ELSEVIER

Artificial Intelligence 100 (1998) 87–123

Artificial Intelligence

Search in games with incomplete information: a case study using Bridge card play

Ian Frank[a,*], David Basin[b,1]

[a] *Complex Games Lab, Electrotechnical Laboratory, Umezono 1-1-4, Tsukuba, Ibaraki, Japan 305*

[b] *Institut für Informatik, Universität Freiburg, Am Flughafen 17, Freiburg, Germany*

Received 9 February 1996; revised 15 March 1997

Abstract

We examine search algorithms in games with incomplete information, formalising a best defence model of such games based on the assumptions typically made when incomplete information problems are analysed in expert texts. We show that equilibrium point strategies for optimal play exist for this model, and define an algorithm capable of computing such strategies. Using this algorithm as a reference we then analyse search architectures that have been proposed for the incomplete information game of Bridge. These architectures select strategies by analysing some statistically significant collection of complete information sub-games. Our model allows us to clearly state the limitations of such architectures in producing expert analysis, and to precisely formalise and distinguish the problems that lead to sub-optimality. We illustrate these problems with simple game trees and with actual play situations from Bridge itself. © 1998 Elsevier Science B.V.

Keywords: Game tree search; Incomplete information; Game theory; Computer Bridge

1. Introduction

In games with incomplete information, the actual "state of the world" is unknown; for example, some playing pieces may be hidden, some of the playing area may not be visible, or the outcome of some moves may not be known. For such games, finding the optimal strategy is typically NP-hard [4] and thus a heuristic approach is often required for timely play. An example of an incomplete information game for

* Corresponding author. Email: ianf@etl.go.jp.

[1] Email: basin@informatik.uni-freiburg.de.

PII S0004-3702(97)00082-9

which good heuristics have yet to be found is Contract Bridge; a history of academic research on the game [5,10,11,17,22,24,27,31,32] and a proliferation of more than two dozen commercial software packages [19,33] have failed to produce solutions or systems capable of competing with even good novice human players [12,23].

The common approaches to automating Bridge card play involve reducing the search space by either considering independently the sub-problems of the *card combinations* in individual suits (first suggested in [3]), or by removing the uncertainty over incomplete information altogether and considering instead the situation where all the players reveal their cards to each other [2,36]. The latter of these approaches has prompted some researchers, for example Levy [20] and Ginsberg [14,15], to suggest Bridge-playing architectures that work by examining a statistically significant number of the worlds that are consistent with a player's knowledge. They speculate that in any given situation the use of search reduction techniques (such as alpha-beta pruning and transposition tables) would enable the minimax value of each possible action to be established in each of these randomly generated sub-problems, and the most promising overall action to be chosen by statistical evaluation based on these values. Recently, Matt Ginsberg has produced what he claims to be a "whole new standard" [35] of Bridge-playing program with an architecture based on such a sampling approach.

Our interest in incomplete information games arose from our own work in designing a system to play Bridge [8,10], and in particular from investigating why this program could produce analyses different from those found in expert texts. To identify the reasons for these discrepancies, we found that we first had to formalise the actual model used by human players when analysing Bridge; surprisingly, we could find no explicit descriptions of such a model in the literature. Thus, the *best defence* model of an incomplete information game—which we formalised by considering the way that experts analyse problems in authoritative Bridge texts—is the first contribution of this paper. We go on to show that an *equilibrium point* for the two players' strategies is well-defined for the best defence model, and describe an algorithm, which we call *exhaustive strategy minimisation*, that identifies such equilibrium points. We then use our best defence formalisation as a tool to investigate the characteristics of sampling algorithms against "best defence".

Whilst others have noted before that computer Bridge architectures may not "play the same way as humans", without a formal model of the assumptions made by experts when solving such problems the qualitative differences in play have been difficult to describe. Our formalisation allows us to identify the two problems that can afflict sampling algorithms, independently of how many worlds they consider. We demonstrate these problems both theoretically, using simple game trees, and in practice, using actual play situations from Bridge itself.

The first of these problems, which we name *strategy fusion*, affects any algorithm that attempts to combine strategies for particular worlds to produce an optimal strategy across all (or some statistically significant subset) of worlds. The flaw in this approach occurs because of the property of incomplete information games that the exact state of the world at any given point of play may not be known to a player. This imposes a constraint on a player's strategy that he must behave the *same* way in *all* possible worlds at such points; a constraint typically broken when combining strategies designed for individual worlds.

