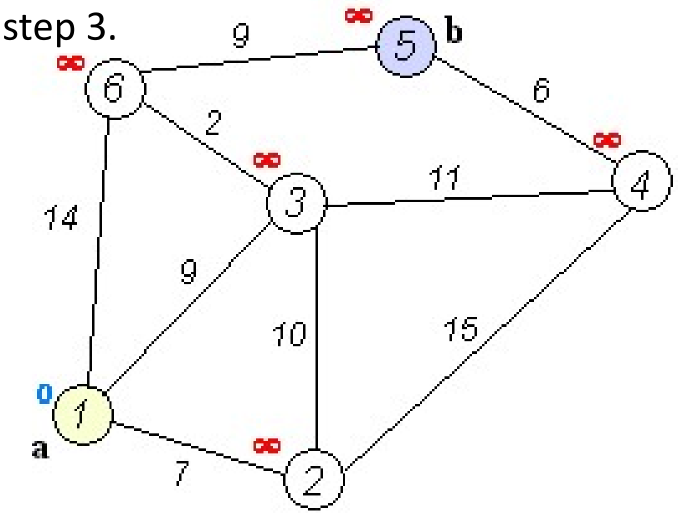
Dijkstra's algorithm

- **A method find shortest distance between nodes in a graph.**
- **Map is represented by weighted graph.**

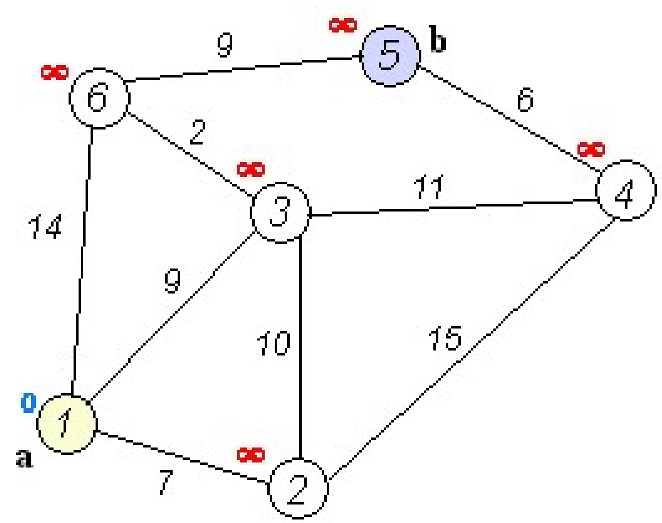
1. Assign all nodes a tentative distance value. Set the start node a value of 0, and all the rest a value of ∞.
2. Keep a list of visited and not visited nodes, call these lists closed and open lists respectively. Add start node to the closed list.
3. For the current node, consider all the unvisited neighbours and calculate g(x) = (distance to the current node) + (distance from current node to the neighbour). Replace their tentative distance, if the calculated one is lower than the current.
4. Once all neighbours are considered add current node to the closed list.
5. If destination node is added to the closed list, terminate.
6. Set the unvisited node with smallest tentative distance as the next current node. Go back to step 3.



A* algorithm

- **Extension to Dijkstra's algorithm.**
- **A method to find the best path on a graph.**
- **Uses best-first search**

Its cost function *f(x)* is a sum of two elements *f(x)=g(x)+h(x)*: the past path-cost function, which is the known distance from the starting node to the current node *x* (denoted *g(x)*), a future path-cost function "heuristic estimate" of the distance from *x* to the goal (usually denoted *h(x)*). For example *h(x)* can be a straight line from the current node to the goal.



A* algorithm

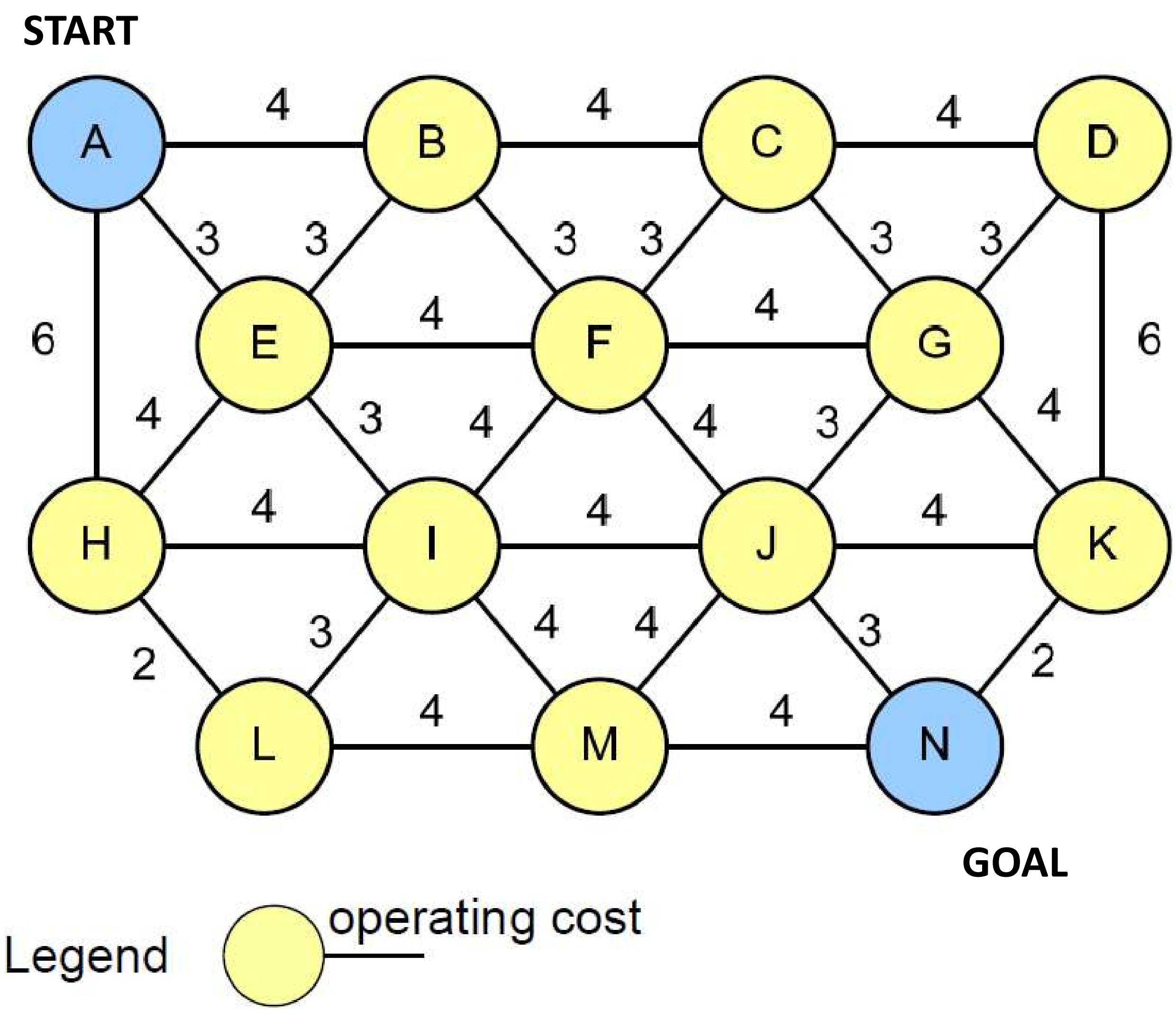
- Cost function f(x) based on actual traversed cost “g(x)” plus an estimate future cost TO GOAL or heuristic “h(x)”.
- Main issue is how to estimate h(x). In a plane its usually direct distance from cell to goal or “Manhattan” distance.
- Use two lists. Closed list: nodes explored. Open list: nodes for expansion.

14	10	14
10	**S**	10
14	10	14

Cost of moving from cell to cell:
A diagonal cell is at a distance of 1.4 blocks (square root of 2)
Others are at a distance of 1 block.

Multiply by 10 to visualize better ie 10 and 14.

