SEER, A SEquence Extrapolating Robot*

D. W. HAGELBARGER†

Summary—**The success of computers in doing routine work formerly done by people suggests that a computer capable of adjusting itself to a changing environment might be desirable. Such a characteristic might be especially valuable to the telephone industry which must service large numbers of people having changing needs and desires. As a step in this direction a relay machine which plays a penny-matching game with human opponents has been built. The machine is described and its behavior against people and other machines discussed.**

Introduction

THERE IS a game played by two school boys in which each of them chooses which side of a coin to expose. One of the boys tries to match the other. If both coins are the same, he wins; otherwise the other boy wins. This is a fair game since each boy wins for two of the four possible combinations. The theory of games says that, if you assume that your opponent is as smart or smarter than you are, a safe strategy is to play randomly with equal probability of heads and tails. This assures your not losing much since you can expect to win on the average half the time and come out even. This is, however, a rather staid approach. If you are smarter than your opponent, you should be able to guess which way he will choose and make your own choice so as to beat him more than half the time. In "The Purloined Letter," Edgar Allan Poe describes a boy who was successful at this game by assuming his opponent's facial expression and observing what he thought with this expression. This paper describes a relay machine built to play this game with human opponents. Rather than playing safe the machine tries to outwit its opponent and thereby win more than half the time. It has achieved a limited amount of success. Out of 9,795 plays against visitors and employees at Bell Telephone Laboratories, it has won 5,218 times and lost 4,577. The odds against the machine's getting this large a lead by chance alone are about 10 billion to one. These figures, however, should not be taken too seriously as there are several effects which bias them one way or the other. Some players use a simple sequence that the machine can beat just to see how fast it "catches on." This tends to make the machine's score higher than it should be. On the other side are people who cheat the machine to see how it behaves when losing. Also people who are consistently beat by the machine refuse to play it very many times, while people who come out even or win will tend to play it a long time. In general, the runs are too short; the machine spends about the first half of each game playing randomly while it is gathering data.

The Game

The front panel of the machine contains:

machine ready lamp
machine play button
plus lamp
minus lamp
opponent's play key
scoring lamps.

The machine plays first so that the player knows it is not cheating. When the machine ready lamp is on, the player announces whether he is going to play plus or minus. (The machine cannot hear.) He then pushes the machine play button and the machine lights either the plus or the minus lamp indicating its play. The player then indicates his choice on the opponent's play key. If both choices are the same, the machine wins; if they are different, the player wins. The machine records the score and is ready to play again.

Response to Simple Periodic Sequences

There are four periodic sequences which the machine can recognize and predict without making any errors once it has caught on. These are:

$$+\,+\,+\,+\,\cdots$$
$$-\,-\,-\,-\,\cdots$$
$$+\,-\,+\,-\,+\,-\,\cdots$$
$$+\,+\,-\,-\,+\,+\,-\,-\,\cdots$$

On most short periodic sequences the machine wins somewhat more than half the time. It tends to recognize a certain part of the sequence each time it occurs even though it cannot remember the entire sequence. Some sequences cause the machine to come out even. I do not know of any periodic sequence which will beat the machine.

Fig. 1 shows a series of learning curves for the machine. The net score is plotted against the play number. The machine was cleared at the beginning of each curve and then the series $++--+--\cdots$ was played against the machine until inspection of the internal memory of the machine showed that it had recognized the sequence and would make no more errors. This point is marked with a *V* on each curve. The phase of the sequence was then shifted by inserting an extra symbol and the sequence continued until inspection of the memory showed that it has assimilated the phase change and would make no more errors. This point is again marked with a *V*. On the average, it takes the machine about 21 plays to recognize this sequence and 7 plays to assimilate the phase change.

Raison d'Être

Why build such a machine? The game which it plays is really not a very exciting one and probably has little

* Original manuscript received by the PGEC, October 28, 1955.
† Bell Telephone Labs., Inc., Murray Hill, N. J.

