Some Remarks on the Game "Dama" which can be played on a Digital Computer

By N. V. Findler*

Dedicated to the memory o,
Professor J. R. Neukomm

The popularity of intelligence-simulating programs is increasing. Several parlour-games have been programmed for computing machines, with the idea of investigating machine learning techniques. This paper describes the strategies employed in a program for playing the game of dama on the SILLIAC.† A learning process is suggested by means of which an optimal grand strategy can be achieved.

1. Introduction

There exists an old ambition to construct a machine which can match any human opponent at such intellectual games as chess. It is a long story from F. Kempelen's "Chess Automaton" (eighteenth century), in which a very clever dwarf was hidden, to the "Chess Program" of a modern electronic digital computer.

So far, noughts and crosses, nim, checkers (Samuel, 1956),‡ and chess [Refs. (1), (2), (3), (4), (5), (7)] are the games which have been selected for programming. A detailed analysis of current chess programs can be found in Ref. (5).

In simple games such as noughts and crosses, the number of distinct situations is comparatively small, and a simple enumeration process will enable the machine to play the best possible game. In more complex games such as checkers, the problem is much more difficult since the number of variations is, for practical purposes, infinite, i.e. there is no question of "looking ahead" to the end of the game. It is necessary, therefore, to devise some means of evaluating the resulting board positions after looking ahead for a limited number of moves.

In this paper we formulate some evaluation functions for use in the game of dama, which is popular in Central Europe. This, like checkers, is also played on a chess-board. We shall, moreover, describe a "learning" process whereby the means of evaluation (and hence the quality of the game) should improve with experience. Similar schemes may be applicable to other situations of economic importance.

2. The Game Dama

Dama is played on a chess-board. Only the white squares are used. The number of pieces and the playing area remain constant throughout the game. Eight pieces of each side stand initially on the first two lines. If the squares are numbered according to Fig. 1, then 1–8 will refer to player A, and 25–32 to player B. A piece can move diagonally forward by single steps, or by double, quadruple, and sextuple steps, the multiple steps occurring only when the piece can jump over other pieces, his own or those of his opponent.

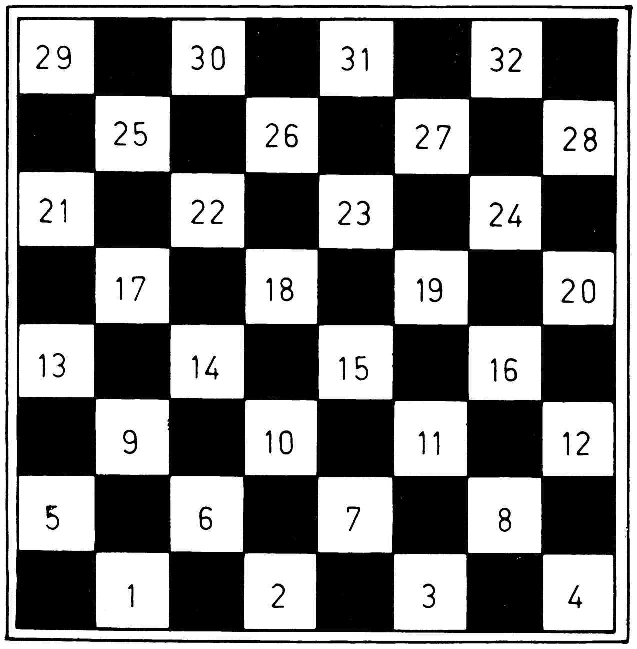

Fig. 1.—The numbering of the used squares.

The aim of the game is to move one's pieces to the opposite pair of lines before the opponent has done so. If a player causes a "blocked position" whereby his opponent has no legal move, the game is assumed to have been won by his opponent.*

* According to the original rules, this was a draw. The change in the rules was desirable in order to exclude trivial games. It also raises the interesting problem of how the machine is to avoid causing a "blocked position," and, if possible, force its opponent to do so instead.

* The experimental part of this work was carried out while the author held a research grant at the Adolph Basser Computing Laboratory, School of Physics, University of Sydney. The author is now with the Colonial Sugar Refining Co. Ltd., Sydney.

