EXPERIMENTAL
STUDY OF
POLITICS

July, 1975
Volume IV, Number 2

CONTENTS

Computer Programs as a Means of Efficiency and Control in Cross-Cultural Experimental Games

Jonathan Pool
Bernard Grofman

State University of New York at Stony Brook

ABSTRACT

We show computer programs to be useful in experimental games as (1) simulated subjects, (2) game organizers which can substitute in important ways for the experimenter himself and can perform certain selection and control functions better than human experimenters, and (3) tools for data analysis. Our examples are drawn from experimental games tested in the U.S. and West Germany. We discuss both the advantages and problems of cross-cultural computer-mediated experiments.

THE ROLE OF THE COMPUTER PROGRAM

The computer program can play up to three roles in an experimental game.*

---

*We shall not attempt here to justify the use of experimentation in political science or to deal with questions of the generalizability of experimental results to "real-world" situations, except insofar as it is necessary to do so in discussing the special

1. The program can act as a player in the game. In this role it can be either an overt or a covert player; i.e. the human subjects playing the game can either know or not know that one or more of the other players is a computer.

2. The program can manage the game. It can give the instructions, teach the players how to play, check that they have mastered the rules, give them whatever information the rules allow them to have, call for their moves, permit and channel whatever inter-player interaction the game allows, enforce all time limits, enforce the other rules of the game, inform the players periodically as to the outcomes, and keep a running account of points or other units of value (e.g. money) won or lost by each player. Moreover, randomization and branching can be incorporated into programs, allowing the experimenter to implement extremely complex experimental designs.

3. Finally, the computer can be used as a tool for storing and analyzing the results of the game or of

---

problems and special opportunities of computer-mediated experiments. For a useful discussion of these points, see Powell, 1973. We shall also not deal in this article with the computer as a teaching tool. On that topic see Pool, forthcoming.

several games together. The same terminals which are used to enter the game program into the computer, and which are used by subjects to play the game, can also be used to enter analysis programs into the computer for the processing of game-generated data, and to print out the results of the analyses. There is in principle virtually no limit to the kinds of game data that can be stored as the game goes on: who sends what messages to whom, who makes what moves, the time at which each player action takes place, attempts by players to violate game rules, and the sequence of winnings and losses are examples. Records can even be kept of the players' performances during the introductory exercises, so that, for example, speed of learning the rules can be used as a variable in analyzing the game results.

TWO EXAMPLES OF COMPUTER-BASED EXPERIMENTAL GAMES

To make our discussion more concrete, we shall describe two games currently being developed by the principal author at computer installations in three countries.* At present, some of the features described

---

*I.e. the Computer-Assisted Instruction Laboratory of the State University of New York at Stony Brook, the Computer-Assisted Instruction Laboratory of Stiftung

below are operational at only one or two of the three locations.

The first game is a generalized, two-person, 2-by-2 game. This is a game in which each player on each round makes a move, without knowing what move the other player is making on the same round. A move consists of a choice between two alternatives, named "1" and "2". The amount that a player wins or loses is a function, constant across rounds, of the combination of the two players' moves on a given round. Naturally, there are four possible combinations.

Within this general paradigm, the program allows the experimenter to modify the details of the game in several ways. First of all, he can choose which of several titles the game will have, since different titles may induce different behaviors. For example, "STRATEGY" suggests the subject will have more control over his winnings than does "JACKPOT." At present, the subject always plays against the computer, but the program allows the opponent to be truthfully described as the computer or deceptively described as a person sitting

---

Rehabilitation, Heidelberg, Federal Republic of Germany, and the Centre de traitement de l'information, Université Laval, Québec, Canada.

at some other terminal. The other player can be described cooperatively, neutrally, or competitively. The experimenter decides what the payoff schedule will be, i.e. how much is won or lost for each combination of moves on a round. The payoff schedule (and the payoffs) shown to the subject can include only his own payoffs, or both his and the other player's. The cells of the payoff matrix can be expressed either in points or in amounts of the local currency (cents or pfennigs). The number of rounds can be fixed by the experimenter, or the subject can be told that the game will end whenever he or the other player wants to stop; in this case, the computer is programmed to have a steadily increasing probability of deciding to stop. If the experimenter wishes to impose a time limit on moves by subjects, he can do so, selecting any number of seconds. As a modification of the basic paradigm, the subject's task can include not only the making of moves, but also the prediction of the other player's moves; or the making of moves can be eliminated from the task, leaving a task that consists simply of predicting what moves the other player will make. In any of these cases, the payoff schedule can be defined as a function of predictions as

well as of moves.

