GRAPH SEARCH

VIDEO LINKS

https://youtube.com/shorts/OULAR3OHgPo?si=QKEJgwPlQbiaUOUb
https://youtu.be/csifG1AM5d8?si=inFe9RMVm4eagxwN

https://youtube.com/shorts/g-fqy3YaN8E?si=lgTqfYCciu6qrzs-

PATH PLANNING

- Before robots could move on their own, scientists studied path planning for factory robots.
- Factory robots, known as **manipulator robots**, have arms with multiple joints, similar to a human arm.
- Planning the movement of a manipulator robot is **more complex than** planning a path for a wheeled robot on flat ground.

WHY IS IT HARDER?

- A robotic arm has many possible movements (degrees of freedom), while a wheeled robot mainly moves forward, backward, and turns left or right.
- **Factory robots** must move **very fast** to keep up with production, so their speed and force (dynamics) are important in path planning.
- In contrast, **mobile robots** usually **move slowly**, so their speed is not a major concern when planning their paths.

WHAT IS CONFIGURATION SPACE OR C-SPACE?

Path planning for robots, whether manipulator robots (robotic arms) or mobile robots (wheeled robots), is often done using a concept called **configuration space (C-space).** This is a **way to represent all possible positions and orientations of a robot in a mathematical space**, making path planning easier.