A*: Example (1/6)

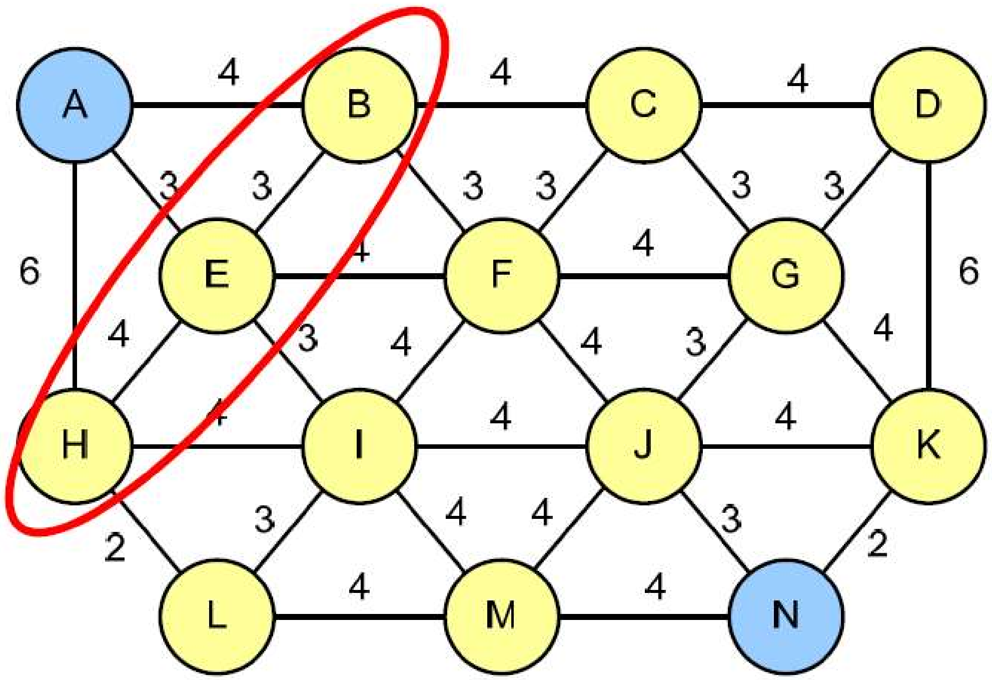


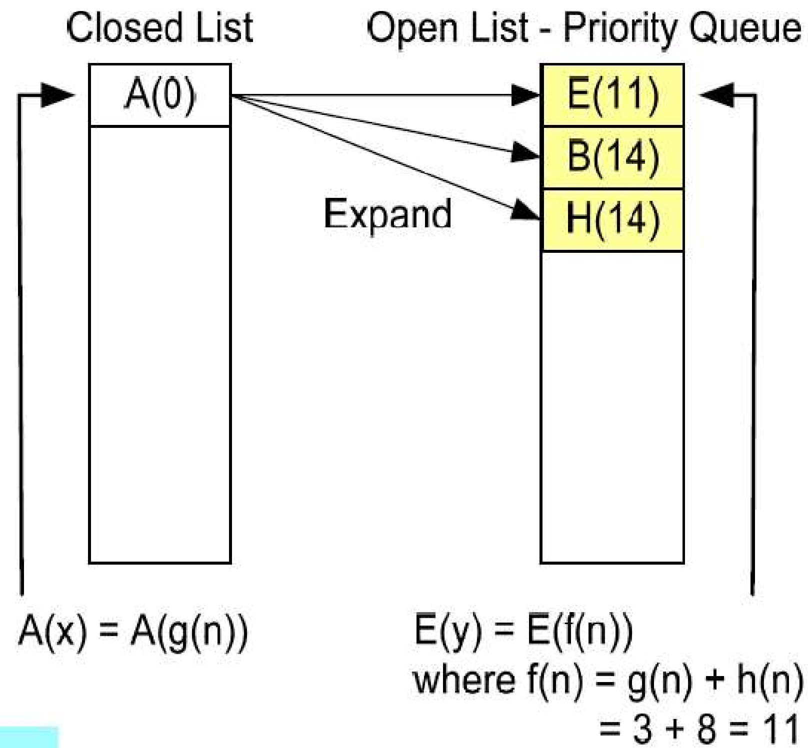
Heuristics

A = 14	H = 8
B = 10	I = 5
C = 8	J = 2
D = 6	K = 2
E = 8	L = 6
F = 7	M = 2
G = 6	N = 0

Slides from H. Choset

A*: Example (2/6)

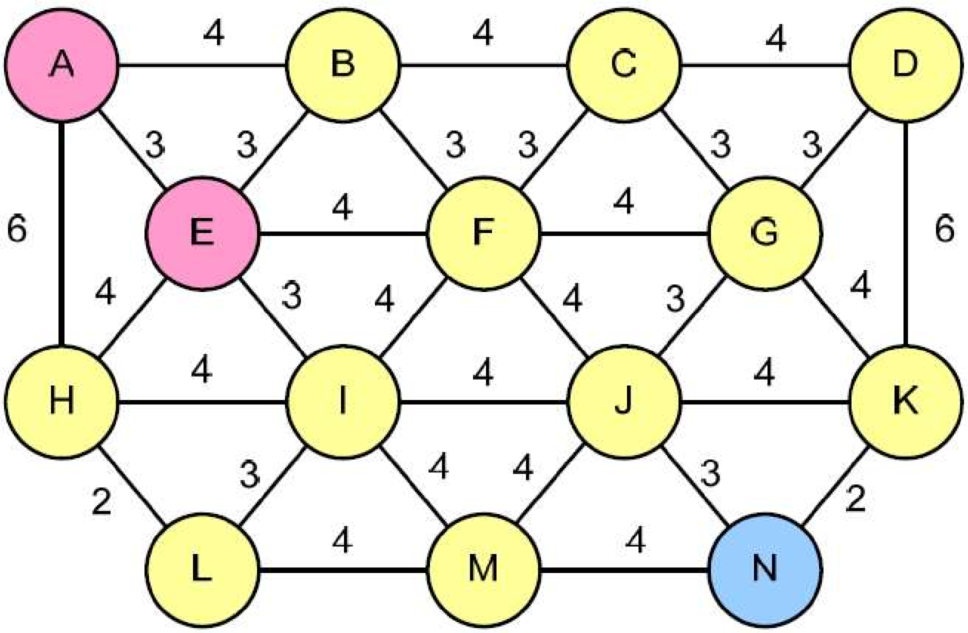


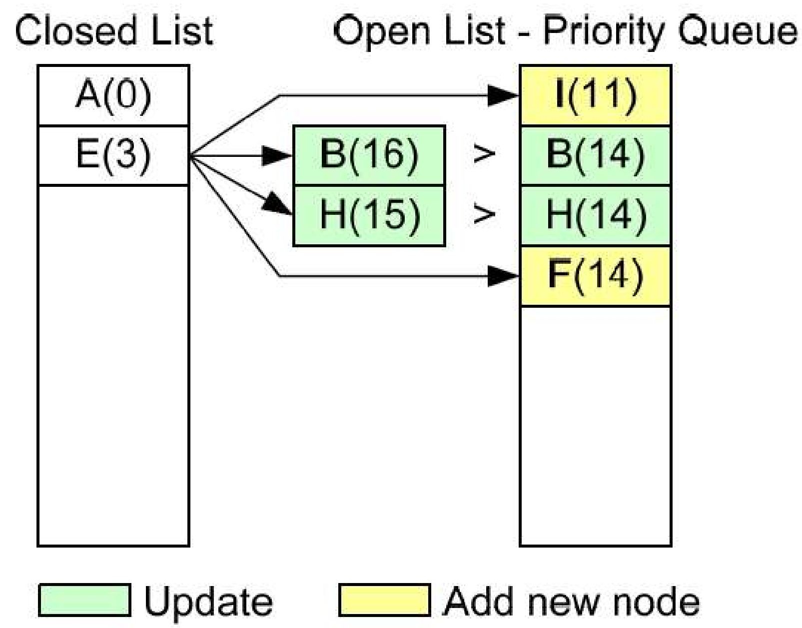


Heuristics
A = 14, B = 10, C = 8, D = 6, E = 8, F = 7, G = 6
H = 8, I = 5, J = 2, K = 2, L = 6, M = 2, N = 0

Slides from H. Choset

A*: Example (3/6)



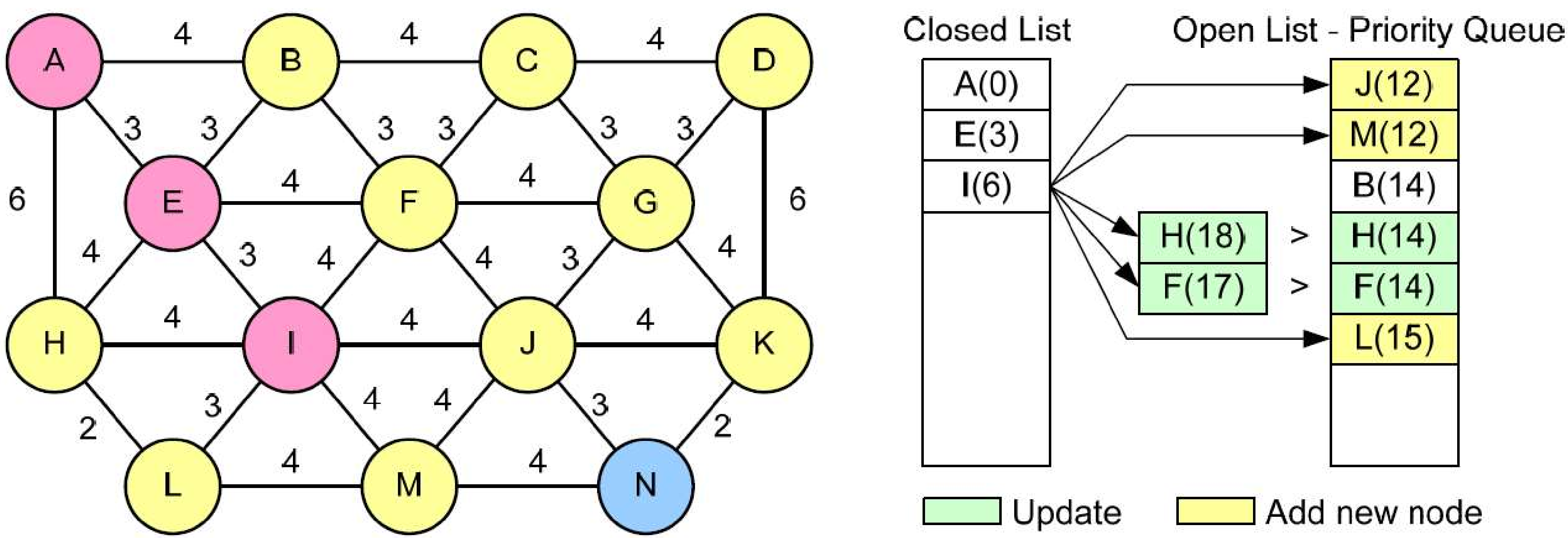


Heuristics
A = 14, B = 10, C = 8, D = 6, E = 8, F = 7, G = 6
H = 8, I = 5, J = 2, K = 2, L = 6, M = 2, N = 0

