Problem Solving By Search

- Dr. Asaad Sabah Hadi

Structures and Strategies for State Space Search

- Introduction
- Graph Theory
  - Structures for state space search
  - state space representation of problems
- Strategies for state space search
  - Data-Driven and Goal-Driven Search
  - Implementing Graph Search
  - Depth-First and Breadth-First Search
  - Depth-First Search with Iterative Deepening
- Using the State Space to Represent Reasoning with the Predicate Calculus
  - State Space Description of a Logical System
  - AND/OR graphs
  - Examples and Applications

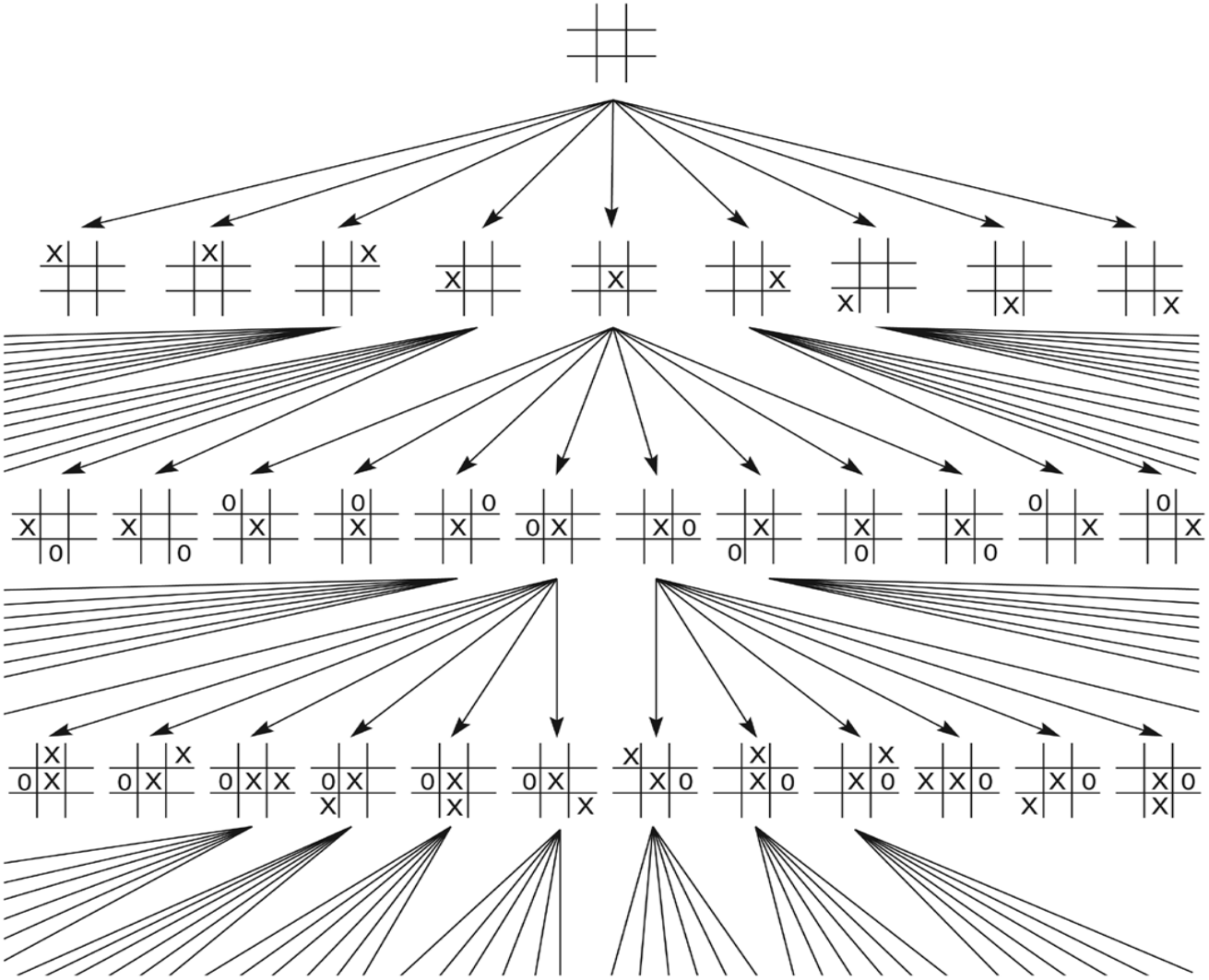
Introduction(1)

- Questions for 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.

