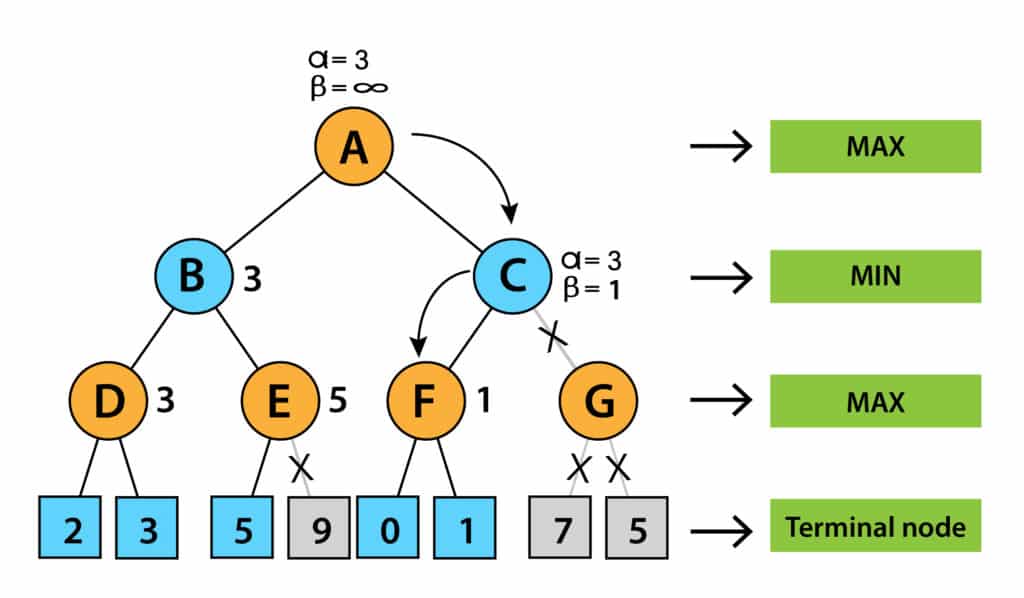
**Min-Max** and **Alpha beta pruning**

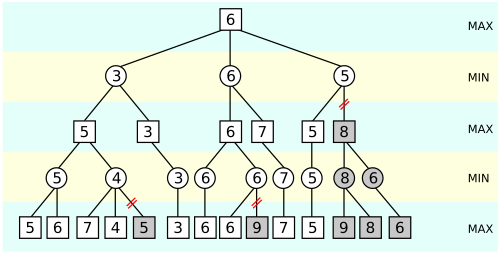
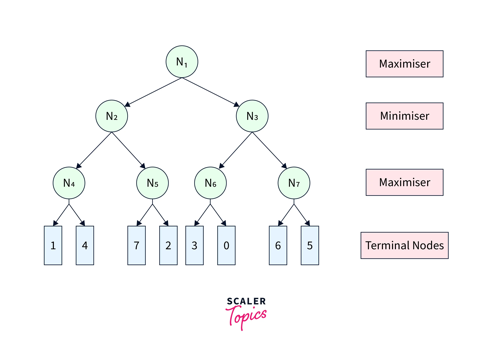
The minimax algorithm is a decision-making algorithm, primarily used in game theory and artificial intelligence, to find the optimal move for a player in a two-player, turn-based, zero-sum game. It assumes that both players are playing optimally, meaning each player aims to maximize their own gain while minimizing their opponent's.

**Alpha-beta pruning** is a search algorithm that optimizes the minimax algorithm by reducing the number of nodes that need to be evaluated in a game tree

How to do alpha-beta pruning?