† SILLIAC, the electronic computer of the Adolph Basser Computing Laboratory, has at present an electrostatic memory of 1,024 forty-bit words. To characterize its speed, we note that the time for an addition is 75 microseconds, and for a multiplication 700 microseconds.

‡ Editor's Note: This contribution was first received before the publication of A. L. Samuel's most recent paper, "Some Studies in Machine Learning using the Game of Checkers," *I.B.M. Journal of Research and Development*, Vol. 3, No. 3, July 1959.

Although from this description it appears that the game is not complicated, nevertheless the number of possible ways in which a player's pieces can reach their final positions with single steps, i.e. when only the different routes are considered, is of order 10^9. This number has to be multiplied by 56! if we distinguish between the different orders of steps. Possible jumps will cause the result to increase by a further large factor.

The game displays many of the characteristic features of more complex games, but has the advantage that the program can be fitted into a comparatively small memory, and a game can be completed in a reasonable time.

The following alternative strategies suggest themselves.

(*a*) Retain the smallest possible amount of information about a game. Hence the evaluation process does not care about the past history of the game but looks for a possible best reply in a given position; "best" in the sense that the given position is the only information. The relative simplicity of this method affords the possibility of playing simultaneously with a number of opponents.

(*b*) Retain as much useful information as possible: recall the machine's and the opponent's intentions, implement and frustrate them, respectively.

Of these, version (*b*) may correspond to the more continuous strategical line followed by most players, while version (*a*) is seldom adopted, except possibly when a chess champion plays with 50 or so persons.

Unfortunately, the speed of the machine, and the limited memory space, meant that only version (*a*) could be realized by the present program. Each board, i.e. each game, requires two words, one representing the machine's, and one the opponent's pieces. Twenty-five games can be played simultaneously.

The geometrical description of the chess-board consists of two parts: (*a*) a record of which squares can be reached from any one square by a legal move; (*b*) a pattern by means of which it can be stated whether at any stage each final square can still be filled in, i.e. whether there is at least one separate piece remaining in the "shadow" of each final square. Thus, each square has a "shadow word" indicating its quality in this respect.

The program is flexible enough to try each of the following alternatives.

(*a*) The number of moves ahead which the machine computes can be:

(1) always six half-moves*;

(2) four half-moves in the opening and end games, six half-moves in the middle game†;

(3) always four half-moves.

(*b*) When looking ahead, the moves considered can be:

(1) all, regardless of their utility and the ordinal of the half-move in question;

(2) all moves in the first two (or four) half-moves and only the jumps in later half-moves;

(3) only the jumps in the opening and end games, regardless of which half-move is in question, and all moves up to two half-moves ahead when in the middle game;

(4) only the jumps, throughout the game.

We must note that the program also looks for single moves if there is no jump reply in the above restricted cases. This is one of the necessary features of avoiding a "blocked position."

With a certain modification of somewhat larger extent, the program can also be made to work in the following, fairly natural way: it computes four half-moves ahead but, in the middle game, it picks out the variations which appear to be the best according to the evaluation of this extent (usually at most two or three) and investigates them further for two more half-moves.

In order to make the machine move fast enough for a demonstration, we altered the program so that it always looks four half-moves ahead; it considers only the jumps in opening and end games, while in the middle game all moves are considered at the first two half-moves and only the jumps at the second two half-moves. With this arrangement, the machine requires 10–150 seconds to produce a move, although in most cases the quality of the move is no better than those of a moderately experienced player. However, a somewhat greater depth of thinking is needed if the learning process (described in Section 5) is to be successful, that is, lead to a useful set of optimum parameters. Indeed, it would be desirable to compute at least six half-moves ahead, and if possible to apply the type of modification noted in the previous paragraph: in this case the time for a move is increased by a factor of 8–10.

3. The Question of Grand Strategy

Since dama is complicated enough to rule out the possibility of looking ahead to the end of the game (except, of course, in the end phases) the question arises: is there a way of playing which provides the best result if we are restricted to think a limited number of moves ahead? If the evaluation is based on such a technique, then the average difference (which we might look upon as a truncation error) between the evaluation functions of a perfect game and the one being played is the smallest possible.

