CS 460/560
Introduction to Computational Robotics
Fall 2019, Rutgers University

Lecture 19
More Planning Problems and Methods

Instructor: Jingjin Yu

Outline

Time varying motion planning problems

Mixing discrete and continuous spaces

Coverage planning

Sensor-based planning methods

- ⇨ Bug algorithms
- ⇨ Gap-navigation trees

Time Varying Problems

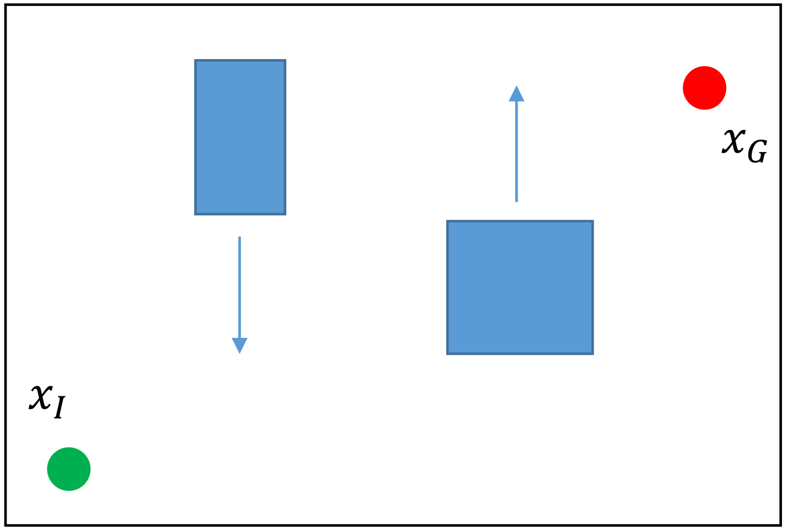
So far we have only discussed **static** problems

⇨That is, the obstacles are static

⇨Also known as **time-invariant** problems

⇨When obstacles may change over time, the problem is **time varying**

⇨E.g.



⇨One may view multi-robot problems as a time varying problem as well

Predictable Obstacles

When obstacle movement is deterministic, the problem can be turned into a problem that looks like a time-invariant problem

⇨This is done via adding the extra "time" dimension

⇨E.g., single obstacle, piece wise linear motion

⇨Configuration: (x_1, x_2, t)

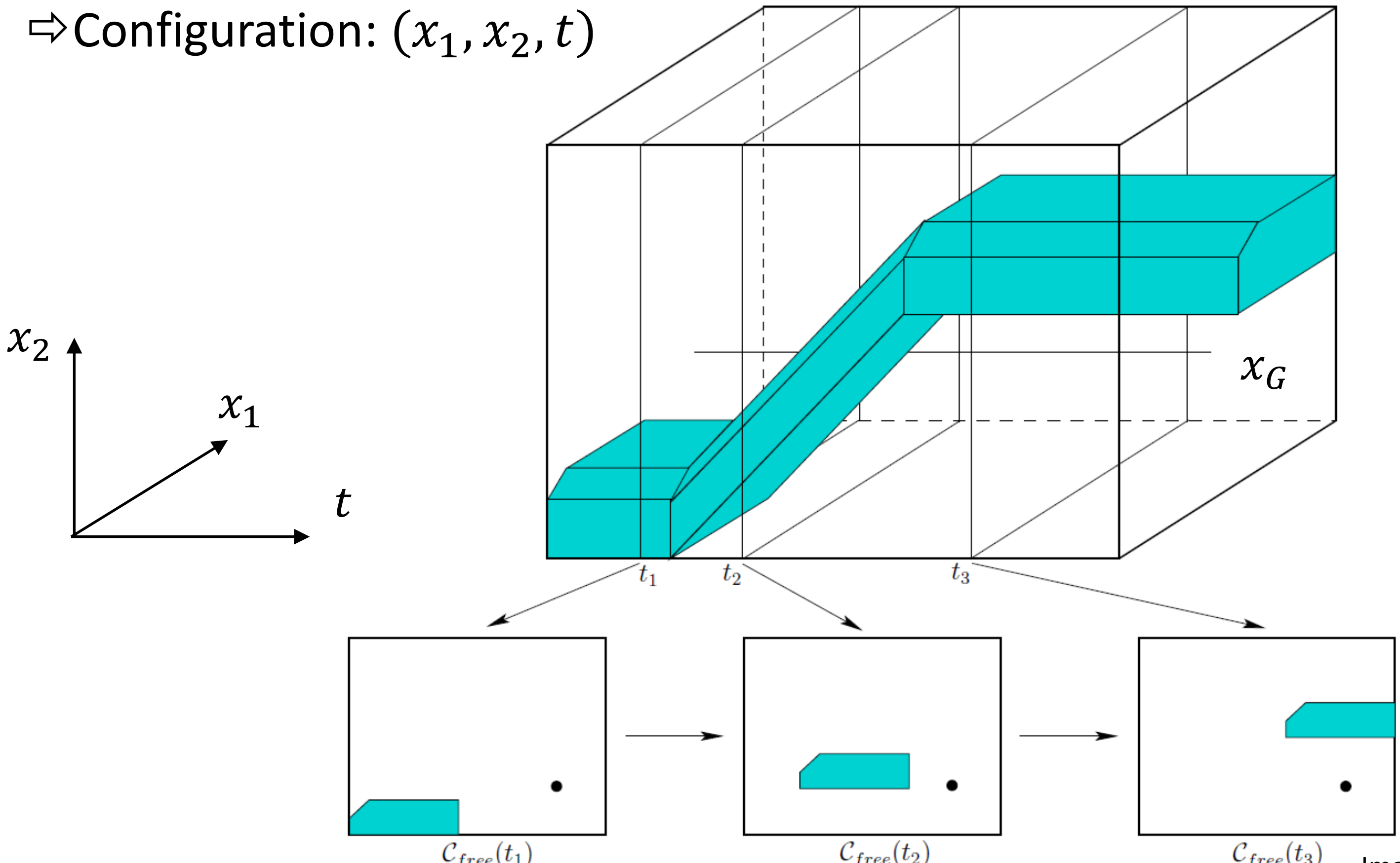
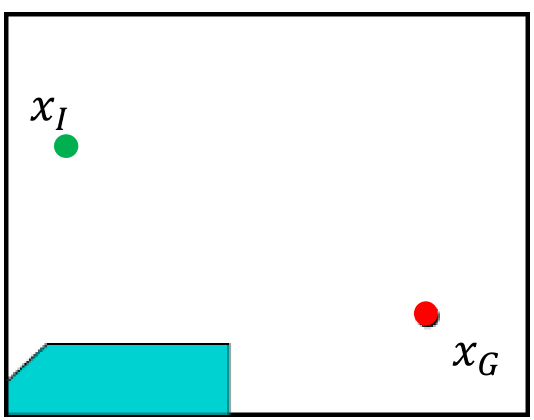


