MASSACHUSETTS INSTITUTE OF TECHNOLOGY
PROJECT MAC

Artificial Intelligence
Memo No. 205 July 1970

LOOK-AHEAD STRATEGIES IN ONE PERSON GAMES WITH RANDOMLY GENERATED GAME TREES

David S. Johnson

Originally published as M.I.T. M.S. thesis August 1968.

Work reported herein was supported by the Warren McCulloch Laboratory, an M.I.T. research program sponsored by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract number NOOO14-70-A-0362-0002.

ABSTRACT

A random method for generated binary trees is presented, and two forms of a class of one person games called "Tree Solitaire" which have such trees as their game trees are defined. After what "look-ahead strategy" means in terms of such games is discussed, a theorem on the most efficient use of unlimited look-aheads is proved, and a collection of strategies involving 0, 1, or 2 look-aheads per move is introduced.

A method involving diagrams is presented for calculating the probability of winning under the various strategies over a restricted class of games. The superiority of one of the 1 look-ahead strategies over the other is proved for games of the _first_ form on this restricted class. For games of the _second_ form in this class, all the introduced strategies have their chances of winning calculated, and these results are compared among themselves, with the result for the first form of the game, and with the results of Monte Carlo estimation of the chance of winning in a particular case.

An approximate method for evaluating strategies from any given position is introduced, used to explain some of the previous results, and suggest modifications of strategies already defined, which are then evaluated by Monte Carlo methods.

Finally, variants on Tree Solitaire are suggested, their general implications are discussed, and using the methods already developed one of the most suggestive variants is studied and the results show a significant reversal from those of the original game, which is explained by the difference in the games on one particular.

TABLE OF CONTENTS

INTRODUCTION

In confronting the problem of programming computers to play games such as chess, where good play seems to require the ability to "look ahead" to future positions, one is immediately aware that, despite the computer's speed, the game tree is much too large to allow looking ahead to the end of the game in making the decision on where to move. Present methods must look to a certain limited depth in the tree using functions which attempt to evaluate "how good" the positions seen are, and use what they find as best they can. Certain techniques such as α-β have been developed to streamline this process, but no theoretic framework for the problem of look-ahead has been constructed.

This paper reports an attempt to begin such a construction by examining a class of one person games with simplified game tree, and evaluating various limited and unlimited look-ahead strategies for it. In doing so, a terminology for dealing with look-ahead is developed, and the kinds of distinctions and definitions that may prove necessary or useful for further work are discussed.

This is an exploratory study with no known predecessors in the literature, and so tight organization and single-minded purpose have been avoided in the interest of presenting all the varied results so far obtained from different approaches, and as many as possible of the conjectures and suggestions for further study that have occurred to the author. In addition much time has been spent on detailed explanations of problems and methods which, though not particularly deep, are probably not particularly familiar either.

Chapter 1 introduces the class of games called Tree Solitaire (abbreviated TS) and its two forms (TSD and TSI), by explaining how the game tree for such a game is generated, and then turned into a game.

Chapter 2 discusses what is meant by "look-ahead" in terms of this class of games, what is meant by "look-ahead strategy", and how such are to be evaluated. It then proves a few simple results and defines the look-ahead strategies that will be dealt with in later chapters: STRAT and NOSTRAT involving no look-ahead, HSTRAT and LSTRAT involving one, and BSTRAT, 2HSTRAT, and 2LSTRAT involving two look-aheads.

In chapter 3 methods are introduced which produce closed formulas for the probability of winning using HSTRAT and LSTRAT in games of form TSD. It is proved that LSTRAT is always the better strategy in such games.

Chapter 4 uses methods similar to those of chapter 3 to evaluate, by means of computer program, all the look-ahead strategies defined in chapter 2, over the class of games of form TSI. (Alternative evaluation by Monte Carlo methods is discussed for HSTRAT and LSTRAT.) The strategies are then compared as to their "effectiveness" in obtaining wins when various parameters involved in the generation of the games are varied.

Chapter 5 introduces the concept of "depreciation factors" for the purpose of evaluating strategies from positions other than the first one in the game. A plausible argument is given for the predominance of LSTRAT over HSTRAT in TSI. Modifications of HSTRAT and BSTRAT are suggested, using these depreciation factors, and evaluated by Monte Carlo methods so that they can be compared to the original strategies.

Chapter 6 introduces variants on Tree Solitaire as a means of suggesting ways of extending results already obtained. In particular a game which seems a much more reasonable idealization of games like chess is introduced, and a start is made toward applying the methods and results of the previous chapters to it.

CHAPTER 1

TREE SOLITAIRE AND ITS GAME TREE

Most game playing programs using look-ahead abstract a game tree from the rules of the game, and assign values to the points in the tree according to an evaluation function defined on the physical configurations of the corresponding positions in the game. In studying the workings of the look-ahead procedure theoretically, one is most interested in just the game tree itself and the values assigned to its points. Therefore in this report the procedure is to generate the game tree, assign values to its points, and then to define, with as much simplicity as possible, a game which the tree represents.

Most consideration is given to a one-person game, or actually class of one-person games of this type, which will be called "Tree Solitaire" (TS), and has been developed from an idea proposed by Seymour Papert.[1] Variants on this game will be discussed in Chapter 6. Trees for TS are binary, and generated as follows:

Start with an initial "fork" as shown in Figure 1.1

[1] In unpublished notes and conversation.

The top point of the form will be called a "split point".

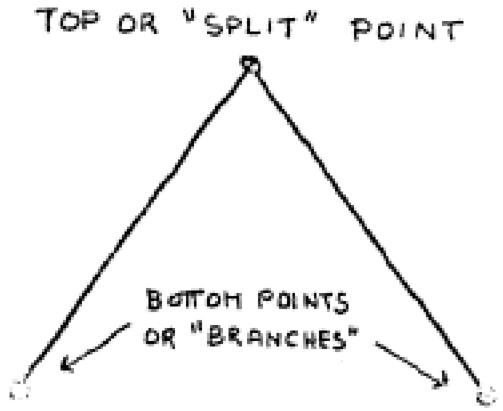


Figure 1.1: a fork

The bottom points of the fork are called the immediate "successors" of the top point, and sometimes "branches." The bottom points of the initial fork may be either further "split points" or simply end points. Whether a bottom point is a split point or an endpoint is decided by some probabilist method, chosen so that the probability that the point is an endpoint is a given number E. The probability that it is a split point is S = 1-E. Whenever no confusion will arise, an endpoint will be referred to as an E, a split point as an S. If a bottom point to a fork is determined to be an S, a new fork is added to the tree with this point as its top point and two new bottom points must be dealt with. As soon as all available bottom points have been determined to be E's, the generation of a binary tree is complete. Figure 1.2 gives an example.

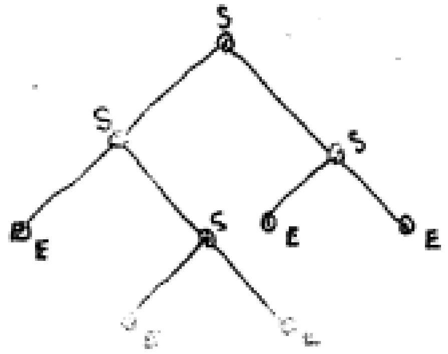


Figure 1.2: A generated tree

The tree will be generated with a finite number of points if $E \geq .5$.[1] The need for this will occasionally be apparent, but many of the results to be obtained will be valid for $E < .5$, and this will be explained when it occurs.

Assign to the top point of the tree the value 1. In some probabilistic way, decide on two number whose sum is this top value, and assign these two numbers to be the values of the branches of the fork. Thus in Figure 1.1, the top point has value 1 and the bottom points might have values .6

[1] Harris, The Theory of Branching Processes, p.7

and .4. For any fork, the ratio of the value of the higher valued branch to the top value is called the "split ratio". The split ratio here would be .6. If a bottom point is an S, we split its value between its two branches in the same probabilistic way. This process is continued until all points of the tree have been assigned values. In a finite tree the sum of the end points will equal 1.

In order to construct a game, some of these endpoints will be called wins, the rest losses. There will be no draws. Two similar methods of assigning wins and losses give rise to two different forms of Tree Solitaire.

The dependent form (TSD) treats the values of the endpoints as a probability distribution, and assigns one win to the game, locating it at an endpoint chosen according to the distribution. All other endpoints are losses. Thus the value of an endpoint is the probability that it is a win. The reason that this form of the game is called dependent is that, if one endpoint of value A is known to be a loss, then the probability that any other endpoint is a win is its original value divided by (1-A), since we know that one of the endpoints must be a win.

In the independent form of the game (TSI), the value of an endpoint is used as the probability that the endpoint is a win , the decision being made independently for each endpoint. Thus there need not be exactly one win. There may be zero, one, two, three, etc. Knowing that one endpoint is a loss gives no information as to the others. This form of the game was introduced because it seems closer to real games than TSD, and also because it was erroneously thought that TSD would be too complicated to analyse. (See Chapter 3).

To complete the definition of Tree Solitaire, one must explain the rules of play. Clearly, if the game tree is to work as game trees should, a "play" of the game will be representable as a path from the top point to an end point, with the points of the path representing positions in the game, and the lines between them representing legal moves. This is exactly what is done.

The "position" corresponding to each point is a certain set of information about that part of the tree:

1. The value of the point itself
2. Whether it is an E or an S
3. If an E, whether it is a win or loss
4. If an S, the values of its two branches, or equivalently, its split ratio.

A move is made from a position corresponding to an S point to a position corresponding to one of its branches. A binary choice is involved and herein lies the opportunity for strategy. The game starts in the position corresponding to the top point of the tree and continues until the position corresponding to an endpoint is reached, and is won if and only if that point is a win. From now on "the position corresponding to" will be omitted and positions will be identified with the corresponding points, hopefully without loss of clarity.

It would be apropos here to explain how this game compares to more complicated ones like chess, and to justify its use as a starting point in the study of the problem of look-ahead. First of all, the simplification to a one-person game is reasonable in the beginning of such a study, as well as convenient due to the nature of the game tree and its valuation. Similarly, having the tree be binary and thus having only two possible moves from any position, seems a reasonable simplification, as does the fact that an ordinary "play" of the game will only comprise a small number of moves. Letting the value of a position be its probability of being a win is a straightforward valuation, even if it is perhaps more powerful and simpler than those used in chess playing programs. In TS the chance of any position being a final one is a constant E throughout the game and this may give rise to doubts, but is perhaps justifiable by the fact that in games like chess a number of the moves from any position are such outright blunders as would create a lost game for the mover -- and a win for his opponent.

The principle objection to TS must arise from the fact that the values of positions of necessity decrease as the play of the game procedes farther down the tree. This is a reasonable criticism, and the weakness referred to does have an appreciable effect on the results, as will be seen when they are compared with those for a variant class of games without this weakness, in Chapter 6.

However, Tree Solitaire is a nice showcase for the various possible limited look-ahead strategies, and allows much straightforward analysis of them. It can serve as a workshop for developing methods of evaluating strategies, and it as well has a certain amount of intellectual appeal all by itself.

