http://orion.journals.ac.za/

ORiON, Vol. 4, No. 1, pp. 47-64 *ISSN 0259-191X*

PROBLEM SOLVING USING ARTIFICIAL INTELLIGENCE TECHNIQUES

by
A.R. Greef and R. Reinecke
Centre for Robotics
Department of Industrial Engineering
University of Stellenbosch

ABSTRACT

Real-world problems often do not lend themselves to an algorithmic solution. Humans, however, cope with these problems despite their fallible problem solving techniques. Instead of trying to construct algorithms to solve problems AI researchers have concentrated on using the more successful methods used by humans. This paper reviews the area of problem solving in the field of Artificial Intelligence. This includes problem representation for computation, "weak" methods of searching for a problems solution, knowledge representations that facilitate more efficient search strategies and planning - an advanced problem solving technique.

0.0 INTRODUCTION

What is Artificial Intelligence (AI) and why should it be of use to Operations Researchers?

Barr and Feigenbaum [1] answer the first part of the question by defining AI to be "the part of Computer Science concerned with designing intelligent computer systems, that is, systems that exhibit characteristics we associate with intelligent human behaviour". Margaret Boden [2] explains that "computers are its (AI) tools, because its theories are expressed as computer programmes that enable machines to do things that would require intelligence if done by people".

The second part of the question is answered when considering that much of Operations Research is concerned with

http://orion.journals.ac.za/

planning activities or actions required to solve problems. The field of AI covers a number of areas of which problem solving is one. Research in this area has been involved with finding solutions to problems that do not lend themselves to an algorithmic approach, a characteristic that many problems display. Furthermore, in many instances the knowledge or information about a problem domain is difficult or unnatural to represent in data structures such as arrays or sets of numbers. Often it is more "natural" to represent a problem as a set of English language sentences or statements describing the problem domain. In these instances AI programming techniques are useful.

There are two difficulties associated with the algorithmic (step-by-step) approach to solving a class of problems (Graham [3]):

a) there are some classes of problems for which there is no algorithm that will solve every problem in a particular class and,
b) even if there is an algorithm that will solve every problem in a particular class, the algorithm may be so inefficient as to be unsuitable for practical problems (programmes that run in exponential time are considered to be impractical).

Humans, however, often cope with a class of problems even though no all-encompassing algorithm exists and if one does exist, it runs in exponential time. This is done in spite of the fact that the problem solving techniques employed have been found to be fallible, approximate and based on quick-and-dirty methods. Instead of trying to construct algorithms to solve problems AI researchers have concentrated on the more successful methods used by humans. As humans can understand and express problem solving knowledge more easily using symbols and natural language, symbolic programming languages have been developed for AI programming which tend to be based on symbol processing rather than number crunching.

This paper reviews the area of problem solving in the field of AI. Part 1 illustrates methods for representing a problem for coding into a computer. Part 2 gives an overview

http://orion.journals.ac.za/

of the "weak" methods that can be employed for finding the solution to a problem. Part 3 describes the production system architecture that most AI programmes adhere to. Part 4 presents methods of representing knowledge to enrich the problem representations outlined in Part 1. Finally, Part 5 contains an overview of planning, an advanced problem solving technique.

1.0 PROBLEM REPRESENTATION

There are two well-used methods for representing a problem (see Winston [4] for details). The first method treats the problem as a network of inherently ordered states. The second sees the problem as a series of sub-problems which in turn have sub-sub problems and so on until a problem with an immediate solution is reached.

Formally these two problem representation methods are known as:

a) the state-space representation, and
b) the problem-reduction representation.

The choice of which representation to use for a particular problem is commented on by Winston [4]:

> "In a practical sense, however, some representations emphasize things that are more important to solving a class of problems. One scheme, therefore, is more powerful than another because it offers more convenience to the user even though theoretically, both can do the same work."

1.1 THE STATE-SPACE PROBLEM REPRESENTATION

A state-space problem representation consists of states which are frozen conditions of a problem at each stage of its solution. This representation should be used for problems whose solutions are characterized by a succession of states. An example is the "shortest path" problem (see Figure 1).

This problem can be represented by a network with nodes representing cities and links representing distances between the cities.