The second problem, which we name *non-locality*, is more subtle than strategy fusion, and we take care to distinguish clearly between the two. Non-locality occurs because an opponent with some knowledge of the actual world state can use this to his advantage. In particular, such an opponent can direct play towards the portions of the game tree that are most favourable in the worlds he expects. Thus, some positions in the game may never be reached under particular worlds (as the opponent may always find better alternatives). In general, determining what nodes in the search space will be reached under what worlds requires examining the entire tree of possibilities (since each move an opponent makes gives him the chance to select different portions of the tree in different worlds). Tree search algorithms, however, are generally "compositional" in the sense that they determine the best play at an internal node of a search space by analysing only the subtree of that node. Such algorithms (e.g., minimaxing) will not take into account the possibility that under some worlds the play may never actually reach the node they are examining. When selecting moves, they may therefore make mistakes by erroneously considering payoffs in world states that are in fact of no consequence at that position in the tree. As in strategy fusion, the problem is one of handling the notion of strategy incorrectly. An algorithm that locally evaluates subtrees considers only *partial* strategies; the complete strategies for the entire game would also have to specify what actions would have been taken in all other nodes outside that portion of the tree.

Thus, we demonstrate exactly how the analysis of sampling algorithms will differ from that of experts. Whether this shortcoming is actually sufficient to undermine their practical playing potential is an issue for empirical testing (such as that apparently being carried out by Matt Ginsberg [13]). Our interest is in clarifying the issues involved in finding solutions to incomplete information games, and in understanding the nature of the models implicitly used by different approaches.

We proceed as follows. In Section 2 we introduce preliminary concepts from game theory and apply these in Section 3 to games with incomplete information; in particular we show how the common model of Bridge play against best defence can be formalised within this framework. We follow this in Section 4 by giving an algorithm for computing optimal strategies in our best defence model. The second half of the paper then considers Bridge in some detail: in Section 5 we present sampling architectures for Bridge card play based on the minimax algorithm, and in Sections 6 and 7 we use our game theoretic framework to identify why such architectures yield suboptimal results against best defence. Finally, Section 8 draws conclusions.

2. Game theory background

In this section we introduce definitions and terminology necessary to make the paper self-contained. This is based largely on the work of von Neumann and Morgenstern [34] and Luce and Raiffa [21].

2.1. The extensive and the normal forms of two-player games

In its *extensive* form, a game is a finite tree in which each node corresponds to a *move* where a selection between the branches is made. Each node is identified as being either a *personal* move or a *chance* move. Personal moves are made freely by one of the players, creatively named "1" and "2", since we will consider only two-player games. Chance moves are decided by some mechanical device (e.g., the shuffling of a pack of cards, or the tossing of a coin) that selects a branch in accordance with definite probabilities. There is one distinguished node which represents the start of the game. A *play*, α, of the game involves starting at this distinguished node and allowing each of the players (or chance) to choose a branch until a leaf node of the tree is reached. The value that each player i attaches to the outcome of a play α, (i.e., a leaf of the tree), is given by a numerical utility function $K_i(\alpha)$. This value is sometimes also called the *payoff* and K_i a *payoff function.*

One complication is that at any particular move a player may not have full knowledge of the choices made prior to that point in the play. For example, in many card games the play begins with the shuffling and dealing of a pack of cards into hands, which are not visible to all players. Also, the outcome of personal moves may be hidden from the other player(s), such as when a card is played face down. At any move, then, it is possible that a player will be unsure of the actual position of the play within the game tree. To precisely formalise the extensive form of a game, therefore, requires the nodes of the tree to be partitioned into sets between which a player will not be able to distinguish. We will follow [21] in referring to these sets as *information sets.* We will also model our actual definition on that of [21, pp. 39–51], requiring the specification of a two-player game in extensive form to include the following:

- A finite tree, t, with a distinguished node (the first move in the game).
- A partition, $\mathcal{P}(n)$, of the nodes, n, of the tree into three sets. These sets tell which of the two players (1 or 2) or chance (0) selects the next move at each node.
- A probability distribution over the branches of each chance move, defined by assigning a probability $\pi(n)$ to each daughter of a chance node.
- A refinement of the player partitions into information sets, $\mathcal{I}_i(n)$, for each player i. Each node, n, at which $\mathcal{P}(n) = i$ is classified by $\mathcal{I}_i(n)$ into one of the sets of nodes (numbered as integers $1, 2, \ldots$) between which player i will not be able to distinguish.
- An identification of corresponding branches for each of the moves in each of the information sets. (Since a player cannot distinguish between nodes in an information set, the possible moves will appear the same to him at each node of a set. When constructing the tree representation of the extensive form, we must therefore indicate which branch at each node of an information set corresponds to the "first" possible move, the "second" possible move, and so on. In our diagrams, we will assume that this identification is in simple left-right order.)
- For each player, i, a numerical utility function, K_i, defined over the set of end points of the game tree.

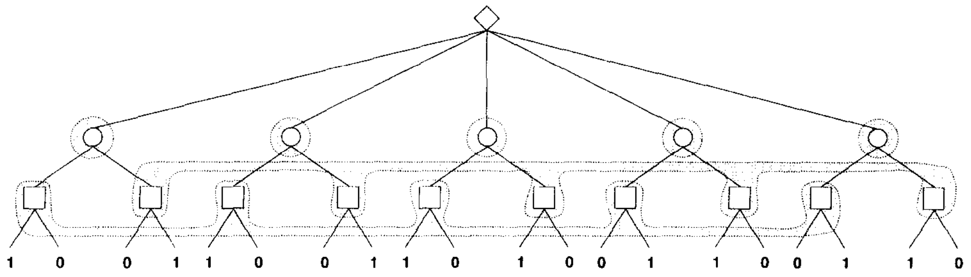


Fig. 1. Information sets in the extensive form of a two player game with one chance move.

Fig. 1 gives an example of the extensive form of a two-player game that starts with a chance move (represented by a diamond). This move has five possible outcomes, and is followed by a personal move of one of the players (represented by a circle). The information set for this player, whatever the outcome of the first chance move, contains only one node, i.e., there is no ambiguity over his actual position in the game tree. This means that the outcome of the initial chance is known to him. For the second player's moves (represented by squares), however, there are two information sets. This is because he is only aware of the outcome of the previous player's move, and not of the outcome of the initial chance move. The actual payoff he will receive (represented by the numbers at the leaf nodes) therefore depends on information that is not known to him.

Most games will be represented by a game tree that is too large to enable the extensive form to be given in practice. In order to facilitate mathematical analysis it is therefore common to work instead with an equivalent formalisation called the *normal form*. This formalisation forces each player, before the game starts, to state in advance what choices they would make in *any situation that could possibly arise during the course of the game*. Such a specification forms a *strategy*. Given the extensive form of a game, an easy way to formulate the possible strategies for each player is to assign a number $1,\ldots,r$, to each branch stemming from a node with r branches. A strategy for a player with q information sets can then be represented by a q-tuple in which each element corresponds to the move to be made in one of the sets. Utilising this notion of strategy, a two-person game in normal form is defined by specifying:

- two *strategy spaces* X_1 and X_2, which are the respective sets from which the two players can choose their strategy, and
- two real-valued payoff functions $K_1(x_1,x_2)$ and $K_2(x_1,x_2)$ that give the utility for each player when strategy x_i is selected by player i. If there are no chance moves in the game, then a given strategy selection $(\widehat{x}_1,\widehat{x}_2)$ determines a unique play of the game, α, and we can define $K_i(\widehat{x}_1,\widehat{x}_2) = K_i(\alpha)$. If the game contains chance moves then $(\widehat{x}_1,\widehat{x}_2)$ instead impose a probability distribution over all the possible plays. If we use *prob*(α) to denote the probability of play α occurring when $(\widehat{x}_1,\widehat{x}_2)$ are chosen, the payoff for each player is expressed in terms of the mathematical expectation $K_i(\widehat{x}_1,\widehat{x}_2) = \sum_\alpha prob(\alpha)K_i(\alpha)$.