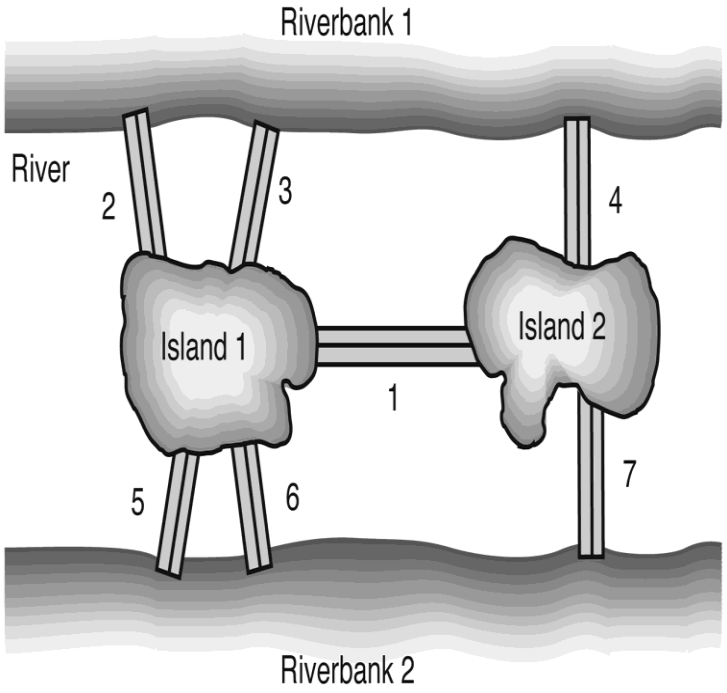
Introduction(2)

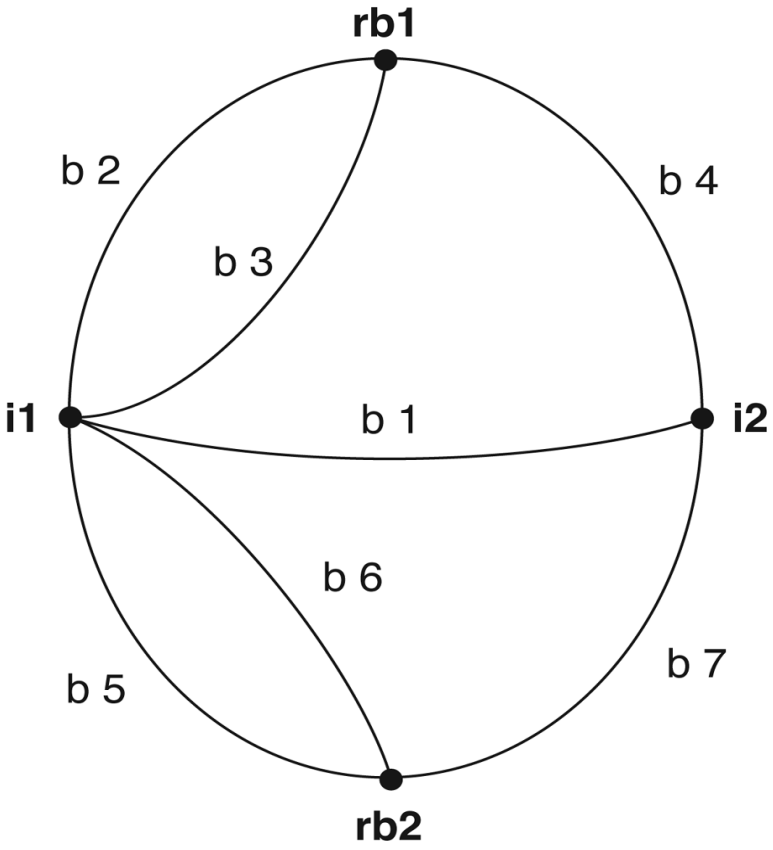
- A graph consists of nodes and a set of arcs or links connecting pairs of nodes.
- Nodes are used to represent discrete states.
  - A configuration of a game board. (tic-tac-toe)
- Arcs are used to represent transitions between states.
  - Legal moves of a game
- Leonhard Euler invented graph theory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

State space representation of Tic-tac-toe



Königsberg Bridge System





Königsberg Bridge System

- Euler focused on the degree of the nodes of the graph
  - Even degree node has an even number of arcs joining it to neighboring nodes.
  - Odd degree node has an odd number of arcs.
- Unless a graph contained either exactly zero or two nodes of odd degree, the walk was impossible.
  - No odd degree node: the walk start at the first and end at the same node
  - Two odd degree nodes: the walk could start at the first and end at the second

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
- 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 children.
- An ordered sequence of nodes [N1, N2, N3, … Nn] is called a path of length n-1 in the graph

Definition of Graph (2)

- 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 them both.