Image source: Planning Algorithms

Solution with Unbounded Speed (I)

First consider problems without a "speed limit"

- ⇨ How can we solve the problem under such assumptions?
- ⇨ Combinatorial methods
  - ⇨ E.g., vertical cell decomposition
  - ⇨ Key: no traveling back in time!
- ⇨ Probabilistic methods
  - ⇨ PRM, RRT
  - ⇨ Need a new metric: $d\big((x_1, x_2, t), (x_1', x_2', t')\big) = \begin{cases} 0 & if\,(x_1, x_2, t) = (x_1', x_2', t') \\ \infty & if\,(x_1, x_2, t) \neq (x_1', x_2', t')\ and\ t' \leq t \\ d\big((x_1, x_2), (x_1', x_2')\big) & otherwise \end{cases}$
  - ⇨ This is a **pseudo-metric**: not symmetric
  - ⇨ Basically, cannot travel back in time



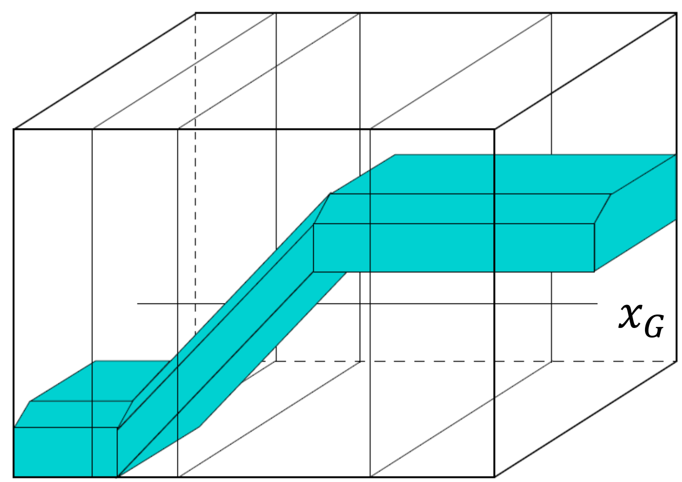
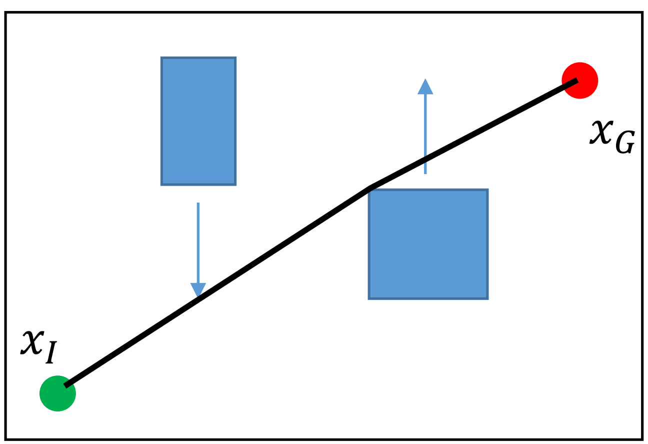


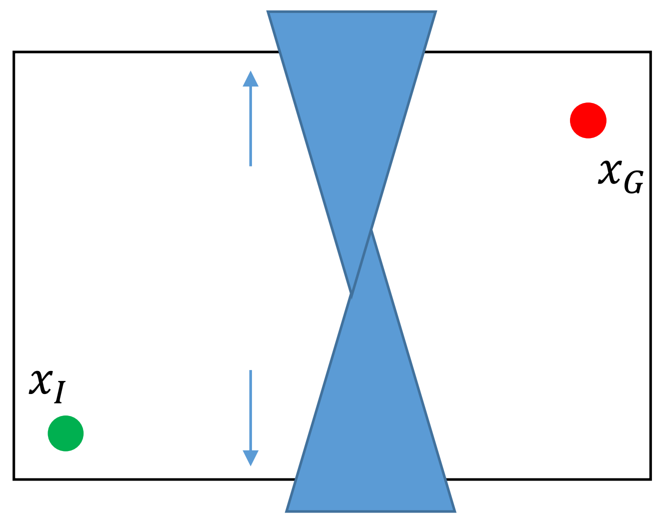
Image source: Planning Algorithms

Solution with Unbounded Speed (II)

Question: given unbounded speed, why not

- ⇨Plan a path at $t = 0$
- ⇨Ask the robot to just go fast to follow the path?
- ⇨Because it does not always work!
  - ⇨ E.g.





Solution with Bounded Speed

For real problems, cannot use arbitrarily fast travel!

- ⇨Otherwise, solution becomes “trivial” if feasible path exists at $t = 0$
- ⇨But, we do not need to much more to make it work

Combinatorial methods

- ⇨Ensure that the slope of any path segment has bounded magnitude
- ⇨For any point in the space-time space, this induces a “speed cone”

Sampling based methods

- ⇨Need to modify the pseudometric
- ⇨Same idea

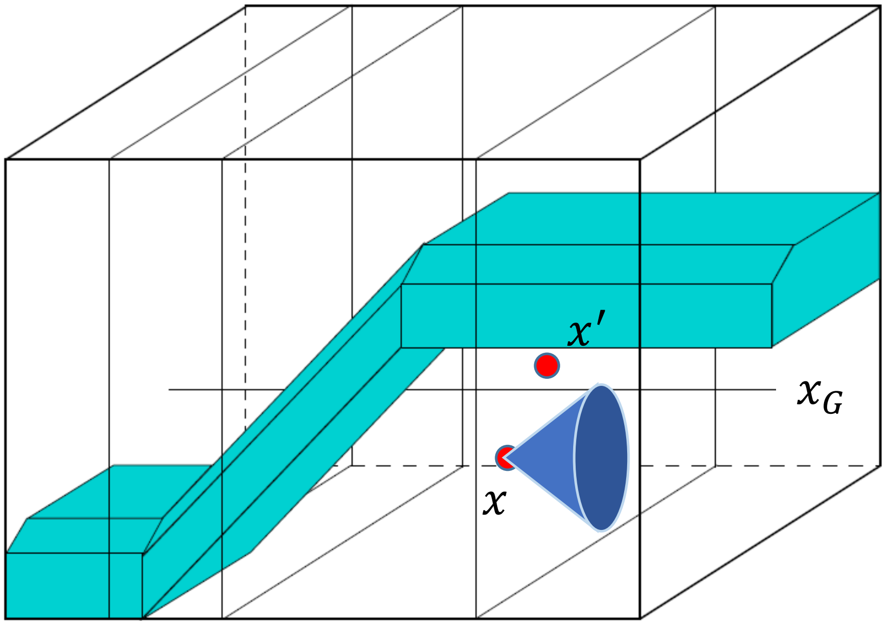


Image source: Planning Algorithms

Hybrid Systems

Hybrid systems is a model for mixing discrete and continuous domains

- ⇨The system generally works with continuous domains
- ⇨But it is also capable of making discrete transitions
- ⇨In this case, the different continuous domains are “modes”
- ⇨For example, the transmission system of a car

