Computer assisted instruction

By F. H. George*

This paper outlines the technique of simulating programmed instruction on the computer. Already there are a number of experimental computer-controlled classrooms, especially in the U.S.A., and the methods outlined here show how it is possible to simulate teaching procedures on the computer.

A good deal of research and development has taken place in recent years, especially in the U.S.A., in the field of Computer Assisted Instruction (C.A.I.).

This is a field for computer applications which is little known so far on this side of the Atlantic (Galanter, 1959) although it is one about which a great deal may be expected to be heard in the next few years.

A variety of teams of computer programmers who are also interested (even primarily interested) in the field of *programmed instruction* have been brought together and developed a number of different approaches to the problem of the "Automatic Classroom," as it is sometimes called. PLATO and SOCRATES at the University of Illinois (Stolurow, 1965) and the Automatic Classrooms at Systems Development Corporation at Santa Monica, California (Coulson and Silberman, 1961, 1962), are among the first known and earliest of such projects. Since then, we have had a whole rash of installations, mostly at Universities and adopting a whole variety of different strategies. This article will attempt to explain some of the work of a closely related kind on the simulation of programming that has been carried out at Bristol (George, 1965). Before doing this, however, let us outline briefly the principles of Programmed Instruction.

Programmed instruction

We can think of programmed instruction as taking one of the following terms.

1. Linear.
2. Branching.
3. Mixed.
4. Adaptive.

The format of a frame (or page) is in each case A (Answer (to previous question)), I (Information) and Q (Question). The problem is to break down information into relatively short and concise steps of a kind that lend themselves to question and answer treatment.

The linear system is illustrated in **Fig. 1** with possible backwards and forward "skips," where each frame is represented by a square and a number. If we use a branching system we use multiple-choice questions, so the viewer has to select the answer to a question from a list of possible answers. This is illustrated in **Fig. 2.**

We can now add sub-routes to make our branching system as complex as we please, as shown in **Fig. 3.**

We now merely have to add Fig. 1 on to either Fig. 2 or Fig. 3 to achieve a "mixed" system. To make our system adaptive we need only to record what relative success has been achieved with respect to different classes of questions, to repeat those most often whose success occurs least often.

In other words, if we have four class exemplars, say, A, B, C and D, then if we find the success ratios (given by ordered pairs of numbers) are, say, as follows: A (13 : 26); B (6 : 10); C (3 : 4); D (2 : 11) then clearly we choose D next, and then as the ordered pair for D becomes other than the minimum value, so we move to A, which is the next smallest, and so on.

Computer simulation of programmed instruction

To simulate the process of instruction we now choose students $X_1, X_2, \ldots, X_n$ who operate with respect to a particular teaching program P, which is composed of questions $Q_1, Q_2, \ldots, Q_m$ with answers $A_1, A_2, \ldots, A_L$.

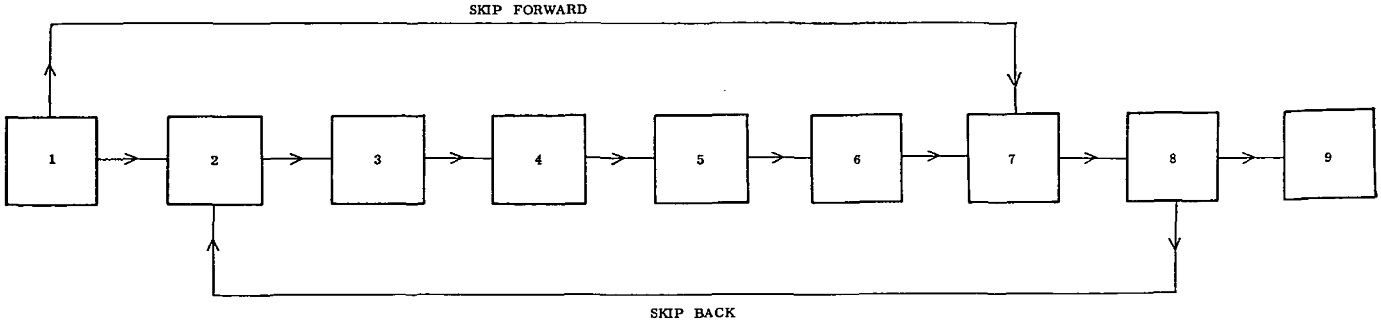

Fig. 1.—Linear system

* *Teaching Programmes Limited, Mark Lane, Bristol,* 1, and *University of Bristol.*

