Example 1: Missionaries and Cannibals

The Missionaries and Cannibals problem is a well-known problem that is often used to illustrate AI techniques. The problem is as follows:

Three missionaries and three cannibals are on one side of a river, with a canoe. They all want to get to the other side of the river. The canoe can only hold one or two people at a time. At no time should there be more cannibals than missionaries on either side of the river, as this would probably result in the missionaries being eaten.

To solve this problem, we need to use a suitable representation.

First of all, we can consider a state in the solving of the problem to consist of a certain number of cannibals and a certain number of missionaries on each side of the river, with the boat on one side or the other. We could rep- resent this, for example, as

3, 3, 1 0, 0, 0

The left-hand set of numbers represents the number of cannibals, missionaries, and canoes on one side of the river, and the right-hand side represents what is on the other side.

Because the number that is on one side is entirely dependent on the number that is on the other side, we can in fact just show how many of each are on the finishing side, meaning that the starting state is represented as

0, 0, 0

and the goal state is

3, 3, 1

An example of a state that must be avoided is

2, 1, 1

Here, there are two cannibals, one canoe, and just one missionary on the other side of the river. This missionary will probably not last very long.

To get from one state to another, we need to apply an operator. The opera- tors that we have available are the following:

1. Move one cannibal to the other side
2. Move two cannibals to the other side
3. Move one missionary to the other side
4. Move two missionaries to the other side
5. Move one cannibal and one missionary to the other side

So, if we apply operator 5 to the state represented by 1, 1, 0, then we would result in state 2, 2, 1. One cannibal, one missionary, and the canoe have now moved over to the other side. Applying operator 3 to this state would lead to an illegal state: 2, 1, 0.

We consider rules such as this to be **constraints**, which limit the possible operators that can be applied in each state. If we design our representation correctly, the constraints are built in, meaning we do not ever need to examine illegal states.

We need to have a test that can identify if we have reached the goal state—3, 3, 1.

We will consider the cost of the path that is chosen to be the number of steps that are taken, or the number of times an operator is applied. In some cases, as we will see later, it is desirable to find a solution that minimizes cost.

The first three levels of the search tree for the missionaries and cannibals problem is shown in Figure 3.6 (arcs are marked with which operator has been applied).

Now, by extending this tree to include all possible paths, and the states those paths lead to, a solution can be found. A solution to the problem would be represented as a path from the root node to a goal node.

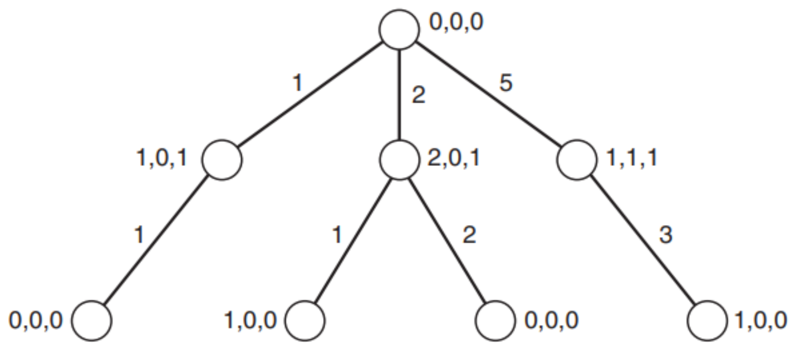

Figure 3.6
A partial search tree for the missionaries and cannibals problem

This tree represents the presence of a cycle in the search space. Note that the use of search trees to represent the search space means that our representation never contains any cycles, even when a cyclical path is being followed through the search space.

By applying operator 1 (moving one cannibal to the other side) as the first action, and then applying the same operator again, we return to the start state. This is a perfectly valid way to try to solve the problem, but not a very efficient one.

Improving the Representation

A more effective representation for the problem would be one that did not include any cycles. Figure 3.7 is an extended version of the search tree for the problem that omits cycles and includes goal nodes.

Note that in this tree, we have omitted most repeated states. For example, from the state 1,0,0, operator 2 is the only one shown. In fact, operators 1 and 3 can also be applied, leading to states 2,0,1 and 1,1,1 respectively. Neither of these transitions is shown because those states have already appeared in the tree.

As well as avoiding cycles, we have thus removed suboptimal paths from the tree. If a path of length 2 reaches a particular state, s, and another path of length 3 also reaches that state, it is not worth pursuing the longer path because it cannot possibly lead to a shorter path to the goal node than the first path.

Hence, the two paths that can be followed in the tree in Figure 3.7 to the goal node are the shortest routes (the paths with the least cost) to the goal, but they are by no means the only paths. Many longer paths also exist.

By choosing a suitable representation, we are thus able to improve the efficiency of our search method. Of course, in actual implementations, things may not be so simple. To produce the search tree without repeated states, a memory is required that can store states in order to avoid revisiting them. It is likely that for most problems this memory requirement is a worthwhile trade-off for the saving in time, particularly if the search space being explored has many repeated states and cycles.

Solving the Missionaries and Cannibals problem involves **searching** the search tree. As we will see, search is an extremely useful method for solving problems and is widely used in Artificial Intelligence.

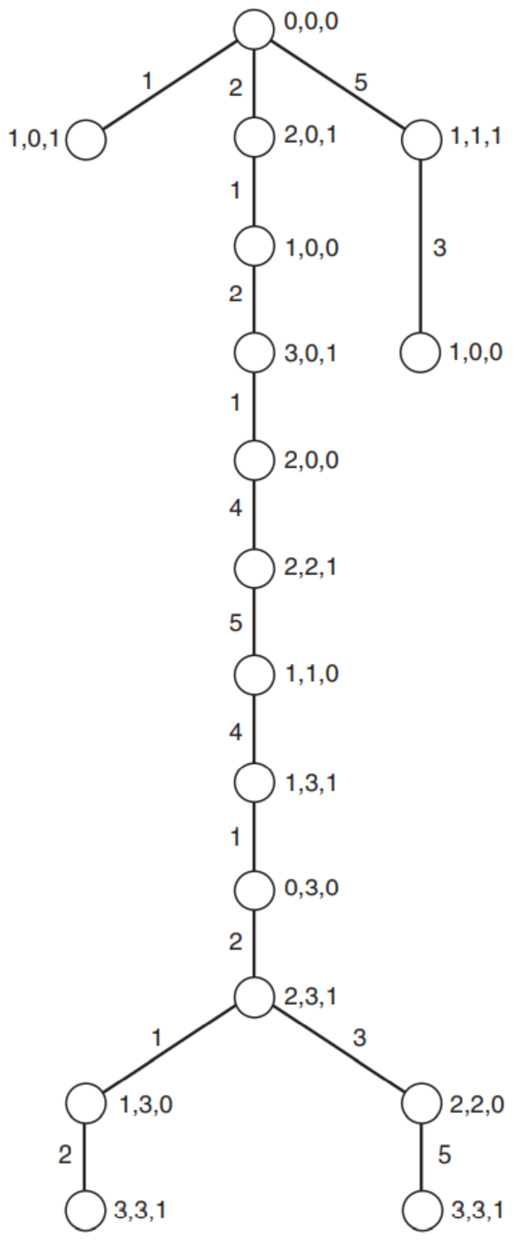

Figure 3.7

Search tree without cycles