Since A → B is smaller than A → E → B, the f-cost value of B in an open list needs not be updated

Slides from H. Choset

A*: Example (4/6)



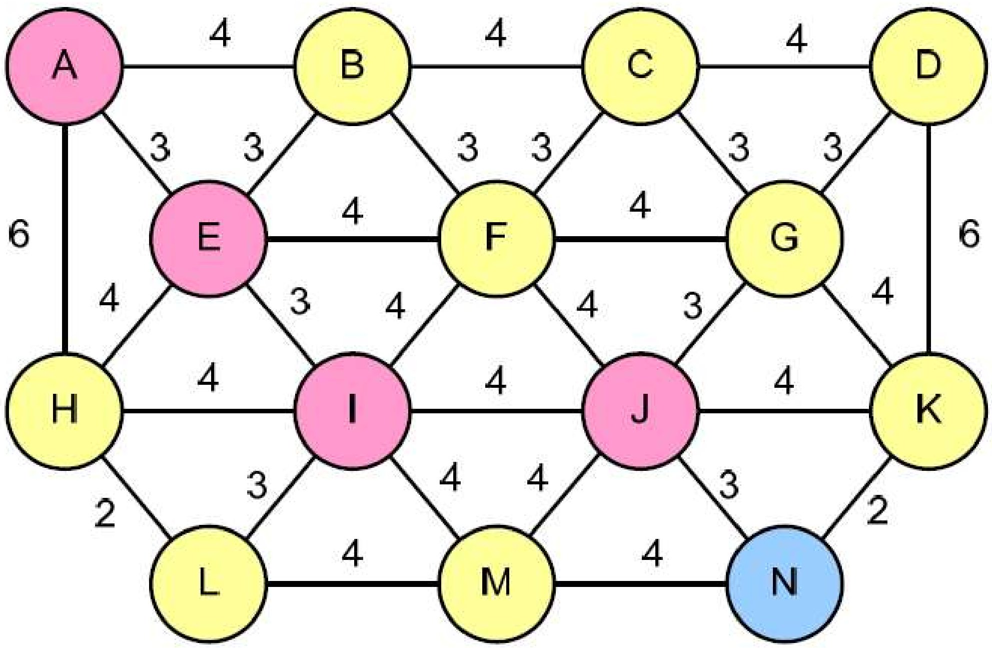
Heuristics

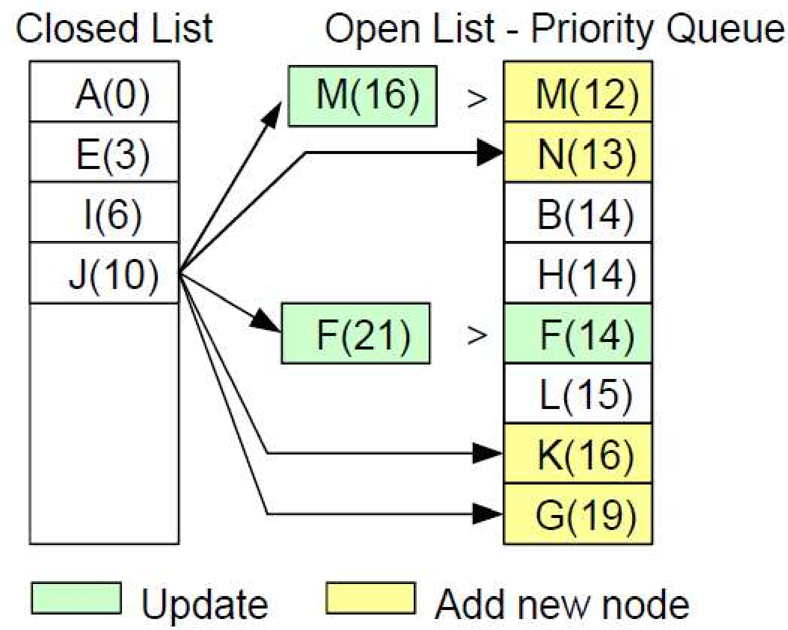
A = 14, B = 10, C = 8, D = 6, E = 8, F = 7, G = 6

H = 8, I = 5, J = 2, K = 2, L = 6, M = 2, N = 0

Slides from H. Choset

A*: Example (5/6)

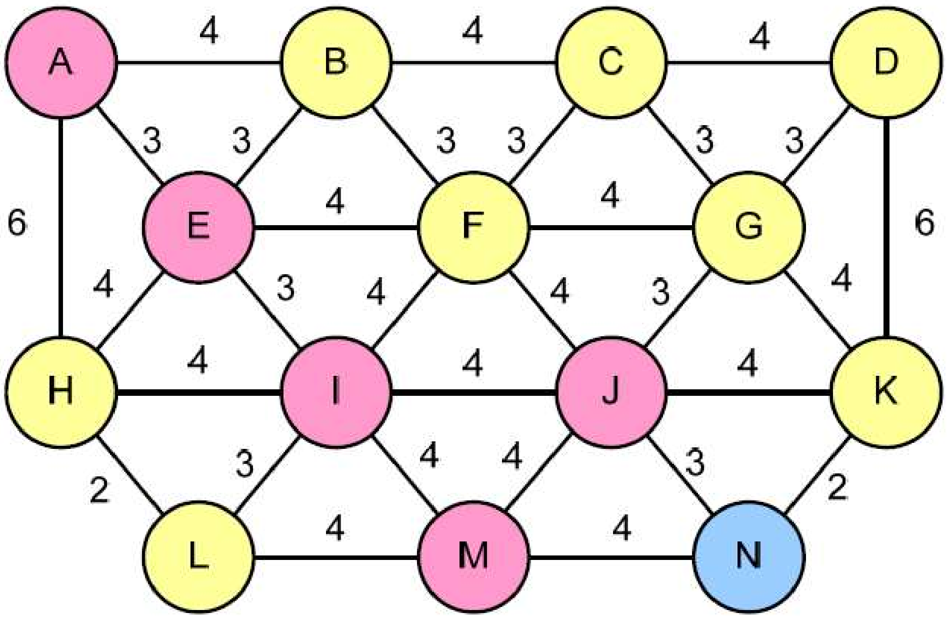


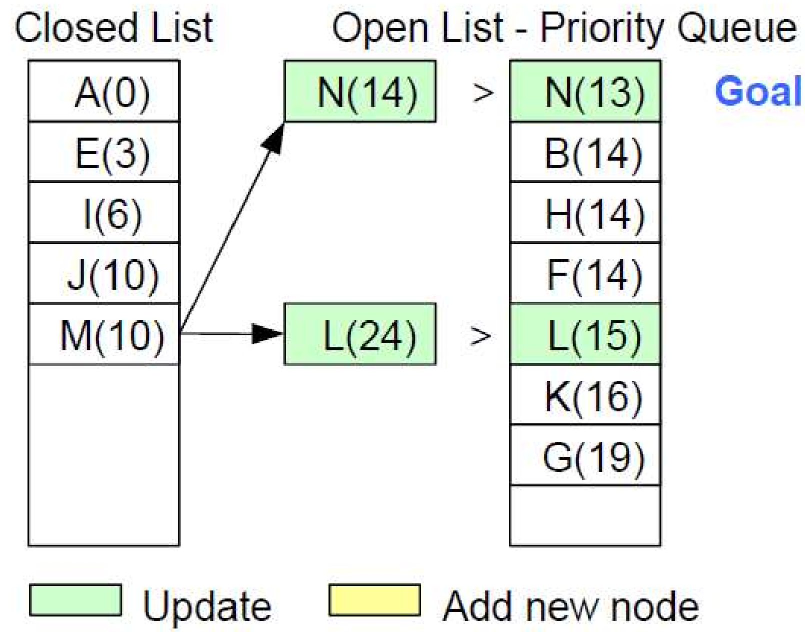


Heuristics
A = 14, B = 10, C = 8, D = 6, E = 8, F = 7, G = 6
H = 8, I = 5, J = 2, K = 2, L = 6, M = 2, N = 0

Slides from H. Choset

A*: Example (6/6)