* Assign the top most layer as the max, the second as min and continue likewise
* Start with the root and explore the left child.
* If the parent is empty, explore the other branch



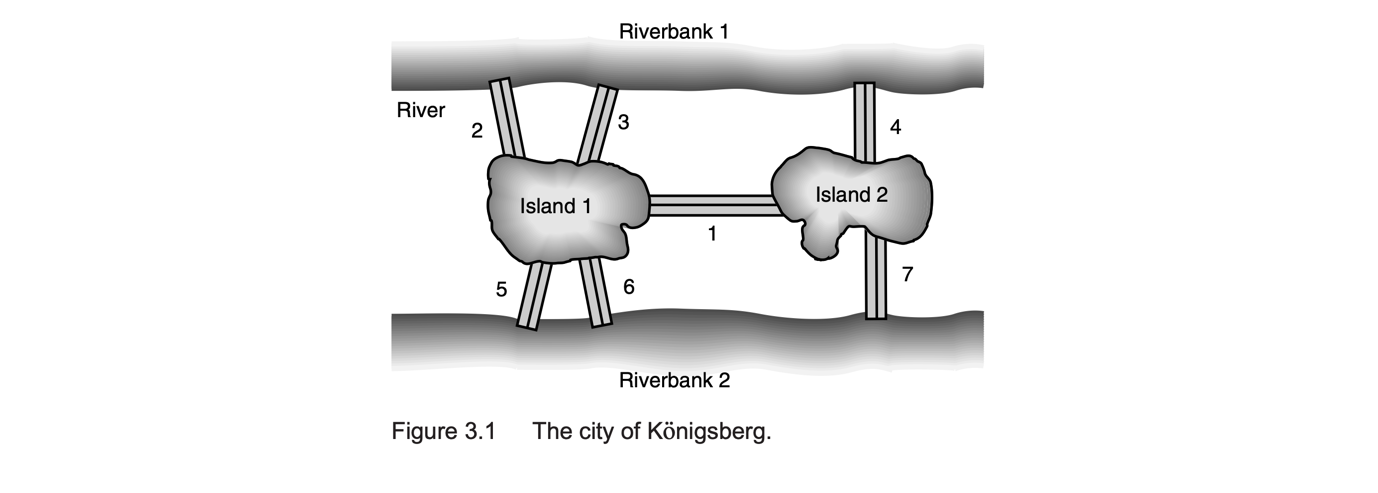
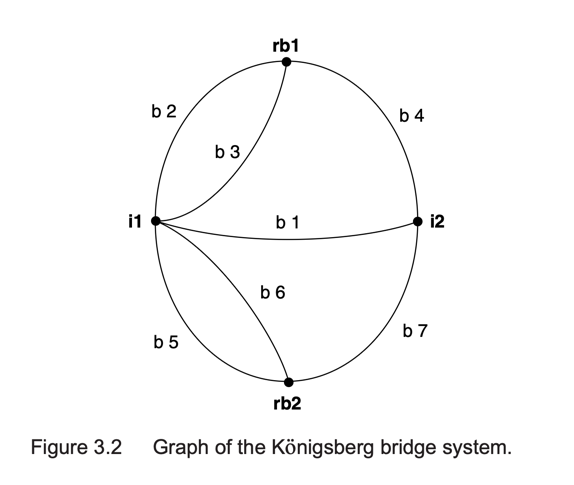
**Structures and Strategies for state space search.**

The theory of state space search uses graph theory to model and solve problems by representing them as graphs with nodes (states) and arcs (transitions).

Each node represents a specific state in the problem-solving process, such as game positions or logical conclusions, while arcs represent actions or inferences that move between states.

This approach is especially useful in **expert systems** where **rules generate new knowledge**.

Graph theory, originally developed by Leonhard Euler to solve the “bridges of Konigsberg” problem, remains a powerful tool for analysing the structure and relationships within such problems. The city of Königsberg occupied both banks and two islands of a river. The islands and the riverbanks were connected by seven bridges, as indicated in Figure 3.1.

**Structures** refer to the way we organize and represent the *state space*. This is typically done using graphs or trees, where **nodes** represent possible states of the problem and **edges** (or arcs) represent actions or transitions between these states. The starting point (**initial state**), the desired outcome (**goal state**.