Formulated in this way, the course of a single play of a game first involves the choice of a strategy by each player. In *non-cooperative* games, this choice is made without any pre-play communication between the players. The basic theory of non-cooperative games is based on the concept of an *equilibrium point*, due to Nash [25]. This views any selection of strategies by each player as being a "solution" to a game whenever no single player can individually increase his payoff, or expected payoff, by changing his strategy selection. That is, a pair $(\widehat{x}_1, \widehat{x}_2)$ is an equilibrium point if the following hold.

$$\begin{aligned} K_1(\widehat{x}_1,\widehat{x}_2) &\geqslant K_1(x_1,\widehat{x}_2) \quad \forall x_1 \in X_1, \\ K_2(\widehat{x}_1,\widehat{x}_2) &\geqslant K_2(\widehat{x}_1,x_2) \quad \forall x_2 \in X_2. \end{aligned} \tag{1}$$

A special case, relevant to our domain, is that of *zero-sum games*, where $K_1(x_1,x_2) + K_2(x_1,x_2) = 0$, for all $x_1 \in X_1, x_2 \in X_2$. Under this condition, we can arbitrarily select a K_i (we choose K_1) and rewrite (1) as

$$K_1(x_1,\widehat{x}_2) \leqslant K_1(\widehat{x}_1,\widehat{x}_2) \leqslant K_1(\widehat{x}_1,x_2) \quad \forall x_1 \in X_1, x_2 \in X_2. \tag{2}$$

This states that K_1 has a saddle point; we consider next when this holds.

2.2. Minorant and majorant games: the minimax theorem

Let Γ represent a non-cooperative, zero-sum, two-player game where the players pick their strategies without knowledge of that of their opponent. To analyse such games, von Neumann and Morgenstern introduce two variations on Γ. The first of these, Γ_1, is defined so that it agrees with Γ in every detail except that player 1 must choose x_1 *before* player 2 chooses x_2, so that player 2 makes his choice in full knowledge of the particular x_1 decided on by player 1. Since player 1 is at a disadvantage in this game compared to his position in the original game, Γ_1 is termed the *minorant* game of Γ. The second variation, Γ_2, is the dual whereby player 2 chooses his strategy first; this game is termed the *majorant* game of Γ.

For these new games, Γ_1 and Γ_2, the identification of the optimal strategy is simplified, as the "best way of playing" may be given a clear meaning. Let us consider the minorant game Γ_1, in which player 1 is the first to select a strategy. For any particular selection, $\widehat{x}_1$, player 2 will choose an x_2 to minimise the value of $K_1(\widehat{x}_1,x_2)$. Thus, when player 1 is choosing an x_1 he can be sure (assuming a competent opponent) that the expected outcome of the game is $\min_{x_2} K_1(x_1,x_2)$. This formula is a function of x_1 alone, and since player 1 is attempting to maximise his payoff, the best expected outcome he will therefore be able to achieve is

$$v_1 = \max_{x_1} \min_{x_2} K_1(x_1,x_2).$$

A similar argument shows that if both players play the majorant game well, the resulting payoff that can be expected by player 1 is

$$v_2 = \min_{x_2} \max_{x_1} K_1(x_1,x_2).$$

Von Neumann and Morgenstern show that these values can be used to establish upper and lower bounds on the value, v, that player 1 can hope for from a play of Γ itself. This result is then refined to produce a theorem that states that there is a subclass of zero-sum two-player games (those with perfect information) for which $v_1 = v = v_2$. This is equivalent to writing

$$\max_{x_1} \min_{x_2} K_1(x_1, x_2) = \min_{x_2} \max_{x_1} K_1(x_1, x_2), \tag{3}$$

which is in turn equivalent to (2) (i.e., it states that K_1 must have a saddle point). The form of (3) has led Von Neumann and Morgenstern's result to be referred to as the *minimax theorem* and it forms the basis of the well-known minimax algorithm suggested by Shannon [30].

2.3. Pure and mixed strategies

The minimax theorem does not hold for all two-player games. To achieve an existence theorem for general games, Von Neumann and Morgenstern extend the notion of strategy as follows. If the sizes of the sets X_1 and X_2 are m and n respectively, the players, rather than choosing an x_1 and an x_2 from these sets, instead specify vectors $\boldsymbol{p} = (p_1, \ldots, p_m)$ and $\boldsymbol{q} = (q_1, \ldots, q_n)$ $(p_i, q_i \geqslant 0,\ \sum p_i = 1,\ \sum q_i = 1)$ where p_i gives the *probability* that player 1 will select the ith member of X_1 as his strategy and q_i is the probability that player 2 will select the ith member of X_2. When the players select their strategies in this probabilistic manner, the natural interpretation of the expected outcome is the mathematical expectation

$$K(\boldsymbol{p}, \boldsymbol{q}) = \sum_{1=i}^{m} \sum_{j=1}^{n} K_1(x_i, x_j) p_i q_j. \tag{4}$$