The problem is that of finding the shortest possible route that a traveller can take in travelling from city "S"

http://orion.journals.ac.za/

to city "G". Each node in Figure 1 depicts a state of the salesman in relation to his start and goal position. The problem's solution can now be given as a suitable succession of states that need to be attained in going from node "S" to node"G".

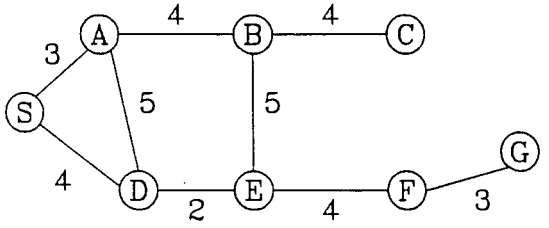


FIGURE 1 : THE SHORTEST PATH PROBLEM

When scanning networks for a solution it is possible to become trapped in a cyclic loop (for example S-D-A-S-D-A in Fig. 1). This can be detected and stopped by extra programmed mechanisms, however, it may be easier to eliminate cyclic paths from the network, the resulting structure is then called a state-space graph or tree (see Fig. 2).

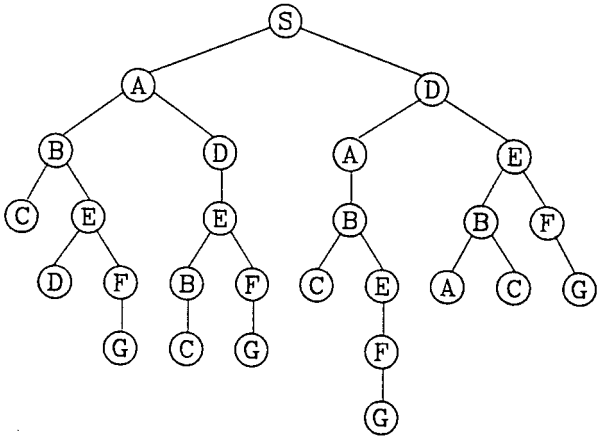


FIGURE 2 : A STATE-SPACE GRAPH FOR THE SHORTEST PATH PROBLEM

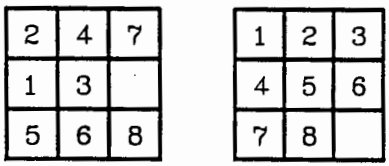
http://orion.journals.ac.za/

Although this increases the size of the problem representational data structure it reduces the amount of difficult code needed to check for cyclic loops. Networks are made into trees by tracing all the possible paths to the point that they re-enter previously visited nodes or they reach a node with no exit path. By convention the tree has nodes connected by branches. Branches directly connect parent nodes with children nodes. The top node, the one that has no parent, is called the root node.

1.2 THE PROBLEM-REDUCTION PROBLEM REPRESENTATION

Using the problem-reduction (or goal-reduction) method, the problem is structured as a set of sub-problems (or sub-goals). The problem's solution is seen as the achievement of sub-goals in going from the initial goal to the final goal. Ah example problem represented using problem-reduction is the 8-puzzle. A tray contains eight tiles numbered 1 to 8 and a space where the ninth tile has been left out as shown in Figure 3.

The tiles are initially positioned in a random state in the tray (Fig. 3(a)). Solving the puzzle requires the ordering of the tiles as shown in Figure 3(b). The tiles can be moved by sliding them either up, down, left or right as the space permits.



(a) (b)

FIGURE 3 : THE 8-PUZZLE PROBLEM

Applying the problem-reduction representation breaks the problem of ordering the initial 8-puzzle into three sub-problems that are easier to solve, namely:

1) The problem of getting the first row in order.
2) The problem of getting the second row in order and keeping the first row in order, given that the first

http://orion.journals.ac.za/

row was in order.

3) The problem of getting the third row in order and keeping the second and first rows in order, given that the second and first rows were in order.

Each of the above sub-problems can be further decomposed into the sub-sub-problem of moving each tile into its respective place in the row being ordered. Problem-reduction continues in this manner until an immediate problem (eg. move a tile either left, right, up or down) is achieved.

The above problem decomposition is one of many ways to solve the 8-puzzle. Ordering columns instead of rows is another alternative. These multiple methods of reducing a problem into smaller problems can be depicted as a generalized tree called an AND/OR graph (Fig. 4).