**Strategies** are the methods or algorithms used to explore the state space in search of a solution. These strategies determine the order in which nodes are visited and include **uninformed search strategies** (such as breadth-first search and depth-first search) and **informed search strategies** (such as best-first search and A\*). Each strategy has its own strengths, weaknesses, and suitability depending on the nature of the problem, such as its size, complexity, and whether or not heuristic information is available.

Definition of a **State Space Search**

A state space is represented by a four-tuple [N,A,S,GD], where:

* N is the set of nodes or states of the graph. These correspond to the states in a problem-solving process.
* A is the set of arcs between nodes. These correspond to the steps in a problem-solving process.
* S, a nonempty subset of N, contains the start state(s) of the problem.
* GD, a nonempty subset of N, contains the goal state(s) of the problem.

The states in GD are described using either:

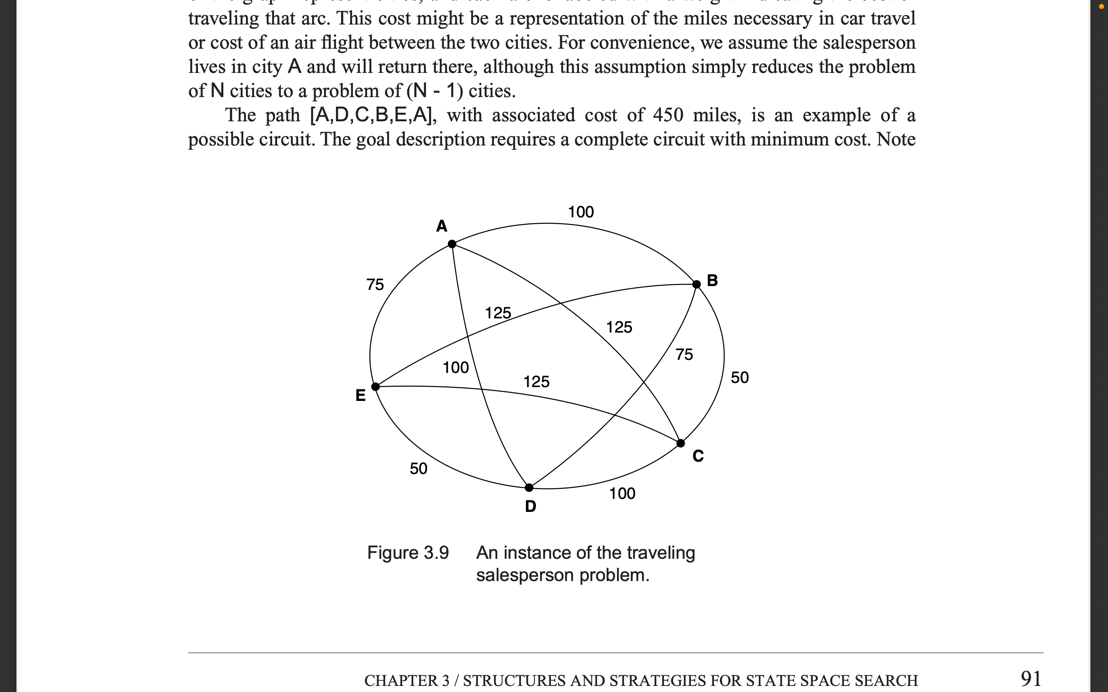
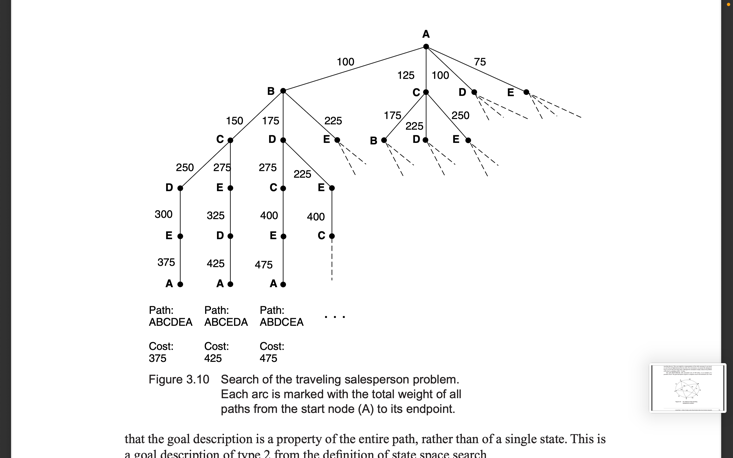
* A measurable property of the states encountered in the search.
* A measurable property of the path developed in the search, for example, the sum of the transition costs for the arcs of the path.