A suitable mathematical formulation of sub-strategies is usually possible with games of middle or more complex nature. For example, in chess the factors to be counted are the different values of the particular pieces, the value of the king's safety, number of squares dominated by pieces, centre control, etc. [cf. Refs. (5), (7)]. We assume that the grand strategy is some linear combination of the sub-strategies.

In the following part of this Section we will try to formulate a similar scheme for dama. Each move is characterized by a number, the linear combination of several figures-of-merit (FOM). Here we propose six possible FOM, though doubtless others could be suggested.

* A "half-move" is either a black or a white move.

† If there is at least one piece in the middle two lines, we shall refer to the state of play as being a *middle game*.

(*a*) A particular position can be characterized in a static way from the standpoint of player A by the number

$$r = \sum_{i=1}^{8} r_A^{(i)} + \sum_{i=1}^{8} r_B^{(i)}$$

where $r_A^{(i)}$ and $r_B^{(i)}$ are the "rank numbers" belonging to the *i*th piece of player A, and the *i*th piece of player B, respectively. The rank number of any piece is its distance (measured in lines) from player A's end of the chess-board. Thus, in the starting position,

$$r_0 = (4 \times 1 + 4 \times 2) + (4 \times 7 + 4 \times 8) = 72.$$

Obviously, player A tries to increase this value, while his opponent wants to keep it as low as possible. If $|r - r_0| > 6$, one side has gained so much that the opponent cannot make up for it by one move (the best possible move is a triple jump).

The first FOM is the major characteristic of a move and can be defined as

$$Q_1 = \Delta r,$$

the change in rank number achieved by that move.

(*b*) Let us denote the current position by P_l, and assume that player A thinks *k* half-moves ahead. There is some difference between the values of the jumps made in the course of *k* half-moves. Thus, at the first half-move a jump over an opponent's piece is more valuable than a jump over A's own piece, for the opponent might remove that particular piece later in order to prevent a good move on player A's part, while player A's piece can be made to remain there. For any later half-move, the position may be reversed, as it is undesirable to leave jumping pieces for the opponent. In other words, player A's tendency is to build up a favourable position while player B tries to destroy it. Consequently a suitable second FOM for any particular sequence of *k* half-moves is

$$Q_2 = (n_B^{(l)} - n_A^{(l)}) - \alpha_2' \sum_{i=l+1}^{l+k}{}' (n_B^{(i)} - n_A^{(i)})$$

where α_2' is a parameter the optimal value of which might be found by the learning procedure described later; and $n_A^{(i)}$ and $n_B^{(i)}$ are, in the position P_i, the number of possible jumps over player A's and player B's pieces, respectively. Accordingly, the value of, say, the first bracket can vary from -3 to $+3$, depending on the number and colour of the pieces being jumped over in the course of the first contemplated move. The sum Σ' is to be extended only over player A's contemplated moves in the course of *k* half-moves.

(*c*) The number of jumps possible to the opponent in the position P_{l+k+1} is unfavourable in contrast with those of player A. Thus, the use of a third FOM is desirable as follows:

$$Q_3 = n_A^{(l+k+1)} - n_B^{(l+k+1)}.$$

(*d*) A certain "levelness" is advantageous for player A's pieces, so that they can more easily block player B's jumps. The smaller the number

$$Q_4 = 8 \sum_{i=1}^{8} (r_A^{(i)})^2 - \{ \sum_{i=1}^{8} r_A^{(i)} \}^2$$

is in the position P_{l+k}, the more level the pieces are.

(*e*) Moves into the final two lines might be of somewhat less value (there being no need to worry about them any more). Clearly, (*d*) partly involves this aspect; however, we can introduce another FOM as follows:

$$Q_5 = n_f^{(l)}$$

where $n_f^{(l)} = 0$ or 1, depending on whether player A's move leads to a final square or not.