CHAPTER 2

LOOK-AHEAD STRATEGIES AND THEIR EVALUATION

In the context of Tree Solitaire, a "look-ahead" will be defined as seeing a successor position to the one you are in, or have already seen. By "seeing a position" will be meant the acquisition of the entire set of information associated with the position, as defined in Chapter 1. The difference between a "look-ahead" and a "move" is that looking ahead to a position, although it gives you as much information as moving there would, does not commit you to that position, whereas moving there would

A look-ahead strategy will work in the following way. Suppose you find yourself in position X. In accord with a given rule you choose those succeeding positions at which you will look -- the rule may allow you to use the result of your earlier look-aheads in choosing where to use your later looks. Once the look-aheads have been completed, you make use of the information to decide according to a second rule where to move, thus arriving at a new position, where, if it is not the end of the game, the process is repeated. Any information you may have gained in previous look-aheads as to the successors of this new position is assumed to be forgotten.

A look-ahead strategy can thus be seen to be defined by giving the two rules that govern it: the one that tells where to look, and the one that tells where to move, given the results of the look-ahead.

Strategies may be evaluated according to a number of criteria. High on the list, however, must be their effectiveness at achieving a win, over the whole class of games of Tree Solitaire. The principles of such evaluation are demonstrated in the following two cases of strategies involving no look-ahead at all.

NOSTRAT is defined by the rule that moves are chosen completely randomly, without any reference to the values of positions. In this case the expected value of the position you move to will be 1/2 the value of the position you are in The chance of winning on a given move is the expected value of the position at the end of that move times the chance of getting that far in the game, under the given strategy, times the chance of the position being an endpoint, i.e., times E. The chance of winning the game at all thus becomes a convergent infinite sum. In this case,

$$\frac{1}{2} E + \left(\frac{1}{2}\right)^2 ES + \left(\frac{1}{2}\right)^3 ES^2 + \ldots = \frac{E}{2}\left(\frac{1}{1-S/2}\right)$$

$$= E/(2-S) = E/(2-(1-E)) = E/(E+1)$$

If $E = \frac{1}{2}$, then the probability of winning $W_{NO} = .3333$.

NOSTRAT is not the best possible strategy involving no look-aheads. That honor belongs to the obvious strategy of always moving to the higher valued of the two branches of a fork, which strategy will be called simply STRAT. To calculate the chance of winning under STRAT, we make use of the expected value A of the split ratio as a prediction for the way values split at every split point we encounter. If the top point of a fork has value X, the expected value of the high branch's value is AX and for the lower valued branch BX, where $B = (1-A)$. Since the value at the top of the tree is 1, the chance of winning under STRAT can be seen to be

$$\sum_{n=o}^{\infty} AE(AS)^N = \frac{AE}{1-AS} = \frac{AE}{1-A+AE} = \frac{AE}{B+AE}$$

If $E = \frac{1}{2}$, $A = \frac{3}{4}$, then the probability of winning $W = .6000$.

Against "effectiveness" must be weighed the "cost" of the strategy, in terms of, say, computer time. Since it may be assumed in the case of look-ahead strategies that the principle time consuming factor would be the number of look-aheads, cost considerations would lead us to want to minimize this number per move over the class of all game trees.

However, in trying to balance this against the importance of having an effective strategy, any theoretical treatment would involve a certain amount of arbitrariness, which might have a considerable effect on the results. In order to avoid this problem, this study has concentrated on comparing as to their effectiveness various strategies whose costs are the same. "Limited look-ahead" strategies involving one look-ahead will first be examined, and then those involving two and possibly more.

But first, for the sake of perspective, we will note a result at the other end of the spectrum -- unlimited look-ahead. With unlimited look-ahead a win is assured (in TSD; in TSI a win is assured if there in fact is one). So what we must do here is pick from a number of equally "effective" strategies, that which is least costly. Such strategies have in common the fact that they use their look-ahead to find the endpoint which is a win (in TSD; in TSI, such an endpoint if it exists), and then the proper choice of moves is a matter of course.

Intuitively, the best of these look-ahead strategies would be the one which looks at the highest valued of the points open to look-ahead i.e., the points which are successors to the points already seen. Thus in Figure 2.1 you would first look at .6, then .5, then .4, .4, .3, .2, .1.

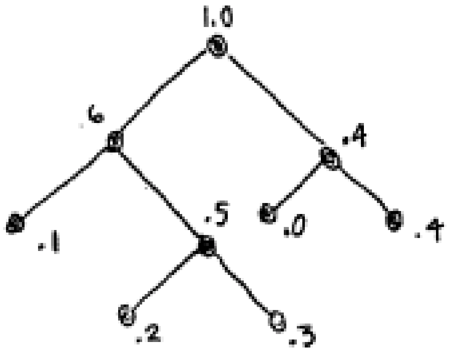


Figure 2.1: A sample game tree with sample values.

Papert[1] proved that this is indeed the best method.

Proof: First note that since any method of selecting what point to look at next is independent of whether that point is an E or not, over the class of all game trees the proportion of E's looked at to the total number of points looked at should be the same, that is, should be the number E. Thus to minimize the total number of positions looked at, over the class of all game trees, one need only minimize the number of endpoints looked at.

[1] In unpublished notes.

The expected number of endpoints to be looked at in order to find a win is, for any strategy in games of TSI:

$$A_1 \cdot 1 + (1-A_1)A_2 \cdot 2 + \ldots + (1-A_1)(1-A_2)\ldots(1-A_{n-1})A_n \cdot n$$

and for TSD

$$A_1 \cdot 1 + A_2 \cdot 2 + \ldots + A_n \cdot n$$

where A_i is the value of the i^{th} endpoint looked at, and n is the number of endpoints. The $(1-A_i)$ are cancelled in the formula for TSD because, if A_i is not a win, then the chance that A_{i+1} is a win is increased to $A_{i+1}/(1-A_i)$.

In both cases, the sum will be minimized if the endpoints are ordered so that $A_1 > A_2 > \ldots > A_n$, which can only be guaranteed by the strategy under consideration. Q.E.D.

We now return to strategies involving a fixed number of look-aheads. The present chapter will conclude with a description of the strategies to be evaluated in the next two chapters. However, first it will be noted that unlike the strategy just described, they are all what will be called "ordinal" strategies. An ordinal strategy is one which does not make use of the precise values of the two branches of a fork, but only notes which one is larger. Only such "ordinal" information is used in deciding where to look next, as well as choosing where to move next. The use of the precise values, either to choose the next move, or to choose where to look next, at first thought would seem quite valuable, however the actual advantage in the case of one or two look-aheads may be minute, and there is a corresponding increase in the amount of calculation needed and hence time used. These and other questions are discussed in chapter 5.

The strategies:

HSTRAT. One look-ahead. Look at the higher branch. If it is an S, move there. If it is an E and a win, move there. Otherwise move to the lower branch.

LSTRAT. One look-ahead. Look at the lower branch. If it is an E and a win, move there. Otherwise move to the higher branch.

2HSTRAT. Two look-aheads (or less). Look at the high branch. If it is an S, look at its high branch. If either of these two is a win, or if both are S's, move to the high branch. Otherwise, move to the low branch.

2LSTRAT. Two look-aheads (or less). Look at the low branch. If it is an S, look at its high branch. If either of these two is a win, move to the low branch. Otherwise move to high branch.

BSTRAT. Two look-aheads. Look at both branches. If either is a win, move there. Otherwise, move to the highest of the two which is an S. If both are losses, accept defeat with a smile.

These five exhaust the possibilities for one and two look-ahead ordinal strategies, in the sense that any other such strategy can be shown to be obviously inferior to at least one of these which agrees with it as to number of look-aheads, whereas the ranking of these is not at first obvious. The next two chapters try to discover this ranking. Although the results in the present chapter hold equally well for both TSD and TSI, the treatment is different when look-aheads are involved, and so one chapter is devoted to each type of game separately.

CHAPTER 3

ONE LOOK-AHEAD STRATEGIES IN THE DEPENDENT FORM OF TREE SOLITAIRE

The results in this and the following chapter will be proved for the restricted class of game trees with constant split ratio A, $.5 \le A \le 1.0$. It is a plausible conjecture that the results are *approximately* true over the class of all game trees, so long as A is the *expected value* of the split ratio.

In this chapter it will be proved that in TSD, LSTRAT is more effective than HSTRAT for all allowable values of A and S. Is this a surprising result? With unlimited look-ahead the least costly strategy that insures a win involves looking at the highest valued position first; with one look-ahead the most effective strategy is to look at the lowest. But this is no contradiction. Also, we must remember the whole strategy, not just the look-ahead rule. LSTRAT, by looking at the smaller branch, insures that if there is a win at *either* branch on this move, you will get it. If this is the virtue of being "aggressive", then HSTRAT has the advantage of "cautious,"for it assures you of not losing on the current move unless both choices are losses. It is difficult to see which factor outweighs the other, so we turn now to the actual evaluation of the strategies.

Now that look-aheads have been invoked, it seems unlikely that the probability of winning under a given strategy will be calculable in as simple a manner as was used in chapter 2. The use of a single expected value for the value of the position on a given move (at a given ply of the game tree) will lead into difficulty. With look-ahead strategies, the probability distribution of position values at a given ply is dependent on the probability distribution for the previous ply. This means that the expected value for the position value at one ply is not a direct function of the expected value for the previous ply, but of the entire distribution of position values. Thus we cannot, as was done for STRAT, have a single estimate of the position value at one ply, and use that to calculate our estimate for the next. Instead we must find a way to retain distribution information.

For games with a constant split ratio this is not too hard, and hopefully the probabilities of winning thus obtained will be good approximations to the probabilities for the class of all games whose expected split ratio is equal to that constant.

With constant split ratio, the number of possible position values at the n^{th} ply (with the top point of the game tree being ply 1) is precisely n. For example, with split ratio A and B = 1-A, the possible position value at the first ply is 1, at the second ply the values are A and B, at the third A^2, AB, and B^2, and so on. To evaluate the chance of winning for a given E and A under a particular strategy we sum, for each ply, the product of each possible position value times the probability that such a position is an endpoint times the probability, over the class of all game trees with split ratio A and ending probability E, that you could get to such a position at the given ply under the strategy. Then we sum the values thus obtained for the probabilities of winning at each ply over all plies from ply 2 on. (This is the same as summing the chances of winning on the first move, second move, etc.) The total is the probability of winning at all, under the given strategy over the class of all games with the particular split ratio and ending probability. The sum will converge, as it is clearly bounded by 1.

To aid in this calculation, it is convenient to have a diagram as in Figure 3.1 for HSTRAT. The diagram is only partially drawn here. However, it is in a sense recursive, and once one understands how it is generated, one does not need to see the rest. As such diagrams are much relied upon in this report, a detailed explanation of this one will be given here, so that the reader can generalize and thus understand later diagrams without much further explaining.

