Study Unit 4: Structures and Strategies for State Space Search

Introduction

Predicate calculus

– provides a means to describe facts and relations in a problem domain mathematically

– Uses rules to infer new knowledge

• The inference rules define a space that is searched to find a problem solution

• State space search theory provides a visual approach for finding a solution to the space search problem

– Represent problem as a state graph

– Use graph theory to analyze the structure and complexity of the problem and search procedure

Historically many people link intelligence with the capability to solve a problem. Problem-solving is one of the factors that demonstrate intelligence. The classical approach that is used is “hit and trial” that check for various solutions to that problem. We use this approach to solve small problems.

For example:

Consider the maze searching problem. The mouse travels through one path and finds that the path leads to a dead-end, it then backtracks somewhat and goes along some other path and again finds that there is no way to proceed. It goes on performing such search, trying different paths to solve the problem until a sequence of turns in the maze takes it to the cheese. Hence, from all the solutions the mouse tries, the one that reached the cheese was the one that solved the problem.

Another Example:

Consider that a toddler is to switch on the light in a dark room. He sees the switchboard having a number of buttons on it. He presses one, nothing happens, he presses the second one, the fan

gets on, he goes on trying different buttons till at last the room gets lighted and his problem gets solved.

Generate and Test:

We call this "hit and trial" approach the "Generate and Test" approach. We generate different combinations to solve our problem, and the one which solves the problem is taken as the correct solution. The rest of the combinations that we try are considered as incorrect solutions and hence are destroyed.

Problem Representation:

For a simple problem or smaller problem, we can use a hit and trial approach to solve the problem. However, when the problem is complex then we need a systematic way to solve the problem. The key to solving a complex problem is its representation. Normally the problems are represented in the form of a diagram (such as graphs or trees).

Questions to consider when designing search algorithms

- Is the problem solver guaranteed to find a solution?
- Will the problem solver always terminate?
- When a solution is found, is it guaranteed to be optimal?
- What is the complexity of the search process?
- How can the interpreter most effectively reduce search complexity?
- How can the interpreter effectively utilize a representation language?

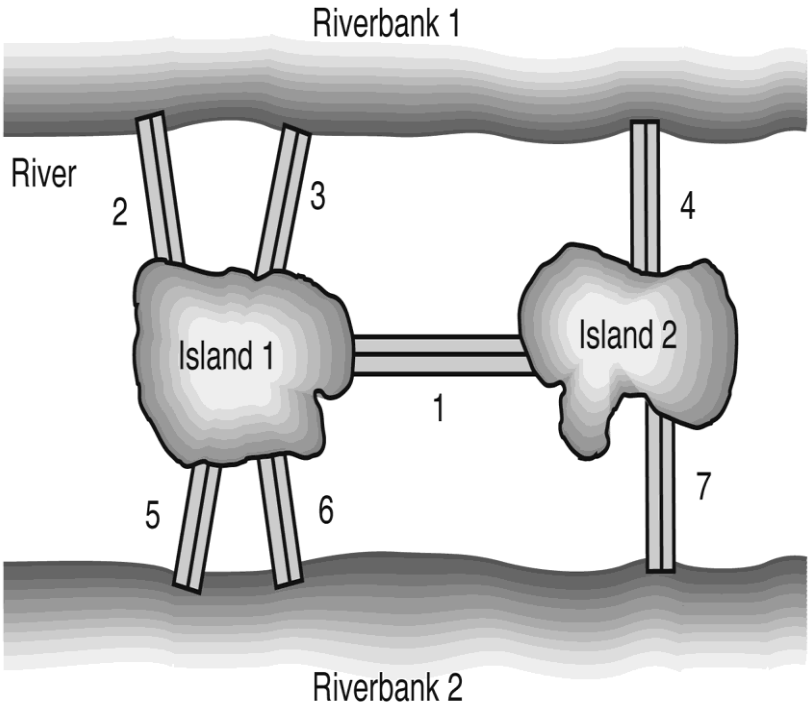
- State space search is the tool for answering these questions.