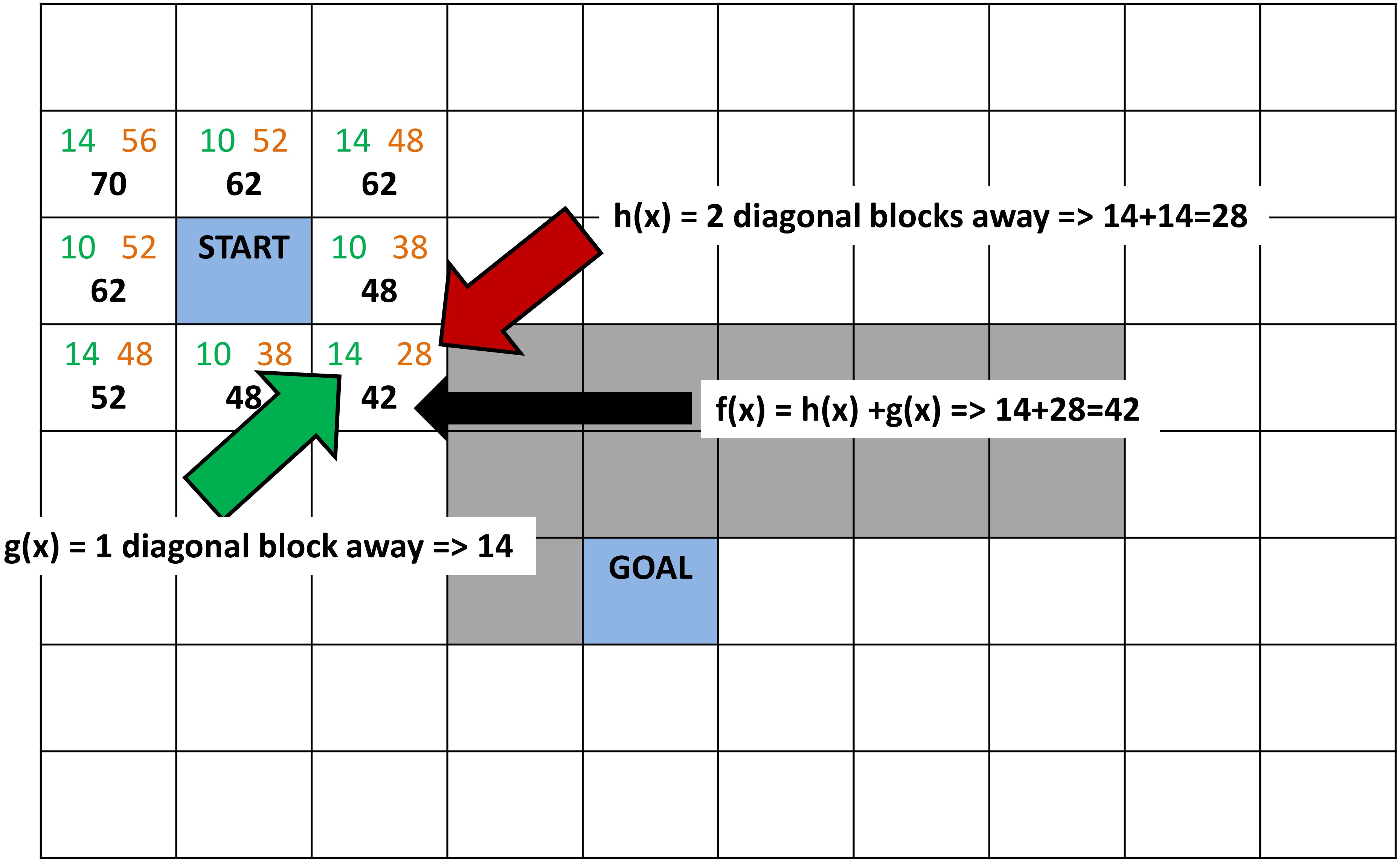


Heuristics
A = 14, B = 10, C = 8, D = 6, E = 8, F = 7, G = 6
H = 8, I = 5, J = 2, K = 2, L = 6, M = 2, N = 0

Since the path to N from M is greater than that from J, **the optimal path to N is the one traversed from J**

Slides from H. Choset

14 56 **70**	10 52 **62**	14 48 **62**							
10 52 **62**	**START**	10 38 **48**							
14 48 **52**	10 38 **48**	14 28 **42**							
				GOAL					


14 56
70
10 52
62
14 48
62
h(x) = 2 diagonal blocks away => 14+14=28
10 52
62
START
10 38
48
14 48
52
10 38
48
14 28
42
f(x) = h(x) +g(x) => 14+28=42
g(x) = 1 diagonal block away => 14
GOAL

14 56 70	10 52 62	14 48 62							
10 52 62	START	24 38 62	14 34 48						
14 48 52	24 38 62	14 28 42							
	28 34 62	24 24 48							
				GOAL					

14 56 **70**	10 52 **62**	14 48 **62**	24 44 **68**	28 40 **68**					
10 52 **62**	**START**	24 38 **62**	14 34 **48**	24 30 **54**					
14 48 **52**	38 38 **76**	14 28 **42**							
	34 34 **68**	24 24 **48**							
	38 30 **68**	34 20 **54**		**GOAL**					

14 56 **70**	10 52 **62**	14 48 **62**	24 44 **68**	28 40 **68**					
10 52 **62**	**START**	24 38 **62**	14 34 **48**	24 30 **54**					
14 48 **52**	38 38 **76**	14 28 **42**							
	48 34 **82**	24 24 **48**							
	44 30 **74**	34 20 **54**		**GOAL**					

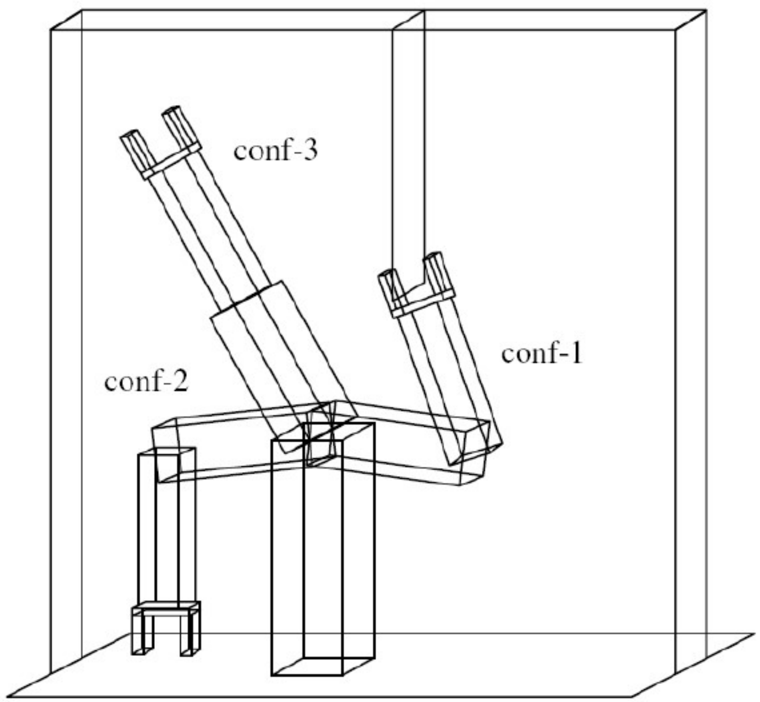
14 56 **70**	10 52 **62**	14 48 **62**	38 44 **82**	34 40 **74**	38 44 **82**				
10 52 **62**	**START**	24 38 **58**	14 34 **48**	24 30 **54**	34 34 **68**				
14 48 **52**	38 38 **74**	14 28 **42**							
	48 34 **82**	24 24 **48**							
	44 30 **74**	34 20 **54**		**GOAL**					
	48 34 **82**	44 24 **68**	48 14 **62**						

Practical motion planning problems

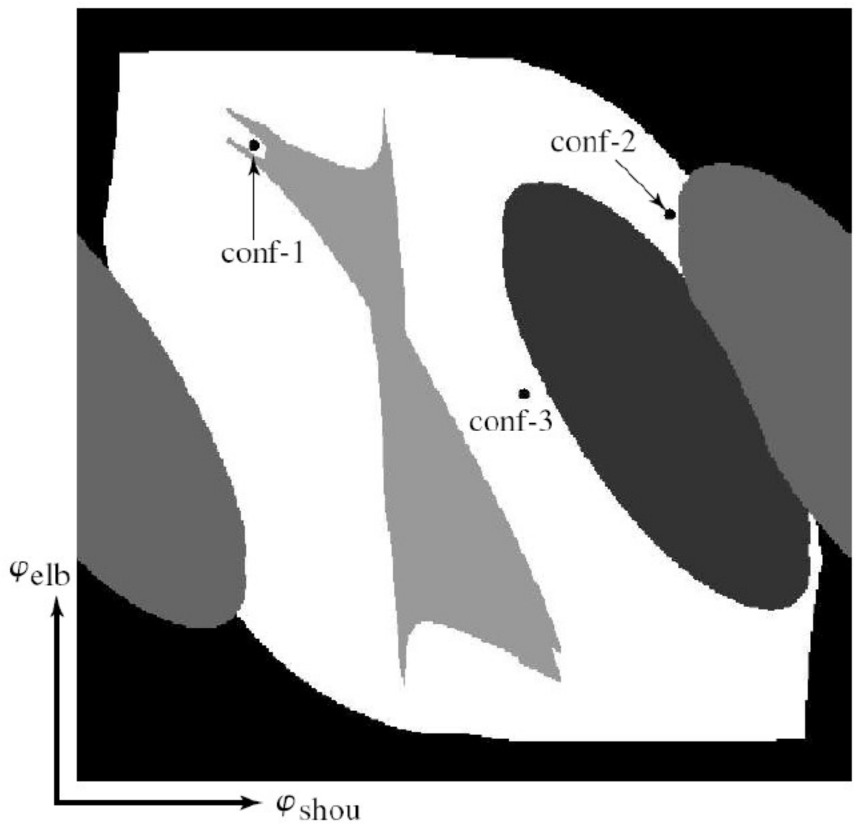
- Heuristic methods trade off completeness with practical efficiency and have weak performance guarantee.
- Complexity grows with the dimensions of the robot's C-space.
- Solution to this problem: (1 project search to lower dimension (2 limit search space (3 sacrifice optimality and/or completeness.

