Lecture 4 CS6800 Artificial Intelligence:

- Graph/Tree terminology
- A general graph-search procedure
- Uninformed graph-search procedures
- Depth-first search
- Breadth-first search

Terminology cont.

- Often it is convenient to assign costs to arcs, to represent the cost of applying the corresponding rule to a database.

  $c(n_i, n_j)$ denotes the cost of an arc directed from node n_i to node n_j. For mathematical reasons, sometimes we wish to guarantee that all $c(n_i, n_j)$ are bounded below by some small positive constant e.

- The ***cost of a path*** between two nodes is the sum of the costs of all the arcs connecting the nodes on the path.

- The object of the graph-search method is to find a path (perhaps having minimal cost) between a member of a given set of start nodes $\{s_i\}$, representing the initial databases, and a member of a set of nodes $\{t_i\}$ that represent databases satisfying the termination condition.
- The set $\{t_i\}$ is called the ***goal set***, and each node t in $\{t_i\}$ is called a ***goal node***.
- The set $\{s_i\}$ is called the ***start set***, and each node s in $\{s_i\}$ is called a ***start node***.

Implicit and Explicit Graphs

A graph may be specified ***explicitly,*** or ***implicitly***. A control strategy is supposed to make explicit a part of an implicitly specified graph. The implicit specification is given by the ***start node*** and the rules that alter databases.

Let us define a ***successor operator*** that when applied to a node generates **all** its successors and the costs of the associated arcs.

The process of applying the successor operator is called ***expanding***.

A General Graph-Search Procedure

1. Create a search graph, *G*, consisting solely of the start node, *s*. Put *s* on a list called *OPEN*.
2. Create a list called *CLOSED* that is initially empty.
3. LOOP: if *OPEN* is empty, exit with failure.
4. Select the first node on *OPEN*, remove it from *OPEN*, and put it on *CLOSED*. Call this node *n*.
5. If *n* is a goal node, exit successfully. The solution is the path along the pointers from *n* to *s* in *G*.
6. Expand node *n*, generating the set *M*, of its successors and install them as successors of *n* in *G*.

7. Establish a pointer to *n* from those members of *M* that were not already in *G*. Add these members of *M* to *OPEN*. For each member of *M* that was already in *G*, decide whether or not to redirect its pointer to *n*. For each member of *M* already on *CLOSED*, decide for each of its descendents in *G* whether or not to redirect its pointer.

8. Reorder the list *OPEN*.

9. Go LOOP.

Example of 7.

Example cont.

Description of Algorithm

- We are simultaneously creating two things: the search graph *G*, and a search tree, that we will denote by *T*.
- The search tree is specified by the pointers of step 7 in the above algorithm.
- Every path to a node is saved in *G*, a ***unique*** path to any node is stored on *T*.

We will now detail step 7

After expansion the graph becomes:

Uninformed Graph-Search Procedures

In the above algorithm, where does information come into the process?

Suppose we have no domain knowledge of the search process. In this case we must make an arbitrary choice. The resultant procedure is called ***uninformed***. There are two very well known types of uninformed search procedures:

-
-

Depth-first Search

The nodes on *OPEN* in step 8 of our algorithm are ordered in descending order of their depth in the ***search tree***: the deepest nodes are put first in the list, and nodes of equal depth are ordered arbitrarily.

Lets look at an example:

Example cont.

Example cont.

Depth Bound

Another important feature to incorporate in the above procedure is a depth-bound. This would prevent the solution from running away along some fruitless path forever.

Note: if the depth-bound is too stringent, you may never discover the solution.

Does this algorithm remind you of some other one we have already studied?

Which one is generally better? Why?

Breadth-first Search

We now look at the opposite ordering scheme, nodes in ***breadth-first search*** are ordered in increasing order of their depth in the search tree.

Example cont.

Example cont.

Heuristic Search

We will next consider ***intelligent search procedures***. These procedures provide information to the search process in the form of ***heuristics***.