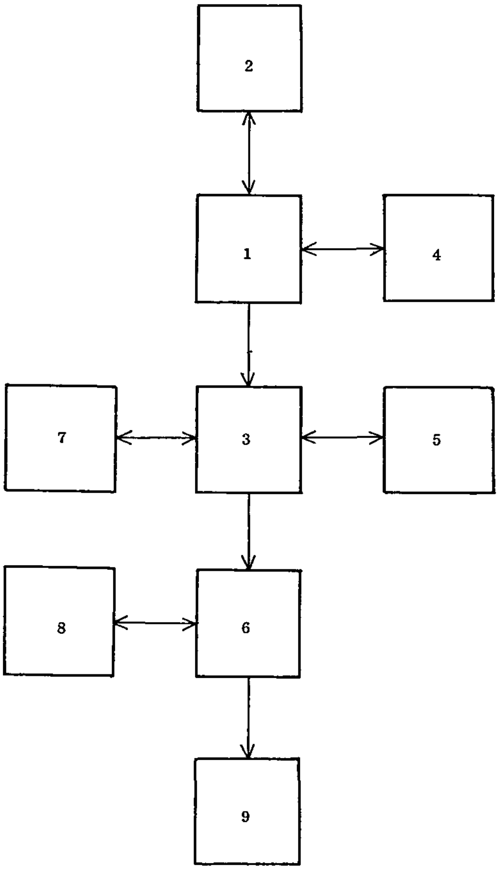

Fig. 2.—Branching system

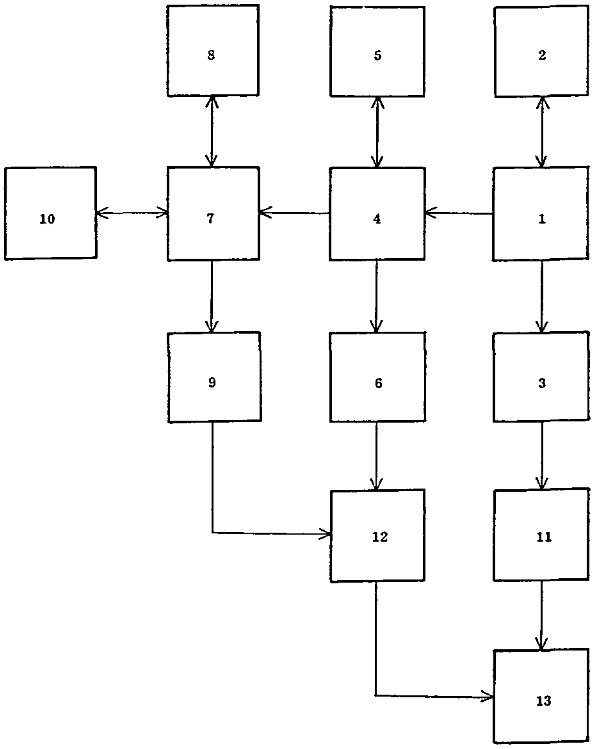

Fig. 3.—Complex branching system

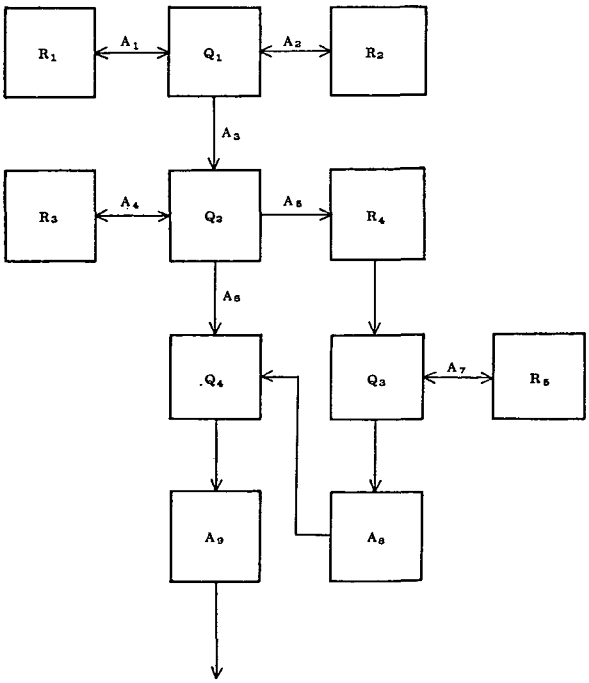

Fig. 4.—Set of questions and answers

If we organize this information in a branching format, we have, say, a set of questions and answers as shown in **Fig. 4**.

Without going into detail we can show with the aid of the flow chart (**Fig. 5**) how a computer may be programmed to process such information

We may allocate stores, say, as follows:

200 (Q_1)	300 (R_1)
201 (Q_2)	301 (R_2)
202 (Q_3)	302 (R_3)
203 (Q_4)	303 (R_4)
	304 (R_5)
320 (A_1)	325 (A_6)
321 (A_2)	326 (A_7)
322 (A_3)	327 (A_8)
323 (A_4)	328 (A_9)
324 (A_5)	

where R_1, R_2, . . ., R_5 are wrong answer frames.

