Instructional Science 16: 137-150 (1987)

A method for sequencing instructional objectives which minimizes memory load

JOHN C. NESBIT & STEVE HUNKA
Faculty of Education, University of Alberta, Edmonton, CANADA T6G 2G5

Abstract. A Gagné-style learning hierarchy often permits a large number of alternate linear arrangements (sequences) of instructional objectives. An alternative is described here to traditional methods of choosing between sequences. Its premise is that, for every sequence, a value termed the *memory load* can be calculated which is theoretically related to the probability that students will fail to recall prerequisite objectives. A graph theoretic approach is taken in presenting an algorithm which generates a minimal memory load sequence from a learning tree, a restricted but frequently encountered type of learning hierarchy. In order to assess the effectiveness of the algorithm in generating low memory load sequences when given hierarchies which are not trees, it was applied to several published examples of learning hierarchies. The results indicated that the algorithm is effective as an heuristic, especially when combined with a hill-descending procedure which attempts to incrementally improve the generated sequence.

Learning hierarchies

Since its emergence in the early 1960's, Gagné's learning hierarchy (Gagné, 1962, 1968; Gagné and Paradise, 1961) has been one of the most widely accepted and influential concepts in the field of instructional science. A learning hierarchy consists of a set of instructional objectives (or tasks, or skills) and a set of prerequisite relationships connecting the objectives. Gagné and Briggs (1974, p. 110) noted that "a prerequisite skill is integrally related to the skill which is superordinate to it, in the sense that the latter skill cannot be done if the prerequisite skill is not available to the learner". In practice, because each objective is associated with a unit of instructional treatment, the learning hierarchy specifies a partial order for presenting units to a student.

Learning hierarchies are normally generated by a subjective process in which the instructional designer starts with goal objectives and recursively breaks them down into prerequisite sub-objectives. This process terminates at sub-objectives assumed to be already mastered by the student population at which the instruction is aimed.

Subject matter experts do not always agree on the partial ordering of a given set of objectives, so several methods, reviewed by Reckase and McKinley (1982), have been proposed for using test data to empirically validate learning hierarchies. Most of these methods (*e.g.*, Dayton and Macready, 1976; Airasian and Bart, 1975; Gagné and Paradise, 1961) are based on the principle that a response pattern

in which an objective is passed and the prerequisite of the objective is failed contributes contradictory evidence. Procedures for automatically generating learning hierarchies from test data (Macready, 1975; Bart and Krus, 1973) are based on the same principle.

Once a valid learning hierarchy has been obtained, it is used to derive a sequence of objectives to be followed by the student. Even when a learner control philosophy prevails, the instructional system should be able to recommend a sequence. Tennyson (1981) found that the achievement of learners allowed to control the amount of instruction was significantly lower than that of learners in a program controlled treatment, unless they were advised on when to terminate instruction on each objective in the lesson. When the objectives are hierarchically related, advice on the best sequence of objectives may have similar value.

The main purpose of this paper is to propose a criterion for selecting a sequence of objectives from the many sequences which are often permitted by a learning hierarchy. Before describing this criterion and a sequence generation algorithm which exploits it, a brief excursion is taken through some simple graph theory definitions as they apply to learning hierarchies. To provide a practical orientation for this discussion, the sequence generation algorithm, together with certain graph algorithms noted in the following section, can be viewed as forming the first rudimentary facilities for an hypothetical computer assisted instructional design toolkit.

Learning hierarchies as acyclic digraphs

As other authors (Heines and O'Shea, 1985; Stelzer and Kingsley, 1975) have observed, a learning hierarchy can be formally represented as a directed graph (digraph). A digraph consists of a set of nodes and a set of ordered pairs (u,v), called arcs, where u and v are distinct members of the set of nodes. When a digraph is used to model a learning hierarchy, the nodes represent instructional objectives and the arcs show the prerequisite relationships between the objectives.