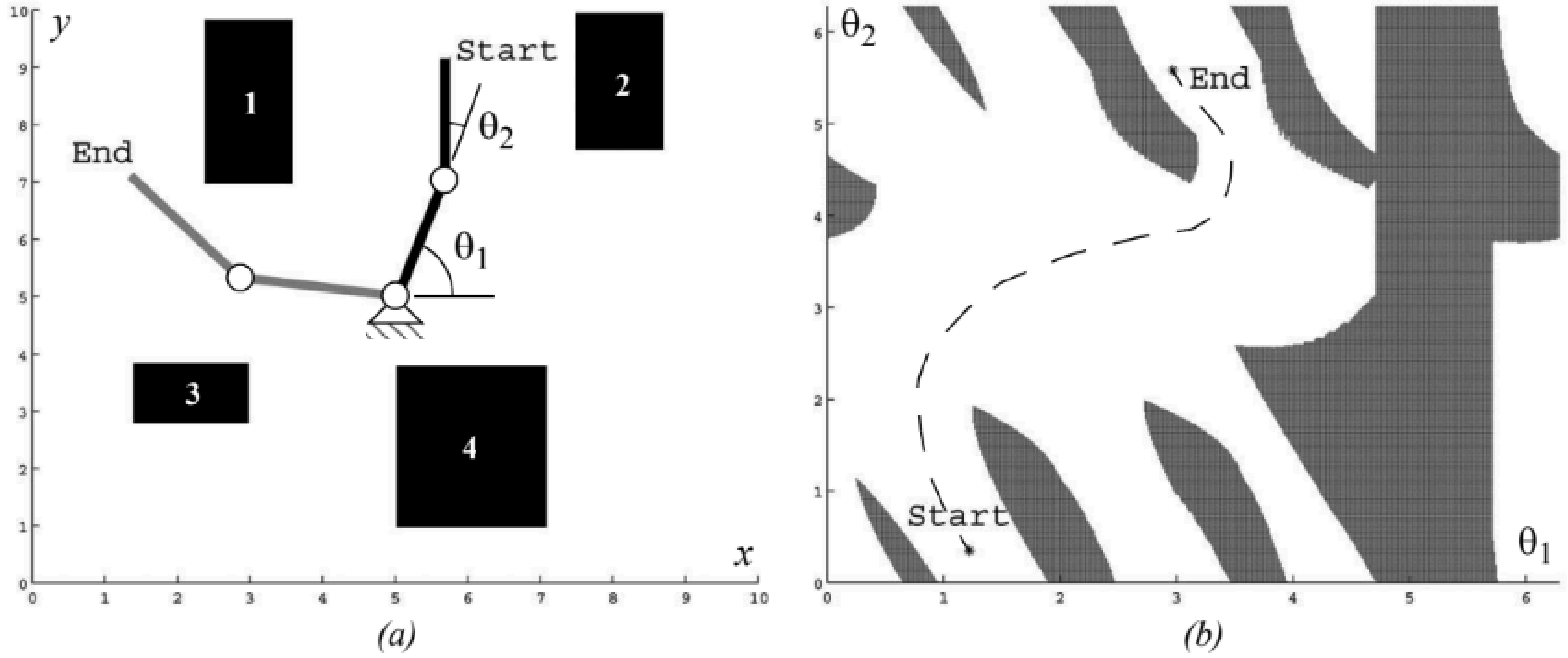


Figure 6.1
Physical space (a) and configuration space (b): (a) A two-link planar robot arm has to move from the configuration *start* to *end*. The motion is thereby constraint by the obstacles 1 to 4. (b) The corresponding configuration space shows the free space in joint coordinates (angle θ_1 and θ_2) and a path that achieves the goal.

- Consider a robotic arm with two joints.
- Each joint's angle can be represented by a number: **θ_1 and θ_2**.
- Together, these two values (θ_1, θ_2) form a point in a 2D configuration space.
- If the **arm touches an obstacle**, it means the robot has entered a **forbidden area** in configuration space.
- The remaining **free space in configuration space** is where the **robot can move safely**.
- **Path planning** in configuration space is **easier** than figuring out movement directly in the **physical world**.

PATH-PLANNING OVERVIEW

- The **robot’s environment** can be represented in different ways, **from continuous geometric descriptions to topological maps**.
- The **first step** in path planning is to **convert the environment into a discrete map suitable for the algorithm.**

Two main path-planning strategies:

1. **Graph Search:** a connectivity graph in free space is first constructed and then searched.The graph construction process is often performed offline.

2. **Potential field planning**: a mathematical function is imposed directly on the free space.The gradient of this function can then be followed to the goal.

GRAPH SEARCH

- **Graph search** is a way for robots to find the **best path from one place to another.**
- Imagine you are playing a game where you have to move from one place to another.
- The robot does the same thing—it needs to find a safe path from its starting point to its goal while avoiding obstacles.
- To help the robot, we **create a graph**, which is like a map made of points (called **nodes**) and lines (called **edges**) that show possible paths.

There are two main steps in this process:

1. **Making the Graph (Graph Construction)**
   - First, we look at the space and decide where the robot can and cannot go.
   - We place points (nodes) and connect them with lines (edges) to create paths the robot can follow.
   - The challenge is to make sure the robot can move anywhere in open space while keeping the graph simple.

2. **Finding the Best Path (Graph Search)**

- Once we have a graph, we use different methods to help the robot find the best path to its goal.
- Some methods focus on making the trip as short as possible, while others focus on keeping the robot safe from obstacles.

Graph construction

1. Visibility Graph – Shortest Path Solution

A **visibility graph** is a method where we connect every pair of points (nodes) that can "see" each other without any obstacles blocking their line of sight.

- The edges in this graph are the shortest possible paths.
- The robot then searches for the best route along these edges using a graph search algorithm.
- The method is optimal in terms of path length.