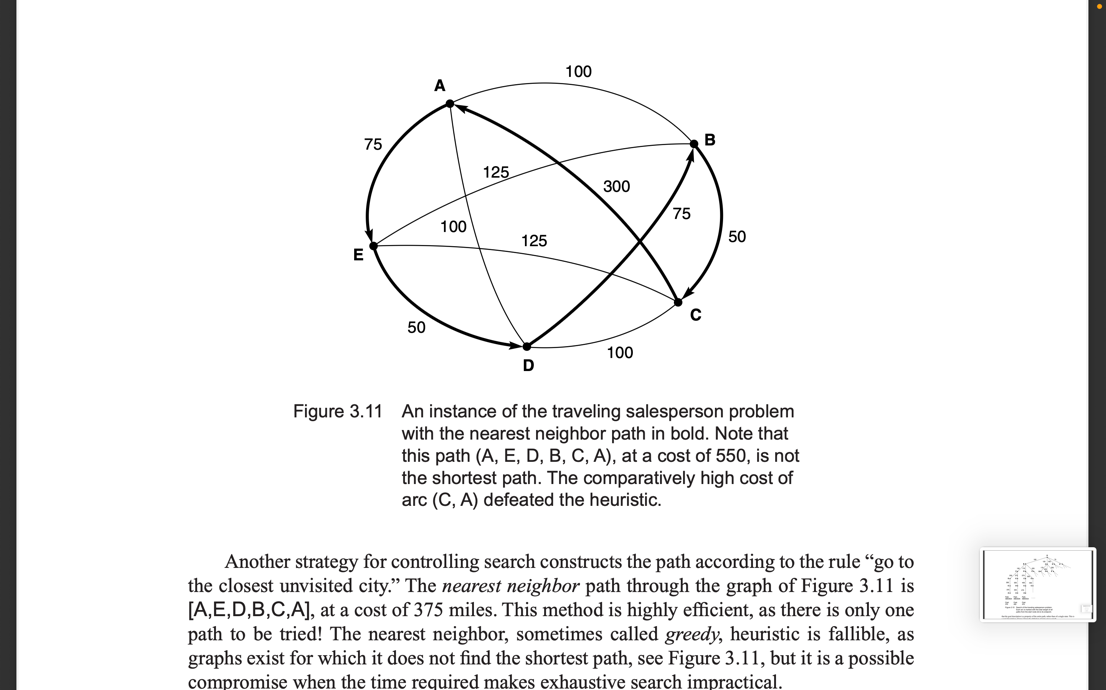
A solution path is a path through this graph from a node in S to a node in GD

**EXAMPLE 3.1.3: THE TRAVELING SALESPERSON**

Suppose a salesperson has five cities to visit and then must return home. The goal of the problem is to find the shortest path for the salesperson to travel, visiting each city, and then returning to the starting city.

Figure 3.9 gives an instance of this problem. The nodes of the graph represent cities, and each arc is labelled with a weight indicating the cost of traveling that arc. **Assume the salesperson lives in city A and will return there.**

** **

Nearest neighbour approach:

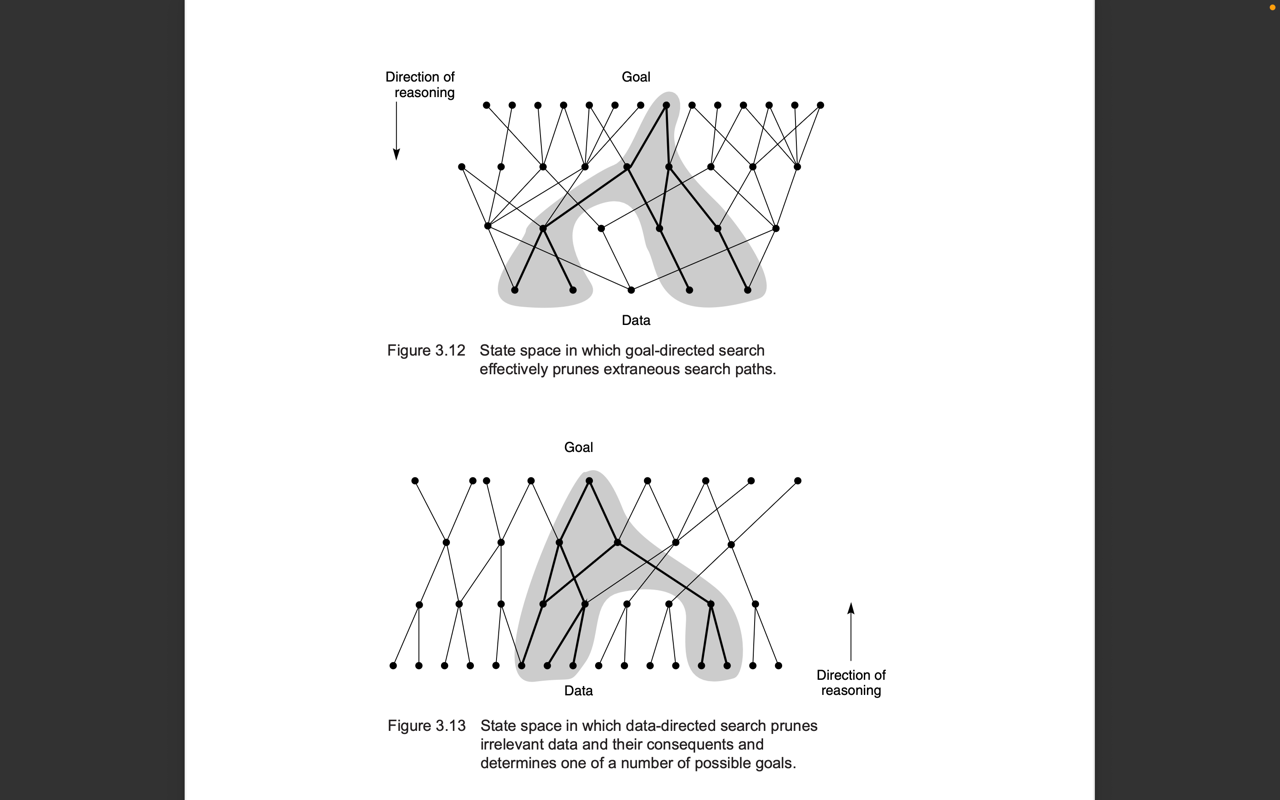
Strategies for State Space Search

A state space may be searched in two directions: from the given data of a problem instance

toward a goal or from a goal back to the data.