As in the previous section, it is possible to show that for this augmented game, the function K has a saddle point. This is a probability theoretic interpretation of the previous saddle point theorem, and illustrates that in some games there is a definite disadvantage to having your strategy "found out" by the opponent. Using a probability vector to select randomly from among a number of possible strategies affords protection from exactly such an occurrence. Strategies in this augmented sense are called *mixed strategies*. The strategies of the previous section are a special case of mixed strategies in which the probability distribution is a 1-point distribution, and are referred to as *pure strategies*.

2.4. Preliminarity and anteriority

Let us consider again the extensive form of a game. If we represent a particular play of a game as a sequence of moves $M_1, M_2, M_3, \ldots$, we can define the moves that are *anterior* to some personal move M_k as being any move M_i with $i < k$. Notice that this property is transitive, i.e., if M_μ is anterior to M_λ and M_λ is anterior to M_κ, then M_μ is anterior to M_κ. We can also look at the amount of information on the outcome of the anterior moves that is available to the player who is called upon to make move M_k. It is possible that this player will know which branch was chosen for each of the moves $M_1, \dots, M_{k-1}$, but it may also be that he has only partial knowledge, or no knowledge at all. The simplest way to describe a player's state of information at move M_k is to form a set of *preliminary* moves, $\mathcal{P}$. This set consists of the moves M_i, for some $i \in \{1, \dots, k-1\}$, such that the branch chosen for any of the $M_i \in \mathcal{P}$ is known to the player, but the exact choice made at any of the other anterior moves is not.

The class of games in which preliminarity and anteriority coincide (i.e., where a player called upon to make a move is informed about the outcome of all the anterior moves) is called *perfect* information games. We have already seen in Section 2.2 that the minimax theorem enables each player's optimal strategy to be interpreted in a precise way for such games. However, in games where anteriority does not imply preliminarity (which we call *incomplete information* games), peculiar features can result. For instance, the property of preliminarity need not be transitive, as illustrated by the following example, which we quote from [34, p. 52]:

> Poker: Let M_μ be the deal of his "hand" to player 1—a chance move; M_λ the first bid of player 1—a personal move of 1; M_κ the first (subsequent) bid of player 2—a personal move of 2. Then M_μ is preliminary to M_λ and M_λ to M_κ but M_μ is not preliminary to M_κ (i.e., 1 makes his first bid knowing his own "hand"; 2 makes his first bid knowing 1's (preceding) first bid; but at the same time 2 is ignorant of 1's "hand".)

This intransitivity of preliminarity involves both players, but it is also possible that preliminarity could be intransitive among the personal moves of one particular player. Bridge provides an example of this, since although it is played by four players, the rules of the game dictate that these players form two teams, which play against each other. Again using a description from [34, p. 53]:

> Bridge is a two-person game, but the two players 1 and 2 do not play it themselves. 1 acts through two representatives A and C and 2 through two representatives B and D. Consider now the representatives of 1, A and C. The rules of the game restrict communication, i.e., the exchange of information, between them. E.g.: let M_μ be the deal of his "hand" to A—a chance move; M_λ the first card played by A—a personal move of 1; M_κ the card played [...] by C—a personal move of 1. Then M_μ is preliminary to M_λ and M_λ to M_κ but M_μ is not preliminary to M_κ. Thus we again have intransitivity, but this time it involves only one player.

Intransitivity of preliminarity raises the possibility of signalling (i.e., the spreading of information to other players). In Bridge, players who form one team but cannot see each other's cards will wish to promote this signalling, and an elaborate system of conventional signals has been developed to enable this. In Poker, the interest of a player lies in preventing this signalling, and this is usually achieved by irregular and seemingly illogical behaviour when making a choice. The first of these two types of procedures is *direct* signalling and the second is *inverted* signalling.

3. A model of the game of Bridge

With the definitions of the previous section behind us, we are now in a position to formalise the way in which Bridge is analysed in the expert literature. We first describe the game and its characteristics in more detail, and then present formal assumptions that model the situation analysed in expert texts. It is the formalisation of this model that allows us to meaningfully assess whether any given algorithm will produce correct, expert-level results in Bridge (as judged by those found in the literature), and more importantly to describe precisely the problems that can lead to sub-optimality.

