Journal of Advances and Scholarly Researches in Allied Education
Vol. 3, Issue 6, April-2012, ISSN 2230-7540

Study on Search and Control Strategies

Parveen Kumar[1] Dr. Anuj Kumar[2]

[1]Department of Computer Science, Accurate Institute of Management & Technology, Greater Noida, (U.P.)

[2]Research Scholar, CMJ University, Shillong, Meghalaya

----------♦----------

Search is one of the operational tasks that characterize AI programs best. Almost every AI program depends on a search procedure to perform its prescribed functions. Problems are typically defined in terms of states, and solutions correspond to goal states. Solving a problem then amounts to searching through the different states until one or more of the goal states are found. In this chapter we investigate search techniques that will be referred to often in subsequent chapters.

PRELIMINARY CONCEPTS

Problem can be characterized as a space consisting of a set of states and a set of operators that map from one state to other states. Three types of states may be distinguished one or more initial states, a number of intermediate states, and one or more goal states. A solution to a problem is a sequence of operators that map an initial state to a goal state. A "best" or good solution is one that requires the fewest operations or the least cost to map from an initial state to a goal state. The performance of a particular solution method is judged by the amount of time and memory space required to complete the mapping. Thus, a solution based on some algorithm A_1 is consider better than one using algorithm A_2 if the time and space complexity of A_1 is less than that of A_2. It is customary to represent a search space as a diagram of a directed graph or a tree. Each node or vertex in the graph corresponds to a problem state, and arcs between nodes correspond to transformations or mapping between the states. The immediate successor of a node is referred to as children, siblings, or offspring, and predecessor nodes are ancestors. An immediate ancestor to a node is a parent.

Search can be characterized as finding a path through a graph or tree structure. This requires moving from node to node after successively expanding and generating connected nodes. Node generation is accomplished by computing the identification or representation code of children nodes from a parent node. Once this is done, a child is said to be generated and the parent is said to be explored. The process of generating all of the children of a parent is also known as expanding the node. A search procedure is a strategy for selecting the order in which nodes are generated and a given path selected.

Search problem may be classified by the information used to carry out a given strategy. In blind or uninformed search, no performance is given to the order of successor node generation and selection. The path selected is blindly or mechanically followed. No information is used to determine the preference of one child over another.

In informed or directed search, some information about the problem space is used to compute a preference among the children for exploration and expansion. Before proceeding with a comparison of strategies, we consider next some typical search problems.

WATER CONTAINER PROBLEM

There is a 4l container and 3l container; neither has any measuring markers on it. There is a pump that can be used to fill the containers with water. Problem to solve is to get exactly two liters of water in the 4l container.

SOLUTION

From initial state to goal state through appropriate sequence of moves or actions such as filling and emptying the containers.

Content of the two containers at any given time is a **problem state.**

Let :

x - Content of the 4l container

y - Content of the 3l container

Then :

(x,y) - Problem state represented by an ordered pair.

The set of all ordered pairs is the **space** of problem states or the **state-space** of the problem.

State-space : { (x,y) | x = 0,1,2,3,4 y = 0,1,2,3 }

Data structure to represent the state-space can be :

- **vectors**
- **sets**
- **arrays**
- **lists**

etc...

Problem statement:

Initial state (0,0)

Goal state (2,y) where y = any possible number.

Moves transform from one state into another state.

Operators determine the moves.

Operators for the problem state-space:

1. Fill the 4l container
2. Fill the 3l container
3. Empty the 3l container
4. Empty the 3l container
5. Pour water from 3l container into 4l conatiner until 4l container is full
6. Pour water from 4l container into the 3l container until the 3l container is full
7. Pour all the water from 3l container into the 4l container
8. Pour all the water from 4l container into the 3l container

Preconditions need to be satisfied before an operator can be applied.