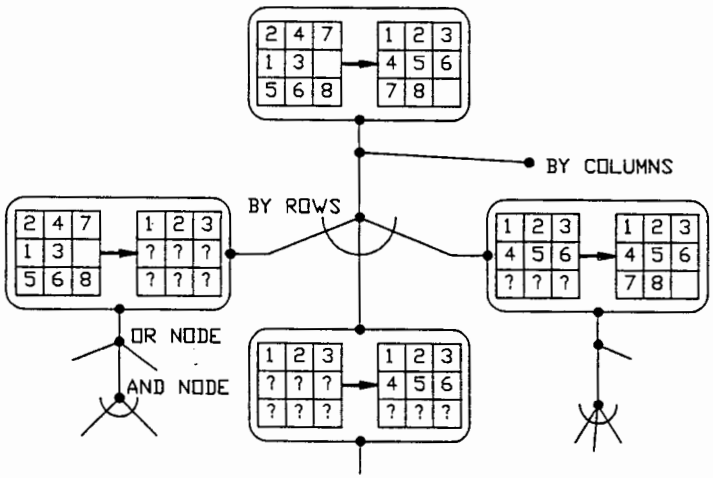


FIGURE 4 : AN AND/OR GRAPH FOR THE 8-PUZZLE

The AND/OR graph consists of nodes representing the decomposed sub-problems and arcs linking the sub-problems in the order that they must be considered. The start node corresponds to the initial problem. If a parent node is solved by solving any one of its children nodes the parent node is called an "OR" node (the start node is an "OR" node). If a parent node is only solved when all of its children nodes are

http://orion.journals.ac.za/

solved then the parent node is called an "AND" node. To distinguish "AND" nodes from "OR" nodes, the arcs leading to "AND" node children are joined by a line as shown in Figure 4.

2. "WEAK" PROBLEM SOLVING TECHNIQUES

Having represented a problem using one of the methods in Part 1, its solution can then be formulated as a search for a sequence of nodes in a graph that connects the root node with a terminal node. This sequence of nodes is the solution to the problem.

If a tree search is conducted by starting with the initial problem configuration(s) at the root of the tree and continues by tracing the solution path to the final problem configuration at a terminal node, then the system conducts forward reasoning (forward chaining). If the search begins at the final problem configuration and ends at the initial problem configuration, the system conducts backwards reasoning (backward chaining).

Two exhaustive search methods are (see Rich [5] for others):

(a) depth-first search and

(b) breadth-first search.

A depth-first search considers each node in a down-up and left-to-right fashion. Breadth-first search traverses across the nodes, also from left-to-right, considering all the nodes on one level before considering any on a lower level. The two search techniques are shown in Figure 5.

These brute-force techniques can be used effectively with small search spaces, however with most practical problems a combinational explosion results with the nodes of the search tree growing exponentially. Another problem associated with brute-force search is that they are not guaranteed to find the problem solution of least cost although they do find some solution. In these cases searches can be made more efficient by employing some "heuristic" that will guide the search in the right direction.

The word "heuristic" comes from the Greek word which means "serving to discover". A heuristic is any hint or rule

of thumb which helps to guide the search for a problem. A search guided by heuristics is called a heuristic search. (This should not be confused with the term "heuristics" as used in knowledge representation).

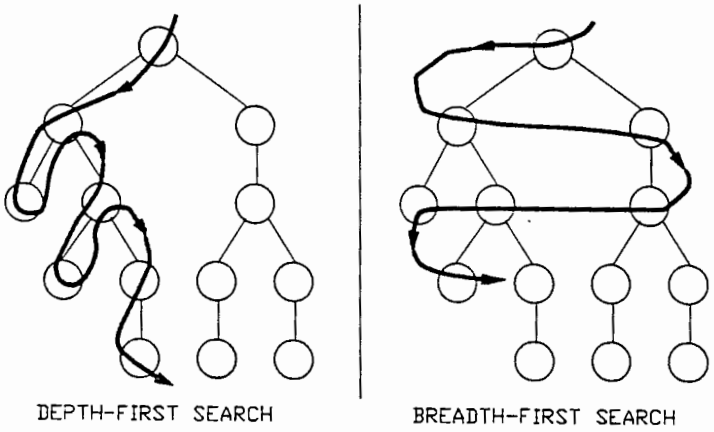


FIGURE 5 : "WEAK" SEARCH TECHNIQUES

A heuristic which can be used to find a solution for the "shortest path" problem in Part 1 is to always choose cities that have the least distance , as the crow flies, to the goal city, as the next step in the search path. A number of generalized heuristic search methods are presented in AI books such as those of Winston [4] and Rich [5].

