Branch and Bound

Branch and bound (BB, B&B, or BnB) is an algorithm design paradigm for discrete and combinatorial optimization problems, as well as mathematical optimization. A branch-and-bound algorithm consists of a systematic enumeration of candidate solutions by means of state space search: the set of candidate solutions is thought of as forming a rooted tree with the full set at the root. The algorithm explores *branches* of this tree, which represent subsets of the solution set. Before enumerating the candidate solutions of a branch, the branch is checked against upper and lower estimated *bounds* on the optimal solution, and is discarded if it cannot produce a better solution than the best one found so far by the algorithm.

Problem: 15 Puzzles

The15-puzzle(invented by Sam Loyd in 1878)consists of 15 numbered tiles on a square frame with a capacity of 16 tiles (Figure (a)).We are given an initial arrangement of the tiles, and the objective is to transform this arrangement into the goal arrangement of Figure(b) through a series of legal moves. The only legal moves are ones in which a tile adjacent to the Empty spot(ES) is moved to ES. Thus from the initial arrangement of Figure(a), four moves are possible. We can move any one of the tiles numbered 2, 3, 5, or 6 to the empty spot. Following this move, other moves can be made. Each move creates a new arrangement of the tiles. These arrangements are called the states of the puzzle. The initial and goal arrangements are called the initial and goal states. A state is reachable from the initial state iff there is a sequence of legal moves from the initial state to this state. The state space of an initial state consists of all states that can be reached from the initial state. The most straightforward way to solve the puzzle would be to search the state space for the goal state and use the path from the initial state to the goal state as the answer. It is easy to see that there are16!(16!=20.9x 10^{12}) different arrangements of the tiles on the frame. Of these only one-half are reachable from any given initial state.

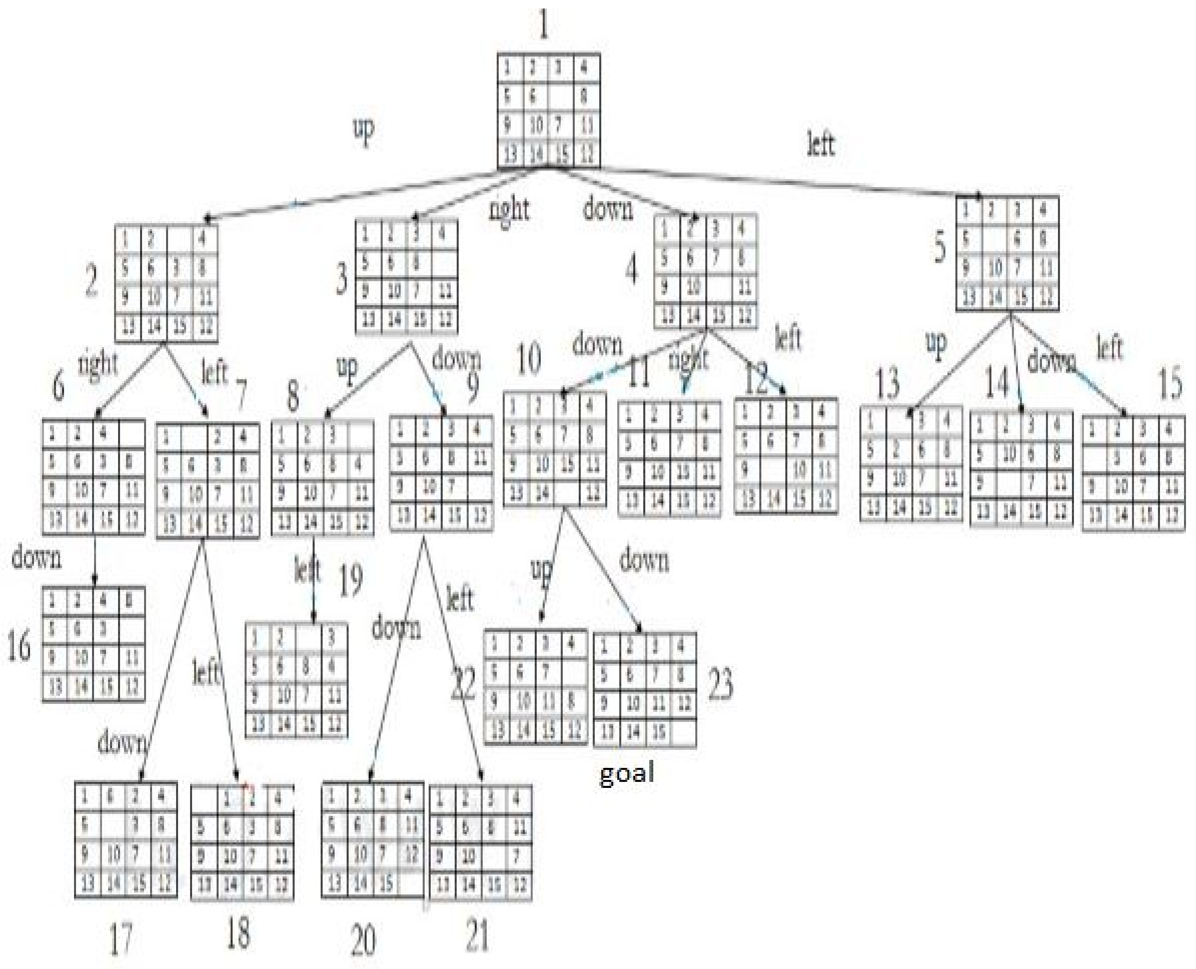
1	3	4	15
2		5	12
7	6	11	14
8	9	10	13

(a) Initial arrange

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

(b) Target arrange

Following figure shows the first three levels of the state space tree of the 15-puzzle beginning with the initial state shown in the root. Parts of levels 4 and 5 of the tree are also shown. The tree has been pruned a little. No node p has a child state that is the same as p's parent. The subtree eliminated in this way is already present in the tree and has root parent(p).As can be seen, there is an answer node at level 4.



We can associate a cost c(x) with each node x in the state space tree. The cost c(x) is the length of a path from the root to a nearest goal node (if any) in the subtree with root x. Thus, in Figure8.3, c(1)= c(4)= c(10)= c(23)= 3. When such a cost function is available, a very efficient search can be carried out. We can arrive at an easy to compute estimate c(x).We can write

c(x) = f(x) +g(x),where f(x) is the length of the path from the root to

node x and g(x) is an estimate of the length of a shortest path from x to a goal node in the subtree with root x. One possible choice for g(x) is g(x) = number of non blank tiles not in their goal position.

Then c(2)= 1+4, c(3)= 1+4, c(4)= 1+2, and c(5)= 1+4, c(10)= 2 + 1,c(11)=2+ 3,and

$c(12)=2+3$