(*f*) Moves away from margin squares may be preferred, since there the pieces have less chance of being advanced in a favourable way. On the other hand, the marginal squares in the final lines are harder to fill. It would seem reasonable to introduce a new quantity, the "mobility" of any given square. It is defined as the sum of the numbers of the different routes along which a piece from a certain square can reach all possible final squares with single steps. Table 1 illustrates the way of computing the mobility of the square "1."

TABLE 1

The Way of Computing the Mobility of the Square "1"

FROM:	TO: ($t =$)	m	r	l	$P_m^{(r,l)}$	d_t	g_t	g_T
	25	6	3	3	20	6	14	
	26	6	4	2	15	1	14	
	27	6	5	1	6	0	6	
1	28	6	6	0	1	0	1	104
	29	7	3	4	35	21	14	
	30	7	4	3	35	7	28	
	31	7	5	2	21	1	20	
	32	7	6	1	7	0	7	

Here m is the number of moves necessary to reach a particular final square ("ro") in the course of which there are r moves to the right and l moves to the left; $P_m^{(r,l)} = \binom{m}{r}$ would be the number of possible different routes if there were not d_t routes to be subtracted due to the finiteness of the chess-board; $g_t = P_m^{(r,l)} - d_t$ and finally, the mobility assigned to the square is

$$g_T = \sum_{t=25}^{32} g_t.$$

In general, it could be said that the change in mobility should be kept low. The sixth FOM characterizes a move by the difference between the mobilities of the squares occupied by the moving piece before and after the move,

$$Q_6 = g_T^{(l)} - g_T^{(l+1)}.$$

Now, a grand strategy is arrived at by allocating a certain weighting factor to each FOM.* The linear

* The formulae for the values Q are constructed so that each weighting factor is to be positive.

combination of the values Q formed according to the grand strategy yields the "utility" of a particular move. Having investigated all the possible variations of moves to a depth of k half-moves, the machine picks out that most favourable for itself, i.e. that which has the largest utility assuming that the opponent also plays his best. (The evaluation of the opponent's possible replies is carried out according to the same scheme.)

An example might render this picture clearer. Fig. 2 shows a particular position in which each of the moves 5–21, 7–21, 11–25 and 11–27 achieves the same gain in rank number, 4. For the sake of simplicity, let $k = 2$, i.e. we think only two half-moves ahead. What can we say about the different sub-strategies?

While the moves 5–21 and 7–21 represent jumps over the machine's own pieces and the opponent cannot possibly foil these by his subsequent move, the move 11–25 is not safe in this respect; finally, the move 11–27 will still be available to the machine next time. Thus, with regard to Q_2, 11–25, 11–27, and 5–21 or 7–21 is the order of preference.

As far as Q_3 is concerned, the move 7–21 is less favourable than the others since this allows the opponent one more possible jump.

The fourth FOM would favour the moves 5–21 and 7–21 because these decrease the "levelness" to a smaller extent than do the moves 11–25 and 11–27.

As to Q_5, the moves are irrelevant; none of them leads to a final square.

As regards the change in mobility, the order of preference is 5–21, 11–25 or 11–27, and 7–21, as a simple calculation will show.

4. The Program

Unfortunately, the available memory space limited the evaluation program to the first, apparently major, FOM. If we drop the possibility of a blocked position, the procedure is as follows.

(1) Find all moves of the machine in the given position and pick out the first. (The order of the recorded moves is not relevant.)

(2) Find all possible replies of the opponent. Pick out the first.

(3) Do the same at each half-move up to the last one (kth). Note which of the last possible half-moves yields the largest change in rank number, and record the change $C(k) = (\Delta r_k)_{max}$. Form the difference $C(k - 1) = \Delta r_{k-1} - C(k)$, where Δr_{k-1} is the change in rank number achieved by the second last half-move.

(4) Pick out the next possible $(k - 1)$th half-move and evaluate for it $C(k - 1)$. Repeat the process with all the $(k - 1)$th half-moves and record the maximum $C(k - 1)$.

(5) A corresponding piocedure with the $(k - 2)$th, $(k - 3)$th, etc., half-moves eventually leads to a maximum value of $C(1)$ [since at each step it is, of course, necessary to re-evaluate higher $C(k - i)$].