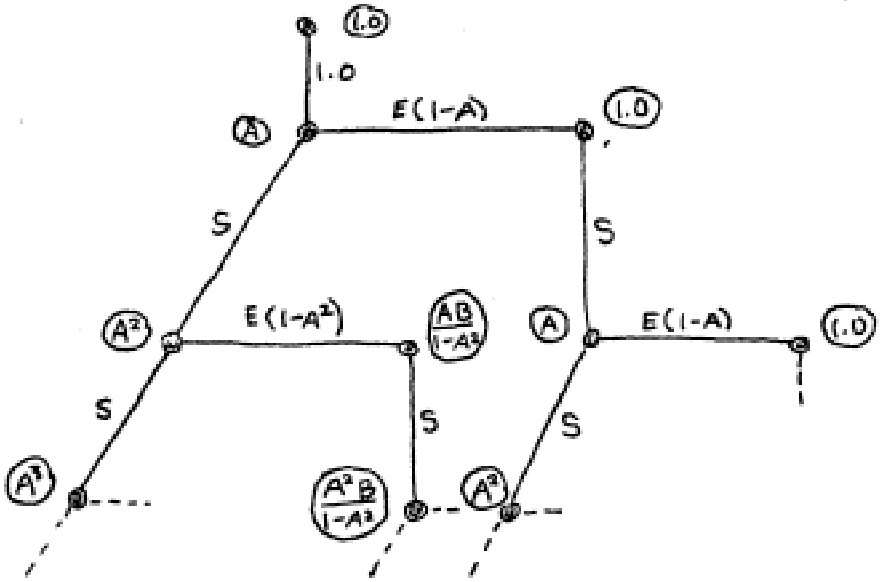


Figure 3.1: Evaluation Diagram for HSTRAT

The points and lines have numbers associated with them. The points represent possible positions you may be in or looking at, and the circled numbers are their values. Lines are directed from left to right or from top down, and the numbers associated with them are the probabilities of going from the starting point to the endpoint (in the case of HSTRAT, of going from *looking* at the high branch to *moving* to the low branch, or of going from the top point of a fork to *looking* at its high branch). The first horizontal row of points represents the first move (or *second* ply), and so on. The probability of being in a given position (i.e., at a given point of the diagram) on a given move (i.e., at a given ply of the game tree) is calculated recursively by associating to each point in the diagram a second number (its "probability") equal to the product of this number for the point's immediate predecessor, with the number on the line connecting the two. (In some later diagrams a point may have more than one such predecessor, in which case we take the sum of such products.) The first point in the tree has 1.00 for its probability.

This particular diagram is rather simple. For instance, on the first move the higher branch has value A. You move their unless it is a loss, the chance of which is E(1-A). If it is a loss, all other probabilities have to be divided by (1-A) because we're in the dependent form of the game, and so the lower branch, to which you move, now has value $\frac{(1-A)}{(1-A)} = 1$, and is so labeled. Essentially this same kind of process goes on at each move.

$A \cdot A^2)$

$A(1-A)$

$A^2)$

Now for the evaluation itself. The chance of winning on the first move is

$$EA + E(1-A)\cdot E\cdot 1 = AE + BE^2$$

where $B = (1-A)$. On the second move it is

$$SA^2E + S(1-A^2)E\cdot\frac{AB}{(1-A^2)} + (1-A)E^2SA + (1-A)^2E^3S$$

(remember $S = 1-E$), and so on. However, cancelling the $(1-A^2)$ and simplifying, the second term is

$$(SA + BES)(AE + BE^2)$$

or $(SA + BES)$ times the chance of winning on the first move. This turns out to be true in general, i.e., study of the diagram and the way the numbers in it are assigned shows that

LEMMA: The chance of winning on any given move (except the first) is $(SA + BES)$ times the chance of winning on the previous move

This means we have a geometric series as before, and so we have proved

THEOREM 3.1: Over the class of all games of TSD with constant split ratio A and ending probability E, the probability of winning under HSTRAT is

$$W_H = \frac{E(A+BE)}{1-S(A+BE)}$$

where $B = 1-A$, $S = 1-E$.

For example, if $E = 1/2$, $A = 3/4$, then $W_H = 7/9 = .7778$.

Similar good things happen for LSTRAT in TSD. The diagram is Figure 3.2, where the possible positions at each ply are located in narrow bands of not quite colinear points.

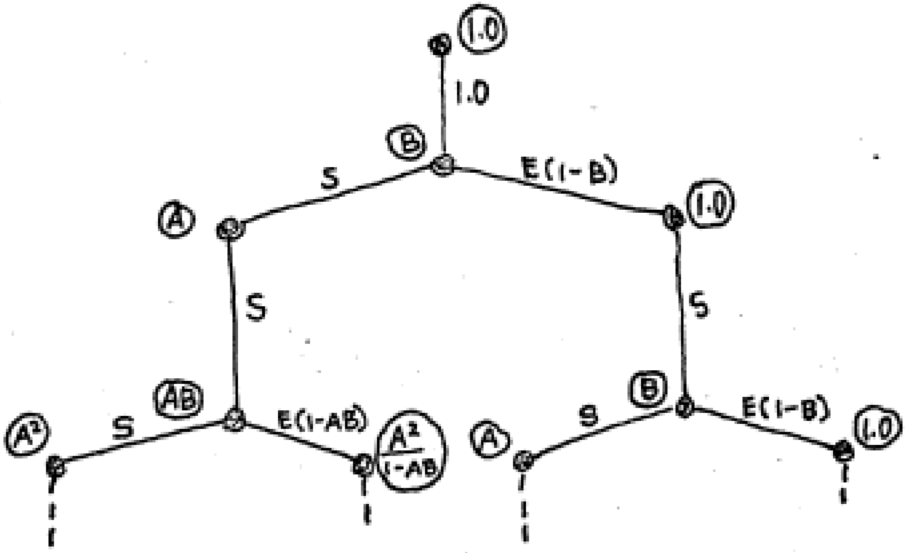


Figure 3.2: Evaluation diagram for LSTRAT

This diagram is explained as follows. The first thing you do is look at the lower valued branch of the initial fork, whose value is B. If it is a win, you move there. If it is a split point, which happens with probability S, you move to the higher valued branch, whose value is A. If it is a loss (probability $E(1-B)$), you still move to the higher branch, but now the probabilities of winning are the original values divided by $(1-B)$ so the value of the higher branch is $A/(1-B) = A/A = 1$. In a similar way the rest of the diagram may be generated.

The chance of winning on the first move is

$$EB + SEA + E(1-B)E \cdot 1 = EB + ESA + E^2A$$

On the next move, the chance of winning is

$$ES^3A^2 + ES^2AB + E^2S^2A^2 + ES^2A^2 + E^2SAB + E^3SA^2$$

or

$$(S^2A + ESA)(EB + ESA + E^2A)$$

The reader may verify for himself that again we have a ratio which holds from move to move throughout the game. The sum of the resulting geometric series is

$$\frac{EB+ESA+E^2A}{1-S^2A+ESA} = \frac{E(B+SA+(1-S)A)}{1-(1-E)SA-ESA} = \frac{E}{1-SA}$$

and we have shown

THEOREM 3.2: Over the class of all games of TSD with constant split ratio A and ending probability E, the probability of winning under LSTRAT is

$$W_L = E/(1-SA)$$

where $S = 1-E$.

If $E = 1/2$ and $A = 3/4$, then $W_L = 4/5 = .8000$.

Now we are at last prepared to prove

THEOREM 3.3: Over the class of all games of TSD with constant split ratio A and ending probability E, for any combination of values of E and A, $.5 \leq E < 1.$ and $.5 \leq A < 1.$, both LSTRAT and HSTRAT are more effective than STRAT (no look ahead), and LSTRAT is more effective than HSTRAT.

Proof: This result is obtained by simply comparing

$W = \frac{EA}{1-SA}$ (for STRAT), $W_H = \frac{E(A+BE)}{1-S(A+BE)}$, and $W_L = \frac{E}{1-SA}$

The first is clearly smaller than either of the second two, so all that need be shown is that $W_L > W_H$, which will be true if

$$\frac{E}{1-SA} - \frac{E(A+BE)}{1-S(A+BE)} > 0$$

That will be true if

$$1 - S(A+BE) - (A+BE)(1-SA) > 0$$

i.e., if $1 - SA - SBE - A - BE + SA^2 + BESA > 0$

i.e., if $1 - SA - SB(1-S) - A - B(1-S) + SA^2 + SAB(1-S) > 0$

i.e., if $1 - S(A+B) + S^2B - (A+B) + SB + SA(A+B) - S^2AB > 0$

i.e., if $1 - S + S^2B(1-A) - 1 + S(A+B) > 0$

i.e., if $S^2B^2 > 0$

But this last is obviously true for $.5 \leq E < 1.$ and $.5 \leq A < 1.$ Hence $W_L > W_H$ and LSTRAT is the more effective strategy. Q.E.D.

CHAPTER 4

STRATEGIES IN THE INDEPENDENT FORM OF TREE SOLITAIRE

The neatness of the formulas for W_L and W_H in TSD is due to an inherent cancellation which does not take place there for the two-look-ahead strategies, so the treatment of these has been postponed to this chapter on TSI. In the case of this class of games, no closed formula has yet been discovered for the probability of winning under any of the strategies involving look-ahead.

However, diagrams such as those in the preceding chapter can still be constructed, and suggest the ideas for recursive computer programs which can calculate the appropriate probabilities of winning (W_H, W_L, and by analogy, W_{2H}, W_{2L}, and W_B) move by move. This process, it turns out, need only be carried on for the first ten moves to get 5-place accuracy, although occasionally a small correction factor must be added to take care of the chance of winning after the 10th move.

A separate program has been written in MAD for each of the five strategies, in addition to one for two limiting cases which will be discussed below. Each program when given values for E and A outputs a sequence D(1), D(2),...,D(10) of numbers representing the probabilities of winning on the 1st, 2nd, etc. move. An output program then prints them and the ratio between succeeding terms, as well as their sum and any correction factor. The diagrams and programs are presented in Appendix A. With the exception of 2HSTRAT and 2LSTRAT, all these programs were checked against hand calculations. Those two were too complicated for hand calculation, but the internal workings of both programs were thoroughly gone over and found satisfactory.

In the case of the one look-ahead strategies, the conclusions of the preceding chapter seem to carry over to TSI, even if their proof does not. Table 4.1 give the calculated values for W_L, W_H (and W) for various combinations of E and A.

As can be seen, $W_L \geq W_H$ for all E and A given here, and both W_L and W_H are greater than W. For fixed E, W_H (and indeed W) approach W_L from below as A approaches 1. In the limiting case they would all be equal, as all the value of the top point of a fork would go to its high branch. Similarly, for fixed A, W_H (but not W) approaches W_L from below as E approaches 1, again with equality for the limiting case, because then both bottom points of the initial fork would be endpoints. Both these conclusions could have been made for TSD, because the numerator of the fraction representing $W_L - W_H$ was $(1-E)^2(1-A)^2$.