Since the subject's opponent is a computer, its strategy is programmed in advance. This does not mean that the computer's moves are determined in advance, because in general a programmed strategy will define moves as a function, among other things, of the previous moves of the subject. Several standard strategies are offered to the experimenter. If he wishes to program his own, he can separately assign four probabilities to the computer's moving "1": one probability for each of the combinations of moves that might have occurred on the previous round. In technical terms, this means the experimenter can program the computer to implement any class 1 (homogeneous Markov) decision rule. In addition to determining the computer's actual strategy, the experimenter can also choose how much information about this strategy the subject will be given.

As the game proceeds, the subject learns the outcome of each round a second or so after he makes his move. In addition, at intervals the computer can display to him the history of the game: his moves and the computer's strung out next to each other, plus summary information about what has been won or lost. The

experimenter specifies the numbers of the rounds on which he wants the subject to see this game history, which can be important for the subject's discovery of the computer's strategy. If the experimenter wants to know how the subject perceived his own strategy and that of the computer, an option may be invoked which at the end of the game automatically prints out a questionnaire for the subject to complete.

Additional options will be built into this game package as work on its development proceeds. The options described above, however, represent some of the more important theoretical variables in the experimental study of decision-making in 2-by-2 games. The most common game designs, such as the Prisoner's Dilemma game, can be reproduced by the appropriate selection of options with this program.

Let us turn to our second example. This is a small-group coalition game, in which the computer makes no decisions as a player, but only manages. Any number from 3 to 20 can play, but the game seems to work best with 5 to 8 players. There are currently two titles and scenarios the experimenter can choose from. In the first, called "Death at Sea," players are told they are

passengers in a sinking lifeboat which cannot hold them all, and that if any are to be saved they must decide who will be thrown overboard. The second, called "Slump," is similar, except that subjects play the roles of co-workers who must decide which of them remain on the job and which are dismissed by their financially devastated employer. In either case, the players, who are identified only by number and each of whom knows only his own number, vote in secret on the membership of the winning coalition. At the end of each round, the computer tells them whether they succeeded in reaching the required consensus, and, if not, how much more time they have before the boat--or the company--sinks.

If the satisfaction of being a winner is deemed inadequately motivating, the experimenter may provide for a monetary reward to the winners, information about which is then included in the game instructions. Several other basic parameters are also subject to experimenter choice. He sets the largest and smallest sizes a winning coalition may have, and the number of players that must vote identically in order to designate a winning coalition of a given size. He determines the

(real) time at which the game will end with no winners if a decision has not been reached by then. He can also vary the amount of post-round information about the outcome that subjects get. At one extreme, they are merely told whether there was an agreement; and at the other, they are given a complete breakdown of how each player voted.

A set of additional options regulates communication among players. Under the simplest option there is no communication at all, except indirectly via the post-round feedback just described. If the experimenter wants to allow communication, he can choose to allow (1) suggestions as to how to vote, (2) promises of side payments, or both kinds of messages. If so, each round includes a period for the exchange of messages prior to the casting of votes. The experimenter can impose on the players any directed probabilistic communication network. In other words, he can determine who can communicate with whom, and S_i may be able to send messages to S_j without S_j being able to send to S_i; furthermore, with each channel in each direction is associated a probability of message arrival, which can be set anywhere between 0 and 1. Subjects can be told

their message arrival probabilities in advance, or left to learn them by trial and error. Absolute limits may also be placed on the number of messages a player can send per round.

In addition to the general deadline for reaching a group decision, the experimenter may also set deadlines for individual actions throughout the game: the maximum number of seconds available for sending messages in a round, and for casting each vote. Although players are made anonymous to each other by the program, which randomizes them across numbers and conditions each game, descriptive labels may also be introduced into the experimental design, so that one player is described to another as, for example, a "speaker of Russian" or "a neighbor of co-worker no. 5."