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.

State Space Representation of Problems(2)

- 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:

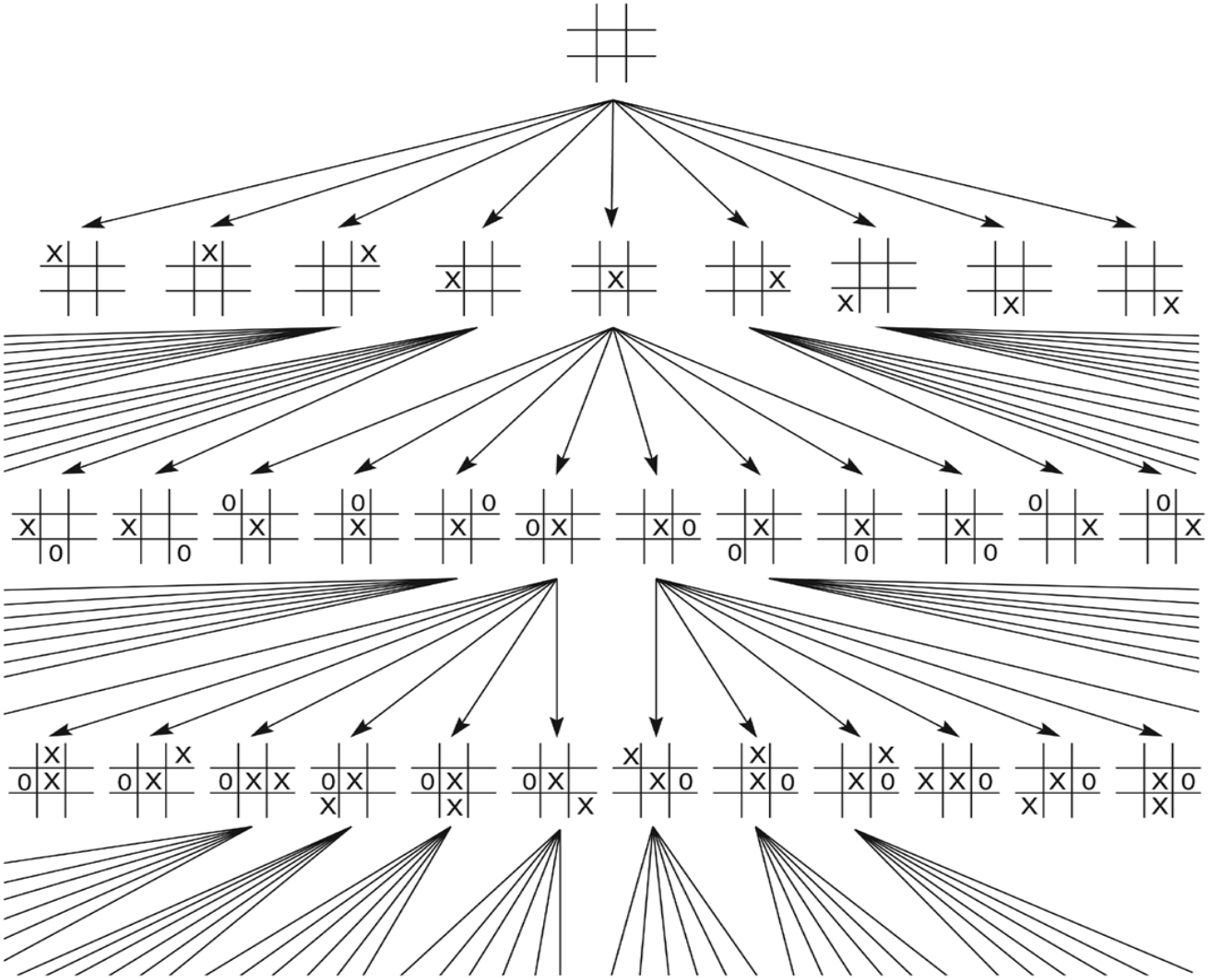
1. A measurable property of the states. (winning board in tic-tac-toe)
2. A property of the path.(shortest path in traveling sales man problem)