E	A	W	W_L	W_H
.5	.5	.33333	.55152	.48110
.5	.66667	.5	.63029	.59068
.5	.75	.6	.68768	.66102
.5	.9	.81818	.83914	.83268
.5	.95	.90478	.91095	.90899
.5	.99	.98020	.98049	.98039
.5	.999	.99800	.99804	.9980[illegible]
.5	.5	.33333	.55152	.48110
.75	.5	.42857	.66890	.64285
.9	.5	.47368	.72041	.71253
.99	.5	.49749	.74717	.74653
.25	.75	.42780	.31847	.46770
.5	.75	.6	.68768	.66102
.75	.75	.69231	.76722	.75745
.9	.75	.72973	.79698	.79371

Table 4.1: Computer results: evaluation of probability of winning under STRAT, LSTRAT, and HSTRAT over the class of games of TSI with constant split ratio.

The case in the table with $E < .5$ can be included because in TSI a game can be played without generating the entire game tree. All that need be generated are the positions that will actually be looked at or moved to under the given strategy. (This is how computer simulation of the game procedes in the Monte Carlo method of evaluation to be discussed below.) The expected length of a path in the tree is only $1/E$, so by such limited generation we can reduce our work to reasonable amounts, as long as E is not too small.

Actually, two factors disturb this estimate of game length when we are dealing with strategies involving look-ahead. Endpoints which are losses may be foreseen and avoided. On the other hand, endpoints which are wins may be foreseen and purposely sought. The results of actual simulation of LSTRAT with $E = .5$ and $E = .25$ show that playing a given amount of games for the smaller E takes only about 1.4 times as long as for the larger E. The factor for BSTRAT is about 2.3, reflecting perhaps the greater ability of that two-look-ahead strategy to avoid losing endpoints.

Since computer simulation of game playing has been introduced, it would perhaps be appropriate here to present the results of an attempt to estimate W_L and W_H in a special case, by Monte Carlo methods. E was set to .5, A to .75. Actual games were played under LSTRAT and HSTRAT, with the game tree being generated as needed with the aid of a MAD random number generator. 20,000 or more games were played to obtain the estimates. A "trial" was a series of 10,000 games. The variation between trials for any particular estimate was at most .00080, indicating considerable stability. The results are summarized in Table 4.2.

	Diagram Method	Constant Split Ratio		Variable Split Ratio	
		I	II	I	II
W_L	.68768	.67990	.69090	.69340	.69910
W_H	.66102	.65930	.66640	.67070	.65930

Table 4.2 Monte Carlo estimates of W_L and W_H for $E = .5$, $A = .75$.

This table, in addition to comparing the Monte Carlo estimates for constant split ratio with the value obtained by diagram method in Table 4.1, gives the estimates for W_L and W_H when the split ratio is <u>not</u> constant, but obeys a probability distribution such that every value between .5 and 1.0 is equally possible (and hence the expected value of the split ratio is .75). Two different value (I and II) for each split ratio alternative are given, because it was found that different utilizations of the random number generator gave rise to significantly different estimates. Under utilization I, low values of the random number caused wins. Under utilization II, high values caused wins.

The difference caused by this effect is not the same for both strategies, or for both split ratio possibilities. Under constant split ratio, the difference for W_L is .01100, for W_H .00710. Under variable split ratio, the difference for W_L is .00570, for W_H .01140. All these are considerably greater than the .00040 margin of error in the individual table entries, thus indicating that these differences are significant, and, although the table 4.1 values do lie between the I and II values for constant split ratio under both strategies, casting a doubt on the use of Monte Carlo methods for precise estimation of probabilities of winning, at least with the present random number generator.[1] However, such methods do, at least here, preserve the ranking of the strategies, and it can be noted that apparently, the given non-constant distribution of split ratios here does yield a greater probability of winning under both strategies than for constant split ratio, even though the expected value for the ratio in each case is the same. An example of a Monte Carlo

[1] Time pressures have prevented the devising of a random number generator which would be for our purposes less biased.

program is listed in Appendix B.

Returning to the diagram method of evaluating strategies, and the assumption of constant split ratio, it is interesting to note how the chances of winning on particular moves (i.e., at particular ply in the game tree) compare for HSTRAT and LSTRAT. Table 4.3 is a listing of D(1) through D(10), as defined above for the two, with E = .5 and A = .75

Move	LSTRAT	HSTRAT
1	.45313	.39063
2	.15253	.15733
3	.05282	.06507
4	.01867	.02735
5	.00670	.01164
6	.00243	.00500
7	.00089	.00216
8	.00033	.00094
9	.00012	.00041
10	.00004	.00018
SUM	.68765	.66088
CORRECTION FACTOR	.00003	.00014
W_L and W_H	.68768	.66102

Table 4.3: Chances of winning under HSTRAT and LSTRAT for the first 10 moves, with E = .5, A = .75.

The ratio between successive terms for LSTRAT begins at .33661 and increases to .37118 by the 10th move. For HSTRAT the ratios go up from .40328 to .43573. This leads to the fact that here LSTRAT's advantage is entirely contained in the first move (For other E and A, it is sometimes contained in the first two.) Then the cumulative effect of "cautiousness" begins to pay off. The fact that the chance of winning on the 6th move is twice as much under HSTRAT as under LSTRAT is due to the cumulative effect of having used that strategy on the first 5 moves, thus reducing the chance of winning early in the game, rather than to any inherent superiority of HSTRAT once you are that far along in the game and position values are small. This will be shown as a byproduct of the work in the next chapter.

Now turning to strategies with more than one look-ahead, we note that in a way 2HSTRAT is an extension of HSTRAT. One could easily continue this extension to a 3HSTRAT, etc., with each new strategy a slight improvement over the others. For instance, in Figure 4.1, 2HSTRAT would move to the A position, then notice the loss at A^3 and move to AB. However 3HSTRAT would see the loss right away and move to B, which is a greater value than AB.

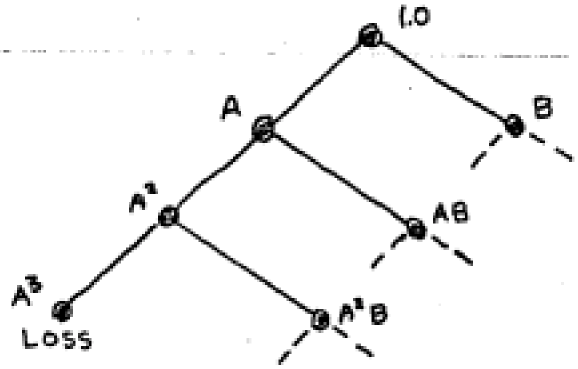


Figure 4.1: A possible game tree, labeled with the values of the points.

So we have the general concept of XHSTRAT, where X = 1,2,3... In the limit we obtain an unlimited look-ahead strategy ∞HSTRAT whose effectiveness serves as a limit for all such strategies. ∞HSTRAT looks down the high branch of a fork, and <u>its</u> high branch, if it splits, and then <u>its</u> high branch, etc., until an endpoint is reached. If it is a win, you move to the original high branch, if a loss, move to the low branch. (A very inefficient way to use unlimited look-ahead, even though it agrees with the one discussed in chapter 2 on the first 2 or 3 looks, depending on A. Also inefficient due to the repeating of previous look-aheads after each move, but this is a common fault of all XHSTRAT's).

Similarly, an ∞LSTRAT may be defined as a limit of XLSTRAT's. Table 4.4 gives the evaluations of these, and the already defined two look-ahead strategies, for various E and A.

E	A	W_B	W_{2L}	$W_{\infty L}$	W_{2H}	$W_{\infty H}$
.5	.5	.61335	.60013	.61583	.54025	.61583
.5	.66667	.66800	.66849	.68669	.61385	.64563
.5	.75	.71438	.71673	.73324	.67301	.68967
.5	.9	.84587	.84783	.85462	.83380	.83551
.5	.95	.91297	.91375	.91620	.90915	.90941
.5	.99	.98058	.98063	.98076	.98039	.98040
.25	.75	.54486	.56359	.61638	.47841	.52694
.5	.75	.71438	.71673	.73324	.67301	.68967
.75	.75	.78183	.78024	.78183	.76470	.76722

Table 4.4: Probabilities of winning under BSTRAT, 2LSTRAT, ∞LSTRAT, 2HSTRAT, and ∞HSTRAT, over the class of games of TSI with constant split ratio.

For $A = .5$, $W_{\infty L} = W_{\infty H}$ because ∞HSTRAT and ∞LSTRAT reduce to the same thing. W_{XL} would not equal W_{XH} for $A = .5$ because they would differ on where to move if the last point looked-ahead to were a split point.

A much more interesting table can be constructed by using Tables 4.1 and 4.4 to rank all the strategies as to effectiveness:

E	A	1	2	3	4	5	6	7
.5	.5	∞L--tie--∞H		B	2L	L	2H	H
.5	.66667	∞L	B	2L	∞H	L	2H	H
.5	.75	∞L	2L	B	∞H	L	2H	H
.5	.9	∞L	2L	B	L	∞H	2H	H
.5	.95	∞L	2L	B	L	∞H	2H	H
.5	.99	∞L	2L	B	L	∞H	2H--"tie"--H	
.25	.75	∞L	2L	B	∞H	L	2H	H
.5	.75	∞L	2L	B	∞H	L	2H	H
.75	.75	∞L	B	2L	∞H--"tie"--L		2H	H

Table 4.4: Ranking of strategies as to probability of winning over the class of games of TSI with constant split ratio.

Table 4.5 speaks for itself. It bears out the conclusion that $W_{XL} \geq W_{XH}$. However it also shows that it is not true that in all cases XLSTRAT is the best X lookahead strategy: For E = .5 and A = .5 and A = .66667

$W_B > W_{2L}$. The wastefulness of the XHSTRAT's "caution" is seen from the fact that one look-ahead LSTRAT is better than 2HSTRAT, and even, in certain cases, better than ∞HSTRAT.

However, rigorous mathematical justifications for such conclusions are yet to be discovered. One of the purposes of the next chapter is to get a start in that direction.

As one last way of looking at the results of this chapter, let us note exactly how much the use of look-ahead in ordinal look-ahead strategies improves the chance of winning. Table 4.5 gives for various combinations of values for E and A the ratios of the probabilities of winning under the most effective one and two look-ahead strategies, to the probability of winning with no look-ahead (under STRAT).

E	A	W	one look-ahead ratio	two look-aheads ratio
.5	.5	.33333	1.67	1.86
.5	.66667	.5	1.26	1.33
.5	.75	.6	1.15	1.19
.5	.9	.81818	1.02	1.035
.5	.95	.90478	1.005	1.01
.5	.99	.98020	1.000	1.000
.25	.75	.42780	1.21	1.32
.5	.75	.6	1.15	1.19
.75	.75	.69231	1.11	1.13

Table 4.5 Improvement in chance of winning due to look-aheads used in ordinal strategies over the class of games with constant split ratio.

CHAPTER 5

DEPRECIATION FACTORS AND NON-ORDINAL STRATEGIES

So far in this report, strategies have been evaluated as to their effect on the whole game. In this chapter we look at them in their effect from any given position, and try to do this without detailed projections of what will happen later on, thus perhaps enabling us to decide which look-ahead strategy to use, move by move.