3.0 PRODUCTION SYSTEMS

Most AI systems have a clear separation between the standard computational components of data, operations and control. Various generalizations of this computational formalism are known as production systems. The major elements of an AI production system are (Nilson [6]) :

a) a global data base;
b) a set of production rules; and
c) a control system.

The data base contains a description of a problem representation and whatever other information that is appropriate. It may be as simple as a small matrix of numbers or as com-

http://orion.journals.ac.za/

plex as a large relational file structure. Some parts of the data base may be permanent, while others may pertain only to the solution of the current problem being solved. The data base is changed during the problem solving process to reflect the problems current status.

A production rule is of the form: IF a certain condition is satisfied THEN do the following. The rule consists of a left hand side (a pattern) that determines the applicability of the rule, and a right hand side that describes the action to be performed if the rule is applied. Production rules are matched against the problem state in the global data base. A forward chaining system matches the left hand side of a rule against the data base and updates the data base as dictated by the right hand side of the rule. A backward chaining system attempts to match the right hand side of the rules to the data base and updates the data base as dictated by the left hand side of the rule. Production rules are then the operators that define the conditions necessary to move from one state or sub-problem (see Part 1) to another during the traverse of a graph representing a problem.

Under direction of the control system, the computer searches through a list of productions, trying to match each one against the data base. The control strategy specifies the order in which the rules are compared to the data base and a way of resolving the conflicts (conflict resolution) that arise when several rules match at once. A typical control strategy is to use a depth-first search to specify the order in which the nodes of a problem graph are to be considered. A conflict resolution strategy would be to consider the first rule that is found. The control system continues to apply production rules until a goal state exists in the data base or until it is discovered that the problem has no solution. Furthermore, it keeps track of the problems solution path.

4.0 KNOWLEDGE REPRESENTATION

In order to solve complex problems using artificial intelligence, a large amount of knowledge and some mechanisms for manipulating that knowledge to create solutions, is

http://orion.journals.ac.za/

needed. So far only a very general method of manipulating knowledge - search, has been presented. Due to their generality, these search techniques are limited and only form the skeleton of practical AI systems. More specific knowledge representation models allow for more powerful inference mechanisms to guide the search for a solution. Emphasis is placed on enriching the knowledge representations using semantics and heuristics that constrain the search for a solution. Heuristics are now embedded in the knowledge representation rather than in the control system as was previously done.

A knowledge base is the collective term for a production system's global data base and set of production rules. Objects, facts and relations describing a problem domain are encoded into the data base. This is termed declarative knowledge. The operators that act on the data base contain knowledge on what to do if certain conditions arise in the data base. This is termed procedural knowledge.

There are two schools of thought concerning how knowledge should be represented (Winograd [7]). One school supports the theory that knowledge is stored in the human brain in a declarative manner and that there are general procedures that manipulate the knowledge. The other school supports the theory that knowledge is contained as procedures (production rules) only and there is no declarative knowledge. Often the best representation schemes use both declarative and procedural knowledge.

The following sections give a brief description of the knowledge representation methods.

4.1 SEMANTIC NETWORKS

Semantic networks were first proposed by Quillian [8] as a psychological model depicting how declarative knowledge is encoded in human memory. A semantic network is a collection of objects called nodes. The nodes are connected together by arcs or links which represent actions, events or relations between the objects. A node-and-link network is not necessarily a semantic network, however. A true semantic network must have semantics. Figure 6 shows a portion of a semantic

http://orion.journals.ac.za/

network depicting the relationships between different classes of machines. In this case lathe-20 is of the class lathe which is a-kind-of (A-K-O) metal remover.

As soon as an object has been identified as lathe-20 for example, it can inherit all the properties of its super-classes lathe, metal-remover and CNC-machine. Inheritance information can be found by searching the network and following the A-K-O links. For example, to find out what lathe-20 is "controlled-by" the A-K-O links are traced to the CNC-machine object node that has a relation showing that all these machines are controlled by a computer. A simple breadth-first search will accomplish this.