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

Tic-tac-toe (1)

- The set of states are all different configurations of Xs and Os that the game can have. [N]
  - 3^9 ways to arrange {blank, X, O} in nine spaces.
- Arcs(steps) are generated by legal moves of the game, alternating between placing an X and an O in an unused location [A]
- The start state is an empty board. [S]
- The goal state is a board state having three Xs in a row, column, or diagonal. [GD]

Tic-tac-toe (2)



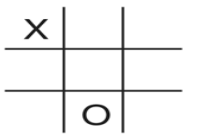
Tic-tac-toe (3)

- There are no cycles in the state space because the directed arcs of the graph do not allow a move to be undone.
- The complexity of the problem : 9!(362,880) different path can be generated.
  - Need heuristics to reduce the search complexity.

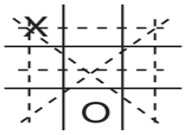
    e.g. My possible winning lines – Opponent's possible wining lines

Chess has 10^{120} possible game paths

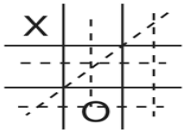
Possible Heuristic for Tic-tac-toe



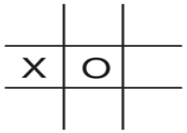
X has 6 possible win paths:



O has 5 possible wins:

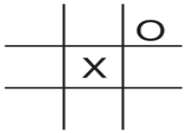