To this end we introduce the concept of "depreciation factors" to aid in our evaluation of the relative contribution of "caution" and "aggressiveness". For the time being, we will be concerned with the class of TSI games, with constant split ratio A, and in addition will concentrate at first on the one look-ahead strategies for simplicity.

Suppose we try to evaluate the probability of winning under LSTRAT from a position which is a split point of value X, about which the only other information available is the value of its branches, which here will be assumed to be determined by the constant split ratio A. The probability of winning is clearly less than X, because we can under the given strategy look only at a limited portion of the game tree below the point, and only <u>move</u> down a single path. This has already been seen to be true for all the limited look-ahead strategies so far considered for the case where $X = 1$ and we are at the beginning of the game. However, suppose we do take X to be the probability of winning in such a case. We might improve our estimate by adding the actual chance of winning under LSTRAT on the first move from the given position, $EBS + (1-EBX)EAX$, to the chance of moving to a split point, $(1-EBX)S$, times the value of that point, AX, which is being used as the estimate of the chance of winning from this second split point. The resulting formula, call it V_L, is

$$V_L = EBX + (1-EBX)EAX + (1-EBX)S(AX)$$

$$= EBX + (1-EBX)(AX) = EBX + AX - EABX^2$$

Evaluating HSTRAT in the same way, we get

$$V_H = EAX + SAX + E(1-AX)(EBX+SBX)$$

$$= AX + E(1-AX)BX = EBX + AX + EABX^2$$

Our over-estimation of the chance of winning from a split point has given rise to the interesting result that $V_L = V_H$. However, rather than puzzle over the exact reason it works out this way, let us try to correct that over-estimation. It would be very convenient if the actual chance of winning from a split point was a constant, depending only on the strategy being used, times the value of the point. Such a constant could be called a "depreciation factor". However, it is really too much to assume (in TSI) that such a constant exists, independent of the value of the position.

What is actually needed is probably a "depreciation function" of E, A, and X. However, we can get an estimate of this function by assuming that it <u>is</u> constant, at least constant from one move to the next. Then letting K stand for the constant, KX would be equal to the estimate of probability of winning which is calculated analogously to the calculations of V_L and V_H.

$$KX = EBX + (1-EBX)[EAX + SKAX]$$

Solving for K:

$$K_L = \frac{EBX + EXA - E^2ABX^2}{X + ESABX^2 - SAX} = \frac{E(1-EABX)}{1 - SA + ESABX}$$

As X approaches 0, K_L (actually a function of E, A, and X) increases to a limit of $E/(1-SA)$

For HSTRAT, we would have

$$KX = EAX + SKAX + E(1-AX)[EBX + SKBX]$$

$$K_H = \frac{EAX + E^2BX - E^2ABX^2}{X - SAX - ESBX + ESABX^2} = \frac{E(A+EB) - E^2ABX}{1 - S(A+EB) + ESABX}$$

As X approaches O, K_H increases to a limit of $\frac{E(A+EB)}{1-S(A+EB)}$

Interestingly enough, the limiting values for K_H and K_L are W_H and W_L for TSD (Chapter 3). *This* surprise has a good explanation, however. As X approaches O, the independent game TSI get closer to the dependent TSD (the $(1-X)$ factors you divide by when you discover losing endpoints, approach 1.) And in TSD W_L and W_H *are* the depreciation factors, which in fact *are* constant throughout the game, as can be seen if one goes over the analysis in chapter 3. Because of this constancy, they give the actual probability of winning when multiplied by the value of the position. The top position has value 1, so the chance of winning the game is simply the depreciation factor itself.

Now, if for given E and A we superscript the K's to let $K_L^\infty = \mathop{\text{limit}}\limits_{x \to o} K_L^X$, we have, from Theorem 3.3

$$\frac{E}{1-SA} = K_L^\infty \geq K_H^\infty = \frac{E(A+EB)}{1 - S(A+EB)}$$

It can be concluded from this that

$$K_L^X = \frac{E - E^2ABX}{1 - SA + ESABX} \geq \frac{E(A+EB) - E^2ABX}{1 - S(A+EB) + ESABX} = K_H^X$$

because exactly the same terms are subtracted from each numerator, and exactly the same terms are added to each denominator.

Now, since in TSI K_L^X is not constant for all position values X, $X K_L^X$ does not give a precise estimate of the probability of winning from position X under LSTRAT, in fact the real probability will be somewhere between $X K_L^X$ and $X K_L^\infty$. It is intuitively plausible from the above inequalities however, that the chance of winning from X under LSTRAT will be greater than that under HSTRAT, no matter what value X. This is in support of the claim made in the last chapter that the fact that under HSTRAT there is a greater chance of winning on the 6th move than under LSTRAT is only indicative of the cumulative effect, and not of any real superiority of HSTRAT at that point in the game.

We can further explain the kind of results summarized last chapter in table 4.2 as follows. Suppose you are at a position of value X small enough so that the limiting formulas K_L^∞ and K_H^∞ approximately hold. Then the probabilities of winning are, respectively

$$\frac{XE}{1 - SA} \qquad \text{and} \qquad \frac{XE(A+EB)}{1 - S(A+EB)}$$

Under LSTRAT the chance of winning on the next move, if X is small enough, is approximately $EAX + EBX = EX$, so the formula can be interpreted as representing the sum of the chances of winning on each successive move, the first term being XE and the ratio between succeeding terms being SA. (This was the way we got the formula in the case of TSD in chapter 3). Similarly, for HSTRAT the first term is XE(A+EB) and the ratio S(A+EB). The ratio for HSTRAT is greater than that for LSTRAT, but this is outweighed by the preponderence of the first term in the LSTRAT series.

If this analysis is correct, the ratio D(I)/D(I+1) described in chapter 4 should approach the ratios described in the last paragraph. This is shown in table 5.1 below

E	A	HSTRAT: $\frac{D(9)}{D(10)}$	S(A+EB)	LSTRAT: $\frac{D(9)}{D(10)}$	SA
.5	.5	.37460	.37500	.24982	.25000
.5	.66667	.41615	.41667	.33165	.33333
.5	.75	.43573	.43750	.37118	.37500
.5	.9	.46774	.47500	.44118	.45000
.75	.5	.21867	.21875	.12486	.12500
.75	.75	.23341	.23437	.18464	.18750

Table 5.1: Comparison between actual ratios between chances of winning on 9th and 10th moves with theoretical limits for HSTRAT and LSTRAT in games of TSI with constant split ratio.

With the introduction of depreciation factors, it becomes possible to discuss intelligently non-ordinal strategies which make use of the actual values of the branches of a fork, not just their relative ranking. To do this we first drop the assumption that the split ratio is constant, and then add the assumption that the depreciation functions just derived for constant split ratio A also hold approximately for a distribution of split ratios merely with expected value A.

The advantage of non-ordinal strategies, if there is any, will come from the possibility that, over the class of game trees, the possibility of winning from a split point whose branches' values are known may vary a little depending on how the top value is split between its successors. Thus, given two positions which are split points and whose split ratios are known, it may be possible that even though position X has value higher than position Y, the chance of winning from position X is lower, because its value splits in an unfavorable way.

The ordinal strategies to which such considerations might apply are HSTRAT, 2HSTRAT, and BSTRAT, for in all these we ordinarily can *move* to a split point whose split ratio we have already seen in look-ahead. The use of this information to modify the strategy in the cases of BSTRAT and HSTRAT will be shown. The part of the strategies which tells where to look will remain the same, but the rules about where to move, given the results of the look-ahead, will be altered. There is also the possibility of strategies with more than one look-ahead, where the results of the first look-aheads (the exact values of the branches seen) may be used to determine where to look next. Such a strategy is the unlimited look-ahead strategy discussed in chapter 2, where that position with the highest value of all those you have not yet seen, but know the value of, is looked at next. However, this is the only example of such non-ordinal strategies studied so far.

One unfortunate aspect of the non-ordinal strategies we do study is that they do not lend themselves to simple evaluation techniques. This is because an expected value of the split ratio, like the A we used for ordinal strategies, is useless when we want to utilize the fact that the split ratio varies according to a probability distribution. Evaluation will have to be done by computer using the Monte Carlo methods whose reliability was brought into question in the last chapter. However if we make consistent use of the random number generator, we should be able to get an estimate of the amount of improvement non-ordinal information allows us. (and also its cost in computer time).

So, to define these non-ordinal modifications: Under BSTRAT the depreciation factor is found as before

$$KX = EAX + (1-EAX)EBX + E(1-AX)SKBX + (1-EBX)SKAX$$

$$K_B^X = \frac{EX-E^2ABX^2}{X-ESBX+ESABX^2-SAX+ESABX^2} = \frac{E-E^2ABX}{1-S(A+BE)+2ESABX}$$

$\lim_{X\to 0} K_B^X = K_B^\infty = E/(1-S(A+BE))$, which combines the best of both HSTRAT and LSTRAT -- the high chance of winning on the move, and the high ratio between such chances on succeeding moves. But this was to be expected, at least in the limit.

By using the depreciation "factor", we can get an idea of the probability of winning from the top of a fork whose split ratio is known. Define

$$K_B(E,A,X) = \frac{(E-E^2ABX)X}{1 - S(A+BE) + 2ESABX}$$

as a depreciation "function" giving the depreciated value of a position with value X. Then let our known fork have top value X and split ratio R. The chance of winning will be approximated by

$$C_B(E,A,X,R) = ERX + (1-ERX)E(1-R)X + E(1-RX)S\ K_B(E,A,(1-R)X)$$
$$+ (1-E(1-R)X)S\ K_B(E,A,RX)$$

(Note that we are still using A as the expected value of the split ratios we don't know about yet.) BSTRAT would be modified only in the case when both high and low branches of your position have been found to be split points. Denote the value of the low branch by L, and the high branch by H, and suppose that their split ratios are R_H and R_L. We then compare $C_B(E,A,H,R_H)$ and $C_B(E,A,L,R_L)$ and move to the branch with the highest value for this. In all probability this would be the H branch anyway, but sample computer calculations show that if $H/(H+L) \leq .55$ there is at least a possibility, correspondingly greater as the ratio approaches .50, that the L branch may be chosen.

Monte Carlo methods, implemented as described in chapter 4, yielded the following evaluation of BSTRAT and its modified version, as to chance of winning. In 10,000 games, BSTRAT won .72435 of the time, with a variation over repeated trials of only .00010. The modified version B'STRAT won .73200, with a variation of .00030 over repeated trials. This is an improvement of approximately .00765. This is a considerable amount when you consider that, in Table 4.3 for $E = .5$, $A = .75$, $W_{2L} - W_B$ is only .00235, and so this modification probably makes BSTRAT the most effective two look-ahead strategy we have. This is born out by a Monte Carlo evaluation of 2LSTRAT which yielded an estimate of W_{2L} at .72810 ± .00020, clearly less than the estimate for W_B.

However B'STRAT, though more effective, is more costly than BSTRAT in the sense that it took slightly more than twice as much time to play 10,000 games, due to the extra calculations involved in B'STRAT. This should probably not be taken to seriously, since in most real games the time involved in such calculations would be much more seriously over-weighed by the time needed to simply generate the position values, and anyway, we could have skipped them except when $H/(L+H) \leq .55$, which is the only time they could be useful.

