The 8-puzzle

An 8-puzzle consists of a 3x3 board with eight numbered tiles and a blank space. A tile adjacent to the blank space can slide into the space. The object is to reach the goal state, as shown in figure 1.21

Example: The 8-puzzle

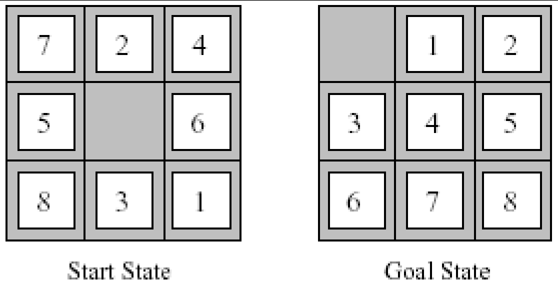

Figure 1.21 A typical instance of 8-puzzle.

The problem formulation is as follows :

- **States** : A state description specifies the location of each of the eight tiles and the blank in one of the nine squares.
- **Initial state** : Any state can be designated as the initial state. It can be noted that any given goal can be reached from exactly half of the possible initial states.
- **Successor function** : This generates the legal states that result from trying the four actions(blank moves Left, Right, Up or down).
- **Goal Test** : This checks whether the state matches the goal configuration shown in figure 2.4.(Other goal configurations are possible)
- **Path cost** : Each step costs 1,so the path cost is the number of steps in the path.
-

The 8-puzzle belongs to the family **of sliding-block puzzles**, which are often used as test problems for new search algorithms in AI. This general class is known as NP-complete.
The **8-puzzle** has 9!/2 = 181,440 reachable states and is easily solved.
The **15 puzzle** (4 x 4 board) has around 1.3 trillion states, and the random instances can be solved optimally in few milli seconds by the best search algorithms.
The **24-puzzle** (on a 5 x 5 board) has around 10_{25} states ,and random instances are still quite difficult to solve optimally with current machines and algorithms.

8-queens problem

The goal of 8-queens problem is to place 8 queens on the chessboard such that no queen attacks any other.(A queen attacks any piece in the same row,column or diagonal). Figure 2.5 shows an attempted solution that fails: the queen in the right most column is attacked by the queen at the top left.
An **Incremental formulation** involves operators that augments the state description,starting with an empty state.for 8-queens problem,this means each action adds a queen to the state. A **complete-state formulation** starts with all 8 queens on the board and move them around. In either case the path cost is of no interest because only the final state counts.

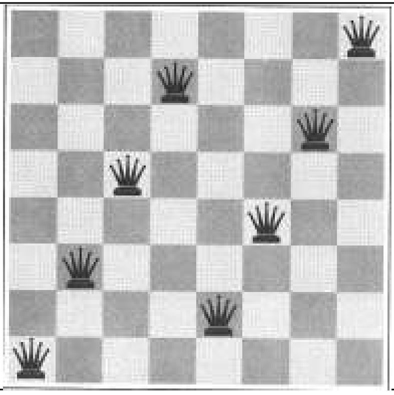
Figure 1.22 8-queens problem

The first incremental formulation one might try is the following :

- **States** : Any arrangement of 0 to 8 queens on board is a state.
- **Initial state** : No queen on the board.
- **Successor function** : Add a queen to any empty square.
- **Goal Test** : 8 queens are on the board,none attacked.

In this formulation,we have 64.63…57 = 3 x 10^{14} possible sequences to investigate.
A better formulation would prohibit placing a queen in any square that is already attacked.
:

- **States** : Arrangements of n queens (0 <= n < = 8) ,one per column in the left most columns ,with no queen attacking another are states.
- **Successor function** : Add a queen to any square in the left most empty column such that it is not attacked by any other queen.

This formulation reduces the 8-queen state space from 3 x 10^{14} to just 2057,and solutions are easy to find.

For the 100 queens the initial formulation has roughly 10^{400} states whereas the improved formulation has about 10^{52} states. This is a huge reduction,but the improved state space is still too big for the algorithms to handle.

1.3.2.2 REAL-WORLD PROBLEMS

ROUTE-FINDING PROBLEM

Route-finding problem is defined in terms of specified locations and transitions along links between them. Route-finding algorithms are used in a variety of applications,such as routing in computer networks,military operations planning,and air line travel planning systems.

AIRLINE TRAVEL PROBLEM

The **airline travel problem** is specifies as follows **:**

- **States :** Each is represented by a location(e.g.,an airport) and the current time.
- **Initial state :** This is specified by the problem.
- **Successor function :** This returns the states resulting from taking any scheduled flight(further specified by seat class and location),leaving later than the current time plus the within-airport transit time,from the current airport to another.
- **Goal Test :** Are we at the destination by some prespecified time?

- **Path cost :** This depends upon the monetary cost,waiting time,flight time,customs and immigration procedures,seat quality,time of dat,type of air plane,frequent-flyer mileage awards, and so on.

TOURING PROBLEMS

Touring problems are closely related to route-finding problems,but with an important difference. Consider for example,the problem,‖Visit every city at least once‖ as shown in Romania map. As with route-finding the actions correspond to trips between adjacent cities. The state space, however,is quite different.
The initial state would be ―In Bucharest; visited{Bucharest}‖.
A typical intermediate state would be ―In Vaslui;visited {Bucharest,Urziceni,Vaslui}‖.
The goal test would check whether the agent is in Bucharest and all 20 cities have been visited.

THE TRAVELLING SALESPERSON PROBLEM(TSP)

Is a touring problem in which each city must be visited exactly once. The aim is to find the shortest tour.The problem is known to be **NP-hard**. Enormous efforts have been expended to improve the capabilities of TSP algorithms. These algorithms are also used in tasks such as planning movements of **automatic circuit-board drills** and of **stocking machines** on shop floors.

VLSI layout

A **VLSI layout** problem requires positioning millions of components and connections on a chip to minimize area ,minimize circuit delays,minimize stray capacitances,and maximize manufacturing yield. The layout problem is split into two parts : **cell layout** and **channel routing**.

ROBOT navigation

ROBOT navigation is a generalization of the route-finding problem. Rather than a discrete set of routes,a robot can move in a continuous space with an infinite set of possible actions and states. For a circular Robot moving on a flat surface,the space is essentially two-dimensional.

When the robot has arms and legs or wheels that also must be controlled,the search space becomes multi-dimensional. Advanced techniques are required to make the search space finite.

AUTOMATIC ASSEMBLY SEQUENCING

The example includes assembly of intricate objects such as electric motors. The aim in assembly problems is to find the order in which to assemble the parts of some objects. If the wrong order is choosen,there will be no way to add some part later without undoing somework already done. Another important assembly problem is protein design,in which the goal is to find a sequence of Amino acids that will be fold into a three-dimensional protein with the right properties to cure some disease.

INTERNET SEARCHING

In recent years there has been increased demand for software robots that perform Internet searching.,looking for answers to questions,for related information,or for shopping deals. The searching techniques consider internet as a graph of nodes(pages) connected by links.