Graph Theory

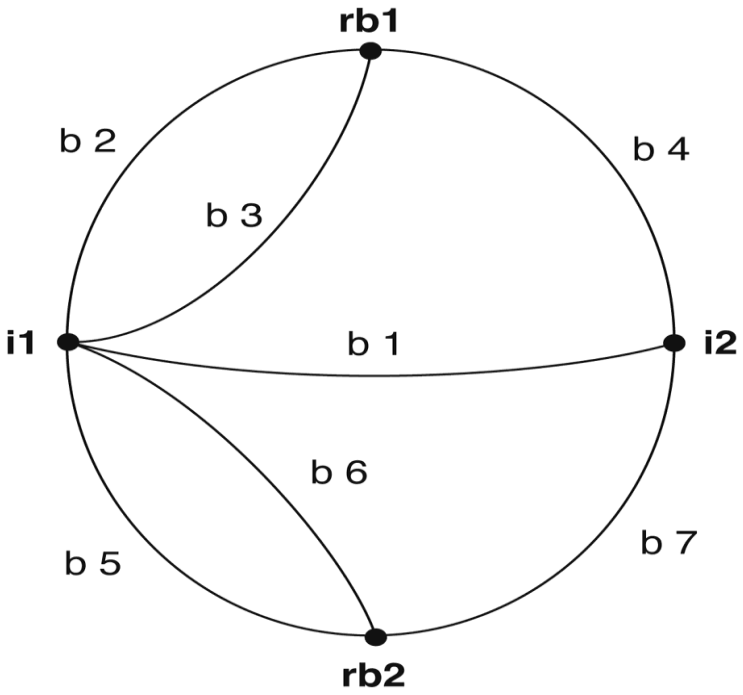
- A graph consists of nodes and a set of arcs or links connecting pairs of nodes.
- Nodes are used to represent discrete states.
  - A configuration of a game board. (tic-tac-toe)
- Arcs are used to represent transitions between states.
  - Legal moves of a game
- Leonhard Euler invented graph theory to solve the "bridge of Königsberg problem"

- Is there a walk around the city that crosses each bridge exactly once.

Konigsberg Bridge System



The city of Konigsberg occupied both banks and two islands of a river. The islands and the riverbanks were connected by seven bridges



- Euler created an alternative representation for the map.
- The riverbanks (rb1 and rb2) and islands (i1 and i2) are described by the nodes of a graph.
- The bridges are represented by labeled arcs between nodes (b1, b2, , b7).

Definition of Graph(1)

- A **graph** consists of nodes and arcs
  - A set of nodes N1, N2, …, Nn … need not be finite.
  - A set of arcs connects pairs of nodes.
- A ***directed graph*** has an indicated direction for traversing each arc(Fig. 3.3, p85)
- If a directed arc connects Nj and Nk, then Nj is called the parent of Nk and Nk is called the ***child*** of Nj.
- A ***rooted graph*** has a unique node Ns from which all paths in the graph originate
- A tip or leaf node is a node without child.
- An ordered sequence of nodes [N1, N2, N3, … Nn] is called a ***path*** of length n-1 in the graph
- On a path in a rooted graph, a node is said to be an ***ancestor*** of all nodes positioned after it (to its right) as well as a ***descendant*** of all nodes before it (to its left).

- A path that contains any node more than once is said to contain ***a cycle*** or ***loop***.
- A ***tree*** is a graph in which there is a unique path between every pair of nodes
- Two nodes in a graph are said to be ***connected*** if a path exists that includes both them.

State Space Representation of Problems(1)

- In the *state space representation* of a problem, the *nodes* of a graph corresponds to partial problem solution *states*, the *arcs* corresponds to *steps* in a problem-solving process.
- *State space search* characterize problem solving as the process of finding *a solution path* from the start state to a goal.

Components of Problem Solving:

- Problem Statement
- Goal State
- Solution Space
- Operators

First of all it must ensure that the given problem is guaranteed to be solve and it is guaranteed to solve with an infinite amount of time. In addition to time complexity, it should be noted that all the resources are available and in order to understand and find the solution all rules must be known.

Problem Solution/ Goal State: It means what is required as an output or what would be considered as the correct solution to the given problem. For example, in the mouse maze problem, the solution state is mouse reached the cheese.

Solution Space: Different states while following a strategy to reach goal state from start state is known as solution space.

For example

When the mouse was in the lower-left corner of the maze, it represents a state i.e. the start state. When it was stuck in some corner of the maze represents a state. When it was stuck somewhere

else represents another state. When it was traveling on a path represents some other state and finally when it reaches the cheese represents a state called the goal state. The set of the start state, the goal state, and all the intermediate states constitute something which is called a solution space.