Note that this non ordinal strategy is quite sophisticated in comparison to what might have been suggested without the aid of depreciation factors. For instance, confronted with a situation such as in Figure 5.1, one might think to go to the lower branch because <u>its</u> high branch has the highest value of all points at its level.

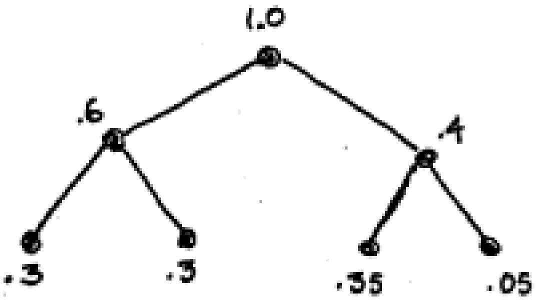


Figure 5.1: Example of part of a game tree for T3I

However, since .6 (.4+.6) > .55, this would probably be wrong assuming that our previous calculations are indeed applicable, and applying such a simple rule over the long run would surely be contra-productive.

Turning to HSTRAT, again we must pass up simple non-ordinal modifications, for instance: after looking ahead to the .6 point, deciding to go to the .4 because .4 is greater than .3. Instead, using the depreciation factor derived above, we define

$$K_H(E,A,X) = \frac{X[E(A+EB) - E^2ABX]}{1 - S(A+EB) + ESABX}$$

and, with R and T as alone

$$C_H(E,A,X,R) = EXR + SK_H(E,A,XR) + E(1-XR)[EX(1-R)+SK_H(E,A,X(1-R))]$$

Suppose we are at a fork whose bottom points are again H and L, and we find by look-ahead that the H branch splits, with split ratio R. Then we can compare $C_H(E,A,H,R)$ to $EL + SK_H(E,A,L)$ and move to the H branch only if the first value is the greater. This modification was evaluated by the Monte Carlo methods already described, with $SK_H(E,A,L)$ replaced by $SC_H(E,A,L,A)$, which <u>should</u> be improvement, although no proof has been found. The resulting fraction of wins was .67265 compared to .67070 as was obtained before for HSTRAT by this method, an increase of about .00200, and nowhere near enough to make it better than LSTRAT, which has an advantage of about .02650 over HSTRAT. The cost was again a doubling of computer time for 10,000 games, but, likewise, again would have been reduced if in the cases where $H/(L+H)$ was too great, the calculations of C had been omitted. Under this strategy the cut-off point would be about .58 for large (in absolute terms) values of H, decreasing to perhaps .56 for $H < .006$. These cut off points were determined by computer evaluations of C for various H and L, and why it should be what it is, and why it should depend on the value of H itself, are questions open to further research.

The results of this research into non-ordinal strategies is summarized in Table 5.2 below

Strategy	Fraction of wins	Fraction of wins when modified	Difference
BSTRAT	.72435	.73200	.00765
HSTRAT	.67070	.67265	.00195

Table 5.1: Comparison of Monte-Carlo Results for Ordinal and Non-Ordinal Strategies in Games of TSI

CHAPTER 6

VARIANTS ON TREE SOLITAIRE
AS A SUGGESTION FOR FURTHER STUDY

The obvious first suggestion for extending the results so far reported would be to study in more detail the ordinal strategies in Tree Solitaire with more than two look-aheads, or perhaps strategies which are not limited as to number of look-aheads but as to the number of ply deep in the game tree they may look.

Going beyond the specific limitation to the rules of Tree Solitaire, the obvious second suggestion for extending the results would be to stop the restriction to binary game trees, and allowing for the possibility of more than two moves from a given position. We could consider the case of ternary, 4-ary, etc. trees, or we could even let the number of branches at a given position vary according to some probability distribution. A kind of tree solitaire could be defined on such trees, in our strategy definitions "low" could be replaced by "second highest", and evaluation could procede in perhaps much the same way.

However, rather than leap to such attempts in this last chapter, we shall continue our limitation to the binary trees for which our terminology has been developed, and consider some variants of tree solitaire preserving this characteristic.

The principle failing of Tree Solitaire as a model for the kind of games involving look-ahead which we usually encounter, appears to be the fact that the values of the positions naturally decrease as you get further on in the game. This is the main reason we had to have "depreciation" factors and the probable explanation of why aggressive LSTRAT did better than cautious HSTRAT. How can we correct this failing?

There are two kinds of modification, which can be effected independently. One is to change the game tree and its generation, the other to change the "valuation" of the tree, the method of assigning values to its points. We would like to keep the basic approach of generating the tree first, independently of the values which may later be assigned to it, although these values may themselves reflect tree structure as is often the case in real games and our evaluations of positions in them.

An example of modifying the tree would be to have E, the probability that a point is an endpoint, be a function of its level in the tree (how many moves deep in the game it is). This would probably not complicate matters much, and indeed, the diagram-computer program methods of chapter 4 might still be useable. In addition the essential feature of each point being judged as E or an S independently of all the other points is retained. However, by itself, this modification does not solve the problem referred to above, and this chapter will now further restrict itself to only those modifications which alter the valuation of the game tree, leaving this suggestion as just that -- a suggestion.

Three possible modified games will be discussed, in ascending order of usefulness.

1. Given some probability distribution of numbers between 0 and 1, assign values to the endpoints of the (finite) tree. The value here represents the probability that the endpoint is a win. For every split point of the tree, let its value be calculated from the values of the set of endpoints which are its successors, immediately or farther down in the tree, and let it be the probability that at least one of these is a win, i.e., the chance that there is a win beneath that split point in the tree. If the set of successor endpoints has 3 numbers, with values A, B, and C, then the value of the point would be $A + B + C - AB - AC - BC + ABC$.

This does not solve the problem raised above, for the values of points decrease as you go down the tree (although not indefinitely, as they can in TS), but it has one interesting facet. Consider the problem raised in chapter 2 of finding the least costly unlimited look-ahead strategy which guarantees a win (if there are any wins in the game). The answer there for Tree Solitaire was to look first at the point with the highest value. In Game 1, if the distribution of endpoint values is such that the values are all equal to some constant A, the least costly strategy, in terms of number of positions you need to look at, is to look at the lowest valued point available.

Proof: From the value of a split point, we can, knowing A, calculate the number of endpoints beneath it. The lower the value the fewer the endpoints. However, the lower the number of endpoints beneath a point in the tree, the higher the proportion of such endpoints to the total number of points beneath that point. (If K is the number of endpoints, there are K-1 split points, counting the top one. K/2K-1 decreases as K increases). Since all endpoints have equal probability of being a win, the least costly look-ahead procedure will be the one which wastes the least time in getting to look at endpoints. This is accomplished by looking at the lowest point, which can be expected, by the above consideration, to be the closest one to an endpoint. Q.E.D.

The reason this works is that the value of a point is not independent of the structure underneath it in the tree, is in fact a function of the number of endpoints. Even if the values of the endpoints were not equal but were assigned according to a probability distribution as explained above, there would still be a dependence, from which some information about the tree structure could be obtained. Such an arrangement is probably not at all unreasonable since such information is probably available in the valuations used in actual games, however, it is an example of the non standard coding phenomena and complicates matters unduly (and arbitrarily) in a theoretic treatment even though it is valuable in giving us an idea of how our expectations may go wrong in such situations. Here the best limited look-ahead strategy would probably have to make use of such information, and the evaluation of <u>any</u> strategy would have to take into account the dependency. An ability to count binary trees with a given number of endpoints might be needed, and there is no easy closed formula for that.

2. Similar problems arise in this second game, due again to the fact that values are assigned from the bottom of the tree up rather than from the top down, and thus can reflect tree structure.

Assign values to the endpoints as in game 1. However this time let each split point have value equal to the average of the values of its immediate successors. We now have avoided the problem of having naturally decreasing position values. However, again problems in evaluating strategies will arise because the value of a position does tell us a little about the tree structure beneath it. Although the expected value of the value of a split point is the same as that of an endpoint, its variance, because it is an average, will be smaller, as a function of the number of end-points beneath it. Therefore a point with a value very far from the mean is likely to have only a few endpoints beneath it, or be one itself.

This game has been mentioned not so much for its own importance, but because it suggests an idea for a game where values for positions are independent of the tree structure beneath them, and yet are "likely" to be the average of their successors.

3. Decide on a probability distribution $\mathfrak{F}$ with mean 0. To the top of the tree assign a number T between 0 and 1. Given a fork in the tree whose top value C has already been assigned, generate independently two numbers A and B according to the distribution. Then the values of the branches are C + A and C + B (i.e., the values of the branches are chosen according to the probability distribution $\mathfrak{F}$ shifted over so that C is its mean). If a point is an endpoint with value P, the probability of its being a win is P if

$0 \leq P \leq 1$, 1 if $P \geq 1$, and 0 if $P \leq 0$.

The valuation in this game is an idealization of the way which computer position evaluation functions might be thought to work with regard to successive positions, and in that sense this game should be a much better approximations to the real games game playing programs are written to deal with. With this hope, the rest of this chapter is devoted to showing how the ideas and techniques of the earlier chapters can be used as a start on the evaluation of strategies in games of form 3.

To construct a diagram that might be appropriate, we note that although the expected value of either branch of a fork is the value C of the top point, the expected value of the higher branch is somewhat greater than C and of the lower branch, somewhat less. Let us suppose that $\mathcal{F}$ is symmetric about its mean; then the expected value of the branches would be C + A and C - A, where A is determined from $\mathcal{F}$. This yields a diagram as in Figure 6.1 based on the assumption that the expected position values are always realized. The values on the lines can be filled in according to the strategy you are considering, and the computer programs of chapter 4 could easily be modified to evaluate all the strategies that were evaluated for TS.

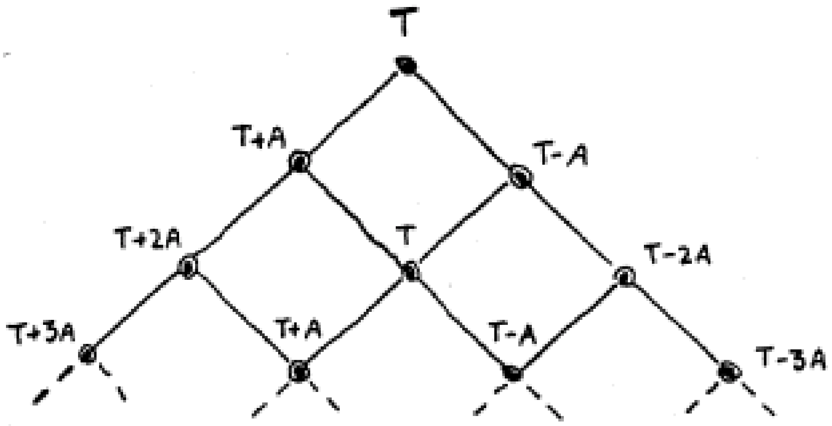


Figure 6.1: Expected position values and possible moves in Game 3.