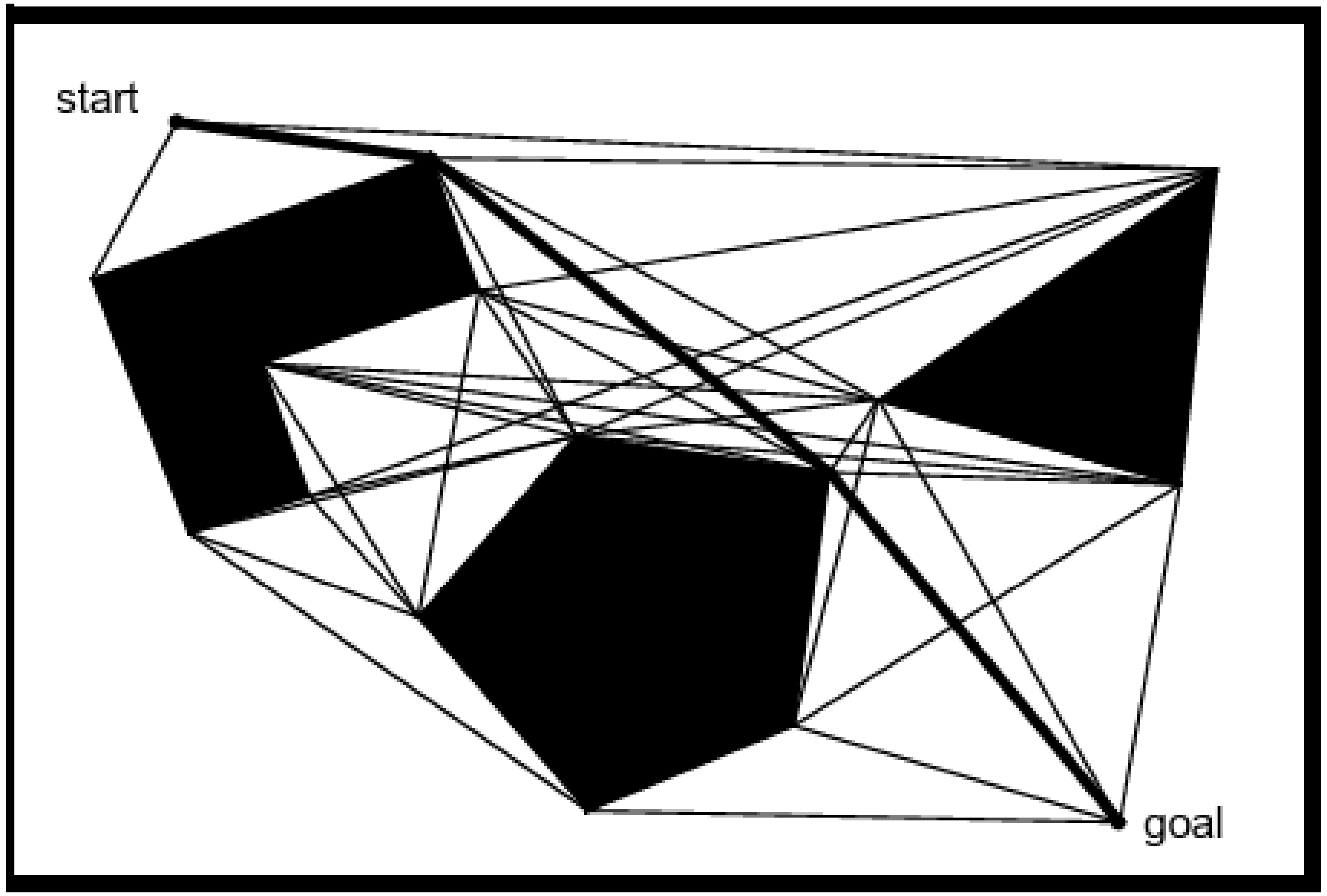


Figure 6.2
Visibility graph [32]. The nodes of the graph are the initial and goal points and the vertices of the configuration space obstacles (polygons). All nodes which are visible from each other are connected by straight-line segments, defining the road map. This means there are also edges along each polygon's

there are two main problems with visibility graphs:

- **Obstacle Proximity Issue** – Since it always takes the shortest path, the robot moves as close as possible to obstacles. This can be dangerous.
- **Computational Complexity** – If there are too many obstacles, the number of edges and nodes increases significantly, making the path search slower.

Note: A common solution to the first problem is to **grow obstacles** (inflate their boundaries) to create a safer path or modify the solution path afterward.

2. Voronoi Diagram – Maximum Clearance Solution

A **Voronoi diagram** is the opposite of a visibility graph. Instead of getting close to obstacles, it tries to maximize the distance between the robot and obstacles.

- It works by calculating the distance of each point in the free space to the nearest obstacle.
- The edges of the Voronoi diagram are formed by the **locus of points that are equidistant from two or more obstacles**.
- This creates a roadmap that keeps the robot as far away as possible from obstacles.

