carrying out locomotion primitives to move between points
  - Navigation includes avoiding obstacles.

- We need to define:
  - The state – a model of the environment
  - The actions – a model of the robot's motion primitives

12

Environment Models

- An environment model is composed of
  - Knowledge of the robots location (Localization)
  - Knowledge of the location of obstacles
- Complicating factors
  - Number of obstacles and complexity of geometry
  - Complexity of robot state
  - Error or uncertainty from sensors

13

It is VERY hard for a robot to know where it is. For now lets assume we solved this problem

It is VERY hard for a robot to know where the obstacles are. For now lets assume we solved this problem

Action Models

- Action models
  - Knowledge of how an action affects the environment
    - preconditions and effects….
  - For planning, model must be known *without* executing the action

- Complicating factors
  - Constraints on robot actions
    - Motion (kinematic) constraints (e.g. car-like robots)
    - Bounded velocity and acceleration
  - Dynamics effects at high speeds
  - Error or uncertainty in actions

Deterministic Path Planning

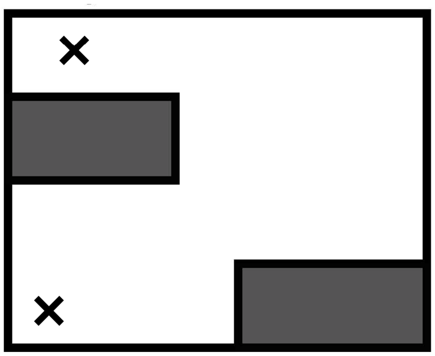
- A*
  - Discretize the state
  - Enumerate a set of actions
  - Search
    - Generate successors of actions
    - Use admissible heuristic
  - Partially successful
- Extensions of A*

15

Discretize the state...perhaps by making state space into a grid

Simple actions: move north/south/east/west. Do higher level planning, and lower level routines can be written to execute the actions. Trade off between branching factor and finding best path when deciding how many actions you have (should you include diagonals, etc?)

Robot Path Planning Problem



Initial location
Goal location
Obstacles

16

Much simpler problem than the real world analogue.

Probabilistic Robot Path Planning

- Continuous state spaces
- Continuous actions

- PRM (Kavraki & many successors)
- RRT (Lavalle & many successors)

17

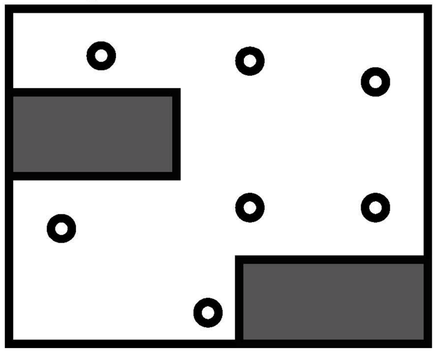
PRM – Probabilistic Road Map

- Separate planning into two stages
  - "Learning" Phase
    - random samples of free configurations (vertices)
    - Attempt to connect pairs of nearby vertices with a local planner
    - if a valid plan is found, add an edge to the graph
  - Query Phase
    - find local connections to graph from initial and goal positions

Local planner can be as simple as just trying to connect nearby vertices with a straight line.

PRM Example – Learning Phase

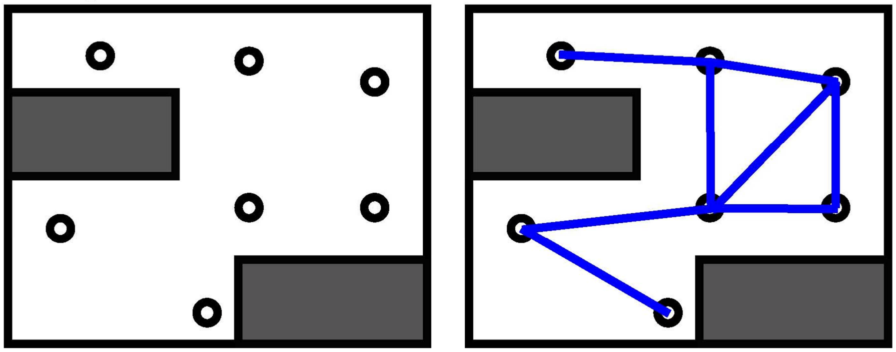
Learning Phase:



19

PRM Example – Learning Phase

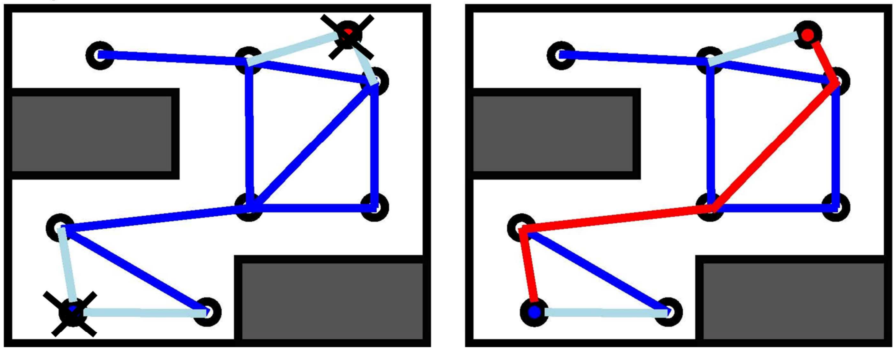
Learning Phase:



20

PRM Example – Query Phase

Query Phase:



21

Connect start and goal states to the graph.

PRM Discussion

- Very interesting approach
  - Continuous spaces
- General learning phase
  - Not targeted at specific initial and goal states
- Not Optimal

22

Instead of discretizing, you are sampling.

Not optimal, or even complete (well, it is probabilistically complete)

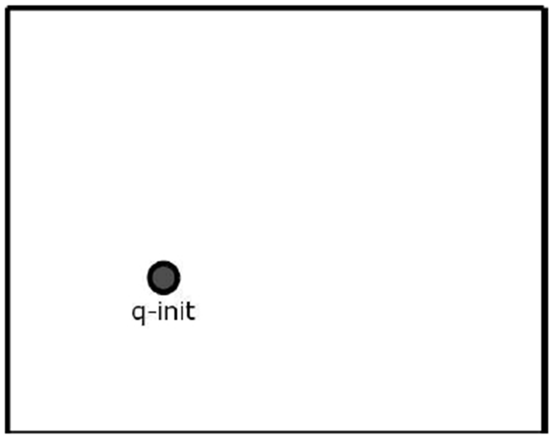
Bad side: you have to make a graph of whole state space, even if you are just moving around in a small area

Rapidly Exploring Random Trees

- RRT
  - Explore continuous spaces efficiently
    - No need for an artificial grid
  - Basic for probabilistically complete planner
- RRT uses random search

Basic RRT Example

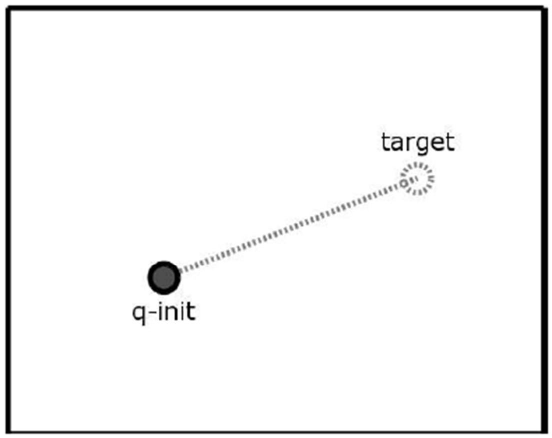
(1) Start with the initial state as the root of a tree