However, the reasonableness of the estimates of probability of winning thus obtained would depend on how the actual distribution 3 scattered its values and what effect this might have. As in the case of Tree Solitaire, it's hard to tell whether using the expected values of position values to construct a diagram leaves out any essential considerations, and here we cannot fall back on the restriction to constant split ratio. The effect could be checked by Monte Carlo Methods.

The results of sample hand calculations for $T = 1/2$, $A = 1/6$ are summarized in Table 6.1, where after move 3 the formulas were too complicated to evaluate.

Move	Probability of winning HSTRAT	 LSTRAT
1	.361	.444
2	.242	.234
3	.149	.104
SUM	.752	.782

Table 6.1: Hand calculations of chances of winning under HSTRAT and LSTRAT over the class of games of form 3.

It is apparent that even though LSTRAT is ahead at the end of 3 moves, HSTRAT will predominate when the sum is taken over, say, 10 moves, by which time the game should be over.

However, a more exact estimate can be obtained by calculating the probability of losing, for that sum turns out to converge quite rapidly. See Table 6.2

Move	HSTRAT	Probability of Losing LSTRAT	STRAT
1	.0556	.139	.167
2	.0191	.035	.042
3	.0069	.000	.000
SUM	.0816	.174	.209
Sum + correction factor	.0850	.174	.174
1 - SUM = Probability of winning	.915	.826	.791

Table 6.2: Hand calculation of chances of losing under HSTRAT LSTRAT, and STRAT over the class of games of form 3

STRAT was thrown in for good measure to show the improvement due to look-ahead over the best one can do without any. The third term for LSTRAT and STRAT is 0 because with A and E as they are, from the 3rd move on under these strategies you can go only to positions with value greater than or equal to 1. The conclusion one draws from this table is that HSTRAT is the best one-look-ahead strategy. It is reasonable to think that this generalizes to all allowable values of A and E. Now that values of points no longer need decrease later on in the game there is a premium on "caution" and "aggression" becomes "recklessness".

Although a proof of HSTRAT's general superiority is not yet known, it can be shown for the limiting case where $A = 0$, i.e. all positions have the same value T. (This is the same as game 2 with all endpoints assigned the same value.)

THEOREM 6.1: Let W_H represent the chance of winning under HSTRAT, and W_L the chance under LSTRAT, over the class of all games of form 3, in which every point of the game tree has the same value T, $0 \leq T < 1$. Then $W_H > W_L$ for all E (the probability that a point is an endpoint) $0 \leq E < 1$.

Proof: Under HSTRAT the chance of winning at <u>any</u> particular position is

$$ET + E(1-T)ET = ET(1 + E - ET)$$

The chance of continuing on in the game is

$$S + E(1-T)S = S(1 + E - ET)$$

So, $$W_H = \frac{ET(1+E-ET)}{L-S(1+E-ET)} = \frac{ET(1+E-ET)}{1-(1-E)(1+E-ET)} = \frac{T(1+E-ET)}{T-TE+E}$$

For LSTRAT the chance of winning at <u>any</u> position is

$$ET + (1-ET)ET$$

and the chance of continuing in the game is

$$(1 - ET)S = 1 - E - ET + E^2T$$

So, $$W_L = \frac{ET(2-ET)}{E(1+T-ET)} = \frac{T(2-ET)}{(1+T-ET)}$$

$$W_H - W_L = \frac{T(1+E-ET)}{T-TE+E} - \frac{T(2-ET)}{1+T-ET}$$

This will be > 0 if

$$(1+E-ET)(1+T-ET) - (2-ET)(T+E-ET) > 0$$

This can be simplified to

$$1+T-ET+E+ET-E^2T-ET-ET^2+E^2T^2-2T-2E+2ET+ET^2+E^2T-E^2T^2$$

$$= 1 - T - E + TE = (1-E)(1-T) > 0$$

(because of the bounds on E and T). Q.E.D.

For A greater than 0 and assuming the game works like diagram 6.1, it can be shown that both the chance of winning on the move and the chance of continuing past it are improved, for both HSTRAT and ISTRAT, but it is not immediately apparent which gains the most. It is at least plausible to believe that HSTRAT maintains its superiority. Perhaps something like depreciation factors will be inventable (although here they might be "appreciation factors") and can be used to solve this problem, and then to go on to propose non-ordinal strategies.

At any rate, the avenues for further study are wide open, and hold prospects of more and varied results.

APPENDIX A: Programs for evaluating strategies in chapter 4

CALLER: This is the calling program for the rest of the programs, on CTSS.

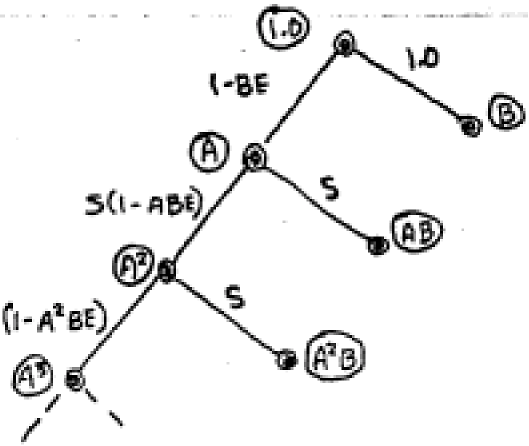
OUTPUT: This MAD external function is used by all the following strategy functions to sum and print their their results: the values D(I) for I = 1 through 10 representing the chances of winning on each of the first 10 moves, and the ratio between succeeding terms. The sum of the D(I) is increased by a correction factor arrived at by assuming that the ratio between successive terms. from D(9) on, is a constant, CORR = D(10)/(1.-D(9)/D(10)) - D(10)

In the following MAD external functions which calculate D(I) for the various strategies discussed in chapter 4 when presented with values for E and A, V(I) represents the value of the i^{th} point from the left in the corresponding diagram, at the level under consideration. T(I) represents the probability of getting to a position with such a value at that level. All the programs start and end with the same statements, which are

```
EXTERNAL FUNCTION
PROGRAM COMMON V,T.D,A,B,S,E,
DIMENSION  V(10),D(10),T(10)
INTEGER J,K,L,I1,I2,I3,I4
  .
  .
  .
  .
FUNCTION RETURN
END OF FUNCTION
```

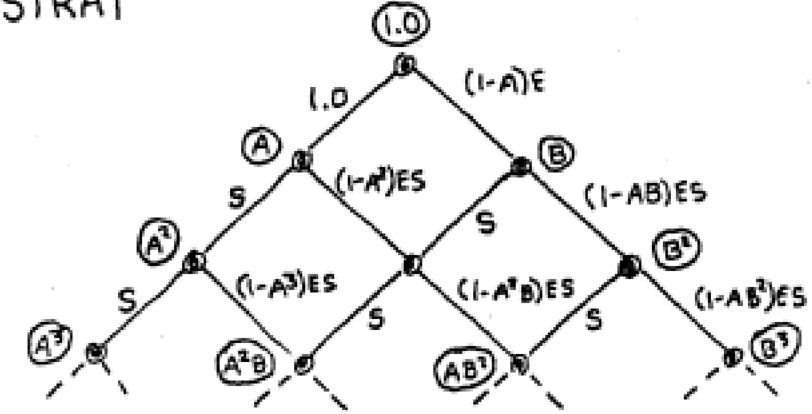
and so only the differing parts will be listed. For each strategy the diagram will be presented, comments if necessary, and then the program listing.

LSTRAT



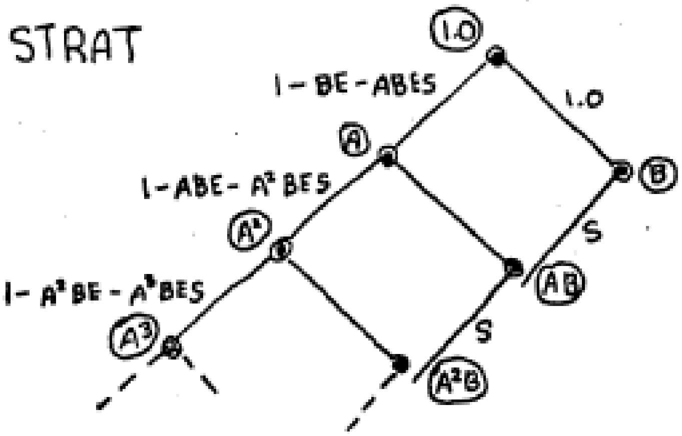
```
     ENTRY TO LSTRAT.
     V(1) = E*A
     V(2) = E*B
     T(1) = 1.-V(2)
     D(1) = V(1)*T(1)+V(2)
     THROUGH M1, FOR J=2,1,J.G.10
     V(2) = V(1)*B
     V(1) = V(1)*A
     T(2) = T(1)*S
     T(1) = T(1)*S*(1.-V(2))
M1   D(J) = T(1)*V(1)+T(2)*V(2)
     PRINT COMMENT $  LSTRAT$
     EXECUTE OUTPUT.
```

HSTRAT



```
    ENTRY TO HSTRAT.
    V(0) = 0.
    V(1) = A
    V(2) = B
    T(0) = 0.
    T(1) = 1.
    T(2) = E*(1.-A)
    D(1) = E*(T(1)*V(1)+T(2)*V(2))
    THROUGH M1, FOR J = 2,1,J.G.10
    V(J+1) = V(J)*B
    THROUGH M2, FOR L = J,-1,L.LE.0
M2  V(L) = V(L)*A
    T(J+1) = E*S*(1.-V(J))*T(J)
    THROUGH M3, FOR L = J,-1,L.LE.0
M3  T(L) = S*T(L)+E*S*(1.-V(L-1))*T(L-1)
    D(J) = 0.
    THROUGH M1, FOR L = 1,1,L.G.J+1
M1  D(J) = D(J)+V(L)*T(L)*E
    PRINT COMMENT $ HSTRAT$
    EXECUTE OUTPUT.
```

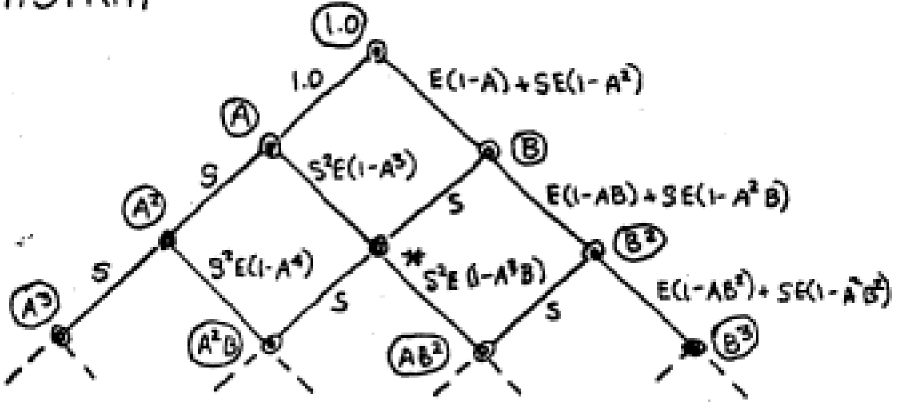
2LSTRAT



The lines are not connected at the AB and A^2B points to emphasize the fact that in TSI, under this strategy, you would not move from B to AB and then to A^2B etc. You move B to AB only when you have seen by previous look-ahead that AB is a win. Otherwise the program is quite similar to that for LSTRAT

```
           ENTRY TO LSTRAT
           V(1) = E*A
           V(2) = E*B
           T(1) = 1.-E*B*(1.-S*A)
           T(2) = 1.
           T(3) = 1.
           D(1) = T(1)*V(1)+T(2)*V(2)
           THROUGH M1, FOR J = 2,1,J.G.10
           V(2) = V(1)*B
           V(1) = V(1)*A
           T(2) = T(1)*S+T(3)*S
           T(3) = T(1)*S
           T(1) = T(1)*S*(1.-V(2)-V(2)*S*A)
     M1    D(J) = T(1)*V(1)+T(2)*V(2)
           PRINT COMMENT $ 2LSTRAT$
           EXECUTE OUTPUT.
```