E(n) = 6 – 5 = 1



X has 4 possible win paths;
O has 6 possible wins

E(n) = 4 – 6 = –2



X has 5 possible win paths;
O has 4 possible wins

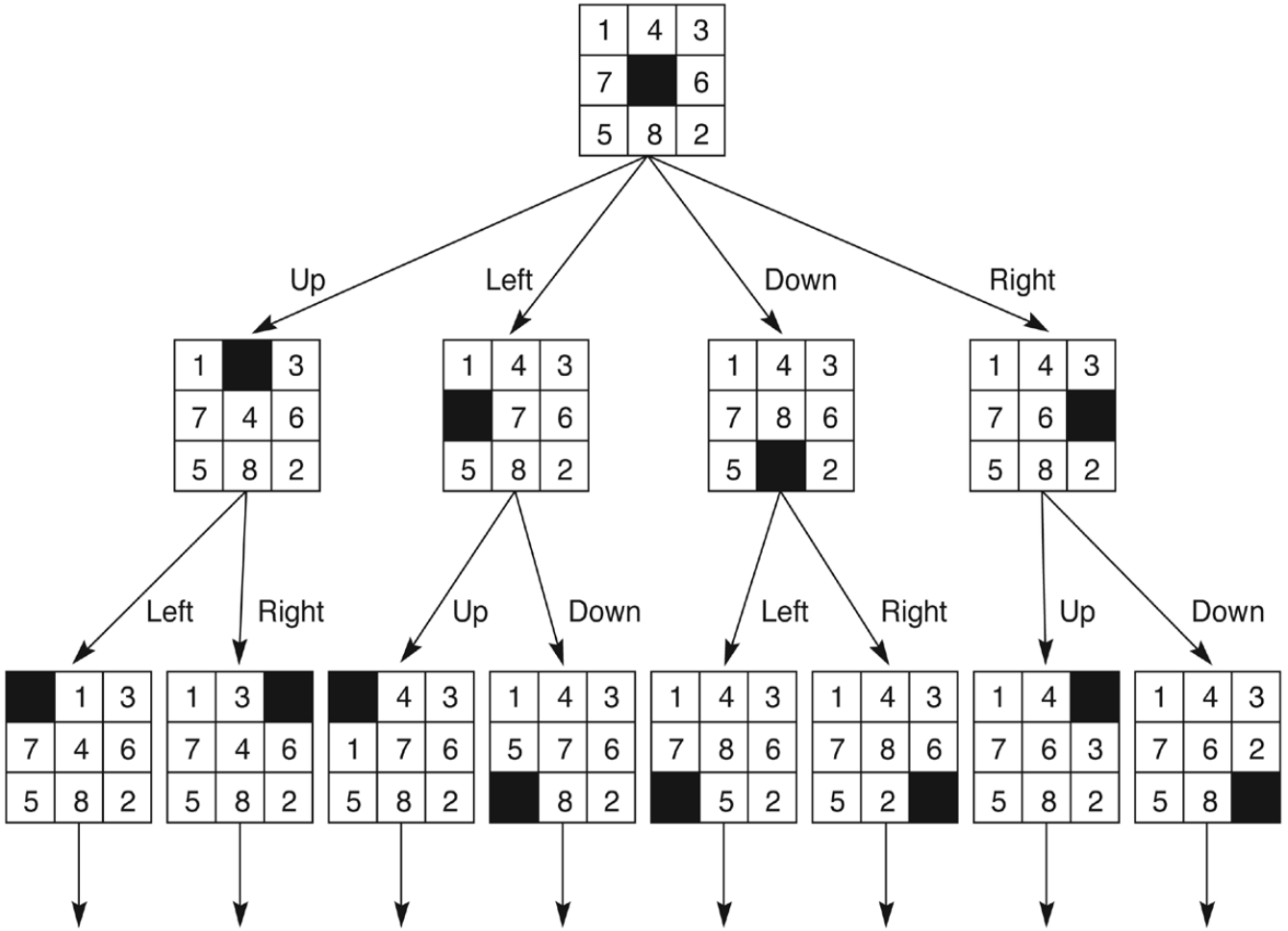
E(n) = 5 – 4 = 1

Heuristic is E(n) = M(n) – O(n)
where M(n) is the total of My possible winning lines
O(n) is total of Opponent's possible winning lines
E(n) is the total Evaluation for state n

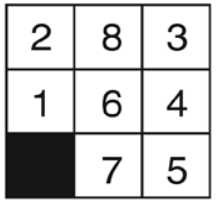
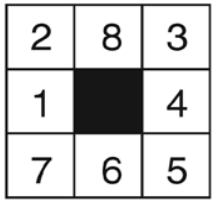
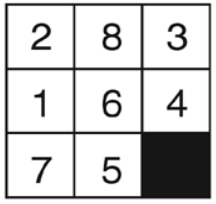
The 8-puzzle (1)

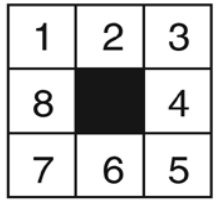
- The set of states are all different configurations of 9 tiles (9!).
- The legal moves are : move the blank tile up(), right(→), down(↓), and the left(←).
  - make sure that it does not move the blank off the board.
  - All four moves are not applicable at all times.
- The start state
- The goal state
- unlike tic-tac-toe, cycles are possible in the 8-puzzle.
- Applicable heuristics

The 8 puzzle (2)



Possible Heuristic for 8-puzzle

	**5**	**6**	**0**
	**3**	**4**	**0**
	**5**	**6**	**0**
	Tiles out of place	Sum of distances out of place	2 x the number of direct tile reversals

