

TESTING CORRECTNESS OF STRATEGIES IN GAME-PLAYING PROGRAMS

Max A. Bramer
Mathematics Faculty,
The Open University
Milton Keynes,
United Kingdom

This paper considers the distinction between winning strategies in game-playing programs which are either 'optimal' or 'correct'. The relative merits of these two types of strategy are considered and methods are proposed for producing correct algorithms by a process of iterative refinement based on an analysis of 'win-trees'. An example is given of the development of a fully correct strategy for the King and Rook against King chess endgame.

1. Correct and optimal strategies

A problem which has emerged from recent studies of endgames in chess (i.e. subgames with only a small number of pieces remaining) concerns the relative importance of constructing strategies which are either 'optimal' (the stronger side plays perfectly in every position) or 'correct' (the stronger side wins eventually from every theoretically won position, but not invariably in the smallest possible number of moves).

The major advantages of aiming for optimal algorithms are that they can readily be verified automatically given a database of the best move in every position [1], and that they are intrinsically stronger than correct ones. However, the development of such algorithms is open to the theoretical objection that consistently optimal play is probably beyond the ability of even the strongest human players, and the practical objection that it is not feasible to construct either the programs or the databases against which to verify them for any but the most simple endgames.

If it is agreed that the development of correct algorithms is a more appropriate aim, there are two important questions which must be answered and these are addressed in the remainder of this paper:

- (i) how can the correctness of an algorithm be tested, either automatically or otherwise?
- (ii) how can a 'nearly correct' algorithm be refined to perform progressively better, based on an examination of its non-winning play?

Judging the optimality of any given move is inevitably an error-prone activity, even for experts. Judging the correctness of a strategy might seem

to be easier, but it is in fact harder, as a result of the fact that a move can be both optimal and non-correct. Correctness refers essentially to sequences of moves, whereas optimality refers to individual positions, independently of one another. The contaminating effect of even a small number of poor (although possibly win-preserving) individual moves can result in non-correct play from many thousands of positions, the great majority of which will pass unnoticed even in extensive expert testing.

2. Testing the correctness of an algorithm

Modifying the basic method of generating databases of shortest-path winning moves for every position in a given endgame [1] gives an automatic method of testing the correctness of a playing algorithm for a chosen side (say White). The revised method generates an implicit win-tree containing all the positions from which the playing program invariably wins for White, although not necessarily in the smallest possible number of moves. The tree is generated by backtracking from terminal positions which are all wins for White, assuming that Black always plays to defer a loss for as long as possible.

The generation method can be summarized as follows (using WTM and BTM to denote White and Black to move, respectively).

- (i) Mark all positions as either a terminal win for White, illegal or 'unevaluated'.
- (ii) For all unevaluated BTM positions, compute and store the number of legal moves available. (If zero, Black must be stalemated, so mark as a draw.)
- (iii) Backtrack one ply, by generating all legal antecedents (WTM) of the terminal positions. Mark those previously unevaluated

(with WTM) as wins in one ply, provided that the program would have selected the move which was reversed.

- (iv) Backtrack from those positions evaluated at the previous level by generating all their legal antecedents (BTM). For those previously unevaluated (with BTM), decrease the counter of legal moves available by one. Mark any position for which the counter is reduced to zero as a win for White at depth two ply.
- (v) Continue backing-up, from the positions evaluated at the previous level only, to White wins in 3,4,5,... ply in this way, distinguishing between backtracking to WTM and to BTM positions as in (iii) and (iv) above. This procedure ensures that every winnable position is included in the tree and gives the smallest number of ply needed for White to win against any play by Black.
- (vi) Terminate when a level is reached at which no new evaluation takes place. The remaining unevaluated positions are not wins for White with the given playing algorithm. If all theoretically won positions are in the tree at this stage then the algorithm is correct.

3. Examining the win-tree

The number of theoretically won positions is known for certain endgames. However an examination of the win-tree can be used to determine whether a playing algorithm is correct, without this knowledge, by the following procedure.

- (a) Examine the BTM positions in the tree at each of the ply depths 0,2,4,... in turn.
- (b) At each level generate all the legal WTM antecedents of the BTM positions examined. If all these antecedents are in the tree (at any level) go on to the next level, otherwise note any WTM positions not in the tree and stop. These positions comprise the error set for the playing algorithm.
- (c) The playing algorithm is correct if and only if the highest BTM level is eventually reached (with no legal antecedents found that are not in the tree).

If the algorithm is not correct, then the WTM positions in the error set are those at the lowest level of the tree where White had the opportunity to play a winning move but did not take it or play any other move which led to a win. Changing White's algorithm to place each such position in the win-tree (by adjusting the move played to give that BTM successor from which the position was backtracked at stage (b)) will generally cause many other positions (both WTM and BTM) also to be included in the tree by

virtue of the normal backtracking process of tree generation. Proceeding in this way iteratively will lead to a fully correct algorithm in a finite number of steps. (Since the refinement process has no effect on the positions in the tree at the BTM level from which the previous error set was generated by backtracking, or at any lower levels, the next time the tree examination algorithm is applied, it is only necessary to begin with positions at the next highest BTM level, and so on.)

