George Isber **Syria**

Tishreen University (Lattakia)

SEARCH METHODS AND LANGUAGES FOR EXPERT SYSTEMS AND THEIR APPLICATION

1- Introduction: Searching for a solution in an expert system (ES) can be divided into the categories of blind searching and heuristic searching. In blind searching no intelligent decision making is required and the choice of a path to follow is completely arbitrary. Heuristic searching involves applying a heuristic function to minimize the number of arcs taken in reaching the solution. How intelligent the heuristic search is depends on the information supplied to the heuristic function.

Once search strategy is chosen a method to control this searching process is decided. Among the popular methods are forward Chaining, Backward Chaining, Bidirectional and Means-End. Forward-Chaining is a goal-driven one. Both are applied together to achieve bidirectional control. The method of control chosen depends very much on the problem being analyses.

Although ES have been created using a wide number of languages, the most popular are the artificial intelligence (AI) language structure.

2- Search methods: Search methods used in ES can be divided into two categories. One is where no intelligent decision making is involved and the choice of the path to follow is completely arbitrary. This type is widely known as blind search. Blind searches are considered weak and are not commonly used unless the problem tackled by the ES is a trivial one. Second methods such as exhaustive, breadth-first, and depth-first belong to the blind search category.

Search techniques which use domain specific information that were not built into the state and operator definitions are considered to be heuristic searches. This domain specific information is known as heuristic information. Heuristic searches are considered strong as they reduce the number of nodes to be searched and shorten the path taken towards reaching the goal state. The features at which heuristic information can be applied in searching include the following:

- deciding which node to expand next, instead of doing the expansion in strictly breadth-first or depth-first.
- In the course of expanding a node, deciding which successor or successors to generate-instead of generating all possible successors at one time.
- Deciding that certain nodes should be discarded, or pruned, from the search space.

In heuristic searches, generally the node which is the most promising is expanded. Heuristic search methods of best-first, hill climbing, A*algorithm and branch-and-bound are explained in this section. But first an explanation of the blind search methods will help to grasp the techniques involved in heuristic searching.

2-1- blind searches: (a) Exhaustive: this technique, as its name suggests considers all the paths in the search space. Exhaustive searching is time consuming and costly to use and is impractical when the search space is large. If completeness is required at the expense of time, money and memory space this is the technique to employ.

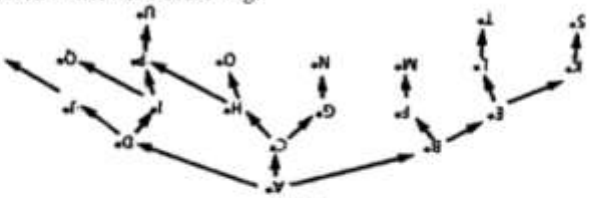

Fig.1

(b) Breadth-first: consider the goal of supplying power from node A to node T in the graph represented by fig. 1.A is the power source and T is considered to be the goal state. A represents level 1 while 5, T and U represent level 5. The breadth-first technique will consider all the paths available from node A to level 2 and will take the shortest path down to that level. At every level it will check to see whether the goal state has been reached. If not it will again check all the available paths from the present level down to the next one. In this way nodes B,C and D at level 2 are checked for possible paths down to level 3. This analyses nodes across the breadth of search space for all possible paths to the nest level below, rather than proceeding along the first available path, and thus it has the name breadth-first searching.

This process is repeated until the goal state is reached. If a dead-end such as node S or U is encountered the method will backtrack to the previous level up and will proceed along the next best path. Hence by employing the breadth-first technique the shortest possible path from A to T can be achieved. As the method involves searching through all the nodes in a graph breadth-first searching technique takes time to reach the solution. It is inefficient to use this technique when the space to be searched is big as it will make excessive demands on memory space.

(c) Depth-first: in depth-first searching, nodes are analyses downwards. If the same problem of supplying power to need T from A (fig.1) is tackled the goal state T will be reached much more quickly using the depth-first technique, but the path chosen may not be the optimal one.

In depth-first, searching nodes are analyses downwards along the first available path until the goal state is reached. In the graph of fig.1 it will proceed from A in level 1 to B in level 2, then to E in level 5. When it reaches node S it will backtrack to the next level up as S is a deed-end. If no alternative paths are found at this level as is the case in the graph of fig.1, it will backtrack to the next level up, to node E in level 3. In this way, an alternative path to level 4 is found at

this node. It then proceeds along the path down to node L and to the goal state of node T.