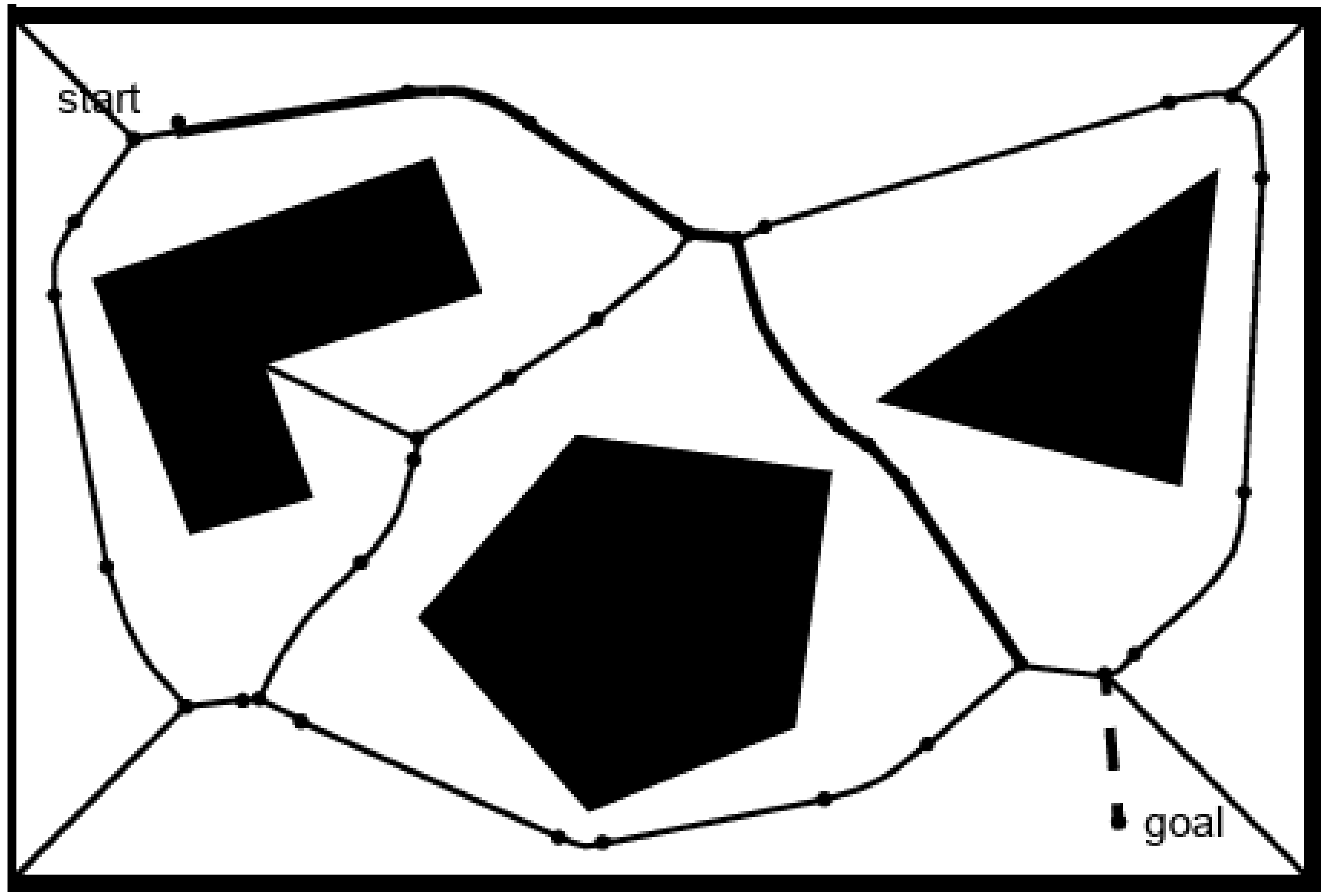


Figure 6.3
Voronoi diagram [32]. The Voronoi diagram consists of the lines constructed from all points that are equidistant from two or more obstacles. The initial q_{init} and goal q_{goal} configurations are mapped into the Voronoi diagram to q'_{init} and q'_{goal}, each by drawing the line along which its distance to the boundary of the obstacles increases the fastest. The points on the Voronoi diagram represent tran-

3. Cell Decomposition – Dividing the Space

Cell decomposition methods break the environment into free and occupied regions.

Exact Cell Decomposition

- The space is divided **precisely**, ensuring a **lossless** representation of the environment.
- A graph is then built by connecting neighboring free cells.