These options in the small-group game can be exercised differently for different players wherever this is sensible. Thus one player can be told just how everyone voted each round, while another is just told whether an agreement was reached. One player might be allowed to send three messages of any type per round, while another may send only one message which must be a promise of a side payment, and still another may not

send any messages at all.*

In both the two-person and the small-group game, provision is made for textual display in an unlimited number of languages. Parallel texts are currently available or under development in English, French, and German. In the case of the small-group game, this means that a linguistically heterogeneous group can play the game with as much ease as a monolingual group. Messages typed in by subjects in accordance with the specified format and syntax requirements are, in effect, translated by the computer into the language of the recipient. Both game packages also provide, of course, for optional omission of the instructional routine for experienced players, and for comprehensive recording of all player actions and interactions. A set of analysis programs is being continually expanded to access and process data generated by these games.

An example will illustrate how such data can be analyzed. At the Heidelberg installation, 41 subjects played a version of the 2-by-2 game for 300 rounds

---

*For a computer-managed game similar to this small-group game in some respects, see Rapoport and Kahan, 1974. On p. 10, they summarize some advantages of computer-managed games, most of which (and others) we discuss below.

apiece, in an experiment designed to determine the effect of the computer's playing strategy on the subjects' behavior. The payoff matrix was constructed in such a way that, the more frequently the computer copied the subject's previous move, the more advantageous it would be for the subject to adopt a strategy of moving "1" rather than "2". One of the hypotheses to be tested was the following: If on any round we count backwards to determine the number of rounds-in-a-row the computer has copied the subject's prior move since the last failure to do so, we shall find that the probability of the subject's moving "1" varies directly with this number. An analysis program was written to scan the results of all 12,300 rounds, classify each round by the length of the immediately prior unbroken series of copying moves by the computer, and tabulate the number of "1" and "2" moves of subjects by series length. The resulting matrix was fed into a plotting program to yield a visual display of the relationship between series length and likelihood of moving "1". Figure 1 shows the result. Since the horizontal variable, the computer's copying behavior, was a true independent variable, being unaffected by