At every node the depth-first technique, like the breadth-first technique, checks whether the goal state has been reached or not. A preset limit on the maximum allowable depth can prevent the depth-first technique from considering depths that are unacceptably large.

3- Heuristic searches: (a) Best-first: in best-first searching all the nodes in the search space are not analysed. As few nodes as possible. At every node, the search space is analysed to find the most promising node to expand on. Whether one node is more promising than another can be defined in a number of ways. Estimating the distance to the goal is one method. This estimate is found by applying a heuristic function, sometimes called an evaluation function. How effective the search algorithm is in reducing the number of nodes expanded depends very much on the formation and effectiveness of this heuristic function. The more informed the heuristic function, the more effective the search algorithm becomes.

The best-first search algorithm keeps a record of the nodes considered. If the heuristic information supplied leads it down a path that does not reach the goal state, it backtracks and expands on the "next-best" node. The record of the nodes analysed is also checked to prevent duplication of paths.

(b) Hill climbing: the name hill climbing suggests that the steepest path should be followed. Hill climbing is the easiest form of heuristic search to implement. The search methodology is very much similar to that of best-first searching. The fundamental difference is that hill climbing does not keep a record of the analysed nodes. This is its main drawback. The search stops when it reaches a node that is better than the one previously considered, causing the search process to get trapped at a local maximum, and fails to reach the goal state (global maximum). One way of overcoming this problem is to inform the heuristic function of this eventuality. But in general hill climbing does not produce the best path to the solution.

(c) A*algorithm: A*algorithm focuses its attention on finding the minimum cost path from a start node to the goal node. This in a way involves finding the shortest path. For A*algorithm to be used, an ordered state-space search is required. The algorithm stands out from other search methods because of the way it defines the evaluation function f*. it considers the cost of reaching the node n from the starting point and the cost of reaching the goal from node n. hence f* is defined as:[2]

$$f^*(n) = g^*(n) + h^*(n)$$

where g* estimates the cost of reaching n from the starting node and h* estimates the cost of reaching the goal from node n. the function h* carries the heuristic information. The node where f*(n) is a minimum is expanded and the procedure is repeated at the next node. If the space to be searched represents a tree-graph then g* will give a perfect estimate. If it is any general graph then the error is in the direction of overestimating the minimum cost. A search similar to A* algorithm under the name of branch-and-bound is popular among operational research scientists.

(d) Branch-and-bound: as with A* algorithm branch-and-bound searching requires an ordered state-space. At every node all the other paths are considered and the path with the shortest distance is chosen. And the same procedure is applied at the nest node and hence more paths are generated. This cycle repeats itself and all the paths are analysed. The searching is terminated when the shortest incomplete path is longer than the shortest completed path. The search method can generally be improved if the heuristic function takes into consideration both the distance traveled and the distance to goal.

4- Control strategies: The control strategy chosen to control the search process depends very much on the problem. The most popular methods are: Forward-chaining, Backward-Chaining, Bidirectional and Means-End.

4-1- Forward Chaining: Forward chaining, sometimes known as data-driven or bottom-up processing, starts with the facts and works its way towards the goal. Rules are applied to facts to generate new facts, which in turn generate more new facts. This is repeated until the goal state is reached. For instance, if following rules

R1 if KVA > KVA_2 then transformer is overloaded.

R2 if transformer overloaded then trip overload alarm.

R3 if overload alarm tripped then transfer load to another transformer

Exists in a rule-based system employing forward-chaining, further consider that KVA_2 was given the value 100, and the new fact KVA = 101 is supplied, the rules will be activated in the order R1, R2 and R3. forward chaining is most effective in situation where:

1- All or most of the data are given in the initial problem statement.
2- The number of potential goals are numerous but the number of ways of using the facts and the domain specific information is limited.
3- It is difficult to form a goal or a hypothesis.

In power system planning, distribution system design and load forecasting, forward-chaining would be a good control strategy to adopt.

4-2- Backward chaining: Backward chaining, and is also referred to as goal-driven or top-down processing. With this type of control the solution starts with the goal and checks to see what rules can be applied to change the state. The new states then become goals or subgoals of the problem. The process is repeated until the problem is solved. Goal-driven processing is useful when:

1- A goal or hypothesis can easily be formulated or when it is given in the problem definition, such as in the proving of mathematical theorems.
2- The number of rules or facts that can achieve a solution is large.

3- The problem data has to be acquired by the problem solver.

Backward chaining is particularly useful in machine fault diagnostics, and power system monitoring and control.