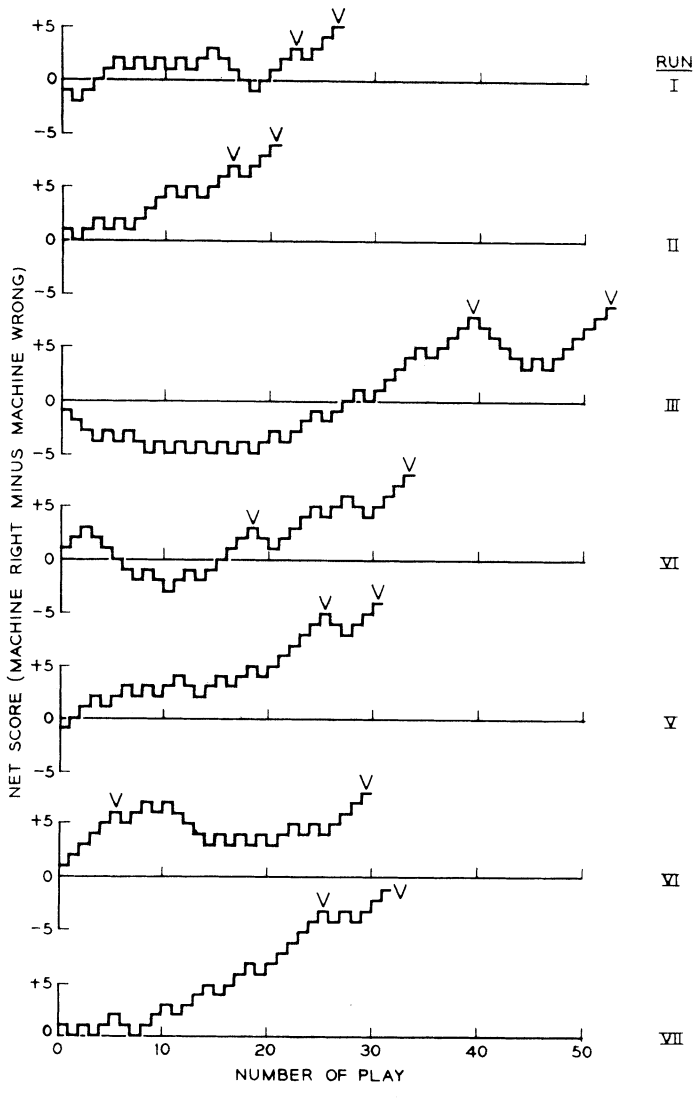

Fig. 1—Learning curves for $++--+--\cdots$

$++++\cdots$

$----\cdots$

$+-+-\cdots$

and having learned them quickly shift among them with few errors. It happens that these three sequences will all fit into the machine's memory with little conflict. Fig. 2 shows this experiment. As before, the symbol V indicates that examination of the internal memory reveals that the machine has "learned" the sequence and will predict it indefinitely without error, if desired. The machine was cleared and started with $+++\cdots$ which it learned at A. It took 4 errors and 9 plays for it to switch to $---\cdots$ at B. It was able to return to $+++\cdots$ at C with 2 errors and 7 plays. It now had a run of bad luck playing randomly and losing 7 plays in a row, but by D it had learned $+-+-\cdots$ in 27 plays with 16 errors. It returned to $+++\cdots$ at E with 1 error and 4 plays, back to $+-+-\cdots$ at F with 2 errors and 5 plays, and so on. From D on it shifted among the three sequences with 1 or 2 errors and 4 to 5 plays per shift. Fig. 3 repeats the same experiment which is similar except bad luck after C is missing.