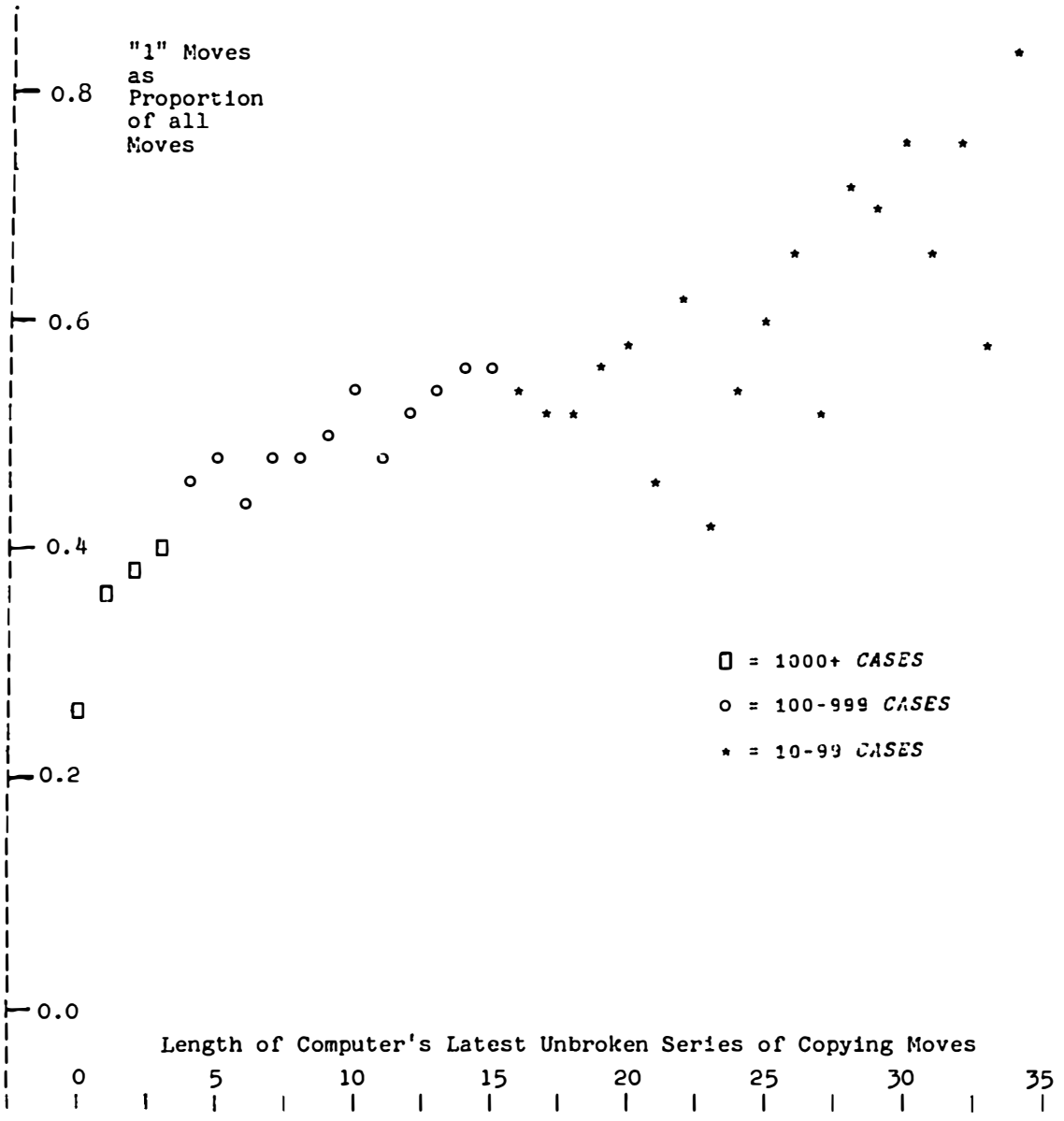

Figure 1--The Effect of Computer's Copying of Player's Behavior on Player's Behavior

any subject behavior, the hypothesized relationship, seen to be confirmed in Figure 1, is a causal one. By minor modifications of a given analysis program, hypotheses may be tested which control for additional variables. One scan is performed and one table or plot is generated for each value of the control variable in question.

THE CONTRIBUTION OF COMPUTER PROGRAMS TO EFFICIENCY

Programs like the ones described above must either be more efficient or better for achieving experimental control, if their use instead of the traditional apparatus of social-psychological experiments is to be justified. Once the scholar has worked out an experimental game conceptually, he must evaluate the alternative ways in which it might be operationalized. Whether this evaluation will indicate that the game should be computer-based depends on many particular facts of each case that we cannot possibly anticipate here. What we can do is present some generally relevant considerations that arise in deciding whether a computer program ought to be used in playing, managing, and/or analyzing a planned game. Our own

experiences will be the basis of what follows. We feel these experiences are relevant because the games described above typify two kinds of experimental designs commonly desired in political science applications, and because the principal author's prior ignorance of programming has given him the kind of experience most political scientists would have if they attempted to program the games they wish to experiment with.

First let us look at the question of efficiency, and then at the question of control. Efficiency considerations can be grouped into three categories: game preparation, game execution, and result analysis.

For most users, computer programming probably will make game preparation more costly than conventional means, such as booths, tables, decision-making forms, message slots, and printed instructions. Much depends, of course, on whether a suitable time-sharing computer system, or a suitable small-group laboratory, is already available. The former cannot be jerry-built, while the latter can. But even if a time-sharing system is accessible, the time involved in learning a programming language, programming the game, and

debugging it can easily reach the order of 200 to 400 man-hours for games of the complexity of those described above.

There are two conditions, however, which would greatly reduce this cost. One is the need for an exceedingly simple game, such as just one version of the 2-by-2 game would be if the instructions and orientation were all mimeographed handouts rather than parts of the program. Such a game could be finished by a hired programmer in a few hours. The other time-saving condition would be the need for a game that has already been programmed by someone else. It is, in fact, with this in mind that the two games described above have so many built-in options and are acquiring more. They are not actually individual games, but rather game packages, which will allow other researchers, including research students, to design and execute a game experiment in a fraction of the time that would otherwise be required. Game packages, like data-analysis packages, achieve this economy at the price of severe limits on the range of alternatives open to the user.