3.1. *Bridge as a game of incomplete information*

Bridge is a card game played with a deck containing 52 cards, comprised of 4 suits (spades ♠, hearts ♡, diamonds ◇, and clubs ♣) each containing the 13 cards Ace, King, Queen, Jack, 10, ..., 2 (we will sometimes abbreviate the first five of these to A, K, Q, J, and T). The game begins with the chance move of shuffling the deck, and the cards are then dealt between four players, traditionally named North, South, East and West. The players form two teams: North/South against East/West. Card play starts when one player lays a card on the table, which all the other players then cover in turn (in a clockwise direction) with a card from their own hand. Each round of four cards is called a *trick*, and the winner of one trick becomes the first person to play a card on the succeeding round. For the purposes of this paper, the only significant rules are:

- The first player in a trick can freely choose which card to play from all those present in his hand.
- Subsequent players *must* play a card of the same suit as the one that started the trick, if they hold such a card—if they do not, they can make a free choice from among the remaining cards in their hand.
- The winner of the trick is the player who plays the highest card (ranked by $A > K > Q > J > 10 > \cdots > 2$) of the suit led. The only exception to this rule is when there is a suit declared as the *trump suit*; if any trump cards are played, then the player playing the highest trump card is the winner.

In addition, play involves one player—the *declarer*—having complete control over two hands of cards, since his partner—the *dummy*—places his cards on the table for all to see, and then takes no further part in the proceedings. For simplicity, our Bridge examples will always assume that South is the declarer, so that North is the dummy.

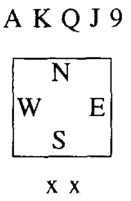
Analysis of the game of Bridge is extraordinarily complicated. The shuffling and dealing of the pack of cards at the start of the game in effect selects one of $52!/13!^4$ possible positions for the subsequent play (the order of the cards in a hand does not matter—hence the dividing factors). Further, each player can initially see only their own hand, so (as we have already seen) preliminarity will be intransitive among the moves of the players, giving the opportunity for both direct signalling between partners, and inverted signalling to confuse the opponents. A related problem caused by this mis-match between the available information is that of predicting an opponent's beliefs. Korf, for example, although not motivated by considering games with incomplete information, examined the situation where two players have evaluation functions that are not known to the other [18]. Such a game can be viewed either as a zero-sum game in which the difference in evaluation functions is due to their heuristic nature, or as a non-zero-sum game in which the evaluation functions represent the payoffs that each player would actually receive, or as an incomplete information game in which the difference in evaluation functions is due to the differing information available to each player. Korf's description of this situation uses the common convention of naming the two players MAX and MIN. In the context of the previous section where we (arbitrarily) chose K_1 as the payoff function, MAX will be player 1, since he tries to maximise the value of K_1. Similarly, MIN will be player 2, since he tries to minimise the value. The decision process in a three-level MAX-MIN-MAX tree is described by Korf as follows:

> MAX's decision will be based on what he thinks MIN will do. However, MIN's decision will be based on what he thinks MAX will do two levels down. Thus, MAX's decision is based on what MAX thinks that MIN thinks that MAX will do. Therefore, the evaluation function that is applied to each of the frontier nodes is MAX's model of MIN's model of MAX's evaluation function, and the nodes with the maximum values are backed up to the MAX nodes directly above the frontier. Next, MAX's model of MIN's evaluation is applied to the backed up nodes, and the nodes with the minimum values are backed up to the MIN nodes directly below the root. Finally, MAX's evaluation is applied to these backed up nodes to determine the final move.

In Bridge, then, we can identify the following four related complications over perfect information games. First, the intransitivity of preliminarity between the moves of the two sides will lead us to encounter the problem of ever-deepening levels of reasoning about the opposing side's beliefs. Second, as we saw in Section 2.4, this intransitivity also makes inverted signalling possible, making it advantageous for one side to attempt to prevent the spread of information about their position to the other side. Third, there is the opportunity for direct signalling, in which the two players who form the opposition play to increase each other's information of their side's situation. Finally, the absence of perfect information will entail solutions that are mixed strategies, since using a pure strategy will typically give the opponents an advantage if they can "find out" what this strategy is. The probabilistic nature of mixed strategies prevents the opponents from knowing which pure strategy will be followed, even if they are aware of the exact mixed strategy that will be used.