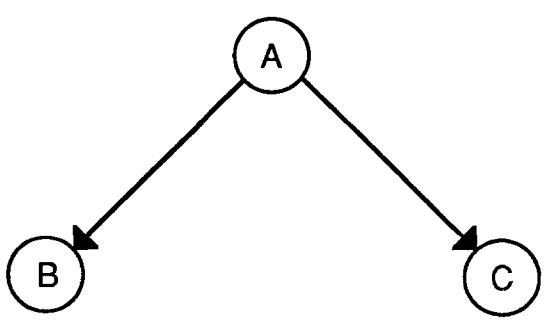

Fig. 1. A simple digraph

Figure 1 shows a simple digraph with three nodes: A, B, and C. In interpreting the arcs, which are represented by arrows, we say that A is the parent of B and C, and that B and C are the children of A. If this digraph were representing a learning hierarchy, it would tell us that both B and C are prerequisites of A. In other words, the learning hierarchy can be used to partition the set of 3! linear arrangements or sequences of the three objectives into two mutually exclusive subsets: the set of instructionally valid sequences (BCA, CBA) and the set of instructionally invalid sequences (ABC, ACB, BAC, CAB). An instructionally valid sequence is an ordered list in which all objectives in the hierarchy appear exactly once, and in which every objective occurs at some position after all its prerequisites. Valid sequences are said to be "permitted" by the learning hierarchy.

With learning hierarchies, the arrowhead can be dropped if we adopt the convention that prerequisites always appear lower on the page than their parents. Figure 2 shows a learning hierarchy cited by Case and Bereiter (1984). In digraphs of this form the number of arcs exiting downward from a node is the outdegree of the node. The number of arcs entering the node from above is the indegree of the node. For example, node A in Figure 2 has indegree zero and outdegree three. In learning hierarchies, nodes with indegree zero represent goal objectives and those with outdegree zero represent entry objectives.

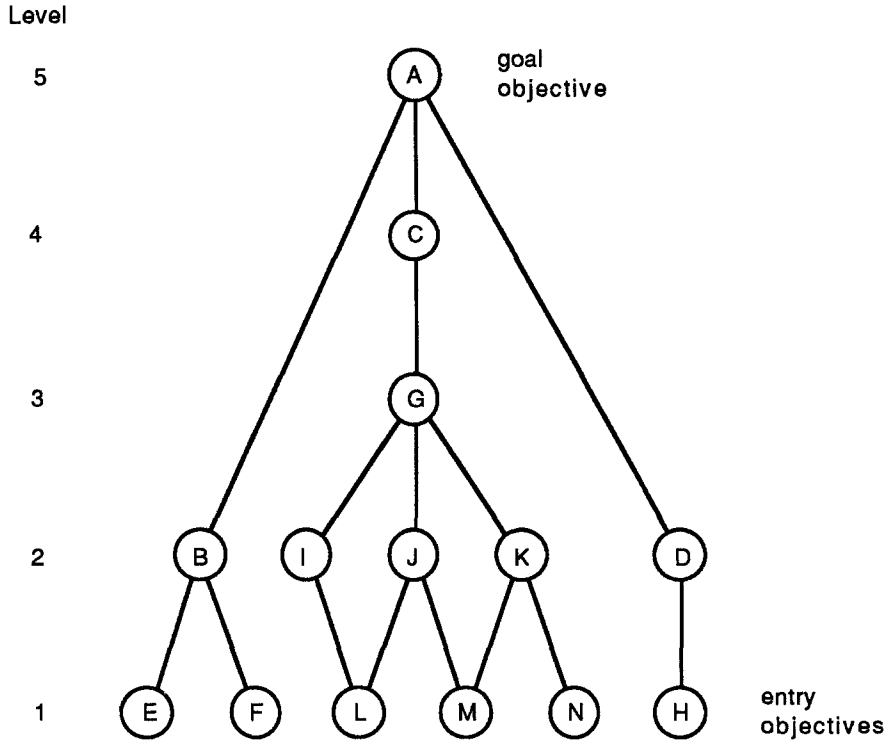

Fig. 2. A learning hierarchy

A path (not to be confused with a sequence) is a list of one or more nodes, starting with any node in the digraph, such that each succeeding member of the list is a child of its predecessor in the list. The length of a path is $n-1$, where n is the number of nodes in the path. A trivial path is a list containing only one node, and therefore having a length of zero.