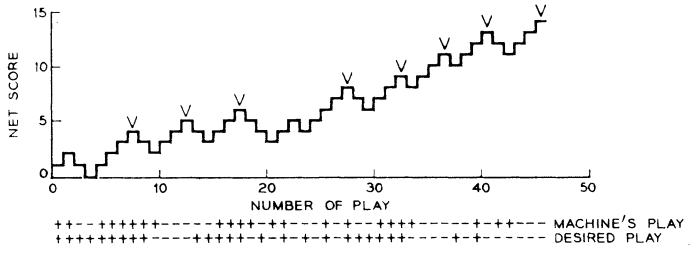

Fig. 3—Experiment of Fig. 2 repeated.

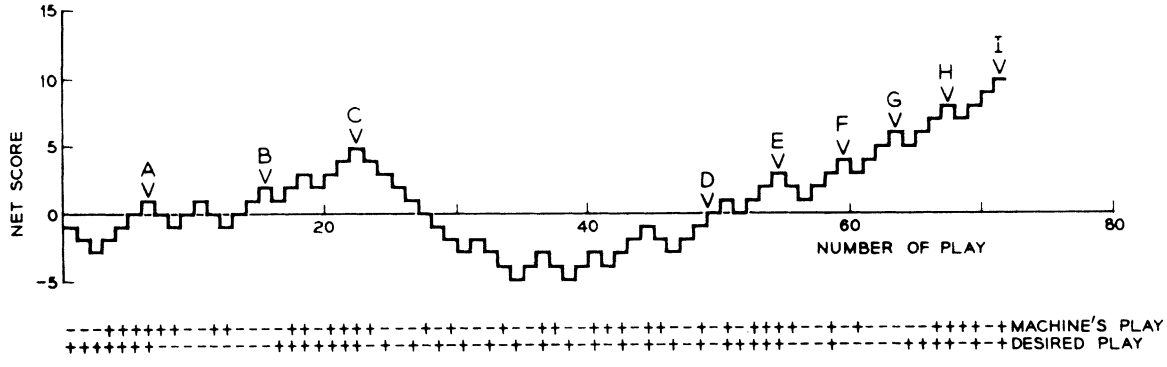

Fig. 2—Learning curve for three sequences.

or no commercial value. By relabeling the panel of the machine we can change it from an adversary to a servant which is trying to please the operator. In fact, the machine was first conceived of as an environment-adapting machine and was suggested by a proposal of J. R. Pierce for a music-composing machine. This environment-adapting behavior will be illustrated by having the machine simultaneously learn three sequences:

As machines get more and more complicated, it seems likely that not only routine matters but some things which we now call thinking[1] will be done by machine. For instance, in a telephone exchange there always ap-

[1] Some people prefer to stop calling it "thinking" as soon as a machine does it. Most of us agree that it is a higher form of intelligence for Newton (or Leibnitz) to invent The Calculus than it is for a school boy to learn it, but is the school boy thinking?

pear to be some jobs for which human operators are required. As telephone systems get more complicated the routine jobs will all be done by apparatus, leaving those requiring judgment and originality. Yet with the increasing use of higher speed devices it is getting harder to connect the person efficiently to the system. A person may easily be the bottleneck in a system which works at the rate of a million pulses per second. It is said that the reason people have been so successful evolutionwise is the ability of people to adjust to a wide range of environments. The ability of a machine to adjust itself to a changing environment will surely be desirable. A new toll alternate routing system is designed according to the measured statistics of the telephone system. If the statistics change the toll system must be redesigned and yet the new system itself may cause the statistics to change. We can imagine an alternate routing system which chooses alternate routes on a basis of how satisfactory previous choices were. Such a system could follow changing statistics.

It is possible, if not probable, that it would be economical to design a telephone central office to measure traffic and adjust itself accordingly. It might observe that most calls from the business district occur during the day and more calls from the residential section during the evening, and connect its apparatus accordingly, yet it would be able to readjust itself if a large fire occurred in the business section during the night.

Perhaps in an extremely complicated situation it might be easier to design a machine which learns to be efficient than to design an efficient machine as such.

Of course we are a long way from anything as sophisticated as this. To give an idea of how much intellectual activity to expect from the out-guesser, consider that a man has 10^{10} neurons, the very dumbest army ant has 200 neurons, and this machine has less than 100 relays.