3.2. The best defence model of an incomplete information game

The problems outlined above present serious difficulties when combined with game trees the size of those generated in Bridge (a lower bound of 1.05×10^{18} can be established on the expected number of legal play sequences even when all the cards are revealed [8]). Yet authors of Bridge texts are able to analyse play situations and to recommend "optimal" strategies for dealing with any given problem. Here, we will examine how this is done, formalising a *best defence* model of incomplete information games that captures the assumptions implicitly made in such expert analyses.



Tricks required:	5
Strategy:	Cash top honors in the hope of dropping the Ten
Chance of success:	87%

Fig. 2. Third card combination from the Bridge Encyclopedia.

3.2.1. Expert analysis of Bridge

As a representative example of expert texts we will look at the authoritative Bridge Encyclopedia [1], published by the American Contract Bridge League. This reference devotes 56 pages to presenting the best strategies (with percentage chances of success) for different card combinations. In Fig. 2, we reproduce one of these examples.

In this game situation we are concerned with only the cards of one suit and North is the dummy, so that South plays both the North and South hands. The remaining cards—the Ten plus five low cards (x)—may be held by either East or West, who see just their own hand and the dummy. The Encyclopedia's solution of "Cashing the top honours" describes the process of playing, one by one, the Ace, King, Queen and Jack. When this is done, the Ten will "drop" (be played by either East and West) unless the player who holds it also has at least four low cards.

To verify the Encyclopedia's solution, we can proceed as follows. First, we enumerate all the possible ways to distribute the unseen cards between East and West. To make this process more efficient, we will refer to these possibilities collectively according to the number of cards held by the two players. For example, a 1–5 split of the cards describes the six possible ways to give just one of the unseen cards to West, and the remainder to East.

For each of the possible splits we then check whether the Encyclopedia's strategy succeeds. For example, the strategy will fail if the unseen cards are split 0–6, since East will be able to play a low card on each of the Ace, King, Queen, and Jack, and then beat the 9 with the Ten. If the cards are split 1–5, on the other hand, playing the four high cards may succeed, but only if the Ten is the single card that is held by West. Note that special care is required in the case of a 6–0 split. In this situation, East will have to play a card from another suit when the Ace is cashed. The situation in this suit then effectively becomes one of complete information, since it is known that West must hold the remaining cards. Under these conditions, winning a trick with the 9 can be guaranteed by playing the low card from the South hand; if West doesn't play the Ten, the 9 will win the trick, but if West does play the Ten it will be beaten by the King, allowing the 9 to win a trick later on.

In the table of Fig. 3 we give, for each possible split of the cards, the probability of the split occurring, the number of ways to produce the split, the actual distributions in which the Encyclopedia's strategy succeeds, and the probability of these distributions occurring. Summing these probabilities results in the chance of success given in the Encyclopedia.

Split	Probability	Total cases	Strategy succeeds	Contribution
0–6	0.007	${}^6C_0 = 1$	none	0
1–5	0.073	${}^6C_1 = 6$	T–xxxxx	$(1/6) * 0.073 = 0.012$
2–4	0.242	${}^6C_2 = 15$	all	$(15/15) * 0.242 = 0.242$
3–3	0.355	${}^6C_3 = 20$	all	$(20/20) * 0.355 = 0.355$
4–2	0.242	${}^6C_4 = 15$	all	$(15/15) * 0.242 = 0.242$
5–1	0.073	${}^6C_5 = 6$	xxxxx–T	$(1/6) * 0.073 = 0.012$
6–0	0.007	${}^6C_6 = 1$	Txxxxx–	$(1/1) * 0.007 = 0.007$
			Total	0.87

Fig. 3. Verifying the chance of success of the Bridge Encyclopedia solution.

3.2.2. *The best defence model*

The probabilistic analysis of all the strategies listed in the Bridge Encyclopedia can be verified in the above way: that is, by examining their outcome under each of the possible distributions of the remaining cards. We verified, too, that this was actually the technique used when the Encyclopedia's solutions were originally generated, by contacting their author, Eric Crowhurst. According to Crowhurst [6], he solved each problem by first using his Bridge expertise to select a small number of promising strategies, and then determining their chances of success as we did above: enumerate the possible distributions of the remaining cards and analyse the return produced when the opponents are allowed to choose their best strategy in each of these perfect information situations. Based on this evidence, we formalised the assumptions implicitly made in expert analysis of Bridge problems as follows:

A-I. MIN has perfect information (i.e., preliminarity and anteriority coincide).