EXAMPLE:
 # 1 can be applied if there is less than 4l water in the container.
IF there is less than 4l in the 4l container THEN fill the 4l container.
Adding pre-conditions to operators => generation of production rules.
Forwarded form of rule # 1:

IF (x,y| x?4) THEN (4,y)

The forwarded set of production rules :

R1	IF	(x,y|		x?4)				THEN	(4,y)
R2	IF	(x,y|		y?3)				THEN	(x,3)
R3	IF	(x,y|		x>0)				THEN	(0,y)
R4	IF	(x,y|		y>0)				THEN	(x,0)
R5	IF	(x,y|	x+y>=4	?	y>0	?	x?4)	THEN	(4,y-(4-x))
R6	IF	(x,y|	x+y>=3	?	x>0	?	y?3)	THEN	(x-(3-y),3)
R7	IF	(x,y|	x+y?=4	?	y>0)			THEN	(x+y,0)
R8	IF (x,y| x+y?=3 ? x>0) THEN (0,x+y)								

In certain states, more than one rule can be applied.

EXAMPLE:
(4,0) satisfies the preconditions of R2,R3 ? R6

4.6 PRODUCTION SYSTEM

Since search forms the core of many intelligent processes. It is useful to structure AI programs in a way that facilitates describing and performing the search process. Production systems provide such structures. A definition of a production system is given below. Do not be confused by other uses of the word production, such as to describe what is done in factories. A production system consists of:

- A set of rules, each consisting of a left side (a pattern) that determines the applicability of the rule and a right side that describes the operation to be performed if the rule is applied.
- One or more knowledge/databases that contain whatever information is appropriate for the particular task. Some parts of the database may be permanent, while other parts of it may pertain only to the solution of the current problem. The information in these databases may be structured in any appropriate way.
- A control strategy that specifies the order in which the rules will be compared to the database and a way of resolving the conflicts that arise when several rules match at once.
- A rule applier. So far, our definition of a production system has been very general. It encompasses a great many systems, including water jug problem solver. It also encompasses a family of general production system interpreters, including:
- Basic production system languages, such as OPS5 and ACT.
- More complex, often hybrid systems called expert system shells, which provide complete (relatively speaking) environments for the construction of

knowledge base expert systems.

- General problem-solving architectures like SOAR, a system based on a specific set of cognitively motivated hypotheses about the nature of problem solving.

All of these systems provide the overall architecture of a production system and allow the programmer to write rules that define particular problems to be solved.

REFERENCES:

- Arbib, M. A., and A. R. Hanson (1990). Vision, Brain, and Cooperative Computation. Cambridge, MA: MIT Press.
- Atmar, J. W. (1976). "Speculation on the Evolution of Intelligence and Its Possible Realization in Machine Form." Sc.D. diss., New Mexico State University, Las Cruces.
- Atmar, J. W. (1979). "The Inevitability of Evolutionary Invention." Unpublished manuscript.
- Atmar, W. (1991). "On the Role of Males," Animal Behaviour, Vol. 41, pp. 195–205.
- Atmar, W. (1994). "Notes on the Simulation of Evolution," IEEE Transactions on Neural Networks, Vol. 5, no. 1.
- Barr, A., and E. A. Feigenbaum (1981). The Handbook of Artificial Intelligence, Vol. 1. San Mateo, CA: William Kaufmann.
- Barr, A., P R. Cohen, and E. A. Feigenbaum (1989). The Handbook of Artificial Intelligence, Vol. 4. Reading, MA: Addison-Wesley.
- Barron, A. R. (1993). "Universal Approximation Bounds for Superpositions of a Sigmoidal Function," IEEE Trans. Info. Theory, Vol. 39:3 .
- Bennett, J. S., and C. R. Hollander (1981). "DART: An Expert System for Computer Fault Diagnosis," Proc. of IJCAI-81.
- Bernstein, A., V. De, M. Roberts, T. Arbuckle, and M. A. Beisky (1958). "A Chess Playing Program for the IBM 704," Proc. West Jut. Comp. Conf., Vol. 13, pp. 157–159.
- Bezdek, J. C., J. Keller, R. Krishnapuram, and N. K. Pal, (eds.) (1999). Fuzzy Models and Algorithms for Pattern Recognition and Image Processing, Kluwer, Norwell, MA.
- Bezdek, J. C., and S. K. Pal (1992). Fuzzy Models for Pattern Recognition: Models that Search for Structures in Data. Piscataway, NJ: IEEE Press.
- Block, H. D. (1962). "The Perceptron: A Model for Brain Functioning," Rev. Mod. Phys., Vol. 34.
- Block, H. D. (1963). "Adaptive Neural Networks as Brain Models," Proc. of Symp. Applied Mathematics, Vol. 15.
- c01.qxd 10/21/2005 7:38 AM Page 27 Block, H. D. (1970). "A Review of `Perceptrons: An Introduction to Computational Geometry,'"
- Information and Control, Vol. 17:5, pp. 501–522.
- Block, H. D., B. W. Knight, and F. Rosenblatt (1962). "Analysis of a Four Layer Series Coupled Perceptron," Rev. Mod. Phys., Vol. 34, pp. 135–142.
- Blume, H. (1998). "The Digital Philosopher," The Atlantic Online, Dec. 9.
- Buchanan, B. G., and E. H. Shortliffe (1985). Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project. Reading, MA: Addison-Wesley.
- Campbell, A. N., V. F. Hollister, R. O. Duda, and P. E. Hart (1982). "Recognition of a
- Hidden Mineral Deposit by an Artificial Intelligence Program," Science, Vol. 217, pp.
- 927–929.
- Cannon, W. D. (1932). The Wisdom of the Body. New York: Norton and Company.
- Carne, E. B. (1965). Artificial Intelligence Techniques. Washington, DC: Spartan Books.
- Cerf, C., and V. Navasky (1984). The Experts Speak: The Definitive Compendium of Authoritative Misinformation. New York: Pantheon Books.
- Charniak, E., and D. V. McDermott (1985). Introduction to Artificial Intelligence. Reading, MA: Addison-Wesley.
- Clark, D. (1997). "Deep Thoughts on Deep Blue," IEEE Expert, Vol. 12:4, p. 31.