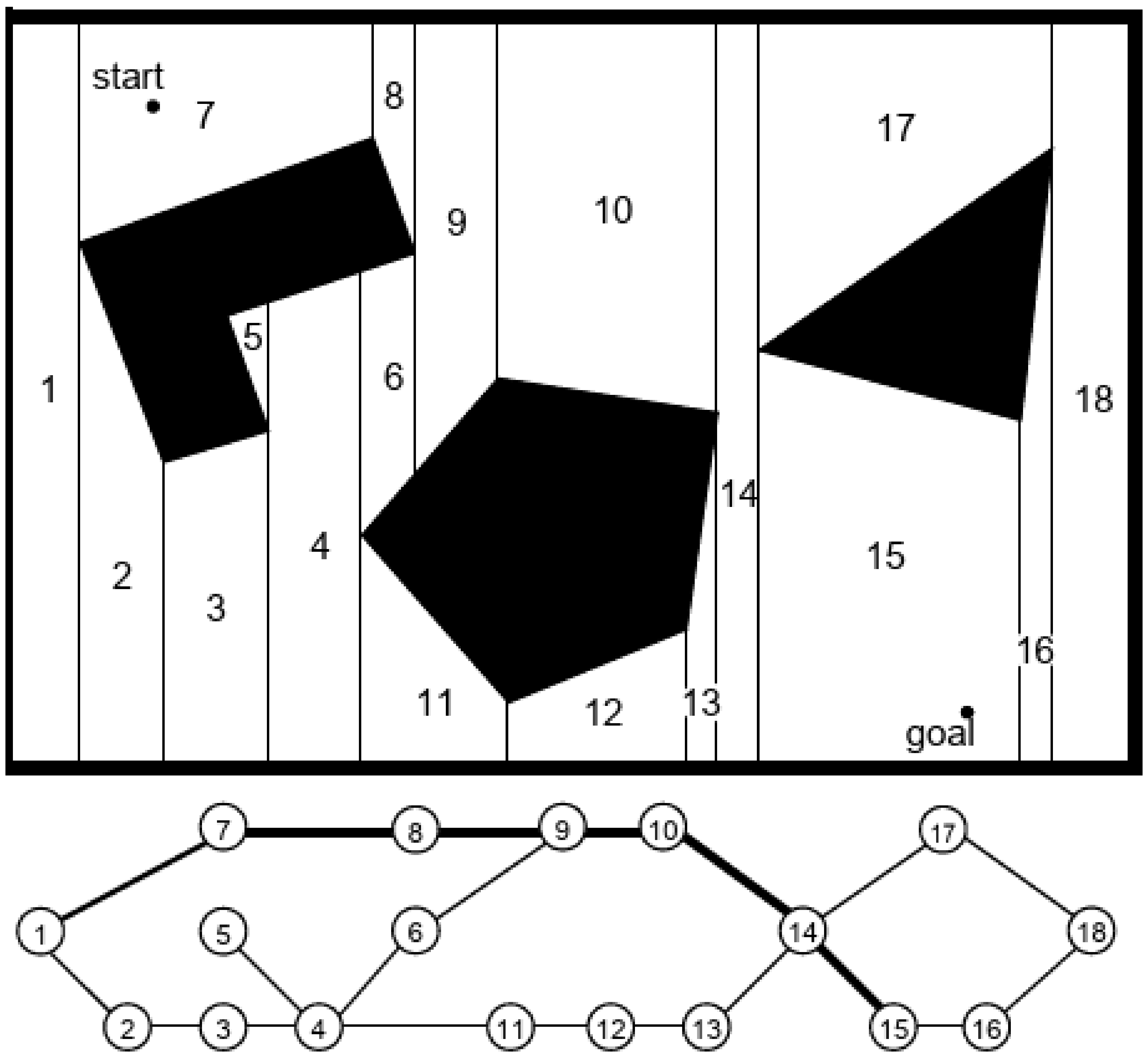