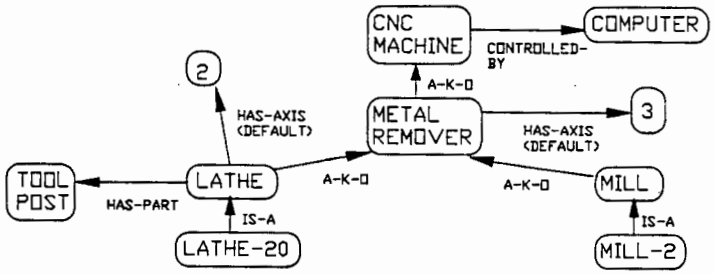


FIGURE 6 : A SEMANTIC NETWORK REPRESENTATION

Default values can be expressed as in the case of the number of axes that the milling machine and lathe have. Here the lathe sub-class has a default value of two axes: however, the metal-remover superclass has a default value of three axes. This is the value inherited by the mill sub-class.

Semantic networks represent declarative knowledge concerning a problem, the solution to which is found using generalized search procedures. The links in the network make class-based inheritance opportunities explicit, thus facilitating the work of class-oriented inference procedures.

4.2 FRAME REPRESENTATIONS

Minsky [9] first proposed the frame representation of knowledge. A frame is a collection of semantic network nodes and links that together describe a stereotyped object, act or

http://orion.journals.ac.za/

event. Stereotyping allows an element of predictability and expectation to be built into the representation. A frame has slots that may contain default values, pointers to other frames, sets of rules, or procedures by which values are found (Harmon and King [10]).

Figure 7 shows part of a frame representation for instructing a robot to move a block from position A to position B. The frames are depicted as nodes in a graph.

The frame "MOVE-BLOCK" has two types of slot: the location slot, representing an expected condition and containing a value which gives the position of the block; and the if-needed slot which calculates a value for the block's mass if it is required. The latter slot filler is called procedural attachment. This inclusion of procedures in frames joins together the declarative and procedural representations of knowledge.

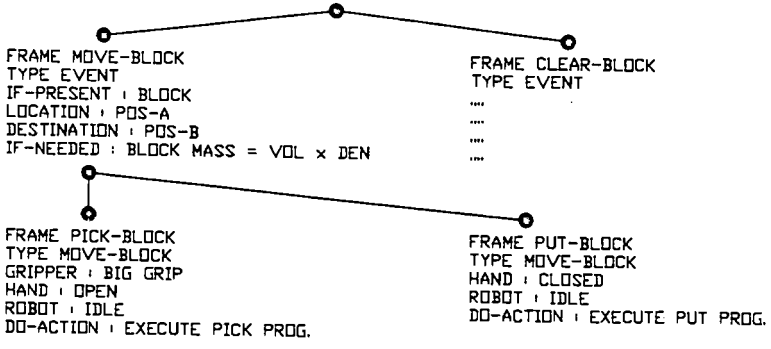


FIGURE 7 : A FRAME REPRESENTATION

Solving a problem that is represented using frames requires a search (depth-first, for example) to traverse the problem representational graph. In this case however, a node is only included into a problem solution if all the frame slots are matched in the global data base.

4.3 RULES

A rule-based system encodes only procedural knowledge of the form:

```
IF      [condition]          premise
  AND   [condition]
THEN    [action 1]           conclusion
  and   [action 2]
```

Rules are used with either attribute-value (A-V) or object-attribute-value (O-A-V) representations. A-V and O-A-V relationships are a specialized case of the semantic network approach. Exotic links are removed leaving two simple relationships. The attribute <- value link is an "IS-A" link and the object -> attribute is a "HAS-A" link.