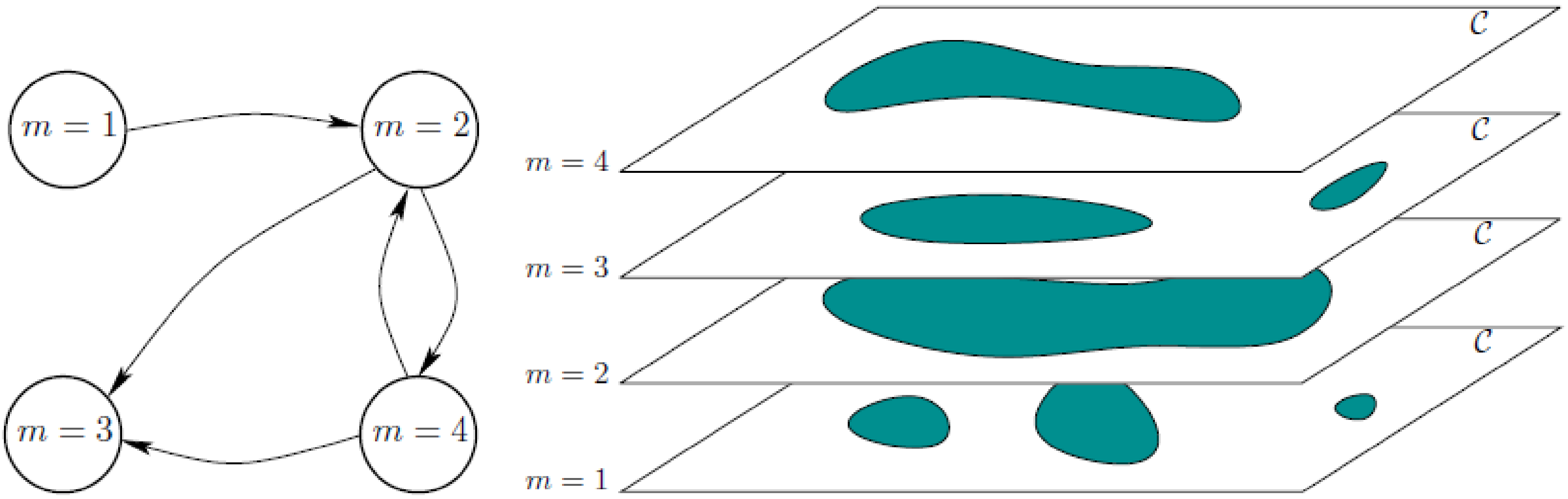


Image source: Planning Algorithms

Hybrid System – A 2D Planning Example

The power of the portiernia

- ⇨Robot must open door 1 and door 2 to access its goal
- ⇨It may retrieve one key at a time from the portiernia
- ⇨Discrete elements:
  - ⇨ Which key the robot has
  - ⇨ The status of the door
- ⇨Continuous elements:
  - ⇨ The 2D domain that may change

Calls for **task and motion planning**

- ⇨Can be modeled as special automata
  - ⇨ Can do planning and verification
- ⇨Or, encode with first order logic
  - ⇨ E.g., planning domain definition language

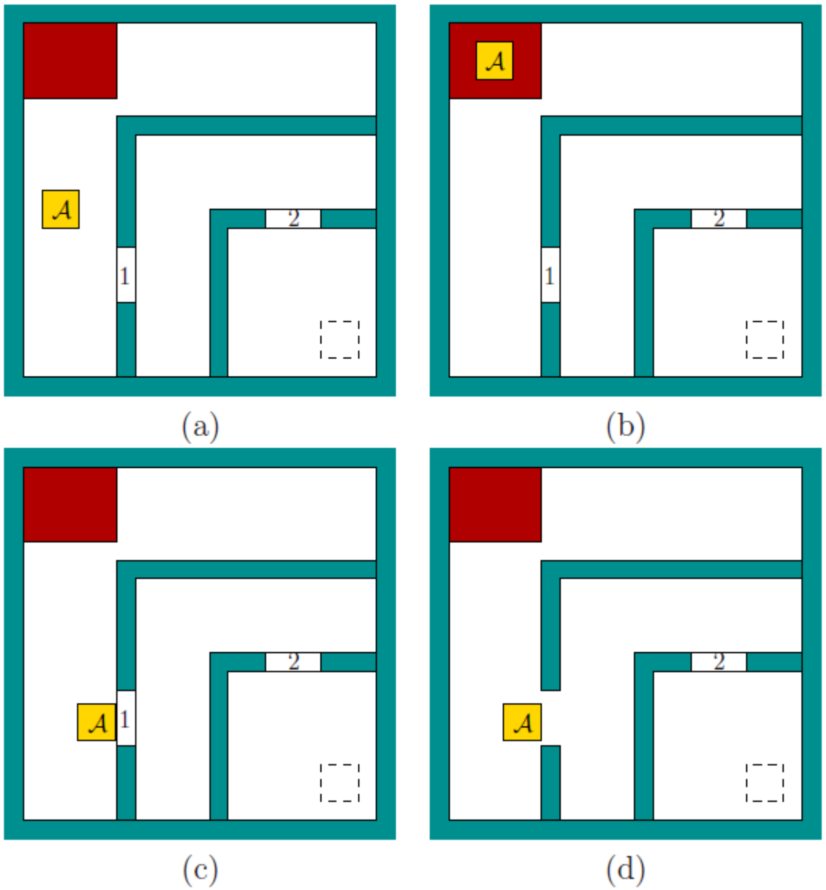
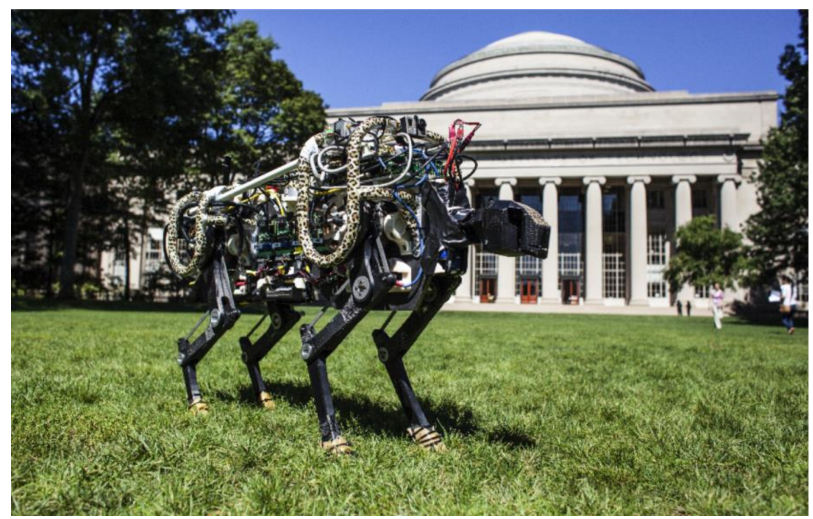