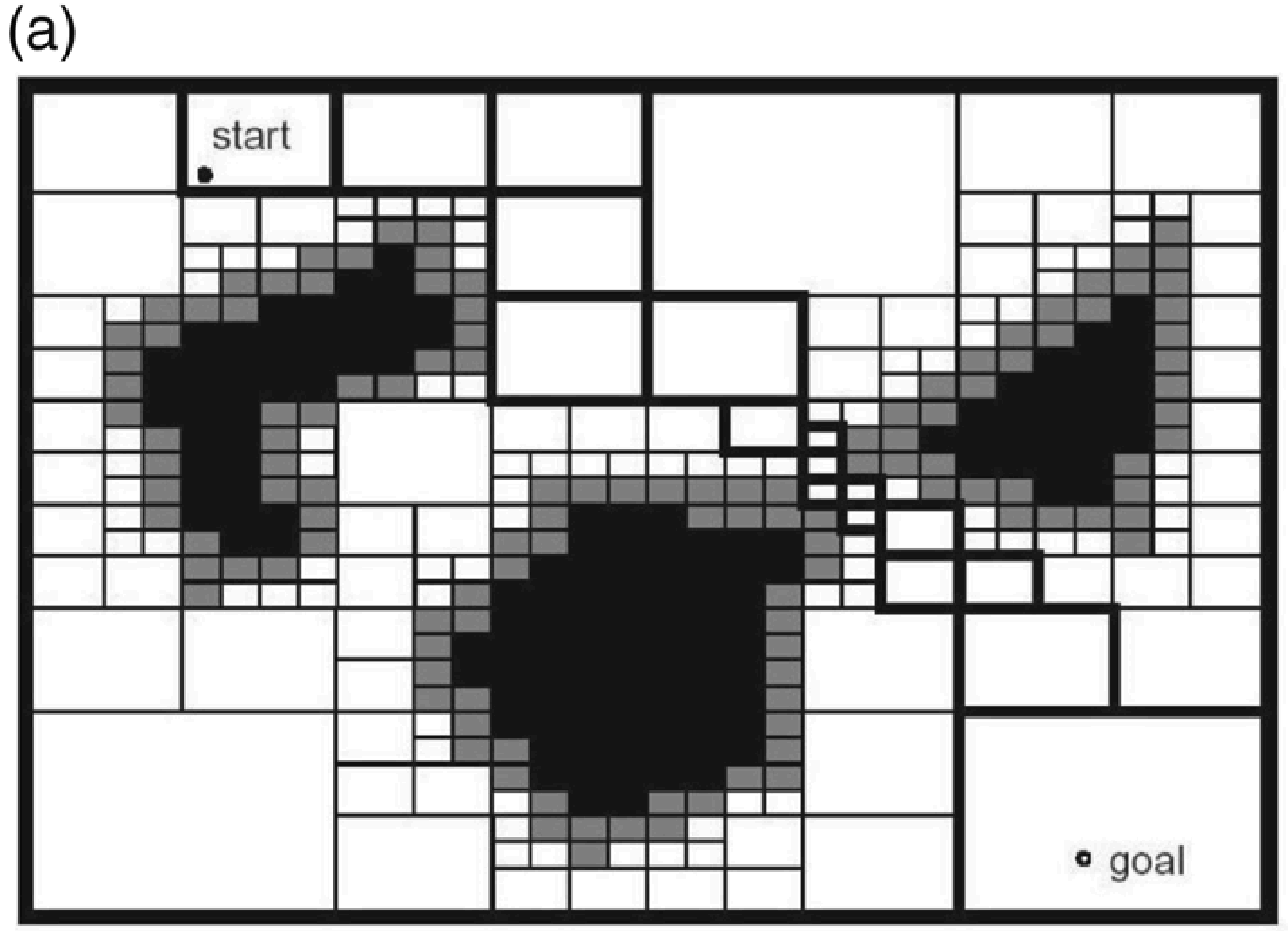