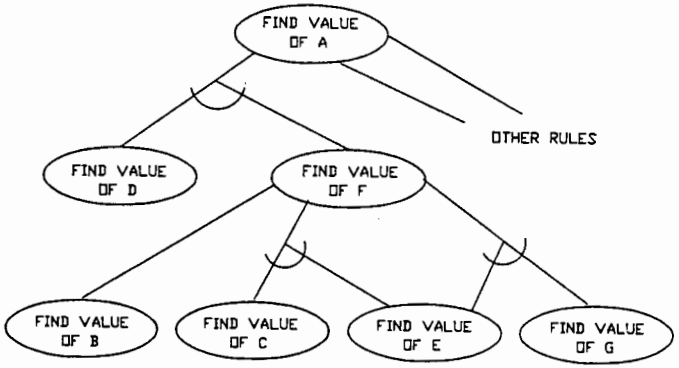


FIGURE 8 : REPRESENTATION USING RULES

Rule-based systems have enjoyed the most success in the construction of Expert Systems. MYCIN (in O'Shea [11]) uses the O-A-V rule representation and most PC based Expert System shells use the A-V form of rule representation. An example of an A-V rule is:

	ATTRIBUTE	PREDICATE	VALUE
IF	milling machine	is	requesting service
&	correct gripper	is	attached to robot
&	part location	is	input buffer
THEN	message sent to robot	is	"EXE MOVE_PART"

The above rule states exactly the correct action to be taken if the premise is true. In some domains it is advantageous to assign a certainty factor to the rule which represents the certainty of the inference contained in the rule

(eg. MYCIN and PROSPECTOR Expert Systems). By using these numbers it is possible to combine several sources of inconclusive information to form an almost certain conclusion.

Rule-based knowledge representations can be diagrammatically represented with AND/OR graphs as shown in Figure 8. Finding the solution a problem is then performed by searching the graph in a forward chaining or backward chaining manner.

4.4 REPRESENTATION USING LOGIC

Real-world facts, relations and actions can be represented by statements written as well-formed-formulas (wff's) in the First Order Predicate Calculus. Logic representations provide a useful method of reasoning with knowledge namely, theorem proving. There are two basic statements (wff's) in logic programming:

A. - an assertion or piece of declarative knowledge

A:- B1 &...& Bn - which is a piece of procedural knowledge stating that the sub-goals B1 to Bn must be satisfied in order to satisfy the sub-goal A.

A fact can be represented in logic as a binary (two-argument) relation. For example:

OBJECT(BOOK, RED)

states the fact that the object which is a book is the colour red. Action representations have n-argument relations. For example:

GIVES(BOOK, JOHN, MARY)

states that John carries out the action of giving a book to Mary. A procedure can be written as

HAS(BOOK, MARY):-

OBJECT(BOOK, RED) AND GIVES(BOOK, JOHN, MARY)

which states that Mary has the book, which is an object, after John carries out the action of giving the book to her.

There are a number of inferencing mechanisms which can be used to infer new conditions from a knowledge base of logical wff's. Modes Pones is one which uses the deduction that if A is true and A implies B, then B is true. Resolution is another inferencing mechanism which tries to prove that the negation of a query statement is inconsistent with the facts contained in the knowledge base. If it is inconsistent then

the statement is true; otherwise, it is false.

Wff's representing a problem can be mapped onto an AND/OR tree. The problems solution is then given as the sequence of logical operators applied to the data during the theorem proving process.

4.5 STRONGER PROBLEM SOLVING STRATEGIES

The control strategy determines the manner in which the problem representational graph is to be searched for a solution. When dealing with complex problems a more efficient search for a solution is facilitated by placing knowledge, specific to the problem, into the production system rules or operators. Heuristics are now coded into the procedural knowledge rather than in the control part of the system. Intelligent procedures working in conjunction with "weak" search techniques prune the search tree rapidly and lead to more efficient problem solving systems.

There are a number of ways in which a knowledge representation can hold information about which procedures (operators or rules) to consider during the traverse of a solution space :

1) Action-centred control systems have procedures that know which other sub-procedures can be considered by the control system. MYCIN, for example, has rules that contain the rule numbers of further rules that must be considered next if the current rules premise exists in the data base.

2) Object-centred control systems use class descriptions to determine which set of rules apply at any given time. For example, if the problem is to determine which mode of transport to use in travelling 50 km, then it is more feasible to go by car, taxi or train and not by bicycle or plane. A class of procedures is thus considered under certain conditions. GPS (General Problem Solver) uses this control.

3) Request-centred control systems have procedures which know their own purpose and volunteer for consideration when they can be used. This is used in blackboard systems as in the Hearsay II Expert System.