4-3- Bidirectional: This method involves the application of both forward chaining and backward chaining. This is useful in situations where the search space grows exponentially. The graph to be searched is divided into two haves and the methods of forward and backward chaining are applied at the same time. The search through one half is controlled by forward chaining, and backward chaining is applied to the other half. When both halves obtain a solution path, they are joined together to obtain the full solution path.

4-4- Means-end: Means-end involves reducing the difference between the goal state and the current state. The reduction mechanism is the means and the differences to be reduced is the end. The difference is divided iteratively until the difference is eliminated. When there is no difference a solution path is determined. The operators chosen to reduce the difference are indexed by the differences they would reduce. If a chosen operator is not applicable, another operator that is more likely to make progress is chosen.

5- Programming languages: The programming language chosen to code the ES depends on the problem being tackled. As ES use symbolic reasoning rather that operations on numeric data, a language that can provide control and reasoning in a symbolic form is essential. Apart from supporting symbolic computation, it should provide flexibility of control, late binding and clear and well defined semantics.

The most fundamental requirement of a programming language is to supply the symbolic reasoning necessary to implement the search algorithms. Search methods such as best-first and A* algorithm employ heuristics to solve the problem. The language chosen should be able to accommodate these heuristic rules without much difficulty flexibility of control is another important feature. Knowledge representation models such as production rules allows these rules to be fired in any order that match given situations. This makes A1 programs different from conventional programs. They allow the flexibility adopted in intelligent thinking to be incorporated rather than the step-by-step approach taken by conventional computer programs. The programming should also allow the withholding of judgment till all the information is received. In other words, the language should allow reasoning to proceed while leaving variables unbounded until enough information has been received.

ES have also been created from conventional programming languages such as Pascal, C and Fortran because of their ease of maintenance and efficiency. But by far the most popular are the A1 languages of Prolog and Lisp. Lisp is more popular in the United States but, in Western Europe and Japan, Prolog is more commonly used.

5-1- Prolog: Prolog was invented in the early 1970s in France. It is the best known example of a logic programming language. Prolog (PROgramming in LOGic) is based on first-order predicate calculus. Programming in Prolog involves: stating facts about an event, object or situation, defining rules and asking questions. The facts are stored in the database. For instance, statements like generator is a machine, machines have moving parts, generator is an electrical machine and GEN1 is a generator can be represented in the database as facts in the following form:

- is_a (generator,machine)
- is_a (generator,electrical-machine)
- is_a (GEN1,generator)
- has_part(machine,moving).

Rules are general statements about objects and their relationships. For example we can have the rule:

X HAS MOVING PARTS IF X A MACHINE

It will be represented in prolog as Has_part(X,moving):- Is_a (X,machine), has_part(machines, moving).

Prolog employs pattern directed search to look for solutions. When a question is proposed it tries to march the query with facts in the database. If a match is not found it will backtrack, and will keep on backtracking until a solution is reached. If the question does generator have moving parts? is put (in Prolog coding it will look like?-has-part(generator,moving)) it will return yes. The main advantage of Prolog is its conciseness. A three page listing of Lisp can be condensed into approximately one page of Prolog. The inability to express negative conclusions is the major drawback of Prolog.

5-2- Lisp: Lisp (LISt Processing) was invented in the 1950s. Lisp deals with symbols which are composed of characters on the keyboard. The list is the basis of its data structure. Lists are symbols or strings of atoms enclosed in brackets. Lisp is a very flexible, interactive language which uses recursion to control the searching process. Although is is one of the oldest computer programming languages, it is still very popular among the-A1 programmers, particularly in the United States. One reason for this is that the program and data take the same form and thus can be interchanged. This allows programs which are written in Lisp to modify other programs. This ability is particularly attractive as it can "learn" while running. Above all, the main advantage of Lisp is its nesting nature which is very suitable for search strategies.

6- Conclusion: Completes the structure of an ES. This search has concentrated on search methodologies and their respective advantages and disadvantages and an introduction to two commonly used ES languages. The short examples given in the text demonstrate the ease with which they can be applied to power systems work.

References:

[1] ALBERICO, R., and MiCCO, M.: "Expert systems for reference and information retrieval" (Meckler Corporation, 1999); [2] LUGER, G.F., and STUBBLEFIELD, W.A.: "Artificial intelligence and the design of expert systems" (The Benjamin/Cummins Publishing Company, Inc., 1989); [3] SHAPIRO, S.C., and ECKROTH. D. (Eds.): "Encyclopedia of artificial intelligence – Vol.1" (John Wiley and Sons, 1987).