Image source: Planning Algorithms

Configuration Space with Varying Dimensions

Another scenario that mixes discrete and continuous spaces has changing configuration space dimensions

- ⇨Bipedal robots, legged robots
- ⇨Why?
- ⇨Manipulators grasping/pushing objects
- ⇨More on these later…



Coverage Planning (I)

An important problem in practice is **coverage planning**

- ⇨Planning paths for Roomba to clean a room
- ⇨Planning paths for harvesting crops
- ⇨What is the main goal here?
    - ⇨ Cover (using limited footprint) the space with least amount of total travel
    - ⇨ A second objective can be minimizing turns, which takes time

Coverage Planning (II)

Boustrophedon decomposition

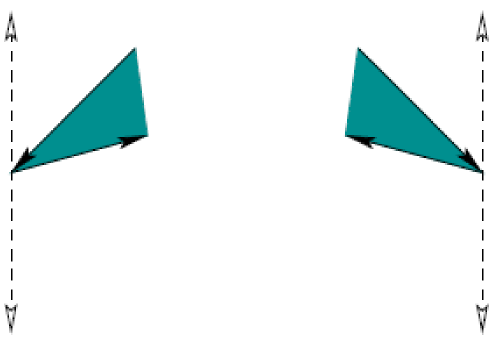
⇨One way to do coverage planning is intuitive: move back and forth!

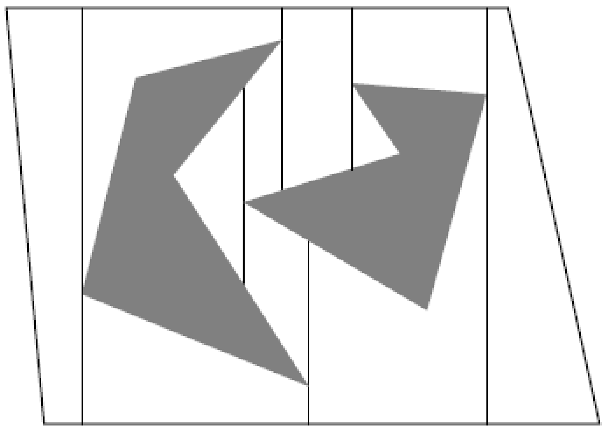
⇨Basically, do a (special) vertical cell decomposition

⇨Then, do boustrophedon paths in each cell

⇨Issue: when cells split, may need to travel back and waste time

⇨Can be resolved using spanning tree double covering





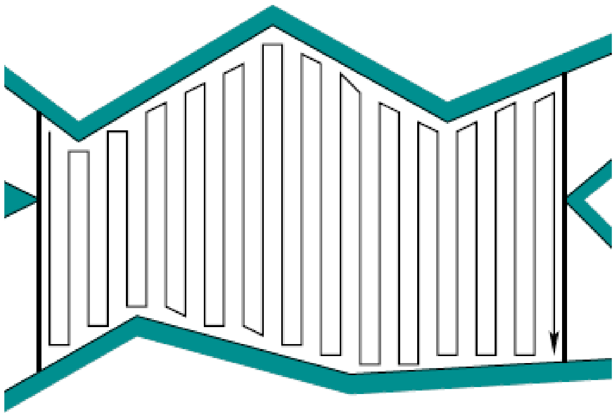
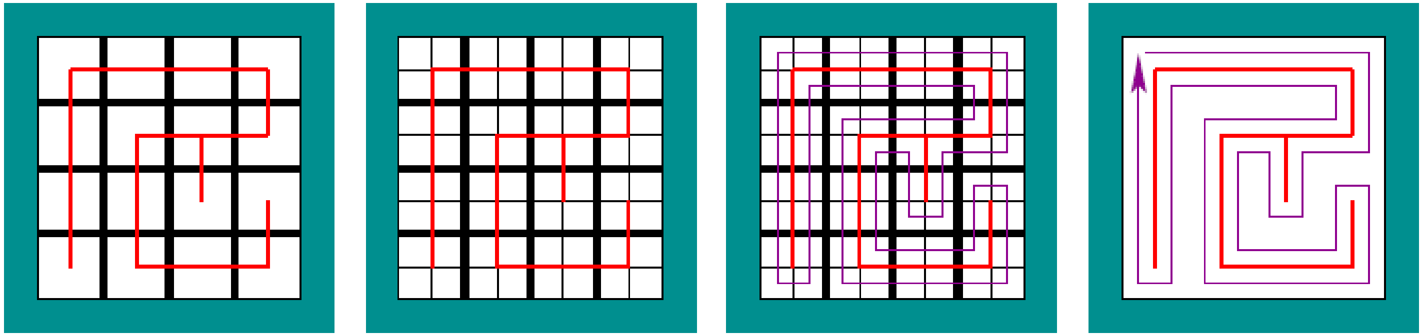


Image source: Planning Algorithms

Coverage Planning (III)

Spanning tree double cover

- ⇨Cover the free space with a grid
- ⇨Find a spanning tree
- ⇨Doubling up the spanning tree to create a single non-overlapping path