Operator:

The traveling inside a solution space requires something called "operators". In the case of the mouse example, turn left, turn right, go straight are the operators which help us travel inside the solution space.

Definition: STATE SPACE SEARCH

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

- N is the set of states.
- A is the set of steps between states
- S is the start state(S) of the problem.
- GD is the goal state(S) of the problem.

The states in GD are described:

1. A measurable property of the states. (winning board in tic-tac-toe)
2. A property of the path.(shortest path in traveling sales man problem)

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

Example of how search is used to solve a more complicated problem, consider the task of diagnosing a mechanical fault in an automobile.

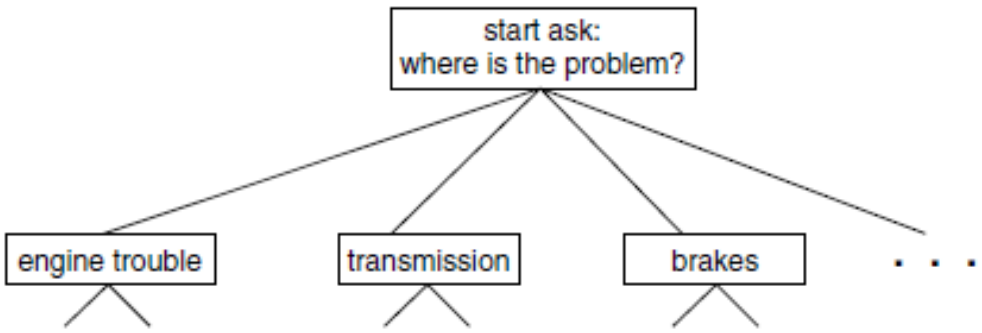
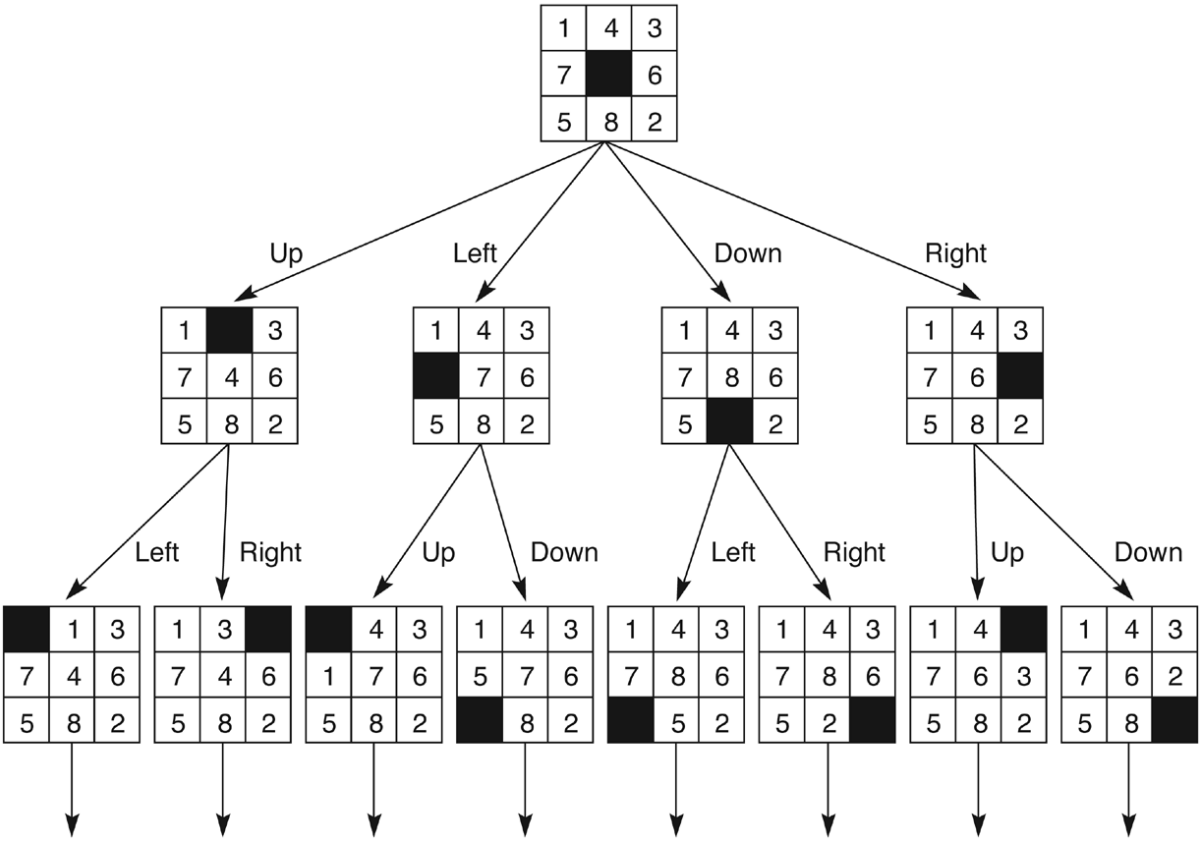