Basic RRT
Just Search, No use of Goal

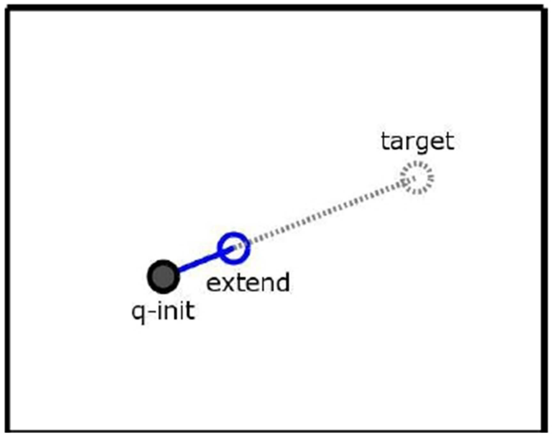
(2) Pick a random state in the environment

(3) Find the closest node in the tree



Basic RRT Search

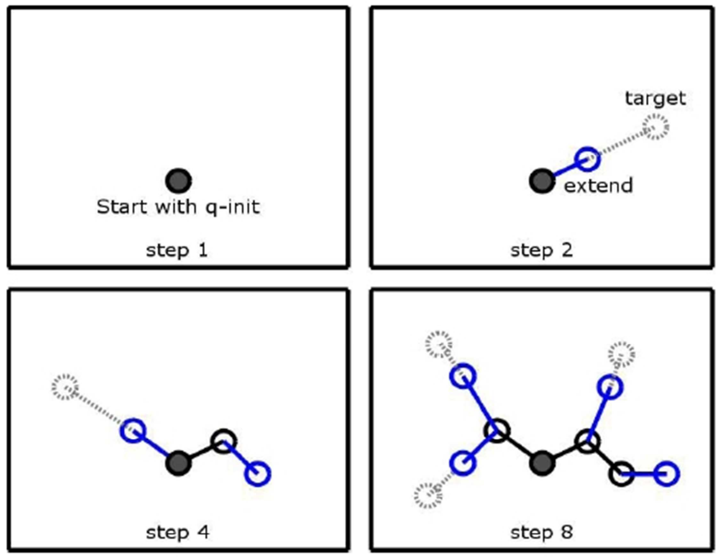
(4) Extend that node toward the target



26

How far do you extend toward the target? Well, you generally just set a step size, and step that far. Sometime RRT algorithms will extend less than the step size if they hit an obstacle.

Basic RRT Search Example

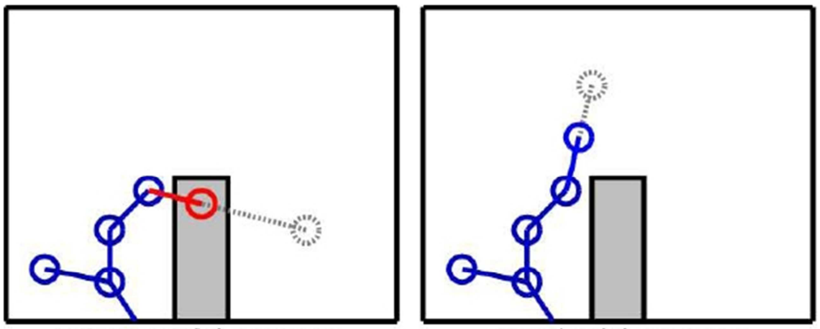


27

No bias (so far, we will add bias in a few slides). We will extend in any direction with equal probability

RRT with Obstacles

- Ignore extensions which hit obstacles
- Resulting tree contains *only* valid paths