Strategy[2]

The strategy of the machine is based on two assumptions:

a) The play of people will not be random. They will be influenced by training and emotions so as to produce patterns in their play. For example, some people after winning twice, say, will tend to "stick with their luck." Others will feel they should not "push their luck" and change. In either case, if they are consistent, the machine should catch them.

b) In order to make it hard to beat, the machine should have its output correlated only when it is winning and play randomly when it loses.

To make its play symmetrical in + and − the machine does all its calculating in terms of whether this play should be the *same* as or *different* from the last play.

[2] For a more detailed description, see Appendix I.

What we will call the "state of play" of the machine is determined by three things:

a) whether it won or lost last play
b) whether it won or lost play before last
c) whether it played *same* or *different* last time.

This information is stored in the state of play relays (see Fig. 4). There are 8 states that the machine can have. Each of these has its own memory register.

At the beginning of any play, the computer is connected to the appropriate memory register by the state of play relays. The memory stores two kinds of information:

a) Should the machine play *same* or *different* in this state in order to win?
b) Has the machine been winning in this state?

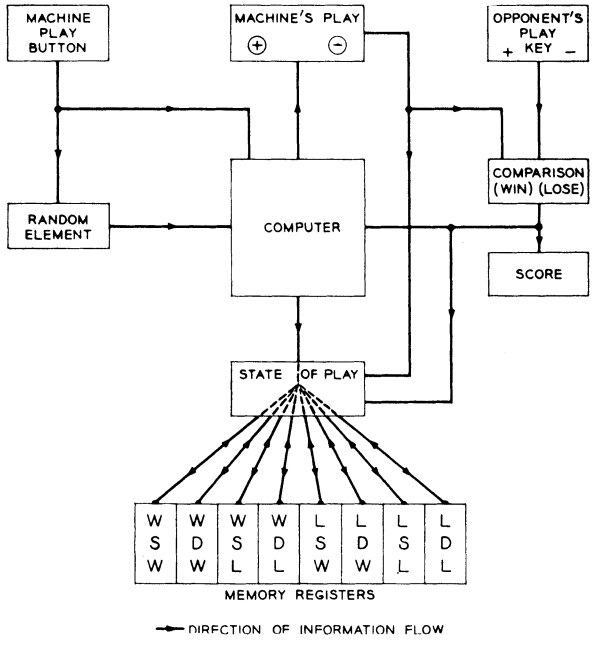

Fig. 4—Block diagram of SEER.

The a) part of the memory is controlled by a reversible counter which starts at zero and can count up to +3 and down to −3. At the end of each play, if the machine should have played *same*, one is added to the counter. If it should have played *different*, one is subtracted. The counter will thus contain the number of times the machine should have played *same* in that state minus the number of times it should have played *different*. The stops at +3 and −3 in effect make the machine forget ancient history. The exact logic of the b) part of the memory is rather complicated when written out in words, but it roughly corresponds to remembering whether the machine has won both, one, or neither of the last two plays in that state.

When the player pushes the machine play button two actions are started:

A random number 1, 2, 3, or 4 is selected by the random number generator (this is determined by the

position of a commutator at the instant the button is pressed).

The computer determines the machine's play from the following rules:

If the machine has lost the last two times in the present state, it plays randomly with equal odds on *same* and *different*.

If the machine has won one of the last two times in this state, it has three-to-one odds that it will follow the instruction in the a) part of the state memory.

If the machine has won both of the last two times in this state, the machine must follow the instruction in the a) part of the state memory.

When the player indicates his play on the opponent play key, the machine calculates whether it won or lost and scores it. The computer makes the appropriate changes in the two parts of the state memory. Then the state of play circuit changes to the new state of play and the computer reads out the memory of this new state and is ready to play again.

Miscellany

There are strategies[3] which will beat the machine. It can be shown that the best possible strategy wins 60–40 over the machine.