- Countess of Lovelace (1842). "Translator's Notes to an Article on Babbage's Analytical Engine."
- In Scientific Memoirs, Vol. 3, edited by R. Taylor, pp. 691–731.
- Covington, M. A., D. Nute, and A. Vellino (1988). Prolog Programming in Depth. Glenview,
- IL: Scott, Foresman. Cybenko, G. (1989). "Approximations by Superpositions of a Sigmoidal Function," Math. Contr. Signals, Syst., Vol. 2, pp. 303–314.
- Dreyfus, H., and S. Dreyfus (1984). "Mindless Machines: Computers Don't Think Like Experts, and Never Will," The Sciences, November/December, pp. 18–22.
- Dreyfus, H., and S. Dreyfus (1986). "Why Computers May Never Think Like People," Tech. Review, January, pp. 42–61.
- Dubois, D., and H. Prade (1982). "A Class of Fuzzy Measures Based on Triangular
- Norms–A General Framework for the Combination of Uncertain Information" Int. J. of General Systems, Vol. 8:1.
- Duda, R., P. E. Hart, N. J. Nilsson, P. Barrett, J. G. Gaschnig, K. Konolige, R. Reboh, and J. Slocum (1978). "Development of the PROSPECTOR Consultation System for Mineral Exploration," SRI Report, Stanford Research Institute, Menlo Park, CA.
- Feigenbaum, E. A., B. G. Buchanan, and J. Lederberg (1971). "On Generality and Problem Solving: A Case Study Involving the DENDRAL Program." In Machine Intelligence 6, edited by B. Meltzer and D. Michie. New York: American Elsevier, pp. 165–190.
- Fogel, L. J. (1962). "Autonomous Automata," Industrial Research, Vol. 4, pp. 14–19.
- Fogel, L. J. (1964). "On the Organization of Intellect." Ph.D. diss., UCLA.
- Fogel, L. J., A. J. Owens, and M. J. Walsh (1966). Artificial Intelligence through Simulated Evolution. New York: John Wiley.
- Fogel, D. B. (2002). Blondie24: Playing at the Edge of AI, San Francisco: CA, Morgan Kaufmann.
- Genesereth, M. R., and N. J. Nilsson (1987). Logical Foundations of Artificial Intelligence. Los Altos, CA: Morgan Kaufmann.
- 28 DEFINING ARTIFICIAL INTELLIGENCE c01.qxd 10/21/2005 7:38 AM Page 28 Giarratano, J. C. (1998). Expert Systems: Principles and Programming, Brooks Cole, NY.
- Gould, J. L., and C. G. Gould (1985). "An Animal's Education: How Comes the Mind to be Furnished?" The Sciences, Vol. 25:4, pp. 24–31.
- Greenblatt, R., D. Eastlake, and S. Crocker (1967). "The Greenblatt Chess Program," FJCC, Vol. 31, pp. 801–810.
- Grossberg, S. (1976). "Adaptive Pattern Classification and Universal Recoding: Part I. Parallel Development and Coding of Neural Feature Detectors," Biological Cybernetics, Vol. 23, pp. 121–134. Grossberg, S. (1982). Studies of Mind and Brain. Dordrecht, Holland: Reidel.
- Haykin, S. (1994). Neural Networks: A Comprehensive Foundation. New York: Macmillan. Hebb, D. O. (1949). The Organization of Behavior. New York: Wiley.
- Hecht-Nielsen, R. (1990). Neurocomputing. Reading, MA: Addison-Wesley.
- Hoffman, A. (1989). Arguments on Evolution:A Paleontologist's Perspective. New York: Oxford Univ. Press.
- Hofstadter, D. R. (1985). Metamagical Themas: Questing for the Essence of Mind and Pattern. New York: Basic Books.
- Hopfield, J. J. (1982). "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," Proc. Nat. Acad. of Sciences, Vol. 79, pp. 2554–2558.
- Hopfield, J. J., and D. Tank (1985). "'Neural' Computation of Decision in Optimization Problems," Biological Cybernetics, Vol. 52, pp. 141–152.
- Hornik, K., M. Stinchcombe, and H. White (1989). "Multilayer Feedforward Networks Are Universal Approximators," Neural Networks, Vol. 2, pp. 359–366.
- Jackson, P. C. (1985). Introduction to Artificial Intelligence, 2nd ed. New York: Dover.