- ⇨Example with obstacles

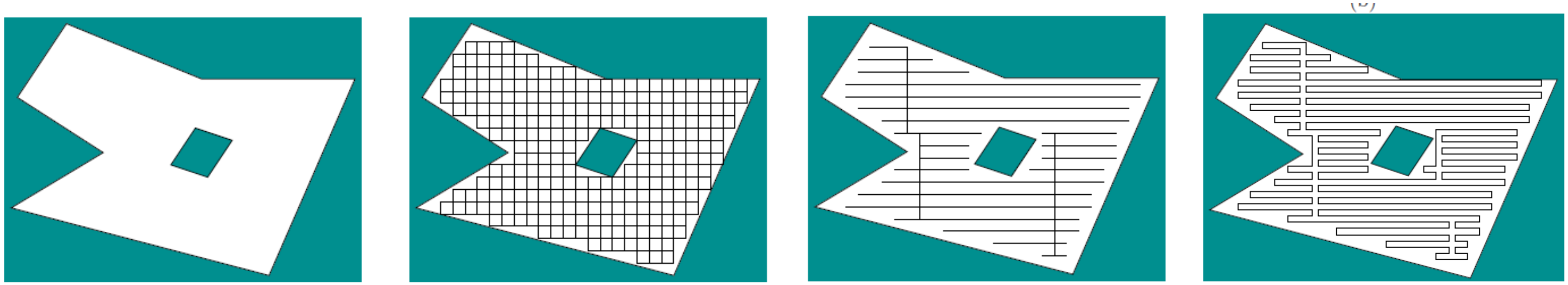


Image source: Planning Algorithms

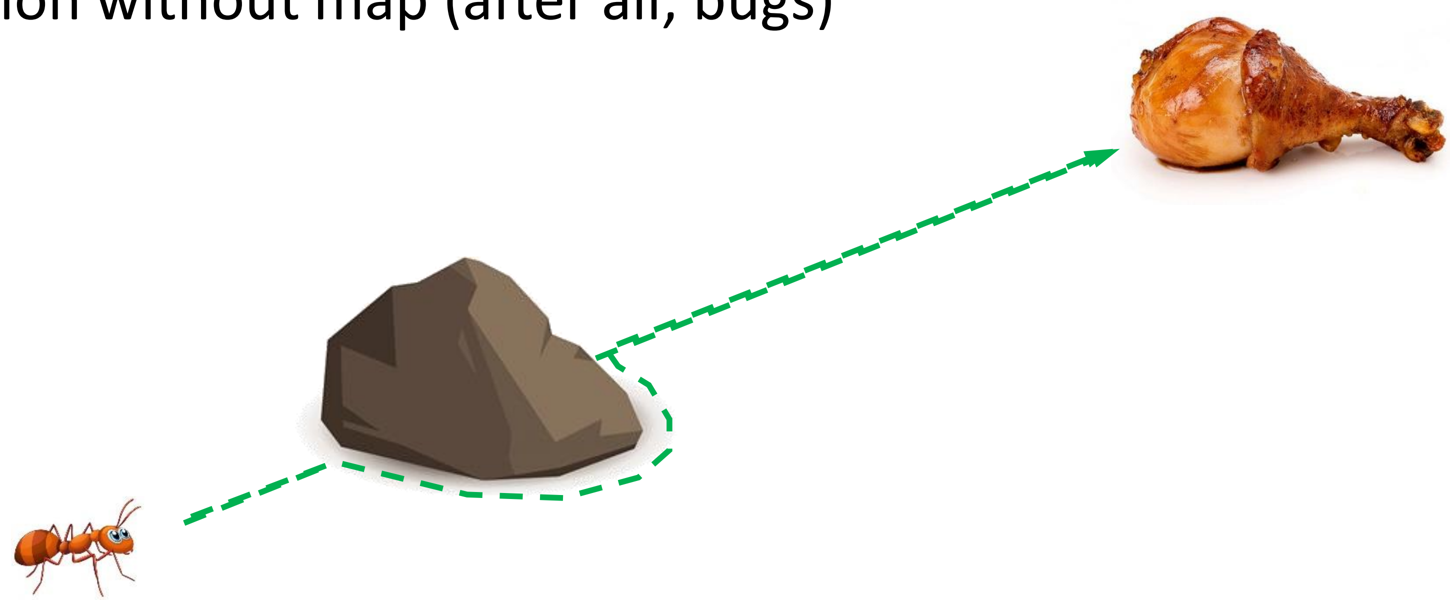
Bug Algorithms

Sometimes robots may have special sensors

For a bug, e.g., ants, the sensors may be

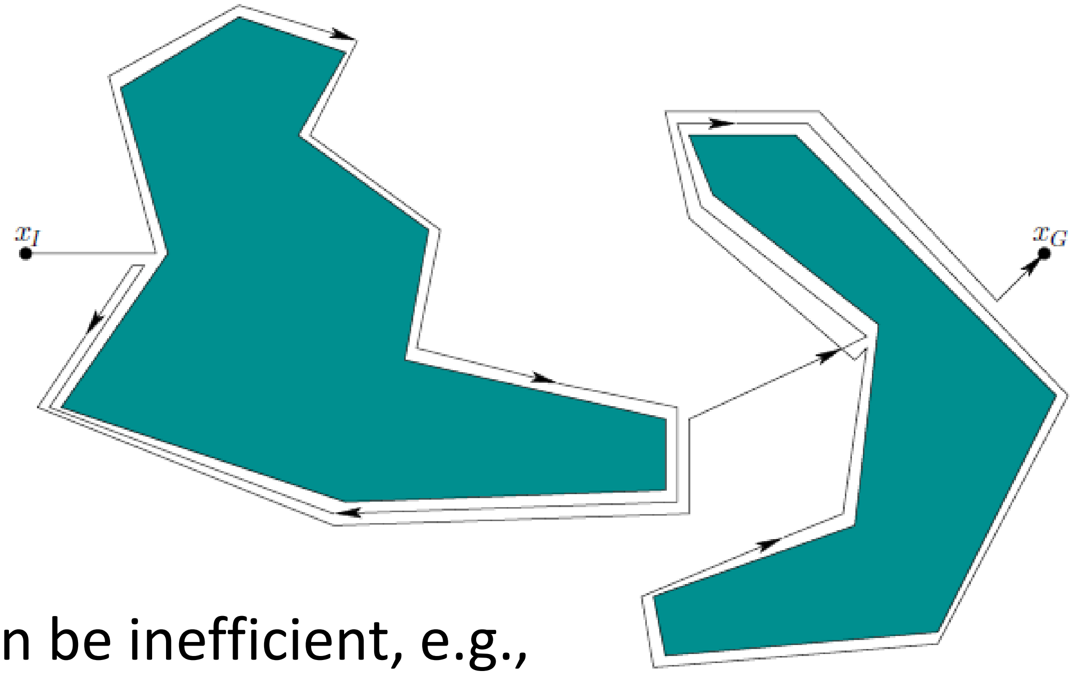
- ⇨A goal sensor: detects the direction of the goal (e.g., food)
- ⇨An obstacle censor
- ⇨Markers: the bug knows when it revisit a place

Navigation without map (after all, bugs)