A digraph is cyclic if and only if it has one or more non-trivial paths which begin and end with the same node. Implicit in discussions by Gagné and others, is the tenet that learning hierarchies are acyclic. Incidentally, the absence of cycles ensures that there will be at least one entry objective and at least one goal objective. So perhaps a graph algorithm capable of checking whether a learning hierarchy entered by a course designer is free of cycles, should be the first to go into a toolkit for computer assisted instructional design. A simple algorithm which tests for the acyclic property is described by Robinson and Foulds (1980, p. 73-75).

Level numbers

Sometimes it is useful to assign a positive integer to each instructional objective indicating its level in the hierarchy: a frequently encountered application being the numbering of university courses (e.g., the high order digit in "Psych 100"). In graph-theoretic terms the level number of a node in an acyclic digraph may be defined as the length of the longest path of which it is the first node. Entry objectives are the last node of every path to which they belong, so every path beginning with an entry objective is trivial and has a length of zero. Therefore entry objectives are always assigned to level zero by the given definition. It may be preferable to increment all level numbers by one so that the lowest level is one rather than zero. Figure 2 illustrates level assignment by this method.

Inreach and outreach

If u and v are nodes in a digraph, and there exists a path from u to v, then we say that v is reachable from u. The inreach of a node v is the set of all nodes from which v is reachable, including v itself. The outreach of a node u is the set of all nodes reachable from u, including u itself. So the outreach of an objective u in a learning hierarchy is the set of all objectives in the hierarchy which must be taken before u, plus u itself.

Interpretation and treatment of inessential arcs

If (u,v) is an arc in an acyclic digraph, then (u,v) is inessential if there is a path from u to v which does not traverse (u,v), and essential if there is no such path. Figure 3 contains an example of an inessential arc. A prerequisite relationship represented by an inessential arc has no effect on the set of sequences permitted by

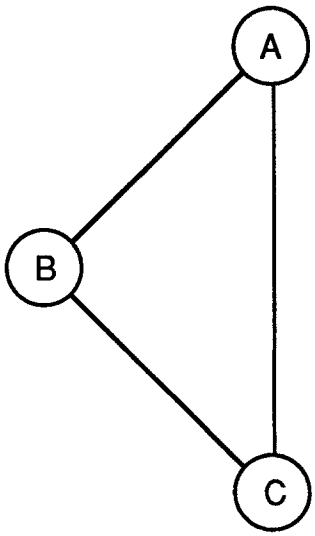

Fig. 3. The arc from A to C is Inessential

the learning hierarchy. Therefore, if the only purpose of the hierarchy is to define this set, an algorithm which simplified the hierarchy by detecting and deleting inessential arcs (Robinson and Foulds, p. 85) would be a useful addition to the toolkit.

Prerequisite relationships are also used to implicate sub-objectives as causes of failure when a student is unable to master an objective. If we assume, as Gagné and Briggs seem to, that mastery of an objective implies mastery of all objectives within its outreach, then inessential arcs are indeed redundant. However, if prerequisites are allowed in the hierarchy which are necessary for learning the new objective, but which are not incorporated and practiced as part of it, then "inessential" arcs may represent information valuable to the diagnostic process. One solution is to enable the author to specify both types of relationships (let us call them integral and non-integral relationships), and to only allow an inessential arc when it does not short-cut a path connected by arcs of the integral kind.

Augmented learning hierarchies

As Gagné (1968) observed, a standard learning hierarchy cannot represent alternative ways of achieving an instructional goal. One hypothesis accounting for the contradictory fail-pass response pattern (in which a prerequisite is failed but its parent is passed) which can thwart the validation of a learning hierarchy, is that a known prerequisite is sufficient but not necessary and that students producing this pattern had mastery of another sufficient prerequisite not represented in the model.

This limitation can be overcome by a representation, used both in the task models of Heines and O'Shea (1985) and in Pask's entailment structures (Pask, Kallikourdis and Scott, 1975), which would indicate that any one of a specified set of subobjectives can serve as a prerequisite. In other words, hierarchies augmented by such a representation are AND/OR graphs, allowing for both conjunction and disjunction of prerequisites where standard Gagné hierarchies allow only conjunction. The work reported here deals only with standard conjunctive hierarchies.