Slides from P. Abbeel

Practical motion planning problems



Serial Robot with elbow and shoulder joints



Configuration Space including obstacles

Slides from P. Abbeel

Sampling-based motion planning

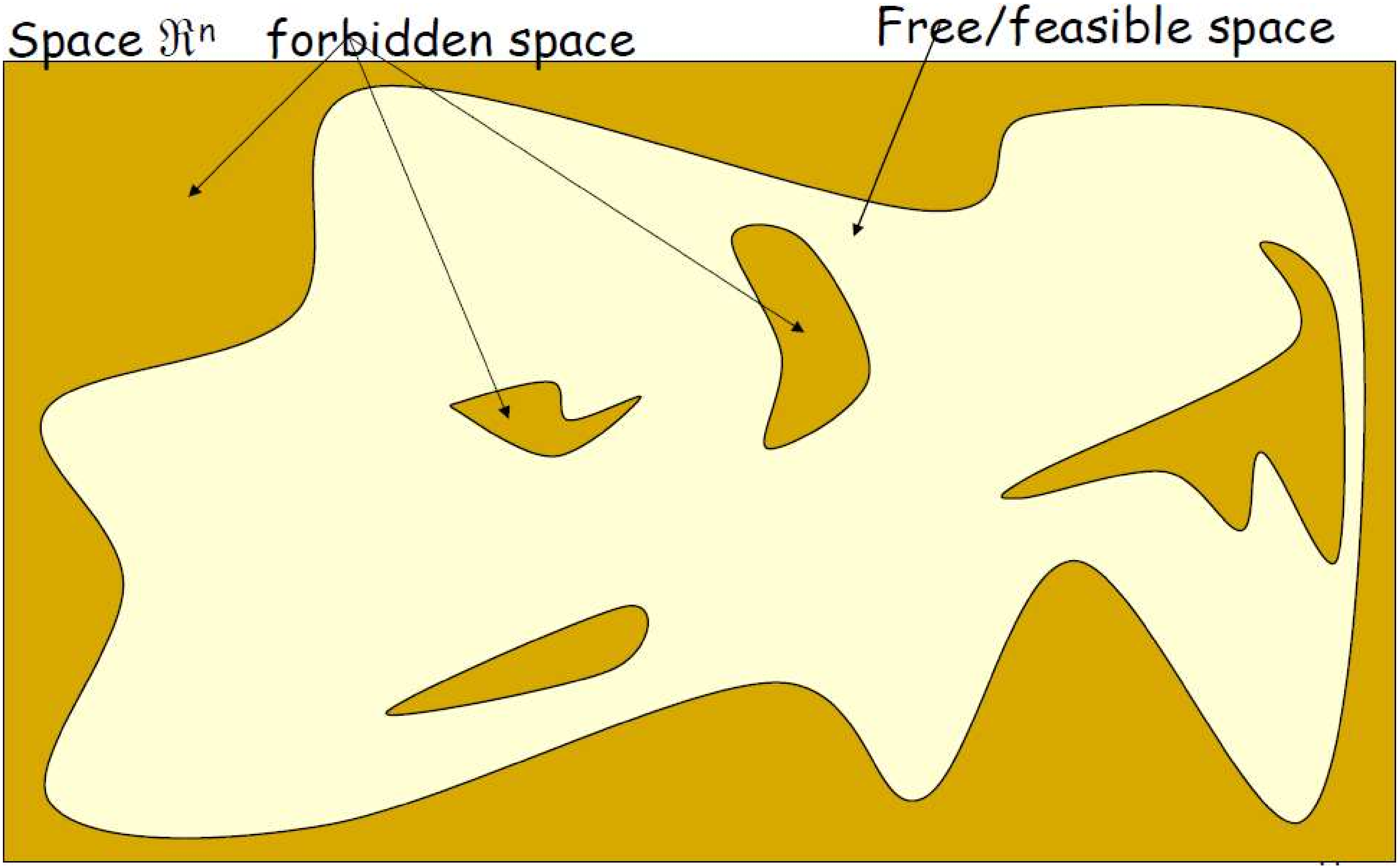
- Explore a subset of possibilities (samples) instead of searching the whole space.
- Classical trade-off between optimality and performance.

Pros:
1. Probabilistically complete.
2. Can be applied to high-dimensional problems.
3. No need to construct complete C-space.
4. Support fast queries.

Cons:
1. No optimality or completeness guarantee.
2. Fail with some problems.