C. E. Shannon has built a machine using about half as many relays which follows a simplified version of the same strategy. The simplifications make his machine more agile and quicker to detect a trend at the expense of security. The best opponent's strategy will beat the smaller machine 75–25 as compared to 60–40 for the larger machine. After much discussion an umpire machine was built which connected the two machines, and they were allowed to play several thousand games. The agility of the small machine triumphed, and it beat the larger one about 55–45.

Appendix I

Circuit Logic and Operation

(To be read one word at a time with one finger on the text and one finger on the circuit diagram, Fig. 5.)

The operation of the circuit will be described by following through one play in detail. At the start of a play stepping switch 3 is in position 20. Since stepping switch 3 controls the sequencing, repeated reference will be made to its position. Relays will be referred to by letter only without saying "relay" every time.

Position 20—Closing the machine play pushbutton operates S. S locks up. R_1 and R_2 lock up in one of their four possible states, according to the position of the commutator when S operates. R_0 also locks up and converts the commutator to a pulse source for stepping switch 3.

Positions 21–25, 1, 2—Empty.

Position 3—N operates and connects WZ_4 to the input of the tree on R_1 and R_2. If a minus comes through the tree, WZ_4 changes state and DS locks up. (WZ_4 is the machine's play; a minus pulse makes it play *different*. DS remembers if WZ_4 played *different*.)

Position 4—N releases, M operates and locks up, lighting either the + or − lamp, removing the pulses from stepping switch 3, and connecting minus to the opponent's play key circuit. The machine remains in this state until the opponent's play key is moved to either + or − causing W or L to operate and lock up. Either the "machine wins" or the "player wins" lamp is lighted and a count is put in either the W or L message register. The opponent's play key is removed from the circuit and pulses from the commutator are again applied to stepping switch 3.

Position 5—A count is put into one of the non-random W or L message registers if either PW_1 or PW_2 (or both) is operated.

Position 6—C_0 operates and locks up. This disconnects $WZ_{1,2,3}$ and the PW circuit from the memory register corresponding to the present "state of play" and clears this memory register. It also operates and holds D if DS and W agree. (Play *different* and win or play *same* and lose.)

Position 7—The previous wins (PW_0, PW_1, PW_2) circuit, which controls the odds in using the random element, is pulsed. If neither PW_1 nor PW_2 is operated the odds are 2:2 and the play is random. If only one of them is operated the odds are 3:1 in favor of following the play corresponding to the sign of the number in $WZ_{1,2,3}$. If both PW_1 and PW_2 are operated the machine must play according to $WZ_{1,2,3}$. Whenever the play agrees with the instructions in $WZ_{1,2,3}$ and the machine wins, the odds are increased. Whenever the play agrees with the instructions in $WZ_{1,2,3}$ and the machine loses, the odds are decreased. For other cases the machine is designed pessimistically (to make it harder to beat). It never increases the odds unless it wins; however, it may leave the odds unchanged. For example, if PW_1 only, say, is operated and $WZ_{1,2,3}$ has a negative number (*different*) in it, the machine has 3:1 odds in favor of playing *different*. If the machine plays *different* the odds are increased to 4:0 if the machine wins and decreased to 2:2 if it loses. If the machine plays *same* the odds are not changed if it wins or loses.

Position 8—A pulse is applied to $WZ_{1,2,3}$ which counts up or down according to whether D is released or operated. The counter has end stops at +3 and −3.

[3] Appendix II.