Memory load

The set of sequences of instructional objectives permitted by a learning hierarchy can be surprisingly large. For example, the learning hierarchy in Figure 2 is a partial ordering of only 14 objectives, yet it allows about 1.6 million different sequences. In fact, the number of permitted sequences is often so large that computer programs attempting to count them by exhaustive search will not terminate within a reasonable period of time. Are there criteria available for selecting the most instructionally effective sequence from this large set?

Posner and Strike (1976) reviewed and categorized many of the principles for sequencing instructional content which have been proposed in the last eighty years. Although the learning hierarchy principle has dominated in recent years, other principles worthy of consideration include the ordering of objectives from most familiar to least familiar, from least difficult to most difficult, from most interesting to least interesting, and so on. If the learning hierarchy is given top priority, the other principles can still be invoked within the constraints it imposes. Gagné and Briggs (1974) suggest that resource availability determine sequencing after the constraints of the learning hierarchy have been satisfied. This paper proposes a new sequencing principle which may be viewed as an extension of the learning hierarchy principle.

Although all permitted sequences ensure that when the student begins to learn a new objective the prerequisites will have been mastered at some previous time, it is possible that some or all of the prerequisites will have been forgotten. The probability of forgetting is known to increase with time as a result of interference from other learning. Therefore, one approach to finding a "best" sequence is to minimize the instructional time elapsing between when an objective is learned and when it is needed for further learning.

Memory load is an attempt to provide, for the practical purposes of the instructional designer, a relative estimate of the retroactive inhibition effects contributing to the forgetting of prerequisites in a sequence with an underlying hierarchical structure. The term memory load was chosen because sequences subject to greater interference presumably place a greater burden on the memory ability of the learner.

Suppose an estimated learning time t is associated with each objective. A memory load value can be obtained by any permitted sequence of objectives:

$$\text{memory load} = \sum_{i=1}^{m} \sum_{j=1}^{q_i} t_{ij}$$

where m is the number of arcs in the hierarchy, q_i is the number of objectives intervening between the parent and child of the ith arc, and t_{ij} is the estimated learning time of the jth objective intervening between the parent and child of the ith arc.

The following sequence is permitted by the hierarchy in Figure 2:

N M K L J I G C F E B H D A

Assuming that all objectives have a learning time of 1.0, the memory load for this sequence is 16.0, which happens to be the minimum memory load for the hierarchy. The arc from A to C is stretched by five intervening objectives, so it contributes 5.0 to the memory load. The arc from D to H has no intervening objectives, so it contributes 0.0 to the memory load.

Deficiencies of memory load as a model of forgetting

The utility of minimizing memory load in the instructional design process depends on the accuracy of memory load as a predictor of forgetting. Four potential sources of error are apparent:

1. Time spent in activities external to the learning hierarchy is a cause of forgetting, but is not captured by the memory load model. So one expects that the accuracy of memory load as a predictor of forgetting will vary with the degree of intrusion of external activities at course delivery time.

2. Some objectives are simply more memorable than others, but the memory load model fails to differentiate between them. Many psychological factors will come into play to vary the likelihood that a specific objective will be remembered by a specific individual. For example, objectives which the student finds interesting are likely to be remembered longer than those which are boring. If an objective is known to be relatively resistant to forgetting, one can afford to allow it greater separation from the objectives having it as a prerequisite in order to shorten the time spanned by more sensitive prerequisite relationships.

3. The probability of forgetting usually increases as a non-linear function of time, but the memory load model assumes linearity. One expects intervening objectives to vary in the strength with which they interfere with memory for the prerequisite. It is conceivable that some intervening objectives may help the student to recall the prerequisite (negative interference). Therefore, one cannot even be sure that the probability of forgetting will be a monotonically increasing function of instructional time.

4. No consideration is given here to the distribution of memory load over the sequence, but this may turn out to be an important factor. For example, it may be that objectives arranged in a sequence with a uniform distribution of memory load are learned more easily than the same objectives arranged with an equal memory load having a peaked distribution.

These deficiencies suggest that a more accurate model could be constructed which would include as parameters some of the factors ignored by the current model. Unfortunately though, the additional information required by such a model is usually unavailable to the instructional designer. However, there is reason to expect that when the instruction is delivered by computer, the frequency with which each prerequisite is forgotten could be recorded and fed back into the sequence planning process.