Slides from P. Abbeel

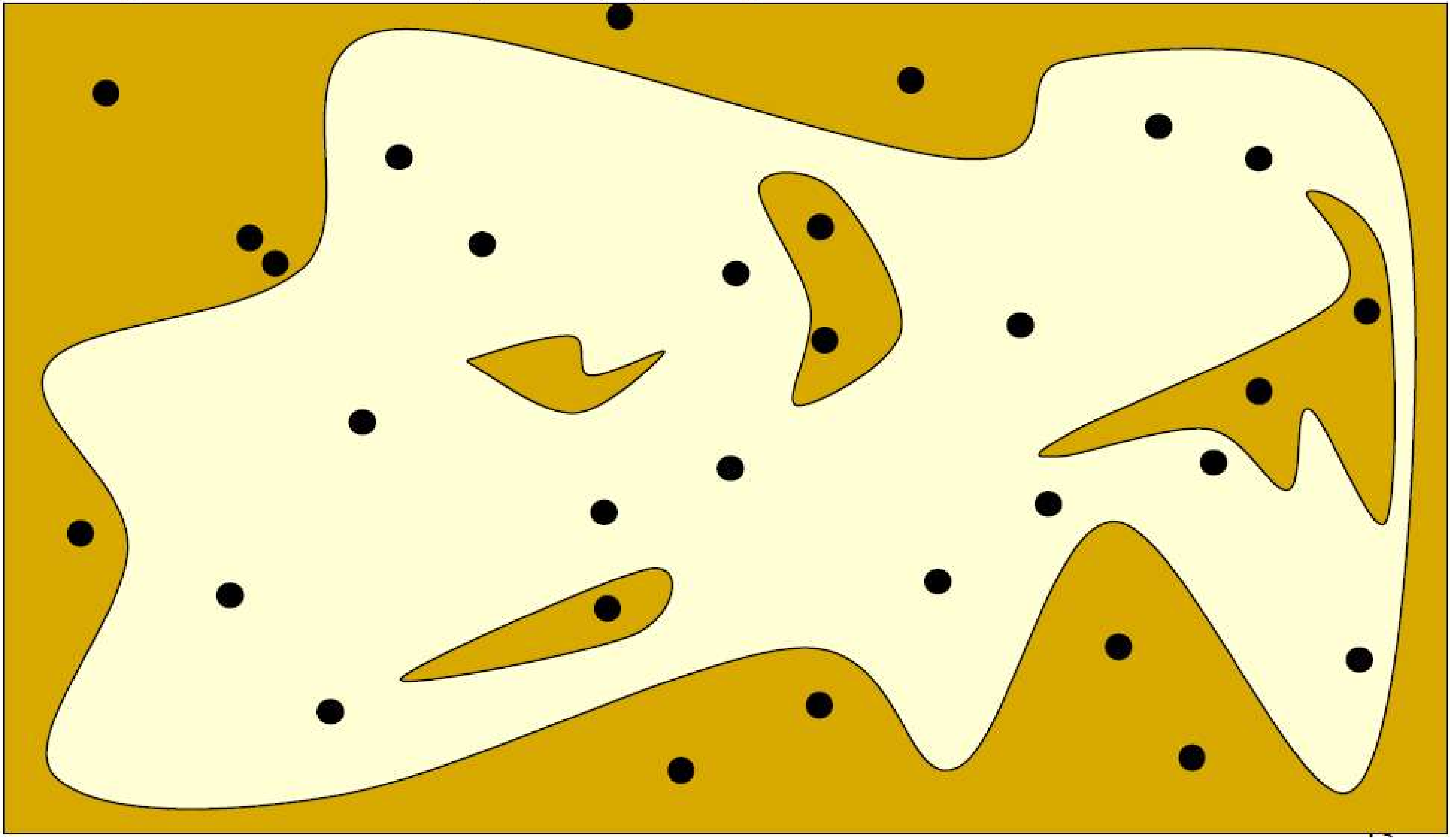
Probabilistic Road Map



Slides from P. Abbeel

Probabilistic Road Map

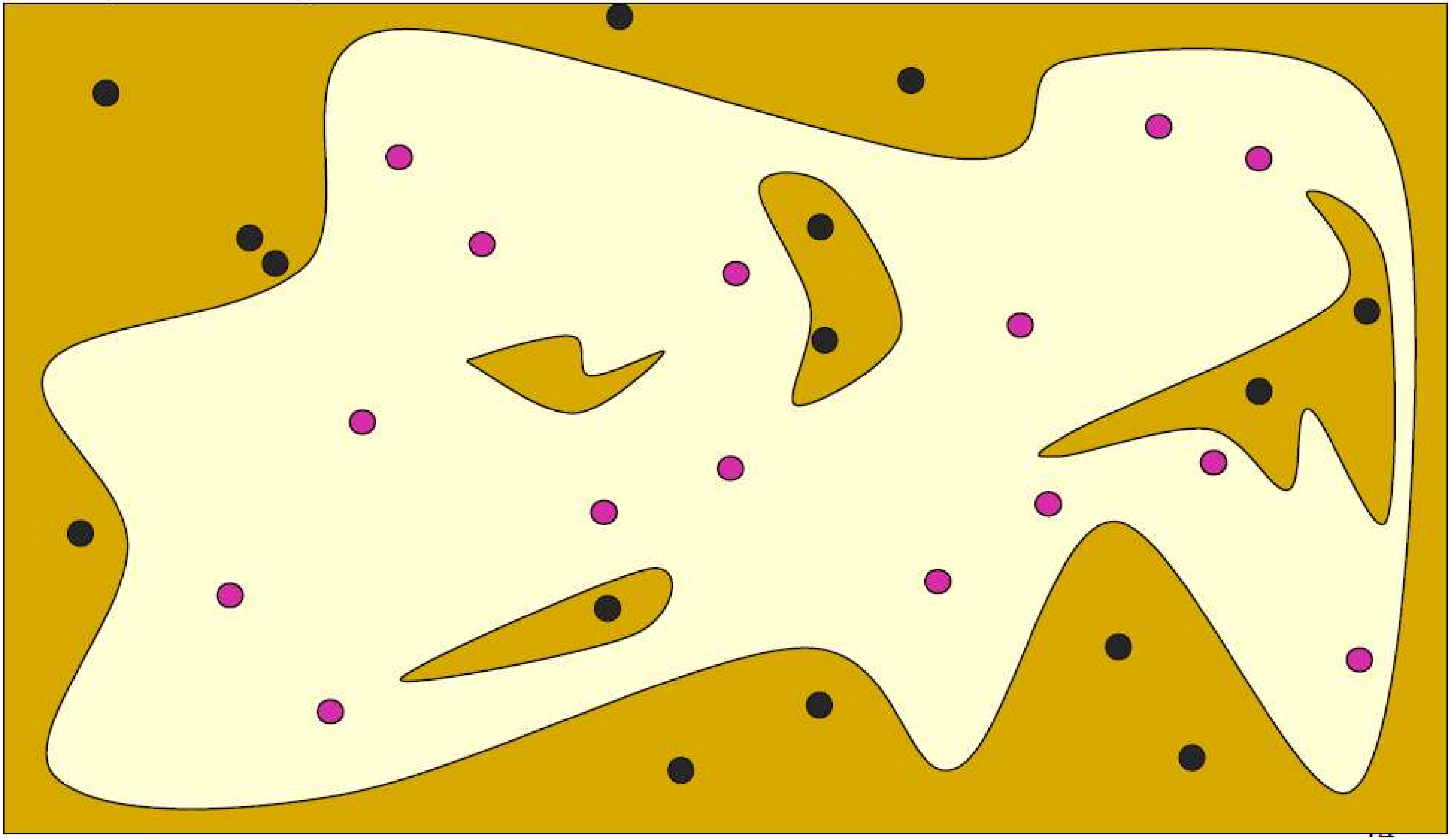
Configurations are sampled by picking coordinates at random



Slides from P. Abbeel

Probabilistic Road Map

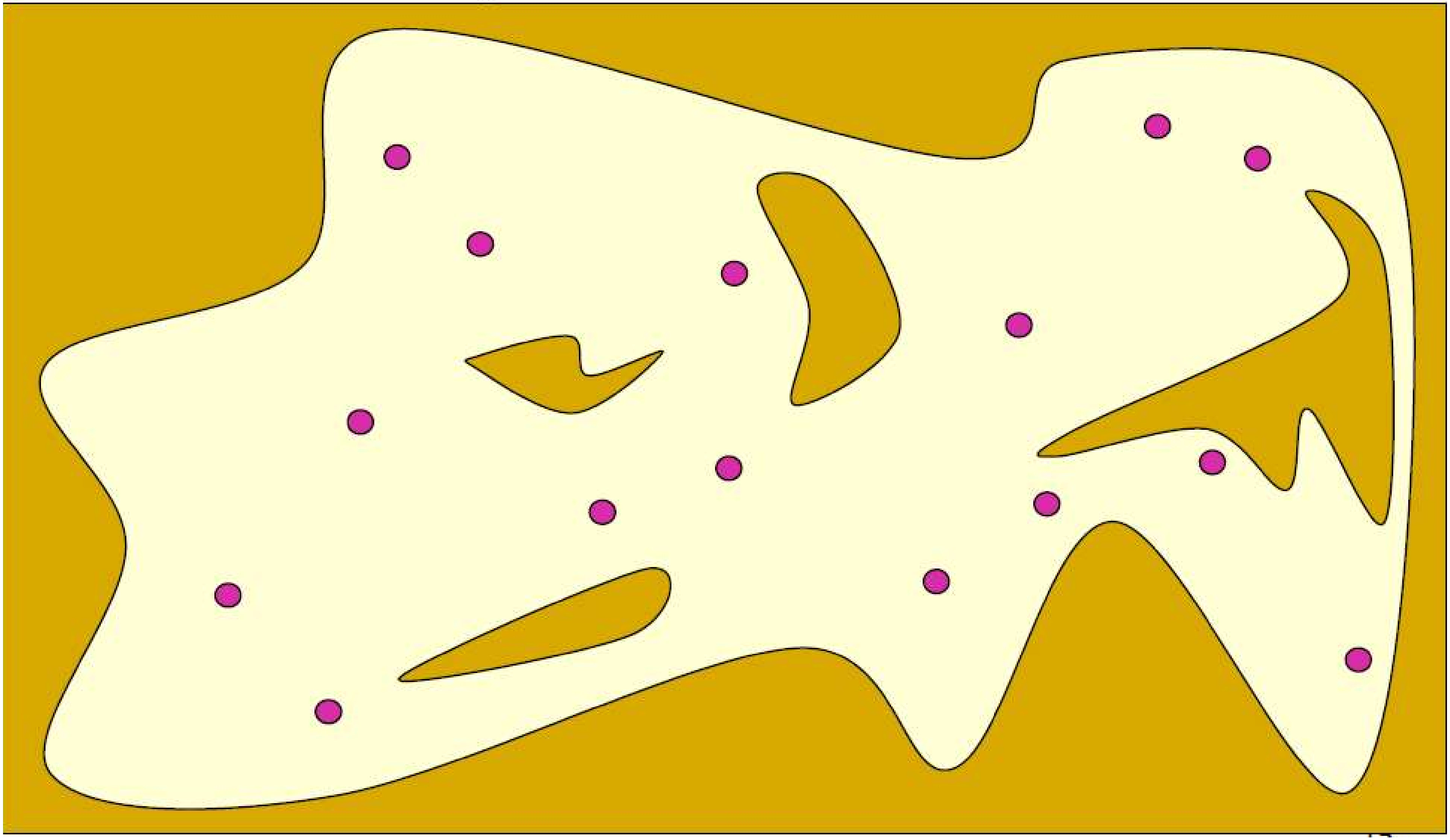
Sampled configurations are tested for collision