Where the investment in game preparation by computer programming is more likely to pay off is at

the second stage: game execution. The first thing to consider is how efficiently subjects learn the game. A computer-programmed game typically is more difficult to learn, because the subject must learn how to use the computer terminal as well as the rules of the game itself. Inexperienced subjects can have considerable difficulty getting used to terminals, depending partly on the type of terminal. Acute "terminal fright" seems to persist, however, in only about one out of 200 subjects, in our experience. So the relative disadvantage of a computer-based game in this respect increases directly with the amount of subject turnover required by the experimental design. Where each subject spends a long time in the experiment and plays the game(s) repeatedly, the disadvantage is small.

But regardless of turnover, a well programmed game has a sequence of questions and drills which ensure that each subject knows those things (and how to do those things) which the experimenter wants him to know before the program allows the subject to begin the game itself. A large number of subjects, limited only by the number of terminals and the capacity of the computer, can be taught the game in an individualized

fashion, with mistake-conditioned feedback, all at once--something that could not otherwise be done without one experimental assistant per subject. Furthermore, a thoroughly programmed game does not need to be pre-learned as well as a conventional game, since attempts to violate the rules are caught immediately by the program, which reminds the subject of the appropriate rule. Programmed games can thus be learned more gradually and experientially (by trial and error) than conventional games.

A computer-programmed game is likely to save the experimenter's own time during execution to an extent that drowns out the cost of the computer time itself. One reason is that interactive computer time is cheap: a 300-round session of the 2-by-2 game, for example, costs about 10 cents per subject. Besides this, a given subject or group of subjects can generally play considerably faster when interim calculations are performed by computer. And one experimenter can supervise several experiments at once when they are programmed. In Heidelberg, for example, the principal author supervised up to 13 subjects at a time playing the 2-by-2 game against the computer, and two groups

playing the small-group game simultaneously. He would have needed the full-time service of 12 assistants to even approximate this performance without the computer.

Furthermore, there are some games that simply could not be executed without being computer-programmed. Where the subject is playing against an opponent whose moves must be a highly complex function of the subject's previous behavior, a computer is essential. Where several players are taking part in a game in which what happens to each one depends on the precise timing of his and/or others' moves, a computer is likewise indispensable. This is also the case where information must be intermittently provided to the players, if the information they get depends in a complex way on what they do. What we have said about complex functions also holds true for probabilistic ones, where the changing conditions to which players are subjected depend partly on chance. Thus computer programming makes the execution of many games more efficient, and makes the execution of some games possible for the first time.

The contribution of programming to efficiency is still clearer at the third stage: data analysis.

Making provision in the game program for a sensible kind of result storage requires extra programming time, but repays itself manyfold if the experiment is carried out more than a handful of times and there is a large volume of data to be analyzed. Since the programming language will have already been learned, it can with little additional investment be used to retrieve the stored data and subject them to any desired kind of analysis. The analysis programs can be stored, too, for use on data from later experiments. It is at this stage where the selection of a powerful programming language may become important, even if a powerful language was not required for the execution of the game itself.

With this in mind, the increasingly popular and highly powerful language APL was selected for the games described above. A general program in APL to print out some desired 2-by-2 game information about each subject, together with the experimental condition that the subject was in, took up 5 lines. The specific information that this program would print out for all subjects was determined by a separate program, which often took only a single line. The program that

computed how many times the subject had moved "1" was:

```
[1]  +/0=⍺[1;]
```

The program that took the subject's 300 moves and the computer's 300 moves and computed how much the subject won in toto, on the basis of the payoff schedule, was:

```
[1]  +/10+ 3 8 -.×1+⍺
```

And the program that computed how many times the subject changed his move from one round to the next was:

```
[1]  +/1=+/[1] 2 301 ⍴⍵, 2 2 ,⍵←⍺[1;]
```

The fact that such programs can be written and executed at a terminal in minutes leads to a more efficient style of data analysis than would be practiced with a batch-processing package such as SPSS or DATA-TEXT. One analysis can be performed at a time, and the results of all prior analyses can be the basis for the hunches and hypotheses guiding the next analysis. Thus even if the preparation and execution of a particular game are no more efficient when programmed, it might well be worth basing the game on the computer for the mere reason that the results can be automatically stored and

immediately analyzed.