A sequencing algorithm which minimizes memory load

Consider the problem of finding any sequence permitted by a given learning hierarchy such that the sequence has the minimum memory load. The huge number of sequences permitted by many learning hierarchies severely limits the usefulness of methods relying on an exhaustive search of the sequence space. A preliminary study which applied A* search[1] using an heuristic based on the minimum possible memory load associated with an objective not yet included in the sequence, found that even this approach is frequently defeated by combinatorial explosion.

The ideal solution would be a more direct one which would avoid searching the sequence space altogether. Unfortunately, no algorithm of this type has been found which succeeds with learning hierarchies in general[2]. However, a simple and efficient algorithm is presented here which is conjectured to generate a minimum memory load sequence for a subclass of learning hierarchies called learning trees.

A tree is an acyclic digraph containing only nodes having indegrees less than or equal to one, and exactly one node having indegree zero. Any learning hierarchy satisfying these restrictions is a learning tree. It turns out that learning trees are a fairly frequent form of learning hierarchy. An informal survey of 49 learning hierarchies cited in 14 documents (Briggs, 1972; Briggs and Wager, 1981; Case and Bereiter, 1984; Dick and Carey, 1978; Edney, 1972; Gagné, 1962, 1965, 1968; Gagné and Briggs, 1974; Gagné and Paradise, 1961; Heines and O'Shea, 1985; Riban, 1969; Walter, 1965; Wollmer and Bond, 1975) found that 31% were learning trees, 59% were non-trees having only one goal objective, and 10% had multiple goal objectives.

GENERATOR, an algorithm which finds a minimum memory load sequence permitted by a learning tree, is given below. When applied to learning trees, the effect of the algorithm is to find a permitted sequence of objectives with a) the objectives belonging to the same outreach positioned contiguously, and b) the

prerequisites of any objective ordered such that a prerequisite having an outreach with a greater total learning time will precede one having a lesser total learning time. The time complexity of GENERATOR is O(n), where n is the number of nodes in the tree.

The GENERATOR Algorithm

1) To each node (objective) in the learning tree assign the value T, the sum of the estimated learning times (t) of all nodes in its outreach.
2) Initialize the sequence to be a null string of nodes.
3) Invoke the following recursive procedure SUB_GENERATOR passing it the goal node of the learning tree.

```
procedure SUB_GENERATOR (α : node);
var β : node;
begin
while α has any unmarked child do
   begin
   β <- unmarked child of α with greatest T;
   SUB_GENERATOR (β);
   end;
append α to sequence;
mark α
end;
```

Generalizing the algorithm to deal with non-trees

When no procedure is known for obtaining an optimal solution to a problem, one is often forced to rely on methods that obtain solutions which are at least better than those resulting from an unguided search. In the hope of obtaining such a method, a version of GENERATOR was developed which differs from the version given previously in only one respect: the node β passed in the recursive call to SUB_GENERATOR is the unmarked child of α having the greatest value T_k/P_k, where node k is any unmarked child of α, T_k is the sum of the estimated learning times of all nodes in the outreach of node k, and P_k is the number of unmarked nodes *not in the outreach of k* which are parents of nodes that are in the outreach of k. It should be clear that $P_k \geq 1$ because the parents of any node k cannot be in the outreach of k, and cannot be marked if k is not marked. In the case of trees, $P_k = 1$ for all k, so the algorithm reduces to its original form.

Consider the application of the modified version of GENERATOR to the example in Figure 4. The nodes are labeled with their T values and initial P values. The sequence generated by the algorithm is: E D C B A. Note, unlike the original version of GENERATOR, the modified version is constrained to choosing E rather than D as the initial node because $T_E/P_E > T_D/P_D$.