Registers 400, 500, 600 and 700 and subsequent registers may be reserved, say, for the four students X_1, X_2, X_3 and X_4.

The list of possible orders is:

Print Q_1
Is answer in 300?
Yes, R_1
R_1, go to Q_1
Print Q_1
No, is it in 301?
Yes, R_2
R_2, go to Q_1
Print Q_1
No, given A_3—
Go to Q_2
Print Q_2
Is answer in 303?
Yes, R_3
R_3, go to Q_2
Print Q_2
Is it in 304?
Yes, print Q_3
No, given A_6—
Go to Q_4
Print Q_4
Q_3, is answer in 306?
Yes, R_5
R_5, go to Q_3
Print Q_3
No, go to A_8
A_8, go to Q_4
Print Q_4
Go to A_9

Now each student goes through this network, and has recorded as, say, in the case of X_1:

400.	200.	322.	Correct
401.	201.	303.	Incorrect
401.	201.	325.	Correct

and so on. We can easily code correct and incorrect answers and arrive at a score for each student. We can, of course, also keep a record of his progress through the program.

We shall not discuss linear methods or mixed methods since it is obvious that they are simply special cases of the above.

Adaptive programming requires no more than a record of success with orders to repeat the least successful answers and information; this too can be clearly programmed easily in the light of what has already been said.

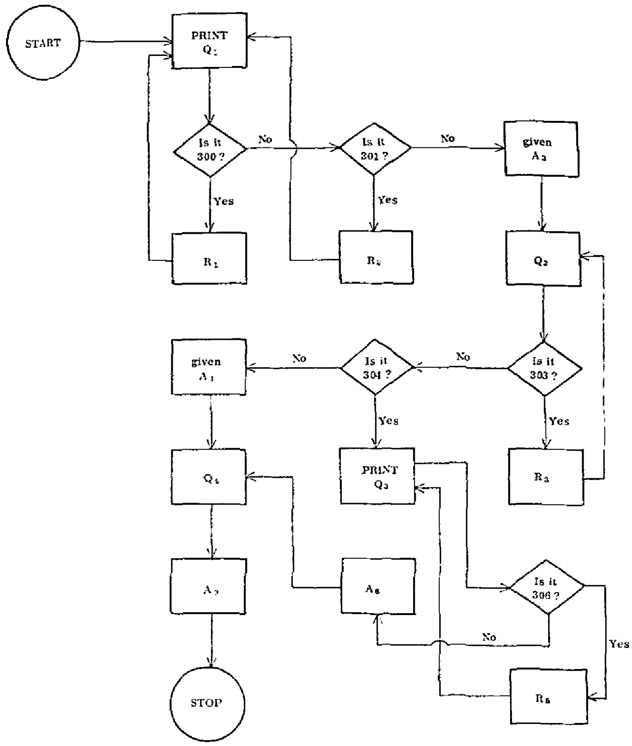

Fig. 5.—Computer flow chart

Finally, it is obvious that we can carry out in the computer any desired statistical analysis of results. There remains, of course, the question of how many different students can carry out how many different learning programs at any one time. The answer to this is, among other things, a function of the size of store.

There are, of course, many other matters that should also be considered. One that occurs immediately as a possibility is that of an open-ended question and a (written) "constructed answer." This can be catered for by storing all (or almost all) possible answers and comparing the given answers with those stored. Unfortunately there may, except in the case of the simplest questions, be far too many ways of phrasing answers, quite apart from the possibility of encountering spelling mistakes. It is these factors that currently make such techniques difficult to program usefully and economically. Such techniques wait on the development of decision-making computer programs, when the much-needed greater flexibility can be more easily achieved.

References

COULSON, J., and SILBERMAN, H. F. (1961). "Automated Teaching and Individual Differences," *Audio-Visual Com. Rev.*, Vol. 9.

GALANTER, E. H. (Ed.) (1955). *Automatic Teaching: The State of the Art*, New York: Wiley, pp. 63–68.

GEORGE, F. H. (1965). *Programming Teaching Machines*. Memorandum 005/P/G. Teaching Programmes Ltd., Bristol 1.

SILBERMAN, H. F., and COULSON, J. (1962). "Automated Teaching." In *Computer Applications in the Behavioral Sciences* (Ed. Harold Borko), New York Prentice Hall, pp. 308–335.

STOLUROW, L. M. (1965). *Proposal for Continuation of Research relating to the TRL System for Organizing Content to Review and Teach Educational Subjects (SOCRATES): Pre-programmed Self Instruction (PSI), and Self Programmed Industrialized Education (SPIE)*, Entelek Inc., Newburypost, Mass.