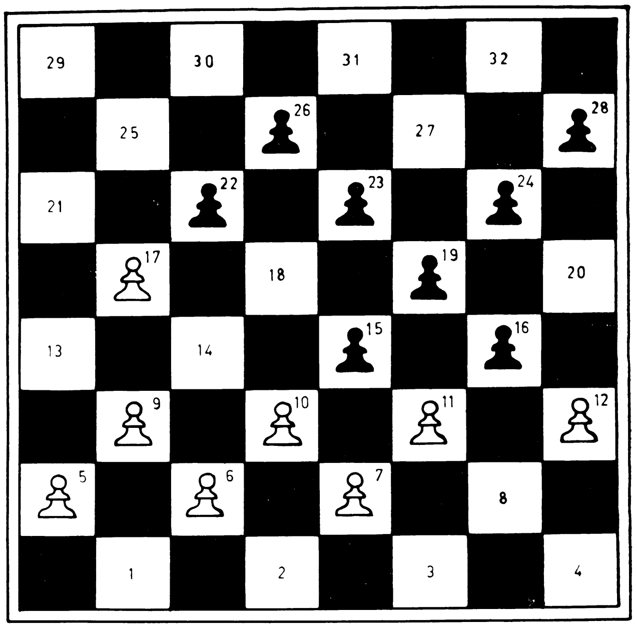

Fig. 2.—A specimen position to illustrate different sub-strategies on the machine's part (white pieces).

The move belonging to this variation will be the best possible move of the machine, whatever the opponent does.

Usually there can be several moves yielding the same $C(1)$. In order to minimize the number of lagging pieces, the program was made to move that piece which stands on the square bearing the smallest number.

5. A Possible Learning Process

If, in some way, the machine can optimize its grand strategy, i.e. improve the set of the weighting factors used in the evaluation, then this would correspond to the human way of acquiring experience, refining the playing skill. An expert chess player must have analysed a large number of games, pondering the different alternatives in given situations. His "playing style" represents a technique characterized by the ratio between the weighting factors in his grand strategy.

There appears to us a mathematical way in which an optimal grand strategy can be obtained. It is an iteration process during which the machine plays with itself. We describe the method in connection with dama, but it can be used also for other games.

We assign a weighting factor of undetermined value at the moment to each FOM. Let us call them α_1, α_2, α_3, α_4, α_5, $\alpha_6 = \{\alpha\}$. We can normalize the most significant of them as $\alpha_1 = 1$ and look for the other values.

For the sake of simplicity, we assume in the following that the optimal grand strategy of a player does not depend on whether he moves first or second at the beginning of a game.

Let us choose a reasonable starting set $\{\alpha\}_{00}$ and a number, say six, somewhat different sets, $\{\alpha\}_{01}, \{\alpha\}_{02}, \ldots, \{\alpha\}_{06}$. The machine makes a first move according to $\{\alpha\}_{00}$ then "moves round to the other side of the chess-board" and replies according to the sets of weighting factors $\{\alpha\}_{01} \ldots \{\alpha\}_{06}$. These second moves (which may not all be distinct) serve to define the start of six different games on six chess-boards, and will be replied to according to $\{\alpha\}_{00}$ again. The fourth moves are made according to $\{\alpha\}_{01} \ldots \{\alpha\}_{06}$, and this alternating type of game goes on until each game is finished. Among the sets $\{\alpha\}_{01} \ldots \{\alpha\}_{06}$ there will be some which lose, some which bring their pieces home within just as many moves as $\{\alpha\}_{00}$ does, and some which win. The machine has to pick out the one that wins within the smallest number of moves or, if there are several of equal quality in this respect, then that in which the opponent is left in the most unfavourable position. Let us call this set $\{\alpha\}_{10}$ and choose again six convenient and somewhat different sets $\{\alpha\}_{11} \ldots \{\alpha\}_{16}$. By repeating the above process we obtain $\{\alpha\}_{20}$, and then $\{\alpha\}_{30}$, and so on, until we find that there is no better set than, say, $\{\alpha\}_{i0} = \{\alpha\}_{opt}$. The actual values of the individual weighting factors depends on the "mesh size" among the sets $\{\alpha\}_{i1} \ldots \{\alpha\}_{i6}$, and on the gradient of the surface which characterizes the utility of the set of the weighting factors in the environment of $\{\alpha\}_{opt}$. The above procedure of finding an optimal ground strategy would fail if one hits upon a local minimum or if the particular game is not sensitive enough to respond to changes in $\{\alpha\}$. Both cases are illustrated symbolically in two dimensions in Fig. 3.