Figure 6.4
Example of exact cell decomposition. Cells are for example divided according to the horizontal coordinate of extremal obstacle points.

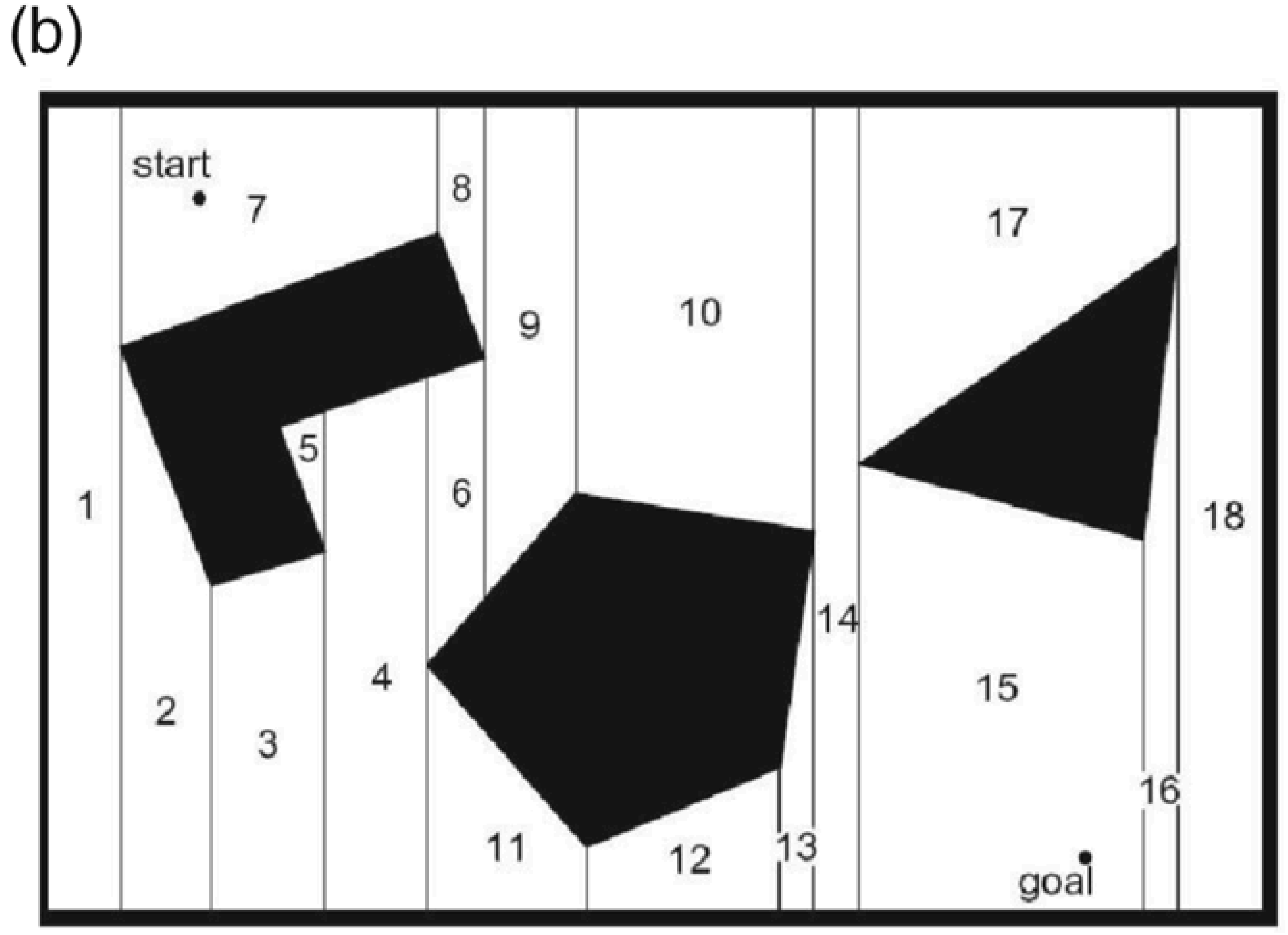
Approximate Cell Decomposition

The space is divided into **predefined geometric** shapes (like squares).

- Some details may be lost, making it an **approximate** solution.
- It is computationally faster than exact decomposition.
- The robot moves between these connected cells to find a valid path.



a) Approximate cell decompostion



b) Exact cell decompostion

4. Lattice Graph – Structured Grid-Based Planning

A **lattice graph** is a type of roadmap where the environment is overlaid with a structured grid.

- The edges are formed by shifting a **base set of edges** over the free space.
- The design ensures that paths are **executable** based on the robot’s motion model.

Since lattice graphs are designed according to the robot’s kinematics, they are directly executable without additional processing.