The efficiency consideration depends most of all on the expected amount of repetition. Computer-programmed games will typically require more of an initial investment, but will be more efficient if repeated often. In the cross-cultural context the question arises as to whether different versions of a program for different cultures constitute repetition. To the extent that the operations of the game are constant, and it is only the language of textual display and of subject responses that changes, programs can be written with replaceable textual content and textual response processing algorithms. It is possible to produce a different-language version of the same game with far less effort than was required to write the original program. The problem of transferring programs across cultures is not only one of efficiency, however. This brings us to the issue of control.

THE CONTRIBUTION OF COMPUTER PROGRAMS TO CONTROL

A programmed game has an additional advantage beyond its efficiency. The computer helps the experimenter treat all subjects equally (or at least

all those who are supposed to be treated equally). Experimenter bias is reduced, first because the experimenter personally plays a less prominent role in the execution of the experiment, thus giving his changes of mood, etc., less opportunity to have an impact, and secondly because a single experimenter is able to run the experiment, eliminating the possibility that different experimenters or laboratory assistants would affect subjects differently. Where subjects are not supposed to be treated alike, the computer can select subjects randomly for different conditions, and can in addition keep the experiment "blind" by leaving the experimenter himself ignorant of who is in which condition until the end. Where subjects should be visually and aurally isolated from each other, the computer can assure complete isolation, while still allowing any degree of mediated contact that the experiment calls for. The computer guarantees anonymity where desired, since individual differences in handwriting are eliminated in message exchanges. The replication of experiments can be performed with more confidence when a large part of the experimental environment is a computer program that can be received

by mail from the originating institution with the knowledge that it will function identically at a different installation having the same computer system.

Whatever their contribution to control, computer programs can never solve the entire control problem. For one thing, they introduce into any experiment an element of technical complexity that may change people's behavior. Computer-programmed experiments raise the question, "Would the same subjects have behaved differently if the game had been played with pencils and paper, or face to face?" This question is especially pertinent where there are known subcultural or interpersonal differences among subjects which interact systematically with technical skill. If, for example, there is an inverse association between body weight and typing speed, then the same person who is most successful in a face-to-face game might be least successful in a computer-based game.

An additional problem arises when the experimenter envisages the cross-cultural application of his game. The computer program can contribute to the structural identity of two experiments conducted on subjects of different cultures. The goal, however, is probably

equivalence rather than identity, in Przeworski and Teune's (1970) terms. Here computer-programmed and traditionally organized games face the same difficulty. The translation of textual components of a game is no easier in computer-programmed games, and it cannot be assumed that the non-textual structure of the game has the same meaning for, or effect on, subjects of different cultures. For example, as strange as the notion of voting on whom to dump out of a sinking lifeboat may seem to Americans, it is apparently even stranger to Germans, according to reactions received from colleagues in Germany. This problem is partly solved, but partly complicated, by the fact that "wählen" in German can mean both "vote" and "choose." In reference to the 2-by-2 game, discussions of the computer's strategy are affected by the apparent fact that the word "Strategie" in German has a more military implication than "strategy" in English, while "Taktik" deals with less comprehensive schemes than are connoted by "strategy." It was found that not just one translation of a text from English to German was necessary, but rather many successive consultations with German informants. A text that one informant, after

corrections, had certified as perfect colloquial German was invariably criticized by the next informant for its Anglicisms.

Even where cross-cultural equivalence cannot be claimed, at least structural identity may be of use, especially when one objective is to determine intercultural differences. To the extent that computer-programmed games are identical across cultures, they will help us discover how members of different cultures respond to identical stimuli. This knowledge, in turn, will enable us to make culturally specific modifications in our games to bring them closer to equivalence.

Cross-cultural applications need not take the form of the same experiment being performed in two different countries. Where members of two or more cultural groups are found in close proximity, they can be successively brought into the laboratory to permit cross-cultural replication. Moreover, they can also be brought in simultaneously to confront each other in the context of a single game. As was illustrated above, when they speak different languages each subject can play a version of the game in his own language. And in

general, different scenarios, different rule-learning routines, or any other modifications introduced to achieve cross-cultural equivalence can be implemented selectively in a culturally heterogeneous group. Of course, the use of these possibilities can lead to an infinity of experimental designs, including many which violate common premises of the "real" world. Subjects who do not share a language, for example, may be enabled to communicate, while subjects who speak the same language may be unable to communicate. This is all to the good. While the advantage of the experimental method is often said to be its ability to limit the variation that naturally occurs in the world, that is only half the story. An equally important advantage is its ability to create new combinations of values of independent variables that are never found naturally (cf. Kelley, 1968:67). Computer programs can be useful in achieving both of these kinds of control, restrictive and creative.