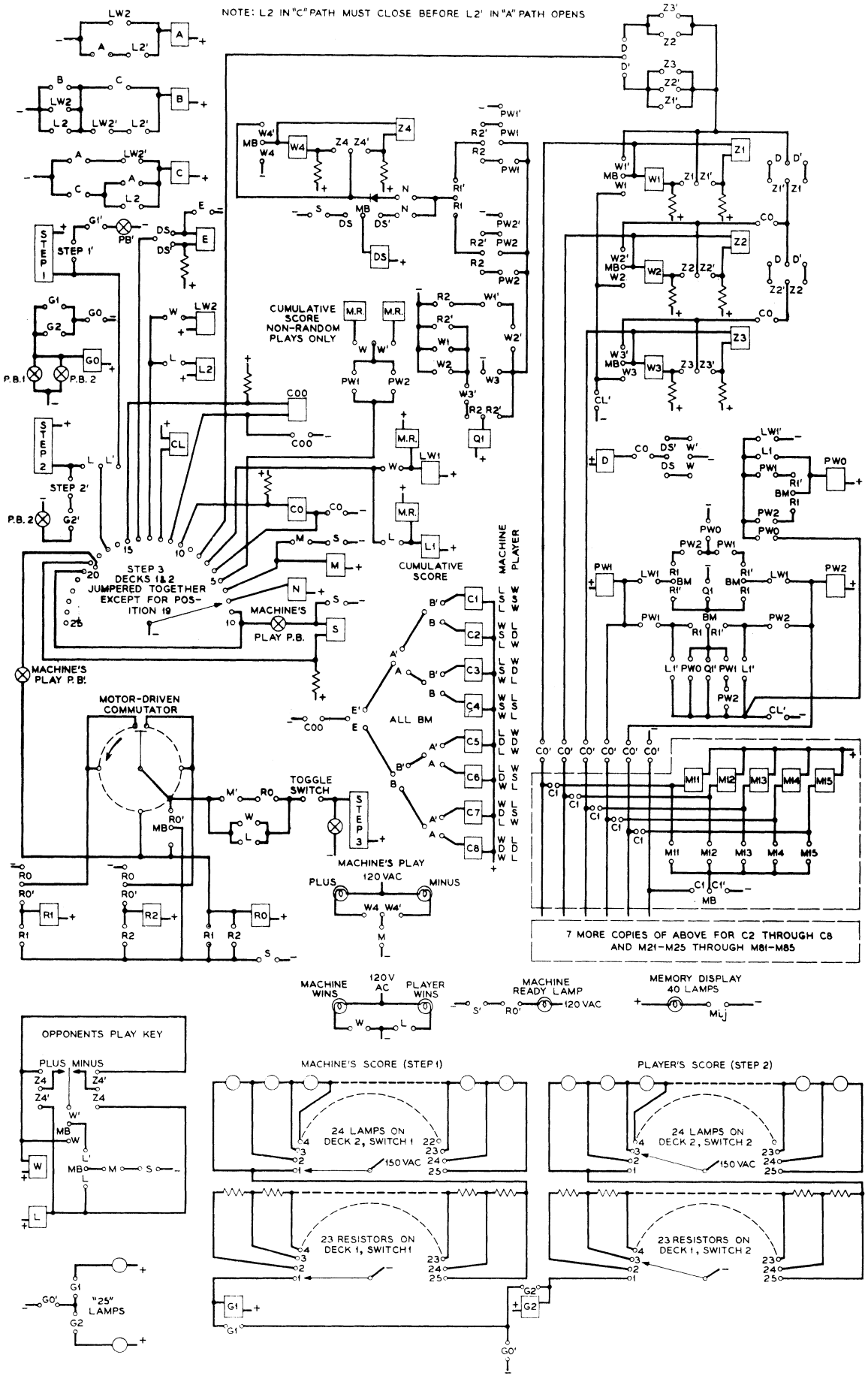

Fig. 5—Circuit diagram of SEER.

Position 9—Empty.

Position 10—C_0 is shunted down. This connects the present "state of play" register to $WZ_{1,2,3}$ and $PW_{1,2}$.

Position 11—C_{00} is operated and locked up. This disconnects the input to the tree on A, B, and E and releases the one of the connecting relays C_1–C_8 that was operated. The memory registers are now all isolated from the computing circuits.

Position 12—CL is operated clearing $WZ_{1,2,3}$ and $PW_{0,1,2}$.

Position 13—A pulse is applied through W or L to bring the "state of play" up to date. A is operated if the machine won this play. B is operated if the machine won the last play.