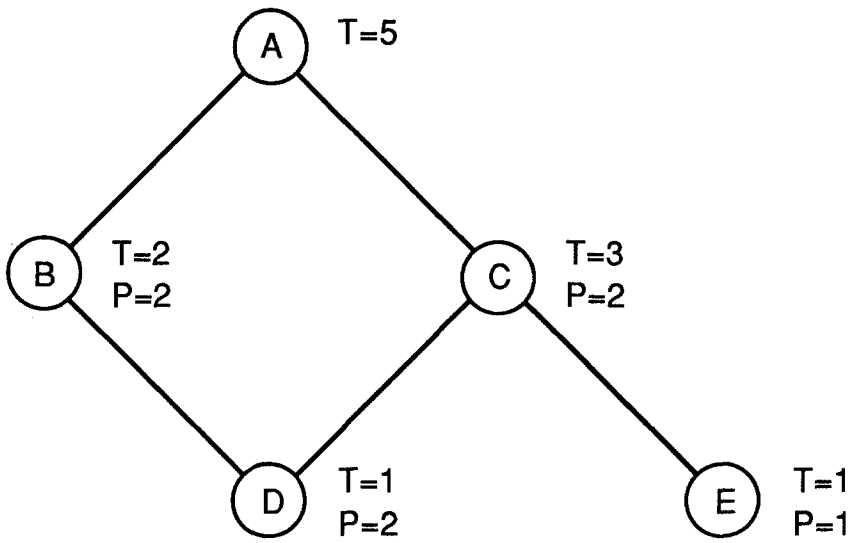

Fig. 4. A hierarchy with nodes labeled by GENERATOR

Tests were conducted with several published examples of learning hierarchies which are not trees to see whether the modified version of GENERATOR holds promise as an heuristic method. The results of the tests are summarized in Table 1. Because it is only hierarchies permitting a large number of sequences which pose a problem, the hierarchies with the largest number of objectives of those hierarchies surveyed were chosen for testing. With the exception of hierarchy 15 (H15), which had one of its two goal objectives deleted, only hierarchies having exactly one goal objective were used. Estimated learning times are not usually given with published examples of hierarchies, so objectives were simply assigned a learning time of one (except for those in H3 and H19 which were assigned random learning times).

All programs were written in Pascal and ran on a Digital Equipment Corporation VAX 11/780 minicomputer. On this machine, GENERATOR consumed less than 100 milliseconds of CPU time per hierarchy.

Entries in the "no. of sequences", "max ML", "ave ML", and "min ML" columns were obtained by a depth-first exhaustive search program which counted sequences and kept track of the maximum, average, and minimum memory loads encountered. An arbitrary upper limit of 12 hours was set on the CPU time required by the depth-first search. In cases where this search was not completed in the allotted time, the accumulated count is presented as a lower bound on the number of permitted sequences, and a random search program was run for another 12 hours to get more accurate memory load estimates.

Table 1. Tests of GENERATOR with some published learning hierarchies

Source	no. of nodes	no. of sequences	max ML	ave ML	min ML	GENERATOR ML	SHIFTER ML
1. Gagné & Briggs (1974, p. 117)	15	90090	29	22.0	9	9	9
2. Edney (1972, p. 103)	29	≥1790000	≥270	241.1	≤122	67	67
3. Edney (1972, p. 103)	29	≥1790000	≥17752	12403.4	≤6883	2733	2733
4. Gagné (1962, p. 359)	10	756	22	17.0	9	9	9
5. Gagné & Briggs (1974, p. 118)	9	480	24	18.2	10	10	10
6. Gagné & Briggs (1974, p. 114)	11	480	24	19.8	15	15	15
7. Riban (1969, p.119)	11	16800	35	26.4	12	12	12
8. Gagné & Briggs (1974, p. 116)	12	44	15	13.3	11	11	11
9. Case & Bereiter (1984, p. 145)	14	1570140	57	43.5	16	16	16
10. Dick & Carey (1978, p. 29)	9	52	12	10.8	9	10	9
11. Walter (1965, p. 52)	16	16200	40	31.6	20	21	20
12. Wollmer & Bond (1975, p. 8)	16	6336	53	45.5	39	41	39
13. Gagné (1965, p. 181)	18	33880	83	72.8	60	64	60
14. Dick & Carey (1978, p.46)	18	≥3990000	≥71	52.7	≤23	23	23
15. Gagné & Paradise (1961, p. 6)	23	≥2700000	≥149	112.3	≤64	46	46
16. Briggs (1972, p. 121)	23	≥2680000	≥104	74.8	≤44	44	44
17. Walter (1965, p. 49)	20	≥3440000	≥134	106.0	≤82	96	82
18. Gagné (1965, p. 150)	20	≥3550000	≥147	129.2	≤98	124	105
19. Dick & Carey (1978, p. 46)	18	≥3990000	≥3906	2743.9	≤1176	1171	1171

A second phase for the sequencing algorithm was developed, called SHIFTER, which incrementally improves a sequence produced by GENERATOR. SHIFTER steps through the sequence, and at each node tests whether the node can be moved to any new position such that memory load is decreased. SHIFTER only terminates when no single node can be moved to improve the memory load. Like other hill-descending methods, SHIFTER gets snagged on local minima.