Bug1

The Bug1 algorithm: scout each obstacle



Complete but can be inefficient, e.g.,

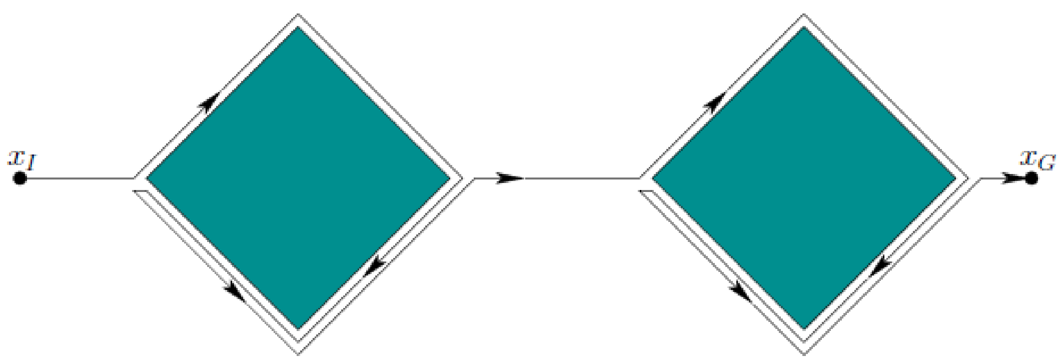
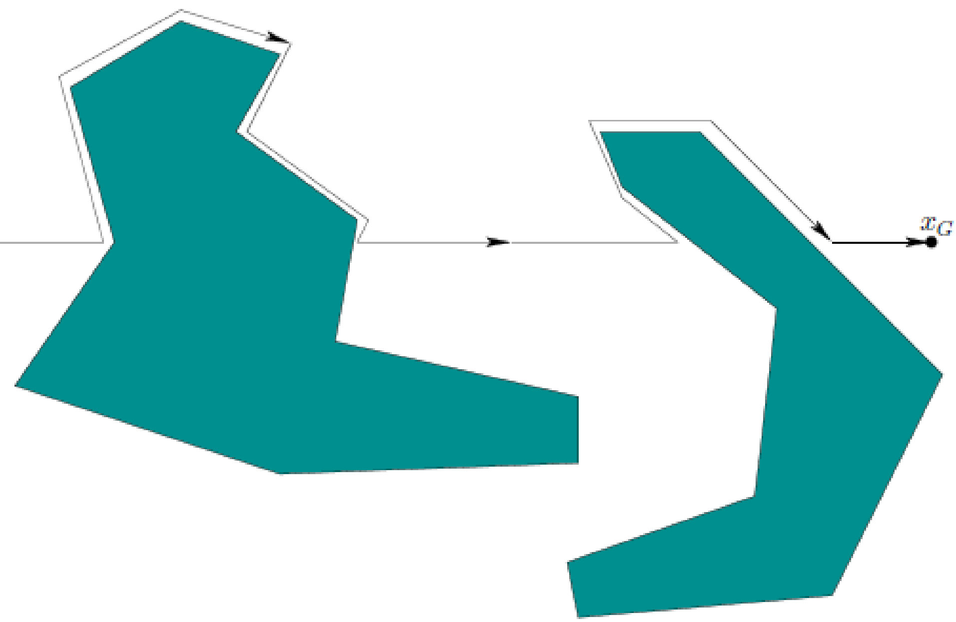


Image source: Planning Algorithms

Bug2

Bug2 algorithm: work with a fixed bearing



A bad case for Bug2

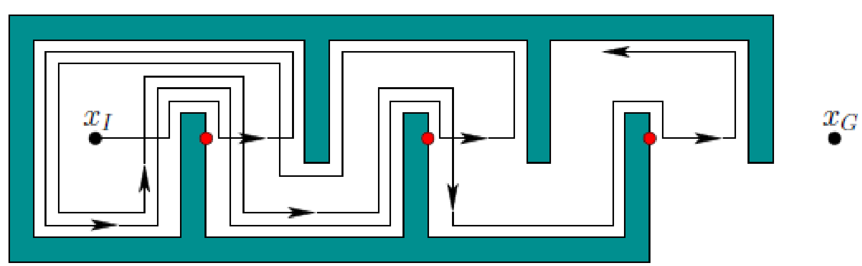
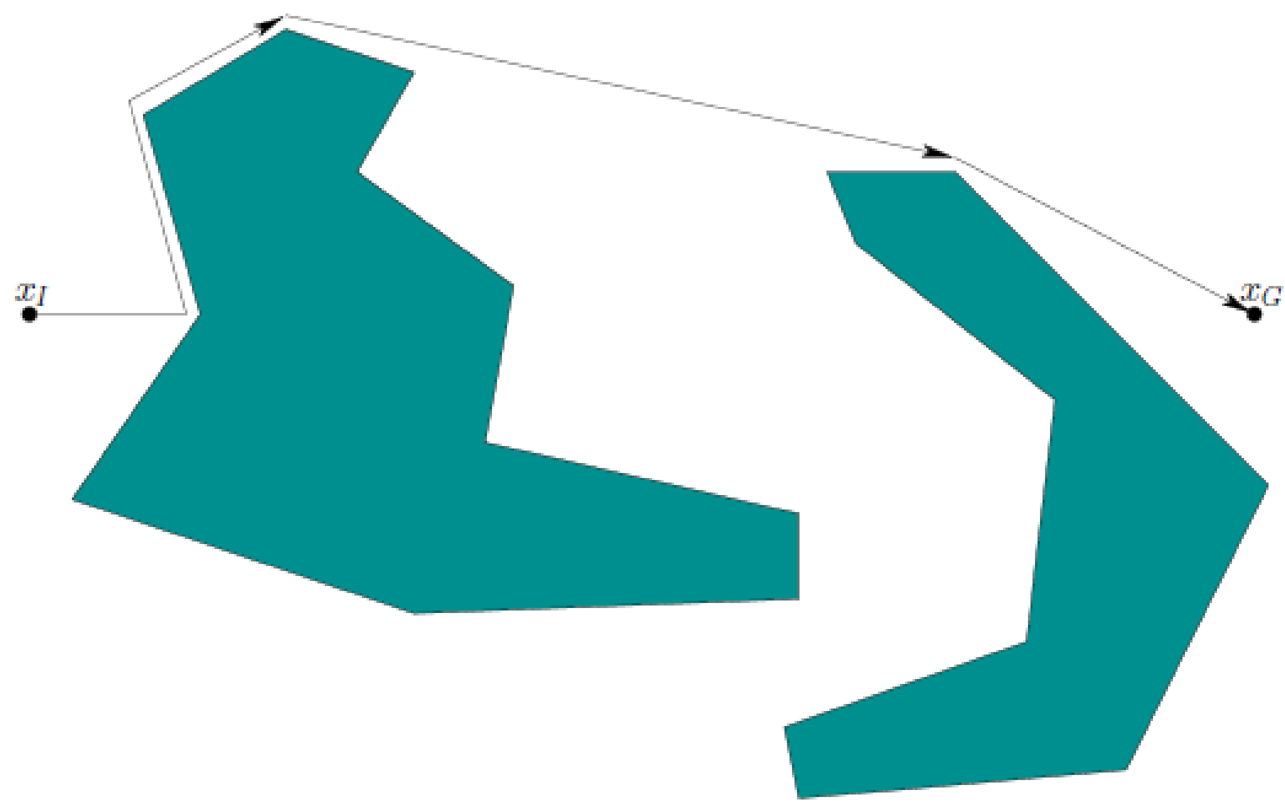