Position 14—E becomes the same as DS. The new "state of play" is now set up on the tree on A, B, and E.

Position 15—C_{00} is shunted down activating the tree and operating one of the relays C_1–C_8 corresponding to the new "state of play." $PW_{1,2}$ and $WZ_{1,2,3}$ become the same as the new registers to which they are now connected.

Position 16—Empty.

Position 17—A pulse is sent to one of the scoring stepping switches.

Position 18—Empty.

Position 19—S is shunted down. This releases R_1, R_2, M, DS, W, and L. If the machine's play pushbutton is not closed, R_0 stays operated and causes stepping switch 3 to step once more to:

Position 20—R_0 is released and the machine is ready for the next play. This last feature prevents the player from holding the pushbutton down and thereby making the machine's play predictable.

Résumé

Relay	*Function*
A	machine won last time
B	machine won time before last
C	transfers from A to B
C_{00}	"state of play" disconnect
C_0	main connector
C_1–C_8	"state of play" connectors
CL	clear PW and $WZ_{1,2,3}$
D	should have played *different*
DS	played *different*
E	played *different* last time
G_0, G_1, G_2	score display, control "25" lamps and reset display
L, L_1, L_2	machine loses
LW_1, LW_2	machine wins
M	machine's play indicated
M_{ij}	memory registers
N	machine's play computed
PW_0, PW_2, PW_2	previous wins
Q_1	counter says play *same*
R_0, R_1, R_2	random number
S	start
W	machine wins
W_1, W_2, W_3	*same—different* counter
W_4	machine's play
Z_1, Z_2, Z_3	*same—different* counter
Z_4	machine's play
M.R.	message register
STEP 1	machine's score display
STEP 2	player's score display
STEP 3	sequence control

Appendix II

Strategies for Beating the Machine

At any time the machine's play is determined by four things: the "state of play," the odds in the corresponding memory register (PW circuit), the sign of the number in the corresponding counter register ($WZ_{1,2,3}$ circuit), and the random element. All of this information except the random element is available to the player if he knows the starting state of the machine and keeps complete records of the play. (Even if he does not know the starting state he can play the machine so as to force it into a known state.) Since the "states of play" are independent and follow the same strategy, we will discuss only one of them assuming that the player keeps 8 records, one for each "state of play." Let a substate be determined by the count in the counter and the odds. (See Fig. 6 for example.) Since the machine's behavior in any substate does not depend on what the player did previously in that substate, the player can use a pure strategy, that is, he can always do the same thing in each substate and does not need a random element.

It can be shown that there is an optimum strategy for beating the machine. There is no better strategy and, except for trivial changes, no other as good. The strategy is described by the substate diagram of Fig. 6. The player plays so as to permit only the transitions shown. Each transition is labeled with its probability and whether the player wins or loses. Following this strategy the player's expectancy of winning is 3/5. The proof that this is the best strategy consists of examining what choice the player has for each substate. Most of these can be eliminated by general arguments. Then all combinations of the remaining choices are tried and the best one picked.

A simpler strategy which beats the machine is to play the opposite to what you did the last time for each of