Slides from P. Abbeel

Probabilistic Road Map

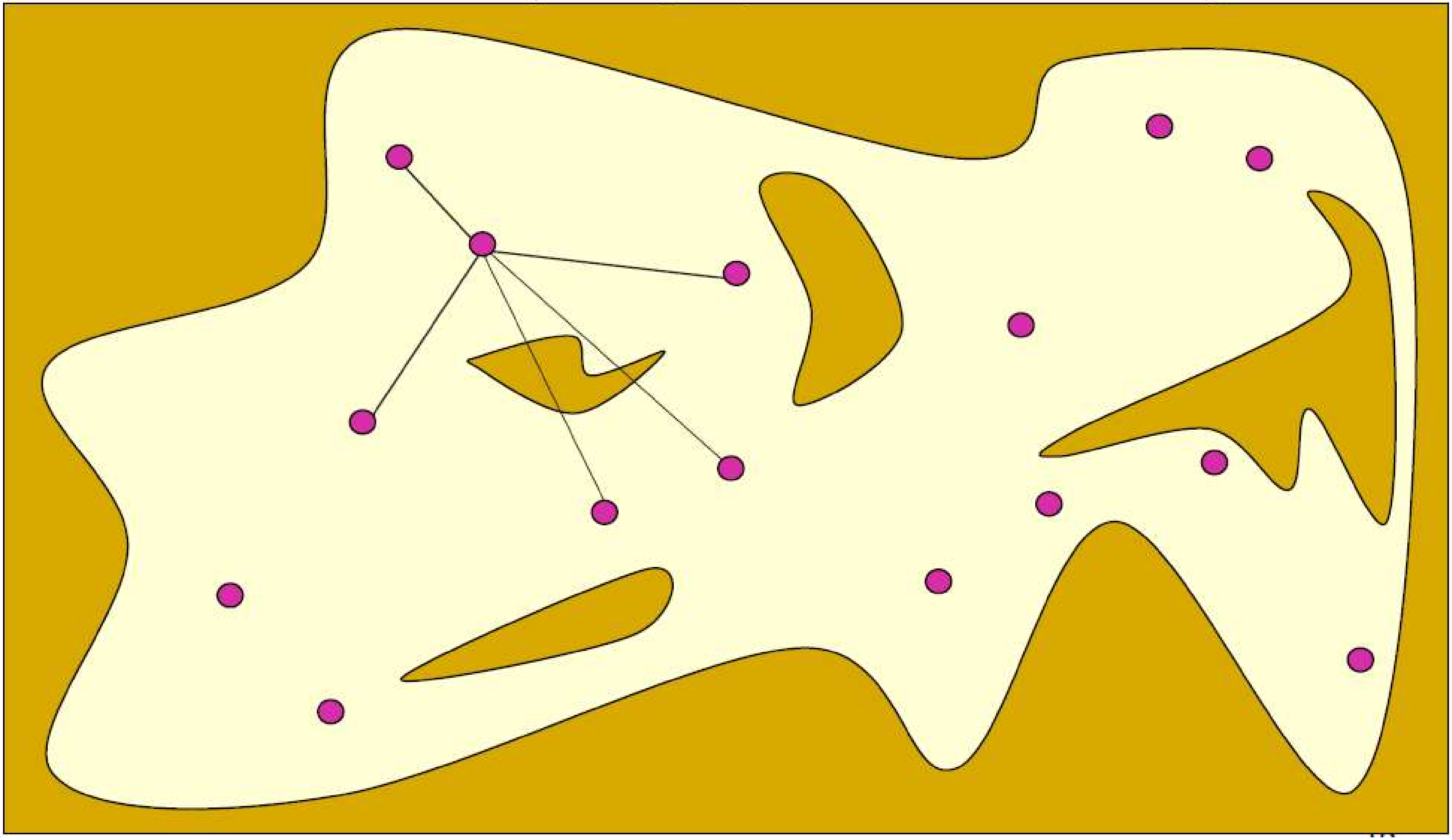
The collision-free configurations are retained as milestones



Slides from P. Abbeel

Probabilistic Road Map

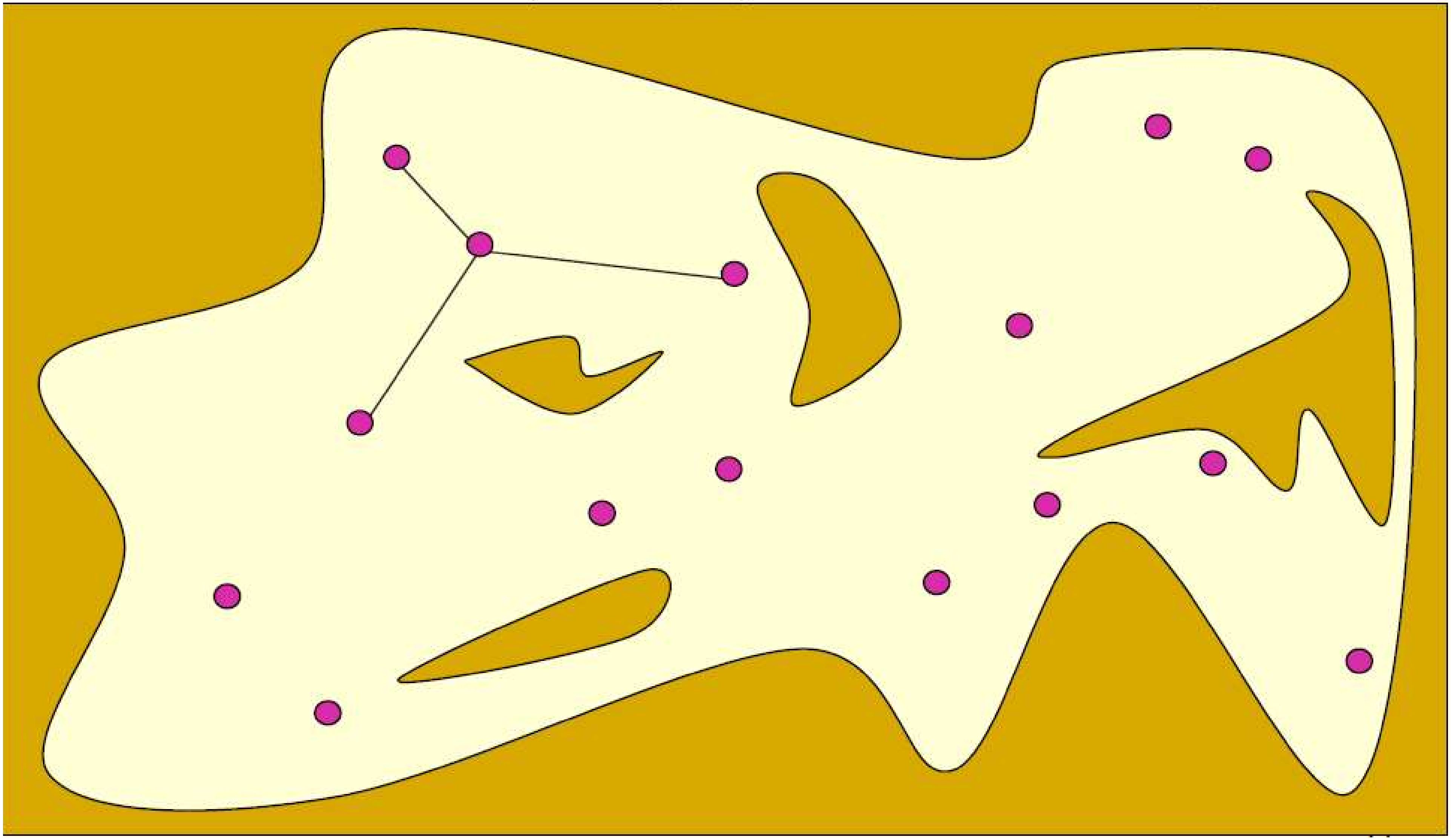
Each milestone is linked by straight paths to its nearest neighbors



Slides from P. Abbeel

Probabilistic Road Map

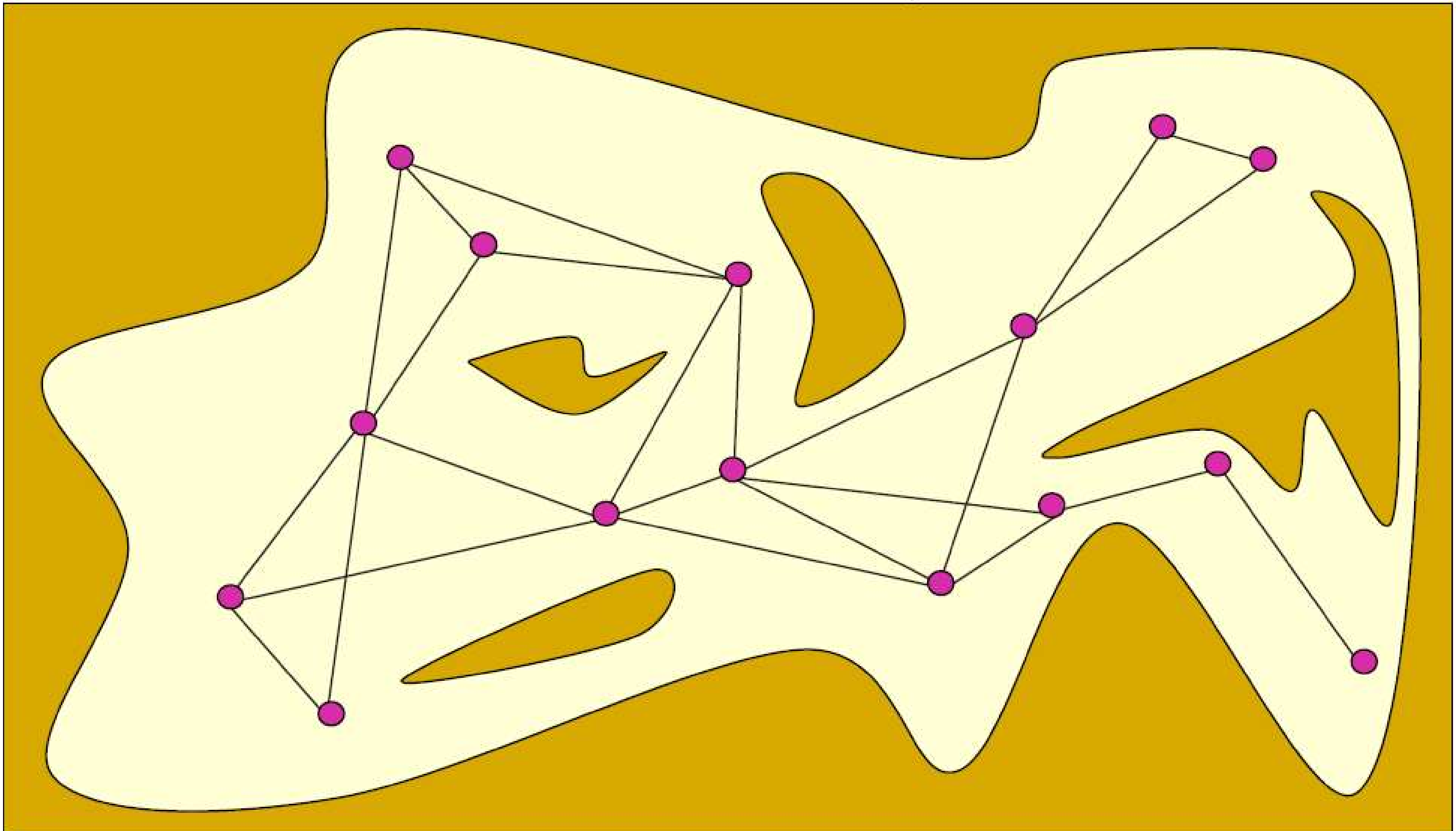
Each milestone is linked by straight paths to its nearest neighbors