2HSTRAT

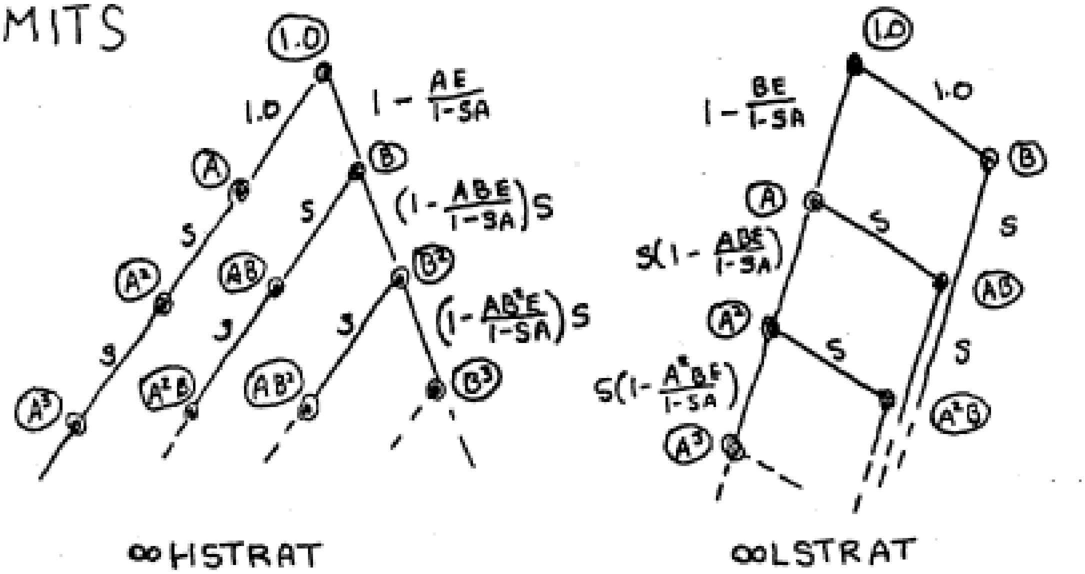


*The diagram does not work like the ones before. Suppose you are at AB and it is an S. If you came from B then you know that A^2B is a split (else at B you would have moved to B^2) and so the chance of moving from AB to AB^2, given that you came from B is $S^2E(1-A^3B)$. However, if you came from A, this chance is $E(1-A^2B) + SE(1-A^3B)$. Therefore you cannot associate to each point in the diagram a single value representing the chance of getting to a point with that value at that level, but need two numbers, one for each way of getting there, and you must keep track of them separately. In the program R(I) is the possibility of coming from the upper left, T(I) from the upper right.

```
DIMENSION R(10)
ENTRY TO HSTRAT
R(0) = 0.
R(1) = 0.
R(2) = E*(1.-A)+S*E*(1.-A*A)
T(0) = 0.
T(1) = 1.
T(2) = 0.
```

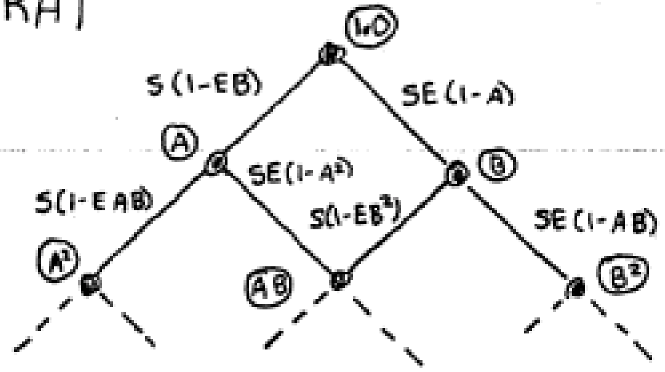
```
      V(0) = 0.
      V(1) = A
      V(2) = B
      D(1) = E*(V(1)+R(2)*V(2))
      THROUGH M1, FOR J = 2,1,J.G.10
      V(J+1) = V(J)*B
      THROUGH M2, FOR L = J,-1,L.LE.0.
M2    V(L) = V(L)*A
      R(J+1) = R(J)*E*(1.-V(J))+S*(1.-V(J)*A))*S
      T(J+1) = 0.
      THROUGH M3, FOR L = J,-1,L.LE.0
      T(L) = S*(T(L)+R(L))
M3    R(L) = T(L-1)*S*S*E*(1.-V(L-1)*A)
     1 +R(L-1)*E*S*((1.-V(L-1))+S*(1.-V(L-1)*A))
      D(J) = 0.
      THROUGH M1, FOR L = 1,1,L.G.J+1
M1    D(J) = D(J)+V(L)*E*(T(L)+R(L))
      PRINT COMMENT $ 2HSTRAT$
      EXECUTE OUTPUT.
```

LIMITS


1.0
1.0
1 - $\frac{AE}{1-SA}$
A
B
S
S
$\left(1 - \frac{ABE}{1-SA}\right)S$
A^2
AB
B^2
S
S
S
$\left(1 - \frac{AB^2E}{1-SA}\right)S$
A^3
A^2B
AB^2
B^3
∞HISTRAT
1.0
1 - $\frac{BE}{1-SA}$
1.0
B
A
S
S
$S\left(1 - \frac{ABE}{1-SA}\right)$
AB
A^2
S
S
$S\left(1 - \frac{A^2BE}{1-SA}\right)$
A^3
A^2B
∞LSTRAT

```
     ENTRY TO LIMITS.
     G = 1./(1.-S*A)
     D(1) = E*A*G
     T(1) = 1.-D(1)
     V(1) = E
     THROUGH M1, FOR J = 2,1,J.G.10
     V(1) = V(1)*B
     C = V(1)*G*A
     D(J) = T(1)*(C*S+V(1))
M1   T(1) = (1.-C)*S*T(1)
     PRINT COMMENT $ HLIMIT$
     EXECUTE OUTPUT.
     T(1) = 1.
     V(1) = E
     THROUGH M2, FOR J = 1,1,J.G.10
     C = V(1)*B*G
     D(J) = T(1)*(C+(1.-C)*V(1)*A)
     V(1) = V(1)*A
M2   T(1) = T(1)*(1.-C)*S
     PRINT COMMENT $ LLIMIT$
     EXECUTE OUTPUT.
```

BSTRAT



This diagram is used a bit differently from the rest. At each point the chance of winning, not at that point, but at its two successors if they exist, is computed. If the value of the point is X, the formula for this is EXB + (1-EXB)EXA. Thus the numbers on the lines indicate the probability of getting to the points and finding that they split. This modification is necessary because of the way BSTRAT works, and is related to the modifications for 2HSTRAT & 2LSTRAT.

```
     ENTRY TO BSTRAT
     D(1) = E*B*(1-E*B)*E*A
     T(0) = 0.
     T(1) = S*(1.-E*B)
     T(2) = S*E*(1.-A)
     V(0) = 0.
     V(1) = A
     V(2) = B
     THROUGH M1, FOR J = 2,1,J.G.10
     D(J) = 0.
     THROUGH M2, FOR L = 1,1,L.G.J
M2   D(J) = D(J)+E*V(L)*T(L)*(B+(1.-E*B*V(L))*A)
     WHENEVER J.E.10, TRANSFER TO M1
     V(J+1) = V(J)*B
     THROUGH M3, FOR L = J,-1,L.LE.0
M3   V(L) = V(L)*A
     T(J+1) = T(J)*E*S*(1.-V(J))
     THROUGH M4, FOR L = J,-1,L.LE.0
M4   T(L) = S*((1.-E*V(L))*T(L)+T(L-1)*E*(1.-V(L-1)))
M1   CONTINUE
     PRINT COMMENT $ BSTRAT$
     EXECUTE OUTPUT.
```

APPENDIX B: Sample Monte Carlo Evaluation Program

The following MAD program was used on a CTSS system to evaluate HSTRAT with non-ordinal modifications, over the class of TSD games, by playing through a series of randomly generated games using that strategy.

```
          INTEGER I,J
          READ FORMAT F,J
          VECTORVALUES F=$I3*$
          THROUGH M1, FOR I=1,1,I.G.J
    M1    A=RANNO.(X)
          READ FORMAT F1,E,NGAME
          VECTOR VALUES F1=$F5.5,F5.0*$
          G=0.
          W=0.
          T=0.
          TN=0.
    M3    WHENEVER G.GE.NGAME, TRANSFER TO END
          X1=1.
    M4    A=RANNO.(X)/2.+.5
          T=T+A
          TN=TN+1.
    M2    X2=X1*A
          X3=X1-X2
          A=RANNO.(X)
          WHENEVER A.G.E, TRANSFER TO M5
          A=TANNO.(X)
          WHENEVER A.G.X2, TRANSFER TO M6
   WIN    W=W+1.
   LOSE   G=G+1.
          TRANSFER TO M3
   M5     A=RANNC.(X)/2.+.5
          T=T+A
          TN=TN+1.
          H=E*X3+(1.-E)*C.(X3,.75)-C.(X2,A)
          WHENEVER H.G.O., TRANSFER TO M6
          X1=X2
          TRANSFER TO M2
```

```
M6   A=RANNO.(X)
     WHENEVER A.G.E, TRANSFER TO M10
     A=RANNO.(X)
     WHENEVER A.G.X3, TRANSFER TO LOSE
     TRANSFER TO WIN
M10  X1=X3
     TRANSFER TO M4
END  SCORE=W/G
     AVE=T/TN
     PRINT FORMAT F2, SCORE, AVE
     VECTOR VALUES F2=$2F8.5*$
     EXECUTE EXIT.
     INTERNAL FUNCTION K.(Y)=Y*(E*(.75+E*.25)-E*E*.1875*Y
   1  (1.-5*(.75+E*.25)+E*S*.1875*Y)
     INTERNAL FUNCTION C.(U,R)=E*U*R+S*K.(U*R)+E*(1.-U*R)
   1  *(E*U*(1.-R)+S*K.(U*(1.-R)))
     END OF PROGRAM
```