Figure II.6 State space description of the first step in diagnosing an automotive problem.

Although this problem does not initially seem to lend itself to state space search as easily as tic-tac-toe or chess, it actually fits this strategy quite well.

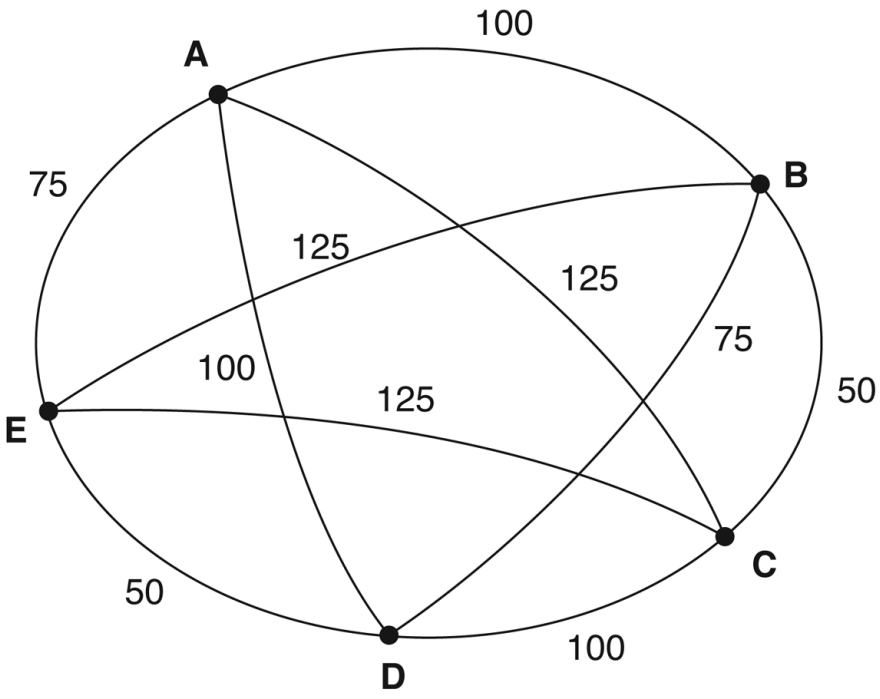
Instead of letting each node of the graph represent a "board state," we let it represent a state of partial knowledge about the automobile's mechanical problems. The process of examining the symptoms of the fault and inducing its cause may be thought of as searching through states of increasing knowledge. The starting node of the graph is empty, indicating that nothing is known about the cause of the problem. The first thing a mechanic might do is ask the customer which major system (engine, transmission, steering, brakes, etc.) seems to be causing the trouble. This is represented by a collection of arcs from the start state to states that indicate a focus on a single subsystem of the automobile