Image source: Planning Algorithms

Others Bug Algorithms

Many other bug algorithms: VisBug, TangentBug, …



More reading: https://www.cs.cmu.edu/~motionplanning/lecture/Chap2-Bug-Alg_howie.pdf

Image source: Planning Algorithms

Gap Navigation Trees (I)

Gap navigation tree (GNT) is another type of navigation algorithm without a map of the environment (similar to bug algorithms)

What is a **gap**?

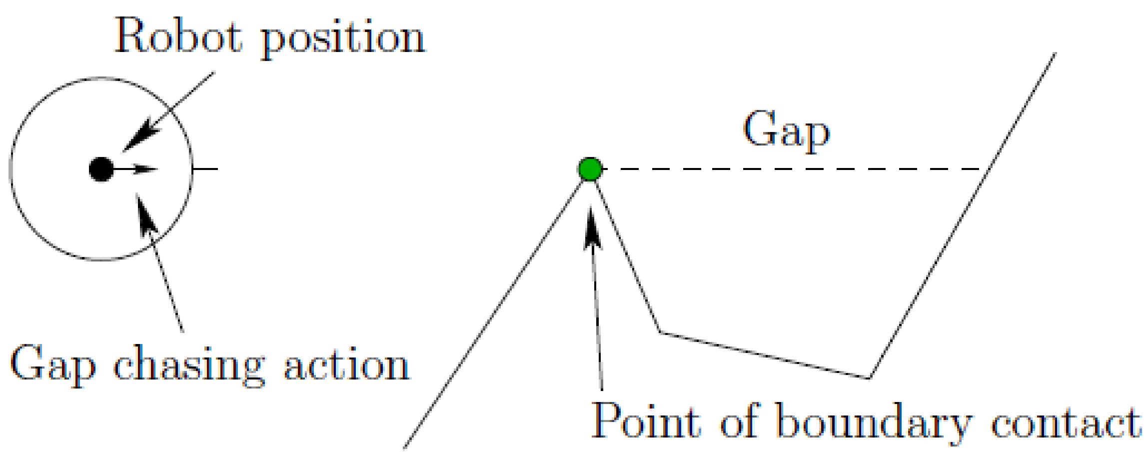
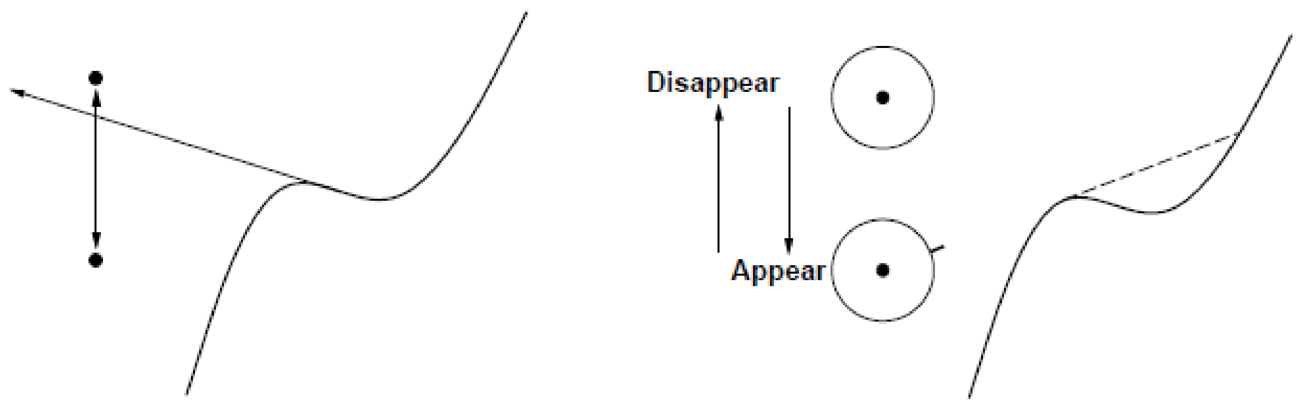


Image source: Planning Algorithms

Gap Navigation Trees (II)

Gaps apply to smooth boundary as well. Four types of **critical changes**

⇨Appear/disappear



⇨Split/Merge

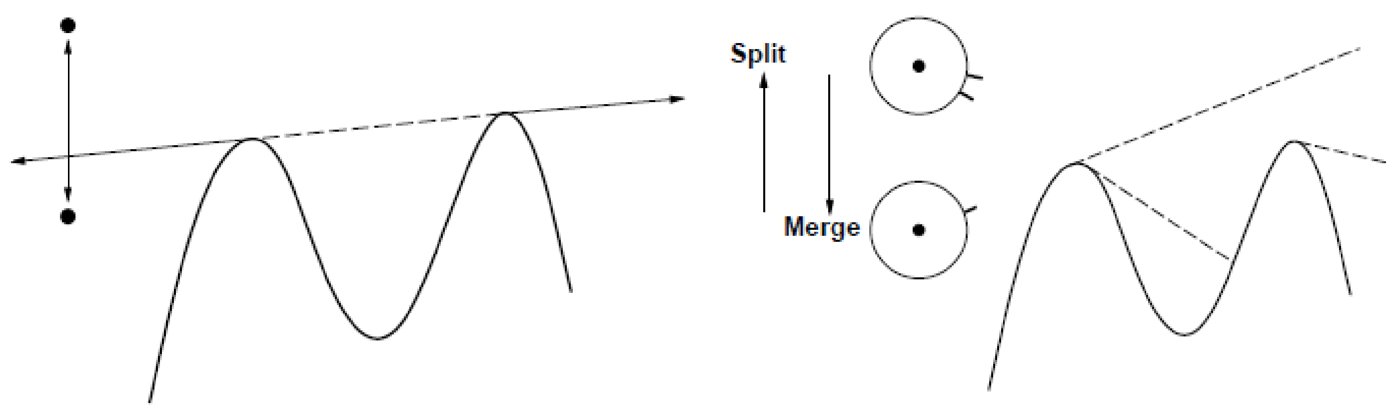


Image source: Planning Algorithms

Gap Navigation Trees (III)