H1 through H3 are the only learning trees in the sample. They were included to demonstrate the effectiveness of GENERATOR when applied to hierarchies of this type. The search program was able to complete an exhaustive search of the sequence space of H1, and a comparison of columns reveals that GENERATOR did produce a minimum sequence. In the case of H2, GENERATOR produced a sequence having a memory load considerable lower than that of the best sequence found by the random search program. H3 is the same hierarchy as H1 with randomly assigned learning times (integers between 1 and 100).

H4 through H9 are hierarchies that could be thoroughly searched and for which GENERATOR produced minimum sequences. H10 through H13 are those that could be thoroughly searched but for which GENERATOR did not produce a best sequence. However, in these four cases SHIFTER was able to improve the generated sequences to obtain minimum sequences.

H14 through H19 are hierarchies whose sequence space could not be thoroughly searched within the allotted time. H19 is the same hierarchy as H14 except that, like H3, it was assigned random learning times. H15 and H18 are particularly noteworthy: the former because its result was considerable better than that of the search, and the latter because its result was considerable worse.

To summarize the results in Table 1, for 9 of the 15 distinct non-tree hierarchies GENERATOR produced a sequence as good as the best found by a depth-first or random search of the sequence space running for up to 12 hours of CPU time. For 5 of the remaining 6 hierarchies, SHIFTER was able to improve the sequence produced by GENERATOR to obtain a sequence as good as that found by the search programs. The sequence produced by GENERATOR was under the estimated average memory load for the hierarchy in all cases.

Conclusion

The evidence seems to support the hypothesis that the generalized version of GENERATOR is useful for finding sequences with low memory loads when given learning hierarchies with a single goal objective. With what is presently known, perhaps the best strategy for sequencing objectives when one has allotted a fixed amount of CPU time for the task is to first obtain a sequence from GENERATOR, improve it with SHIFTER, then spend the remaining time randomly searching the sequence space. Whenever a better sequence is found an attempt should be made to improve it with SHIFTER. Although it is slower and cannot examine as many sequences, random search is preferable to an ordered depth-first

traversal because it is not localized to one region of the sequence space, and the sequences it sees will have a wider range of memory loads.

There are several problems which might be included on an agenda of future research in the area. One essential but arduous enterprise will be empirically validating the utility of memory load as a criterion for sequencing instructional objectives. Several studies, involving instructional treatments covering various subject domains, will be required before a convincing conclusion emerges.

When memory load is viewed as a model of forgetting in the instructional process, inherent sources of error become evident. By including relevant information that can be known or estimated at course design time, it is possible that a better model could be developed which is still usable as a tool for instructional planning. Throughout this paper, the relation between instructional objectives, and the surface manifestations of these objectives which are presented to the learner, has been assumed as one-to-one. However, there is no essential incompatibility between the approach followed here and systems which are concerned with deciding the appropriate number of objectives to be covered by the next instructional frame or problem (*e.g.*, Smallwood, 1962; Westcourt, Beard and Gould, 1977).

There is room for more work on algorithms for finding minimum memory load sequences. A major disadvantage of GENERATOR is that it cannot plan the remainder of a partially completed sequence. An algorithm capable of finding a low memory load completion of a partial sequence could be used at course delivery time to fit the instructional plan to the current state of the student model.

Notes

1. See Barr and Feigenbaum (1981, p. 64) for an introduction to A*.

2. Even and Shiloah (1975) proved that this problem, the optimal arrangement of nodes in an acyclic digraph, is NP-Complete. NP-Complete problems are those that have no known algorithm which will always provide a solution in polynomial time, and are equivalent to other NP-Complete problems in the sense that if an efficient algorithm is ever found for just one NP-Complete problem, efficient algorithms for all the others could be immediately derived (Garey & Johnson, 1979).