Finally, we remark that it should be possible to devise a more general learning process, in which the machine determines not only the weights of the evaluation terms, but also the nature of the terms themselves.

6. Conclusions

In principle, a general-purpose digital computer can be made to play any "game," in the sense that the machine is given a position to be evaluated and has to find by trial the best possible action in order to achieve a maximum gain. Naturally, the quality of the game will depend on the depth of "thinking" employed, and this in turn depends on the speed and storage capacity of the machine. While special-purpose analogue machines may be more adequate in certain cases, digital computers are more flexible and versatile. Their detailed method of operation may be different from natural thought processes, but the results themselves are similar: they can thus serve as models for certain human activities.

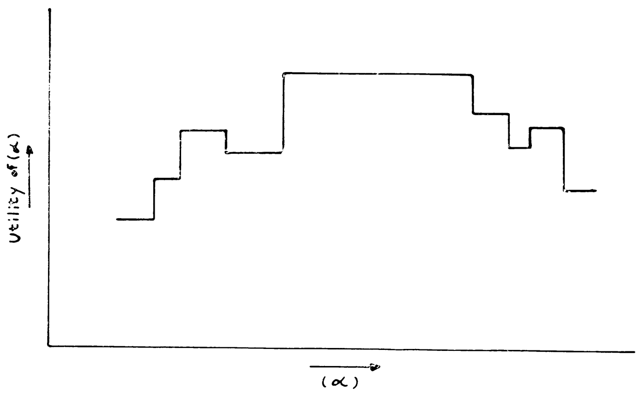

Fig. 3.—Two-dimensional illustration of the cases of local optima and insensitiveness with regard to the changes in weighting factors.

Programming experience has brought out three major points:

(*a*) The limited size of the rapid-access store (1,024 words) was responsible for a considerable amount of time spent in packing and unpacking data, and other red-tape operations. A rough estimate indicates a reduction in computing time by a factor of 10–15, if the memory is "infinitely" large.

(*b*) Even if there is a satisfactory procedure for evaluating the quality of the moves to be considered, thinking two half-moves deeper requires considerably more storage space and computing time.

(*c*) Finally, certain advances in machine design, e.g. suitable instructions contributing to pattern recognition, could also reduce the time for a move.

Acknowledgements

The author wishes to express his gratitude to Professor H. Messel for his interest in this work, and for the research facilities provided in the Adolph Basser Computing Laboratory. Further, he also thanks Dr. J. M. Bennett and Mr. J. C. Butcher for many stimulating discussions, and the visiting Professor, D. H. Lehmer (Berkeley, California), for valuable comments.

References

1. Bernstein, A., Roberts, M. de V., Arbuckle, T., and Belsky, M. A. (1958). *Proc. of the* 1958 *Western Joint Computer Conference*, May 1958.
2. Bowden, B. V. (1953). *Faster than Thought* (Pitman, London).
3. Kister, J., Ulam, S., Walden, W., and Wells, M. (1957). *J. Assocn. for Computing Machinery*, Vol. 4, p. 174.
4. Newell, A. (1955). *Proc. of the* 1955 *Western Joint Computer Conf.*, March 1955.
5. Newell, A., Shaw, J. C., and Simon, H. A. (1958). *IBM Journal of Research and Development*, Vol. 2, p. 320.
6. Samuel, A. L. (1956). *I.E.E. Proc.*, Vol. 105, Part B, p. 452.
7. Shannon, C. E. (1950). *Phil. Mag.* (7), Vol. 41, p. 256.