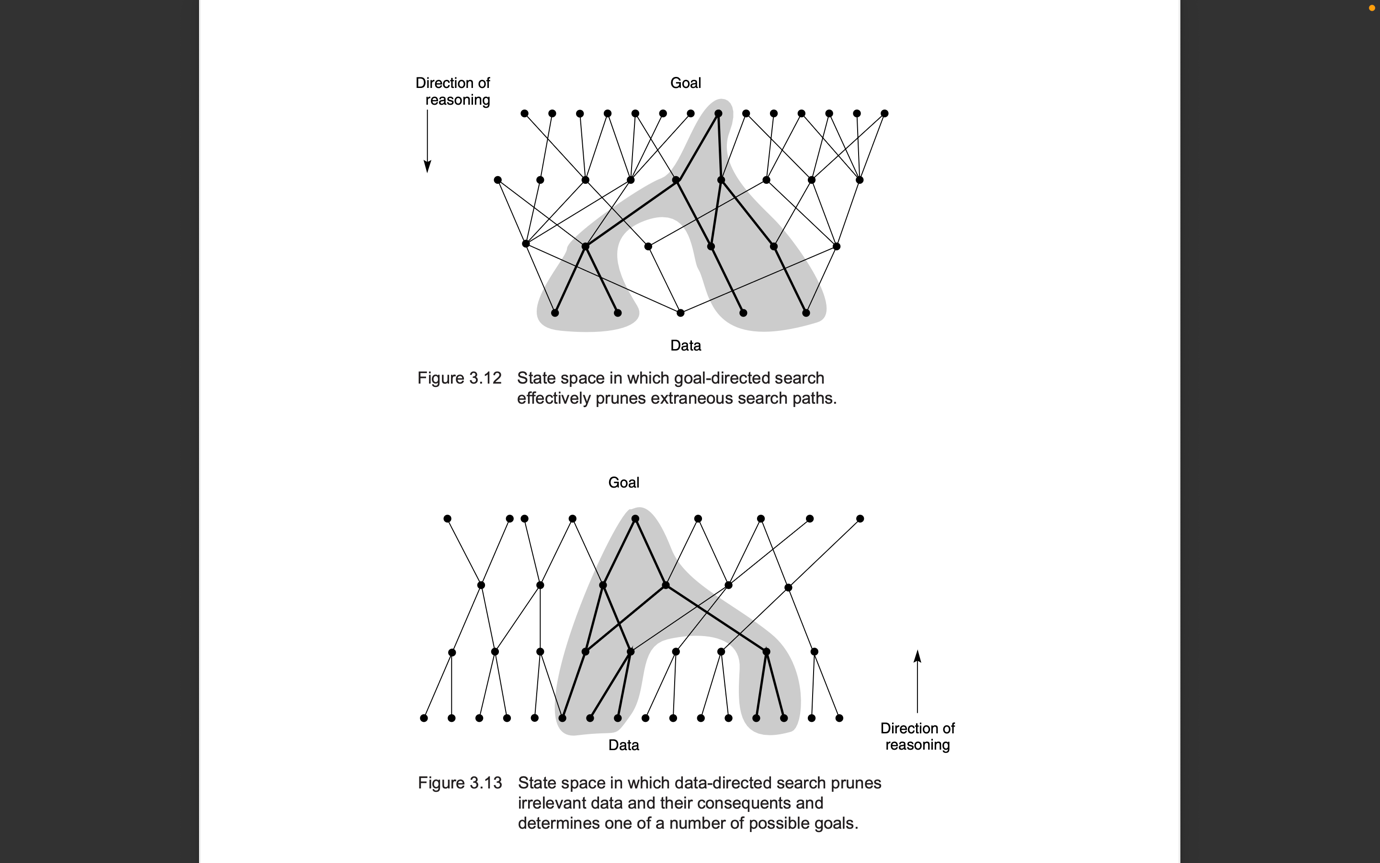
In data-driven search, sometimes called forward chaining,

* The problem solver begins with the given facts of the problem and a set of legal moves or rules for changing state.
* Search proceeds by applying rules to facts to produce new facts, which are in turn used by
* the rules to generate more new facts.
* This process continues until it generates a path that satisfies the goal condition.



In goal-driven search, sometimes called backward chaining,

* Take the goal that we want to solve.
* See what rules or legal moves could be used to generate this goal and determine what conditions
* must be true to use them.
* These conditions become the new goals, or subgoals, for the search.
* Search continues, working backward through successive subgoals until it works back to the facts of the problem. This finds the chain of moves or rules leading from data to a goal, although it does so in backward order



**To summarize:** data-driven reasoning takes the facts of the problem and applies the rules or legal moves to produce new facts that lead to a goal; goal-driven reasoning focuses on the goal, finds the rules that could produce the goal, and chains backward through successive rules and subgoals to the given facts of the problem.