References

Airasian, P. and Bart, W. (1975). Validating *a priori* instructional hierarchies. *Journal of Educational Measurement, 12*, 163-173.

Barr, A. and Feigenbaum, E. (1981). *The Handbook of Artificial Intelligence* (Vol. 1). Los Altos, CA: Kaufmann.

Bart, W. and Krus, D. (1973). An ordering theoretic method to determine hierarchies among items. *Educational and Psychological Measurement, 33*, 291-300.

Briggs, L. (1972). *Student's Guide to Handbook of Procedures for the Design of Instruction.* American Institutes for Research.

Briggs, L. and Wager, W. (1981). *Handbook of Procedures for the Design of Instruction.* Englewood Cliffs, N.J.: Educational Technology Publications.

Case, R. and Bereiter, C. (1984). From behaviorism to cognitive behaviorism to cognitive development: steps in the evolution of instructional design. *Instructional Science, 13*, 141-158.

Dayton, C. and Macready, G. (1976). A probabilistic model for validation of behavior hierarchies. *Psychometrica, 41*, 189-204.

Dick, W. and Carey, L. (1978). *The Systematic Design of Instruction.* Glenview, IL.: Scott, Foresman & Company.

Edney, P. (1972). *A Systems Analysis of Training.* London: Pitman.

Even, S. and Shiloah, Y. (1975). *NP-Completeness of Several Arrangement Problems..* (Department of Computer Science, Technical Report #43). Haifa, Israel: Technion Institute.

Gagné, R. (1962). The acquisition of knowledge. *Psychological Review, 69(4)*, 355-365.

Gagné, R. (1965). *The Conditions of Learning.* New York: Holt, Rinehart, & Winston.

Gagné, R. (1968). Learning Hierarchies. *Educational Psychologist, 6(1)*, 1-9.

Gagné, R. and Briggs, L. (1974). *Principles of Instructional Design.* New York: Holt, Rinehart, & Winston.

Gagné, R. and Paradise, N. (1961). Abilities and Learning Sets in Knowledge Acquisition. *Psychological Monographs, 75(14)*, (Whole No. 518).

Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco: Freeman.

Heines, J. and O'Shea, T. (1985). The design of a rule-based CAI tutorial. *International Journal of Man-Machine Studies, 23*, 1-25.

Macready, G. (1975). The structure of domain hierarchies found within a domain referenced testing system. *Educational and Psychological Measurement, 35*, 583-598.

Pask, G., Kallikourdis, D. and Scott, B. (1975). The representation of knowables. *International Journal of Man-Machine Studies, 7*, 15 134.

Posner, J. and Strike, K. (1976). A categorization scheme for principles of sequencing content. *Review of Educational Research, 46(4)*, 665 689.

Reckase, M. and McKinley, R. (1982). *The Validation of Learning Hierarchies* (Research Report ONR 82-2). Iowa City: The American College Testing Program.

Riban, D. (1969). *An investigation of the relationship of Gagné's hierarchical sequence model in mathematics to the learning of high school physics.* Doctoral dissertation, Purdue University (University Microfilms No. 70-8957).

Robinson, D. and Foulds, L. (1980). *Digraphs: Theory and Techniques.* London: Gordon & Breach.

Smallwood, R. (1962). *A Decision Structure for Teaching Machines.* Cambridge, MA: MIT Press.

Stelzer, J. and Kingsley, E. (1975). Axiomatics as a paradigm for structuring subject matter. *Instructional Science, 3*, 383-450.

Tennyson, R. (1981). Use of adaptive information for advisement in learning concepts and rules using computer-assisted instruction. *American Educational Research Journal, 18(4)*, 425-438.

Walter, K. (1965). *Authoring individualized learning modules: A teacher training manual* (Title III, E.S.E.A). Kensington, MD: Project Reflect.

Westcourt, K., Beard, M. and Gould, L. (1977). Knowledge-based adaptive curriculum sequencing for CAI: Application of a network representation. *Proceedings of the Annual Conference of the Association for Computing Machinery*, Seattle, WA, October, 234 240.

Wollmer, R. and Bond, N. (1975). *Evaluation of a Markov-decision model for instructional sequence optimization* (ARPA Order No. 2284). Los Angeles: University of Southern California.