Building GNT

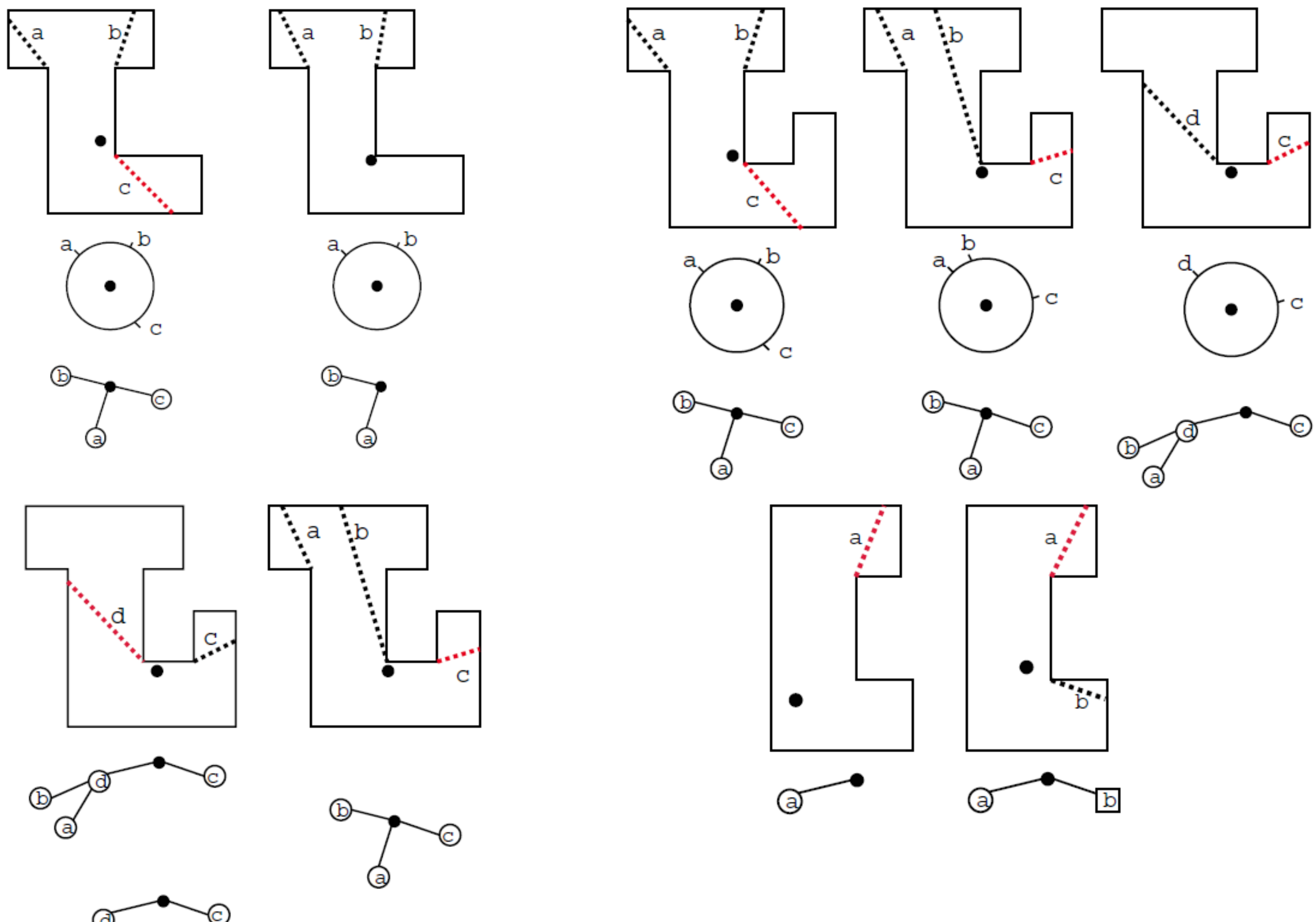


Image source: Planning Algorithms

Gap Navigation Trees (IV)

Using GNT, it becomes possible to navigate regions of an environment without the actual map of the environment via “chasing a gap”

- ⇨In particular, GNT induces equivalence class on the environments
- ⇨E.g., all following environments are equivalent under GNT

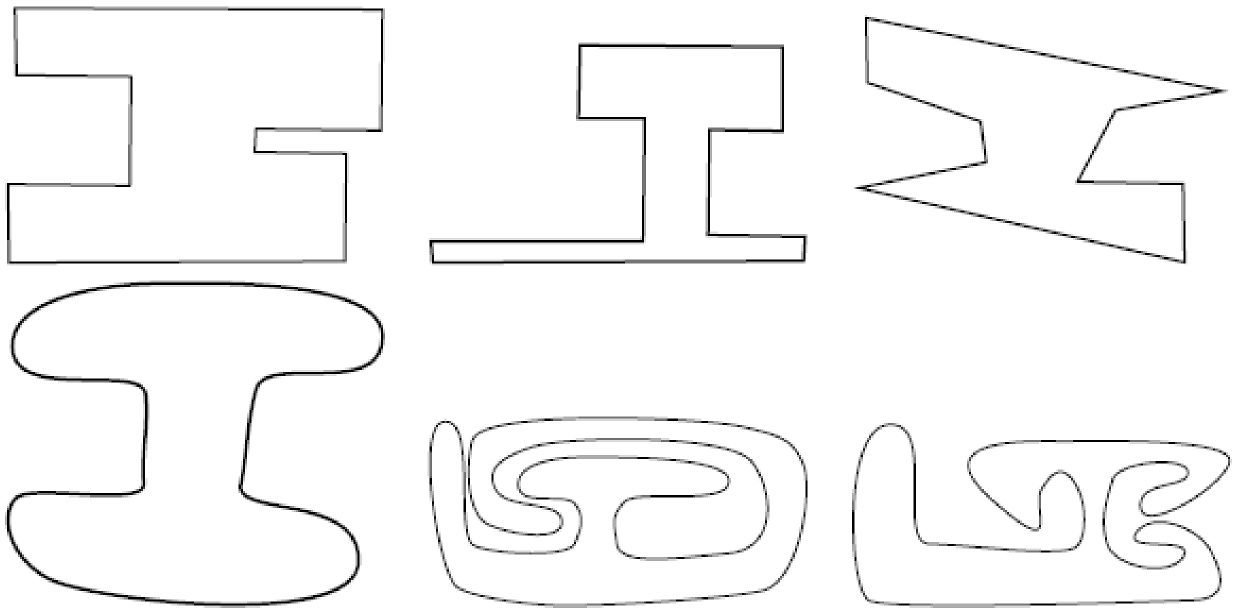


Image source: Planning Algorithms