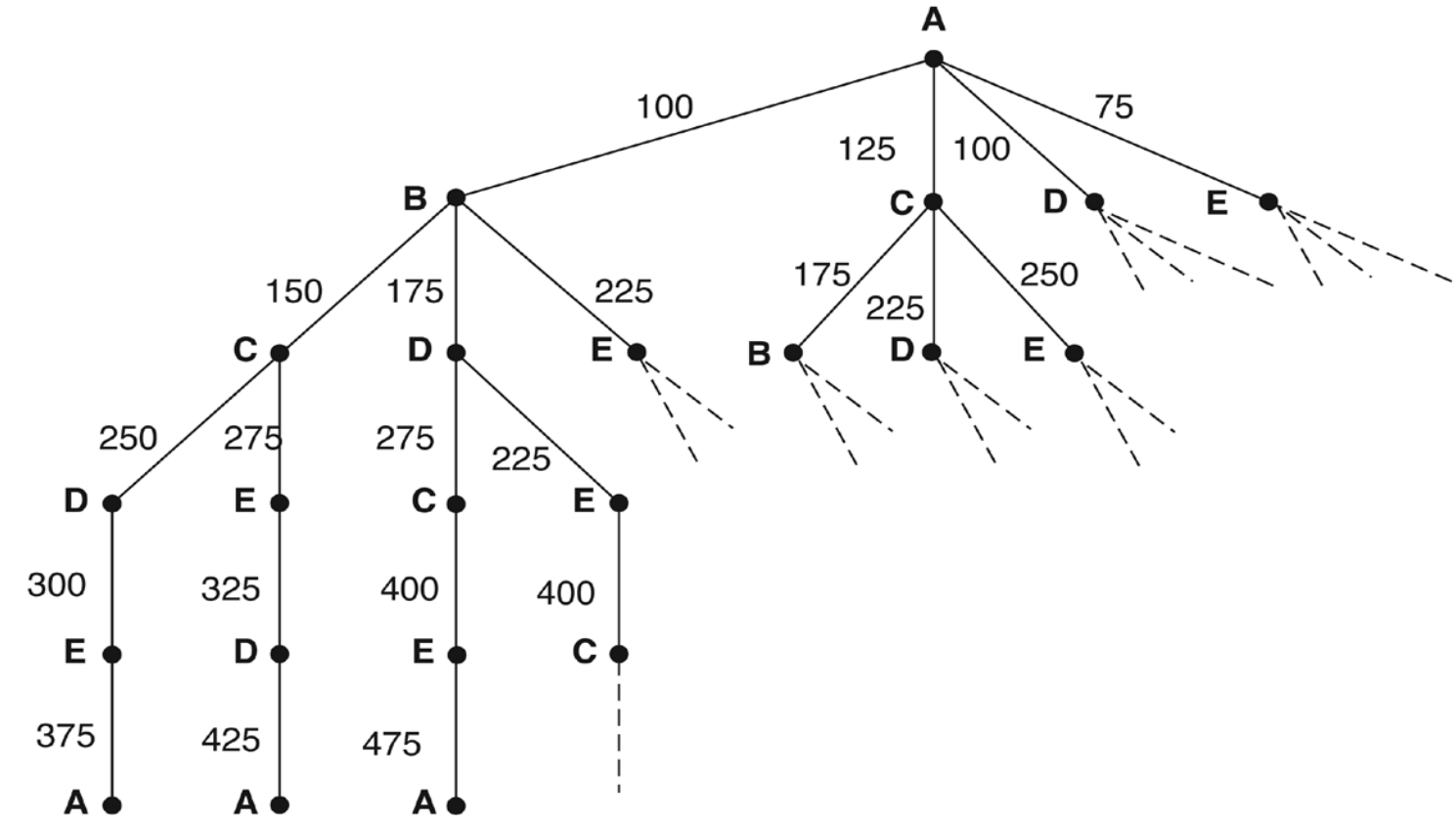


Goal

The Traveling Salesperson(1)

- 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
- Techniques for reducing the search complexity.
  - Branch and Bound, The Nearest neighbor.

The Traveling Salesperson(2)



Path: ABCDEA
Cost: 375

Path: ABCEDA
Cost: 425

Path: ABDCEA
Cost: 475

. . .

The Traveling Salesperson(3)

- 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
- Other examples of State Space representation
  - Blocks world ,FWGC problem

The Traveling Salesperson(4)

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*.

Data-Driven and Goal-Driven Search (2)