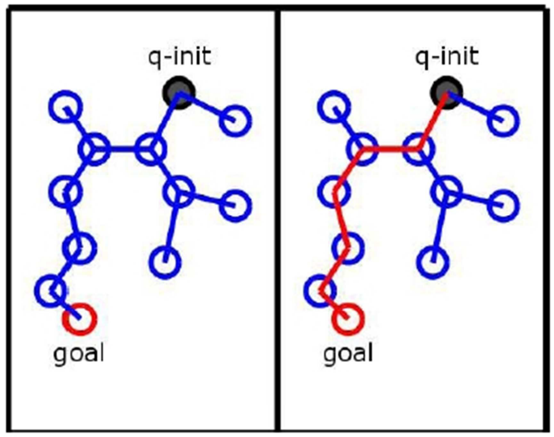


Ignore invalid extension Record valid extension

28

RRT As a Planner

- Once a node of the tree is a *goal*, the plan is the path back up the tree



29

Stop when you find the goal, because you know there is a path! We don't care that it might not be optimal.

RRT-GoalBias Algorithm

1) Start with initial state as root of tree
2) Pick a random target state
   - o Goal configuration with probability p
   - o Random configuration with probability 1-p
3) Find the closest node in the tree
4) Extend the closest node toward the target
5) Goto step 2

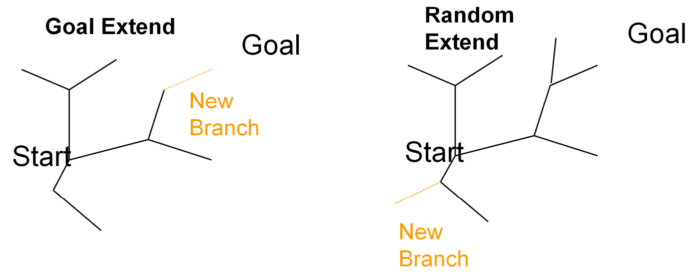
30

Even better: add bias towards the goal!

RRT for Planning

Probability *p* : Extend *closest node* in tree towards goal

Probability *1-p* : Extend closest node towards a random point



31

Planning and Replanning

- Environments and planning
  - Value of p?
- Dynamic environments
- When failure, what to do?

32

Replanning is very important when the environment (other obstacles, goals, etc) is moving. Maybe an obstacle moved into your way. No need to throw away all your previous work, you can reuse it

ERRT – RRT with Replanning

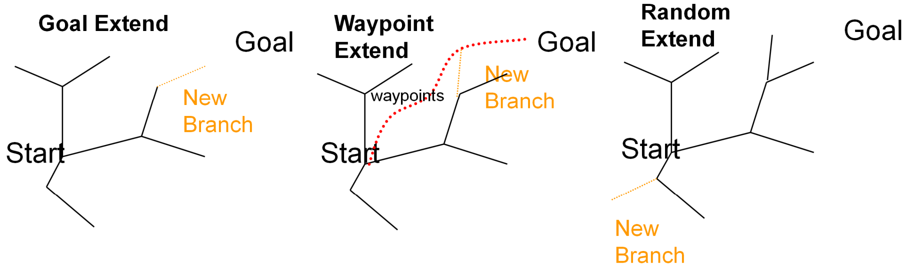
(Bruce & Veloso 2002)

Introduce past path as a bias!

1) Start with initial state as root of tree
2) Pick a random target state
   - Goal configuration with probability p
   - **Random item from waypoint cache with probability q**
   - Random configuration with probability 1-q-p
3) Find the closest node in the tree
4) Extend the closest node toward the target
5) Goto step 2

ERRT: Replanning with Advice

Probability *p* : Extend closest node in tree towards goal
Probability *r* : Extend closest node in tree towards random cache point
Probability *1-p-r* : Extend closest node towards a random point



Summary

- PRM
  - Sampling and search among sample nodes
- Planning with RRT
  - Extend towards random target, or towards goal
  - High p – few known obstacles
  - Low p – many known obstacles
- Replanning with ERRT
  - Extend towards random target, goal, or past plan
  - High q – small dynamics (no state change)
  - Low q – high dynamics (lots of state change)
  - ERRT – bias to use previous plan; but could be any other bias
- RRT and ERRT – probabilistic convergence

35