FURTHER REMARKS AND CONCLUSIONS

The generalizations offered above suggest that computer-programmed games are likely to have considerable

advantages over traditionally executed games for experimental purposes. These advantages will not always outweigh the disadvantages we have discussed, but it seems clear to us that computers are currently used as a base for experimental games in only a small fraction of the instances where it would be optimal to use them. This lag is largely due to the unavailability of suitable facilities. Even as time-sharing systems become more common on college campuses, it cannot be taken for granted that they will be well suited for experimental game applications.

The would-be programmer of games will be interested in having access to a computing system with certain features. Foremost among these is interactive, time-sharing computing with terminals. For small-group experiments, at least one room with enough terminals for all subjects is very handy. The system should support a high-level language like APL. It should be possible to program the computer so that terminals can interact not only with the computer, but also with each other during program execution, through the sharing of files, the sharing of variables, or the executable transmission of messages. The system should also

support a time-out feature; this allows continuous programmed monitoring of a subject's terminal while he is entering input requested by the program, and enforcement of a specified time limit for the completion of this input. The programming of games is made far easier by this time-out feature, since in group games the experimental design normally does not allow one subject to hold up the game indefinitely by delaying his input.

While all these features are gradually becoming more common, intervention by game programmers in the computer-selection process would solidify and hasten the trend. Another trend that has far to go is toward program interchangeability across computer systems. Each system, for example, has its own slightly different version of APL, and they are not all compatible. The potential for shipping a game tape as easily as one can ship a game box is certainly not yet realized.

In spite of these current problems, the programming of a computer to manage, play, and analyze an experimental game has definite attractions. Although it may be more costly to prepare an experiment this way the first time, the subsequent preparation of similar

games or cross-cultural versions of the same game is far less expensive and time-consuming. Efficiency is increased during the execution of the experiment, and especially in the analysis of its results. Some types of experiments are made possible by computer programming that would otherwise be unthinkable. Computer control cannot be equated with experimental control, but programs can be used to reduce experimenter effects and increase replicability.

Beyond these considerations, we would also argue that the experimental design itself often benefits from being programmed. Programming constitutes a precise statement of the design--something many investigators never get around to formulating until after the experiment is over. As our experience testifies, the experimenter may not really know what he wants his game to be until he tries to program it.*

---

*This article is based on research which has received direct and indirect support from the Research Foundation of the State University of New York, the Computer-Assisted Instruction Laboratory of the State University of New York at Stony Brook, the Universität Mannheim, the Stiftung Rehabilitation, the Deutscher Akademischer Austauschdienst, the Council for European Studies, and the Alexander von Humboldt-Stiftung. Current further development of the game packages described above is taking place as part of a research training program supported by a fellowship to the

REFERENCES

Kelley, E.W.
1968 "Techniques of Studying Coalition Formation," MIDWEST JOURNAL OF POLITICAL SCIENCE 12:62-84.

Pool, Jonathan (ed.)
forthcoming COMPUTER-ASSISTED INSTRUCTION IN POLITICAL SCIENCE. Washington: American Political Science Association.

Powell, Charles A.
1973 "Validity in Complex Experimentation," EXPERIMENTAL STUDY OF POLITICS 2(no. 2, March): 61-98.

Przeworski, Adam and Henry Teune
1970 THE LOGIC OF COMPARATIVE SOCIAL INQUIRY. New York: Wiley.

Rapoport, Amnon and James P. Kahan
1974 "Computer Controlled Research on Bargaining and Coalition Formation," presented at the meeting of the Public Choice Society, New Haven, 21 March.

---

principal author by the Social Science Research Council. Additional support for this purpose is being furnished by the Centre international de recherches sur le bilinguisme and the Centre de traitement de l'information, Université Laval, and by McGill University. An earlier version of this paper was delivered at the 1974 Annual Meeting of the American Political Science Association, Chicago, 29 August to 2 September, 1974.