- Kandel, A. (1986). Fuzzy Expert Systems. Boca Raton, FL: CRC Press.
- Kasabov, N. K. (1996). Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering, MIT Press, Cambridge, MA.
- Keller, H. B. (1961). "Finite Automata, Pattern Recognition and Perceptrons," Journal of Assoc. Comput. Mach., Vol. 8, pp. 1–20.
- Kesler, C. (1961). "Preliminary Experiments on Perceptron Applications to Bubble Chamber Event Recognition." Cognitive Systems Research Program, Rep. No. 1, Cornell University, Ithaca, NY.
- Kohonen, T. (1984). Self-Organization and Associative Memory. Berlin: Springer-Verlag.
- Lenat, D. B. (1983). "EURISKO: A Program that Learns New Heuristics and Domain Concepts," Artificial Intelligence, Vol. 21, pp. 61–98.
- Levine, D. S. (1991). Introduction to Neural and Cognitive Modeling. Hillsdale, NJ: Lawrence Erlbaum.
- Levy, D. N. L., and M. Newborn (1991). How Computers Play Chess. New York: Computer Science Press.
- Lindsay, R. K. (1968). "Artificial Evolution of Intelligence," Contemp. Psych., Vol. 13:3, pp. 113–116.
- Lindsay, R. K., B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg (1980). Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project. New York: Mc- Graw-Hill.
- Manuel, T. L. (2003). "Creating a Robot Culture: An Interview with Luc Steels," IEEE Intelligent Systems Magazine, May/June, pp. 59–61.
- Maynard Smith, J. and E. Szathmáry (1999). The Origins of Life: From the Birth of Life to the Origin of Language, Oxford University Press, New York.
- Mayr, E. (1988). Toward a New Philosophy of Biology: Observations of an Evolutionist. Cambridge, MA: Belknap Press.
- McCarthy, J., P. J. Abrahams, D. J. Edwards, P. T. Hart, and M. I. Levin (1962). LISP 1.5 Programmer's Manual. Cambridge, MA: MIT Press.
- McCarthy, J. (1997). "AI as Sport," Science, Vol. 276, pp. 1518–1519.
- McCulloch, W. S., and W. Pitts (1943). "A Logical Calculus of the Ideas Immanent in Nervous Activity," Bull. Math. Biophysics, Vol. 5, pp. 115–133.
- Mead, C. (1989). Analog VLSI and Neural Systems. Reading, MA: Addison-Wesley.