5.0 ADVANCED PROBLEM SOLVING - PLANNING

In order to solve most nontrivial problems, it is necessary to combine some of the "weak" search methods with one or more knowledge representation techniques. It is also necessary to divide the large problem into smaller problems that are solved separately. The separate solutions are then combined to form the full problem's solution. This is unlike the 8-puzzle presented in Part 1 where the problem is simple enough to allow the complete state description at each node of a graph. The methods used to decompose the original problem into sub-parts, and the ways of recording and handling interactions between the sub-parts as they are detected, are often collected under the heading of planning.

A linear planning system determines a complete solution for a sub-part before considering another as used in the STRIPS (Rich [5]) programme. For some problems, however, it is necessary to perform some work on one goal, then on another and then some more on the first. Systems which solve these problems are called non-linear planners. If a planner needs to solve particularly difficult problems it may be advantageous to consider the problem at different levels of abstraction. For example, when planning a route to get from one city to a particular place in another city, it would be best to consult a large scale national highway map initially. The lower level detail of a street map for the city could be consulted at a later stage. Systems that consider problems in this manner are called hierarchical planners.

There are more detailed planning systems which are specific to the particular types of problem that they solve, a discussion of which is not necessary here. All planning systems, however, derive their power from a great deal of domain specific knowledge. Due to this fact the most sophisticated planners are found in Expert Systems where a constrained problem is usually considered.

6.0 CONCLUSION

Artificial Intelligence problem solving techniques provide a range of tools which can be combined in any number of combinations to construct programmes for solving problems

that do not lend themselves to algorithmic solutions. Most real world problems fit into this category.

The most difficult part of applying these techniques is in deciding which to use in solving a particular problem. Initially, a problem representation - state-space or problem-reduction - must be chosen to make explicit those factors that can be utilized when searching a solution space. For small problems it is sufficient to use a simple knowledge representation (not necessarily symbolic based) with a heuristically guided control strategy which will efficiently search the problem graph. The larger and more complex the problem, the more sophisticated the knowledge representation scheme needed.

Most of the AI problem solving strategies have evolved by being designed to solve specific problems which have arisen in particular domains. As a result, it would be highly unlikely that one type of knowledge representation and control strategy would suit any other large, real-world problem. It is more appropriate then to look for similarities between well established problem solvers and the problem to be solved, and then to tailor the techniques accordingly. Many strategy combinations could fit a problem to be solved; however, as is the goal of AI researchers, it is necessary to pursue that combination which will yield the most efficient solutions in terms of computational time and resources.

REFERENCES

[1] A. BARR, E.A. FEIGENBAUM, "The Handbook of Artificial Intelligence", Volume 1, William Kaufmann, 1981.

[2] M. BODEN, "Artificial Intelligence and Natural Man", The Harvester Press Limited, USA, 1977.

[3] N. GRAHAM, "Artificial Intelligence. Making Machines Think.", Tab Books Inc., U.S.A., 1979.

[4] P.H. WINSTON, "Artificial Intelligence", (2nd ed), Addison-Wesley Publishing Company, Inc., U.S.A., 1984.

http://orion.journals.ac.za/

[5] E. RICH, "Artificial Intelligence", McGraw-Hill, Japan, 1983.

[6] N.J. NILSON, "Principles of Artificial Intelligence", Springer-Verlag, Berlin, 1982.

[7] T. WINOGRAD, "Frame Representations and the Declarative/Procedural Controversy", in BORROW, G D (Ed), "Representation and Understanding", Academic Press, New York, 1975, pp. 185-210.

[8] M.R. QUILLIAN, "Semantic Memory", in MINSKY, M (Ed), "Semantic Information Processing", MIT Press, England, 1986.

[9] M. MINSKY, "A Framework for Representing Knowledge" in WINSTON, P H (Ed), "The Psychology of Computer Vision", McGraw-Hill, 1975, pp. 211-277.

[10] P. HARMON, D. KING, "Expert Systems. Artificial Intelligence in Business.", John Wiley & Sons, Inc., New York, 1985.

[11] T. O'SHEA, M. EISENSTADT, "Artificial Intelligence. Tools, Techniques, and Applications.", Harper & Row, Publishers, New York, 1984.