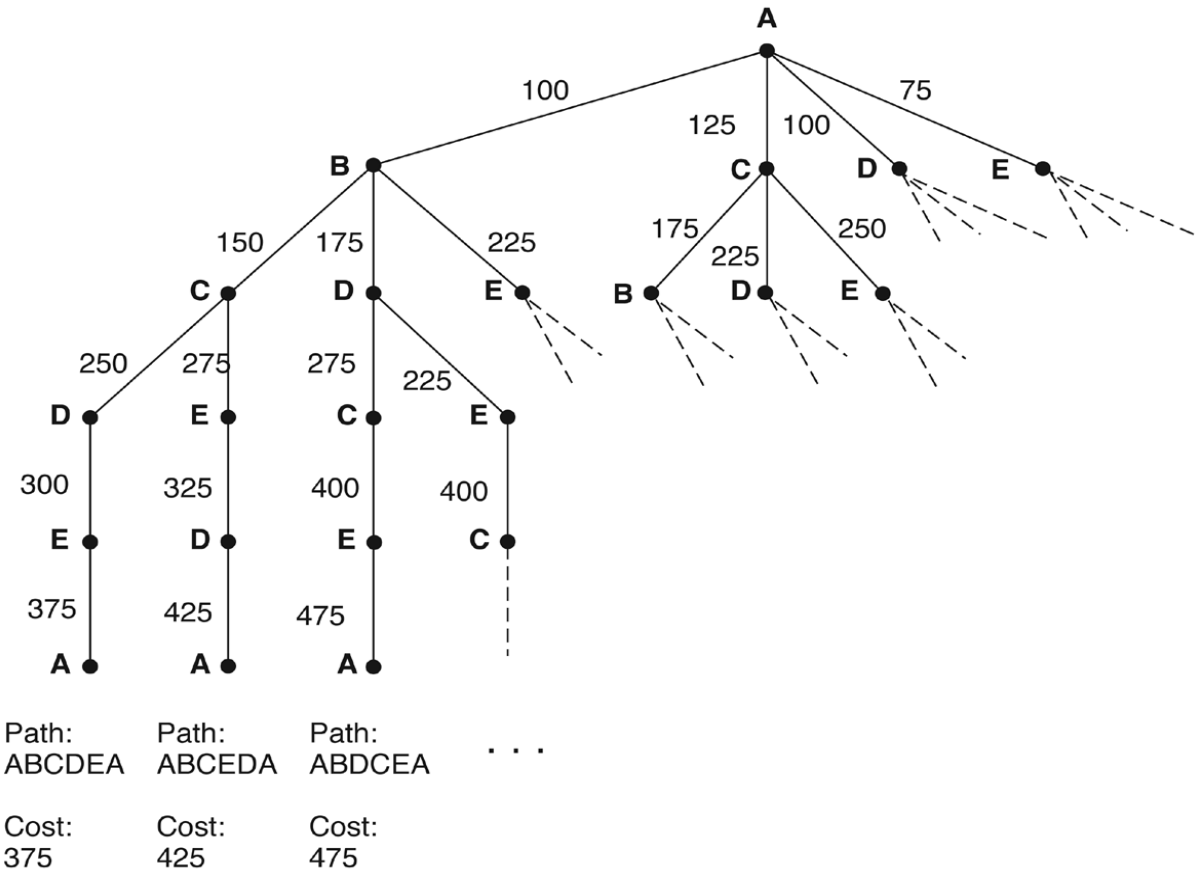


The travelling salesman problem

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.
- The goal description requires a complete circuit with minimum cost.
- The complexity of exhaustive search is (N-1)!, where N is the number of cities(Fig 3.8, p93)
- Techniques for reducing the search complexity.
    - Branch and Bound, The Nearest neighbor.





- **Branch and Bound technique**

  1. Generate paths while keeping track of the best path found so far.
  2. Use this value as a bound
  3. As paths are constructed one city at a time, examine each partially completed path by guessing the best possible extension of the path (the branch).
  4. If the branch has greater cost than the bound, it eliminates the partial path and all of its possible extensions.

- **The "Nearest Neighbor" technique**

  Go to the closest unvisited city.

  (A E D B C A) is not the shortest path

Data-Driven and Goal-Driven Search (1)

- *Data-driven search*(forward chaining) takes the facts of the problem and applies the rules and legal moves to produce new facts that lead to a goal.

- *Goal-driven search* (backward chaining) focused 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.

- Both problem solvers search the *same* state space graph. The *search order* and the actual *number of states searched can differ.*

Searching:

All the problems that we have looked at can be converted to a form where we have to start from a start state and search for a goal state by traveling through a solution space. Searching is a formal mechanism to explore alternatives.

Most of the solution spaces for problems can be represented in a graph where nodes represent different states and edges represent the operator which takes us from one state to the other. If we can get our grips on algorithms that deal with searching techniques in graphs and trees, we'll be all set to perform problem-solving in an efficient manner.