Slides from P. Abbeel

Probabilistic Road Map

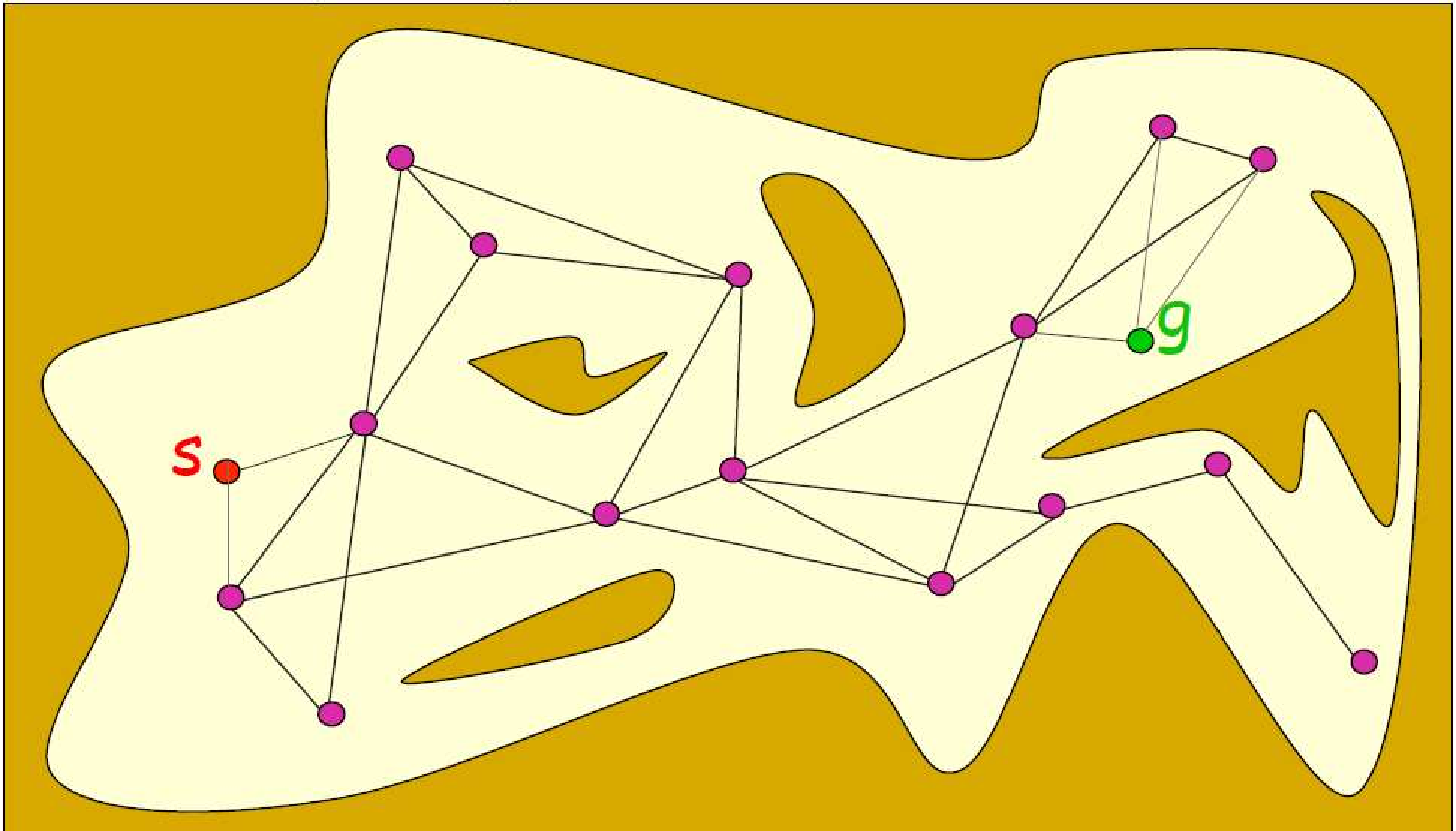
The collision-free links are retained as local paths to form the PRM



Slides from P. Abbeel

Probabilistic Road Map

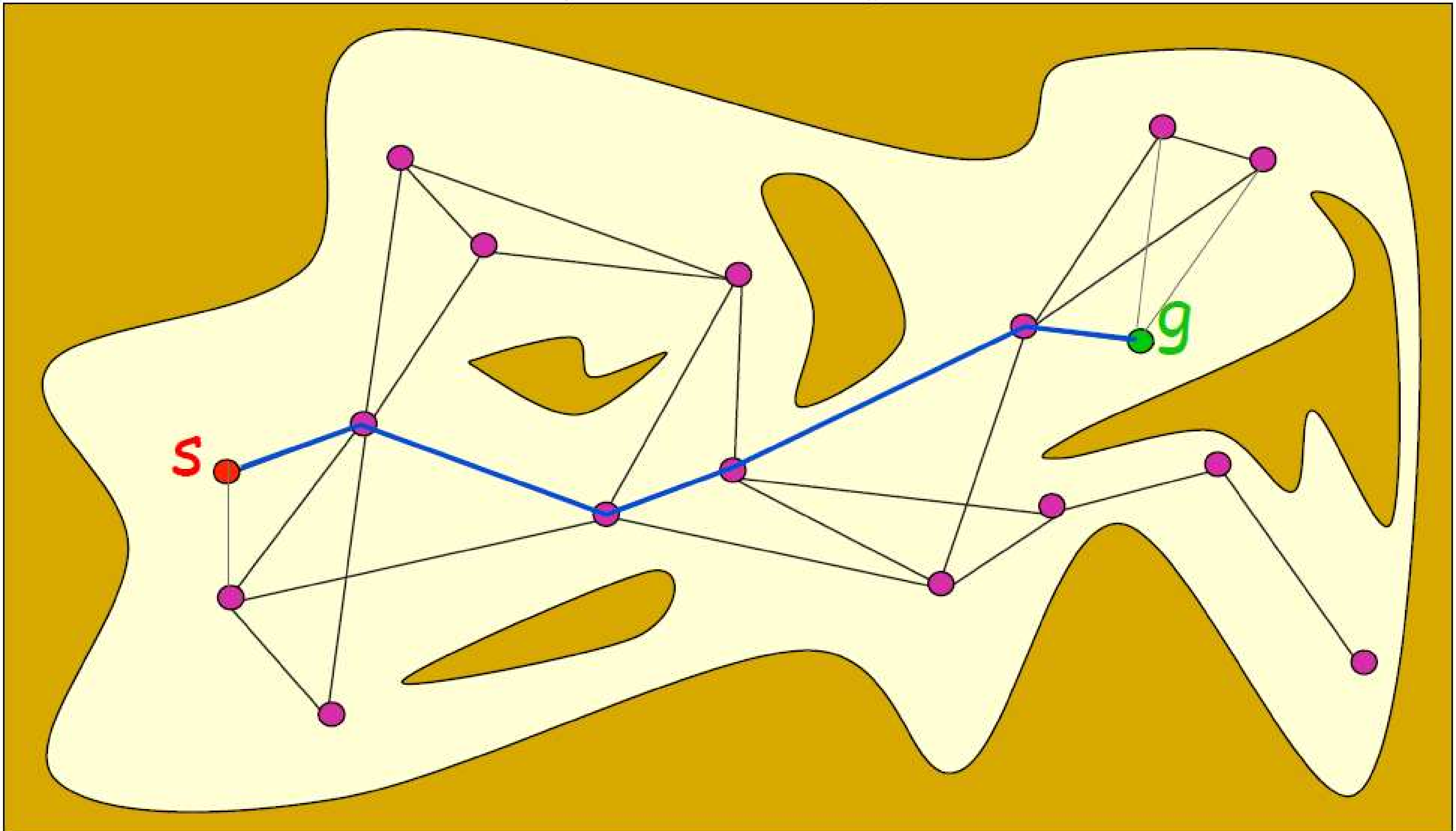
The start and goal configurations are included as milestones



Slides from P. Abbeel

Probabilistic Road Map

The PRM is searched for a path from s to g



Slides from P. Abbeel

Probabilistic Road Map

- Initialize set of points with Start and Goal
- Randomly sample points in configuration space
- Connect nearby points if they can be reached from each other
- Find path from Start to Goal in the graph
- Challenges:
  - Connecting neighbouring points: Only easy for holonomic systems (i.e., for which you can move each degree of freedom at will at any time).
  - Collision checking is expensive Computationally.