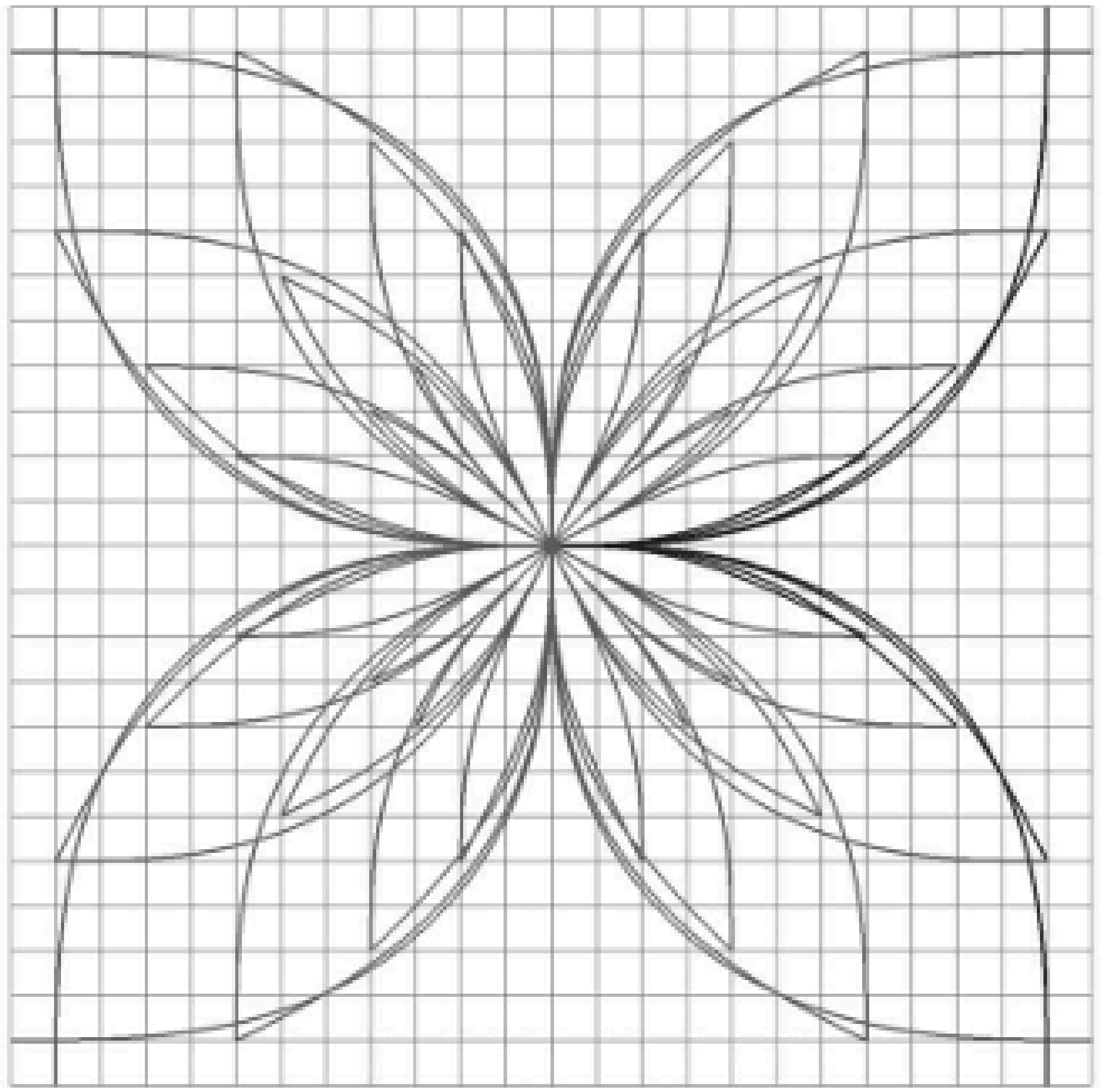


Figure 6.5 16-directional state lattice constructed for a planetary exploration rover. The state includes 2D position, heading, and curvature (x, y, θ, k). Note that straight segments are part of the lattice set but occluded by longer curved segments. The lattice is 2D-shift invariant and partially

THANK YOU