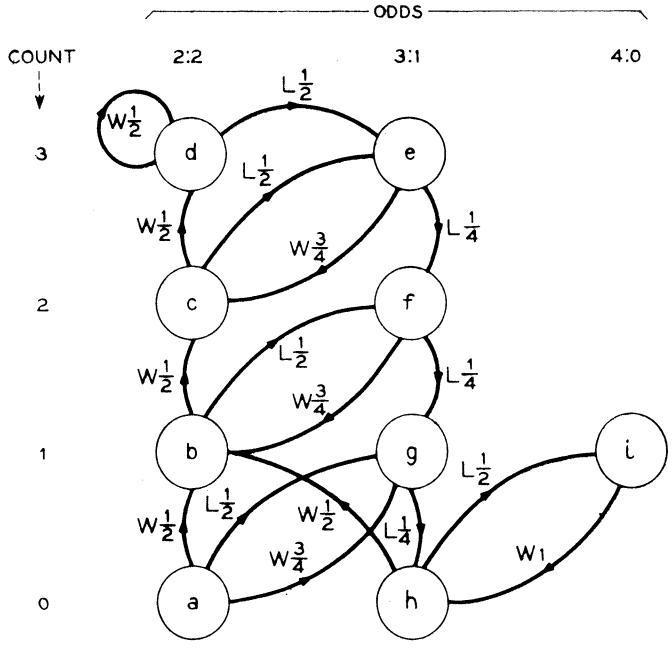

Fig. 6—Substate diagram for optimum strategy.

the eight states. With the proper initial state this follows the substate diagram of Fig. 7. The player's expectation of winning is 23/40 = .575. A machine for playing

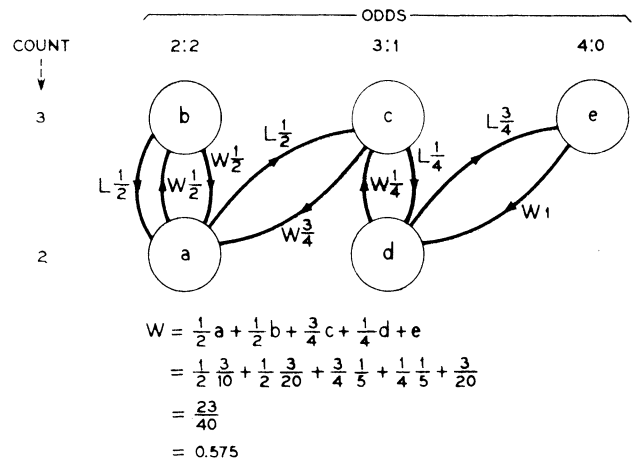

Fig. 7—Substate diagram for "Opposite from last time" strategy.

this strategy against the outguessing machine was assembled and a run of 1,043 plays taken. (The outguesser was provided with a table of random numbers on tape since the commutator was no longer valid as a random element against another machine.) The strategy won 598 times or 57.3 per cent of the time compared with an expected 57.5 per cent.

Acknowledgment

The author wishes to thank C. E. Shannon and E. F. Moore for many helpful discussions and suggestions.

Automatic Data-Accumulation System for Wind Tunnels*

J. J. WEDEL†, A. HUNTINGTON†, AND M. B. BAIN†

Summary—**A new high-speed data-accumulation system has been designed for a supersonic wind tunnel. The data are recorded on punched paper tape for direct input into an Electrodata digital computer. Extensive presentation of data is available to the wind-tunnel operators. All data are typed by an electric typewriter, and an automatic plotting machine plots several of the data words as functions of the independent-variable data word. Special codes to control the computer are automatically punched into the tape. The preliminary source of the data may be either manually operated keyboards or shaft-position digitizers. The new system increases the wind-tunnel pace, eliminates intermediate data handling before computation, and lowers the cost of data reduction.**

* Original manuscript received by the PGEC, September 22, 1955; revised manuscript received, December 27, 1955. This paper presents the results of one phase of research carried out at the Jet Propulsion Lab., Calif. Inst. Tech., under Contract No. DA-04-495-Ord 18, sponsored by the Dept. of the Army, Ordnance Corps.

† Jet Propulsion Lab., Calif. Inst. Tech., Pasadena, Calif.

Introduction

THE JET Propulsion Laboratory (JPL) operates two supersonic wind tunnels and is constructing a third. These tunnels generate a large volume of data which are reduced by the digital-computing facility of the Laboratory. Equipment has been built to record, on a punched paper tape, the data obtained from force tests. The tape is suitable for use in the digital computer with a minimum of editing. The system is designed to work with a computer processing center in which the larger portion of computer time is taken up by computation for research programs other than the wind tunnel. The present data-accumulation system is to be expanded to a complete automation of pressure