Changing White's moves at a relatively low level of the tree (i.e. in the 'error set' positions, rather than some other arbitrary set of positions) is likely to introduce the greatest number of new positions during the subsequent backtracking and thus to lead to a relatively short sequence of iterations before a correct strategy is reached. Moreover, the total number of positions for which White's move is modified (equivalent to introducing "special cases" in the original algorithm) will be substantially less than the number of incorrect positions which occurred with the original version of the playing algorithm.

A semi-automatic alternative to the above approach is to make use of an analysis of the error set (or the union of two or three error sets produced by the 'automatic' refinement process) to suggest modifications to the playing algorithm within its original framework (of patterns, say), on a heuristic basis, with the complete win-tree recomputed at each stage of the iteration.

It is reasonable to suppose that most or all of the positions in an error set will have some feature in common to serve as the basis for a change to the algorithm, for example, the required White move in many positions in the error set will often lead to the same BTM successor. When well chosen, changes to the program's pattern-knowledge are likely to have greater beneficial effect and lead to more rapid convergence than those to individual positions. Naturally, however, if the changes are badly chosen there is no longer any guarantee that the process will terminate.

4. Refinement of a correct algorithm for King and Rook against King (K RK)

Both the automatic and the semi-automatic refinement processes described above were applied to an existing algorithm for K RK, which was thought to be correct, although non-optimal, as a result of extensive testing against human opponents. The algorithm made use of a ranking

of all legal BTM positions for that end-game based principally on a classification of positions into fourteen simple patterns, to find moves for White (the side with the Rook) only. Further details are given in [2]. Surprisingly, the initial version of the win-tree contained only 14,978 WTM and 7,654 BTM positions out of totals of 27,352 and 34,968 legal positions respectively (in a suitable standard orientation). Since it is known that all WTM positions and all but 3,495 (drawn) BTM positions are theoretical wins for White, it follows that in as many as 45% (WTM) and 68% (BTM) of all legal positions the algorithm failed to win when it could.

The first attempt at refining the algorithm was by means of the fully automatic process described previously. The following table summarizes the first six iterations. At all stages of the refinement process the maximum depth of positions in the win-tree remained as 52 ply, compared with the theoretical maximum required for an optimal algorithm of 32 ply.

Iteration	WTM moves adjusted		Total added to win-tree		No. of incorrect positions remaining	
	Depth					
	No.	(ply)	WTM	BTM	WTM	BTM
1	1	9	2,339	3,398	10,035	20,421
2	8	13	1,956	2,466	8,079	17,955
3	89	15	3,000	5,184	5,079	12,771
4	176	17	1,188	2,124	3,891	10,647
5	269	19	2,542	6,138	1,349	4,509
6	190	21	621	1,882	728	2,627
Total	733	-	11,646	21,192	-	-

Although the iterations were not followed through to a final correct algorithm, the table above gives a good indication of the contaminating effect of a relatively small number of wrongly played positions.

The semi-automated approach to modifying the original algorithm began by examining the cause of incorrectness in the 9 positions in the first two error sets identified above. The first change made simply involved a restriction on the highly-valued pattern "King two files apart, with the Rook on the file between them", which led to a dramatic reduction in the number of incorrect positions from 10,035 to 3,291 (WTM) and from 20,421 to 9,513 (BTM). Subsequent minor modifications to a second pattern, again suggested by examination of error sets, gave a final algorithm which plays correctly in all positions, with a maximum depth of 60 ply. The program plays optimally in 18,386 out of the 27,352 legal positions for KRR (67%), and requires only two ply more than the theoretical

minimum in a further 4,694 positions. Further details of this experimentation are given in an extended version of this paper [3].

To summarize, the method of win-tree examination can be used not only as a means of testing the correctness of an algorithm without any knowledge of the total number of theoretically won positions, but also to enable a fully correct algorithm to be refined, either automatically or by semi-automatic means, taking account of the overall framework of patterns etc., of an original algorithm. Using these methods would therefore seem to make the development of correct as opposed to optimal algorithms a feasible proposition in future research.

Although the above discussion has been presented solely in terms of simple endgames in chess, it is equally applicable to other problems which can be represented as games of perfect information with two players who move in turn, such that the result is either a win for one player (and a loss for the other) or a draw.

REFERENCES

- [1] Bramer, M.A. Computer-generated databases for the endgame in chess. Mathematics Faculty Technical Report, The Open University, Milton Keynes, U.K., 1978.
- [2] Bramer, M.A. and Clarke, M.R.B. A model for the representation of pattern-knowledge for the endgame in chess. Forthcoming in *Int. J. Man-Machine Studies*, September 1979.
- [3] Bramer, M.A. Correct and optimal strategies in game-playing programs. Mathematics Faculty Technical Report, The Open University, Milton Keynes, U.K., December, 1978.