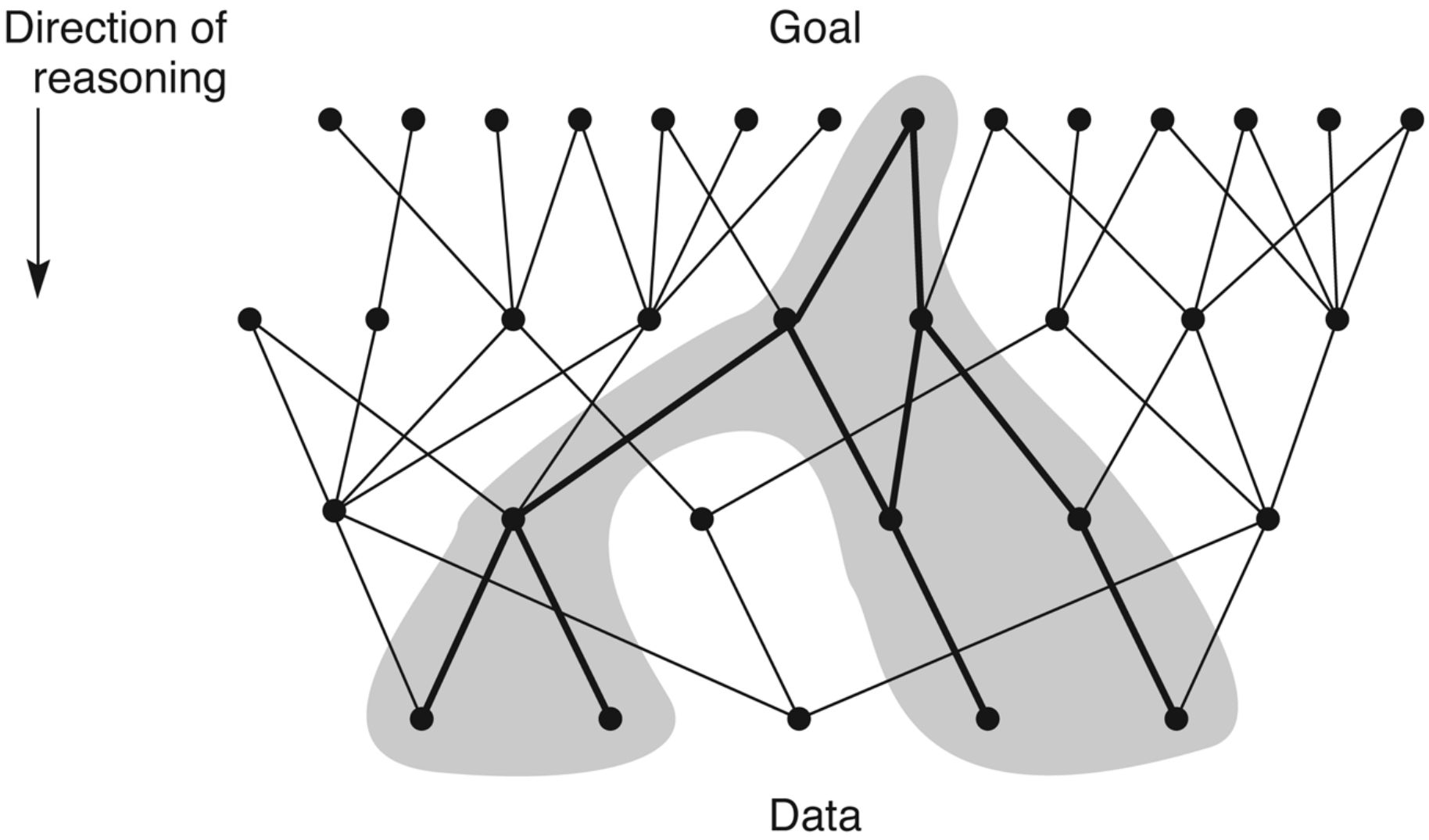
- The preferred strategy is determined by the properties of the problem: complexity of the rules, "shape" of the state space, the nature and availability of the problem data.
- Confirming or denying the statement

  "I am a descendant of John."
  - Assume that John was born about 250 years ago and that 25 years per generation and that 3 children per family
  - I -> John(backward) 2^{10} ancestors
  - John -> I (forward) 3^{10} nodes of family

Goal-Driven Search is Suggested(1)

- A goal or hypothesis is given in the problem statement or can easily be formulated.
  - Mathematics theorem prover, diagnostic systems
- Early selection of a goal can eliminate most branches, making goal-driven search more effective in pruning the space.
  - Mathematics theorem prover: total # of rules to produce a given theorem is much smaller than # of rules that may be applied to the entire set of axioms
- Problem data are not given but must be acquired.
  - A medical diagnosis program: Doctor order only diagnostic tests that are necessary to confirm or deny a hypothesis

Goal-Driven Search is Suggested(2)



Data-driven Search is Appropriate

- All or most of the data are given in the initial problem statement.
  - PROSPECTOR interpreting geological data.
- There are only a few ways to use the facts and given information.
  - DENDRAL finding the molecular structure of organic compounds. For any organic compound, enormous number of possible structures. The mass spectrographic data on a compound allow DENDRAL to eliminate most of possible structures.
- Branching factor, availability of data, and ease of determining potential goals are carefully analyzed to determine the direction of search.

Backtracking Search Algorithm

- Backtracking is a technique for systematically trying all paths through a state space.
- Search begins at the start state and pursues a path until it reaches either a goal or a "dead end". If it finds a goal, returns the solution path. If it reaches a dead end, it backtracks to the most recent node on the path
- Figure below :

A→B→E→H→I→F→J→C→G

Backtracking Search Example