A-II. MIN chooses his strategy after MAX.

A-III. The strategy adopted by MAX is a pure strategy.

We call the result of transforming any game by making these modifications its *best defence* form. It should be obvious that A-II follows directly from our discussion above. A-I also follows since allowing the opponents to choose the best strategy for each possible card distribution in effect assumes that they will always *know* this distribution. Further evidence that the Encyclopedia makes an implicit assumption of complete information is that all of the problems are presented only one way around. That is, a card combination such as that of Fig. 2 is never presented with South as the dummy and North as the hidden hand. From the perspective of East or West, these problems should be significantly different, unless we assume that they have complete information.

Under assumptions A-I and A-II, the optimal strategies for MAX will in general be mixed strategies, since this will afford some degree of protection from his actual moves being completely predicted by MIN. We justify the restriction to pure strategies in A-III by the observation that pure strategies form the solutions typically given in expert texts (for example, the Encyclopedia *only* gives pure strategies), and it is this implicit expert model that we are trying to capture. However, we should point out that the ability to identify the best pure strategies for MAX can in turn be used to generate good mixed strategies. For example, the identification of a small number of good (pure) strategies for each player makes it feasible to form a matrix of the payoffs produced for each possible pair of strategy selections. The vector maximisation problem set up by this matrix can be reduced to a set of constrained scalar maximisation problems, which can be solved by techniques of mathematical programming. Strategies produced in this way would be solutions to the probability theoretic interpretation of the minimax theorem introduced in Section 2.3.

Note that as well as capturing the assumptions made by experts in solving Bridge problems, the best defence form is also amenable to formal analysis. A-I cuts the cycle of reasoning about beliefs discussed in Section 3.1, since now MIN knows exactly what information MAX possesses at any stage of the game, and MIN's model of MAX's evaluation function will be MAX's function itself. Further, A-II is the same assumption as that made by von Neumann and Morgenstern in reducing a game to its minorant form, for which (as we saw in Section 2.2) the outcome can be specified as a function of MAX's strategy only. Finally, A-III enables MAX's optimal strategy to be found from among a finite (although possibly very large) set, in contrast to the set of possible mixed strategies. In the next section, we will make use of these properties in discussing how the best defence form of a game may be solved.

4. Solving the best defence model

In order to use the best defence model to analyse proposed architectures for play under incomplete information we now introduce an algorithm that can compute optimal solutions relative to the assumptions of the model. We do not suggest this algorithm for practical use (although we analyse its complexity in Section 4.5). Rather, this algorithm will form the basis for a rigorous evaluation of architectures that *have* been proposed for Bridge card play, allowing us to understand their limitations in producing expert analysis, and to formalise the two types of sub-optimality from which they will suffer.

4.1. Exhaustive strategy minimisation

As we mentioned in the previous section, assumption A-II (that MIN chooses his strategy after MAX) makes the best defence model similar to the minorant form of a perfect information game. From Section 2.2, the value to be expected by MAX (player 1) in such a game is

$$v_1 = \max_{x_1} \min_{x_2} K_1(x_1, x_2). \tag{5}$$

For the minorant game, $\min_{x_2} K_1(x_1, x_2)$ is a function of x_1, so it is easily maximised. In the best defence model, however, there is asymmetric information. Specifically, MIN

Algorithm ***esm***(Γ): Returns optimal strategies for player 1 (MAX) in the best defence, extensive form, two-player game Γ.

- Form the set of player 1's strategies, S, as q-tuples in which the ith entry represents the branch that is chosen at all nodes, n, for which $\mathcal{I}_1(n) = i$.
- For each $s_j \in S$, calculate $E_j = esm(t, s_j)$.
- Return the strategy (or strategies) s_j for which E_j is maximum.

Here, the result of $esm(t, s)$ is defined as follows:

Condition	Result
t is leaf node	$K_1(t)$
$\mathcal{P}(node(t))$ is 2 (i.e., MIN to move)	$\min_{t_i \in sub(t)} esm(t_i, s)$
$\mathcal{P}(node(t))$ is 1 (i.e., MAX to move)	$esm(t_i, s)$, where i is the $\mathcal{I}_1(node(t))$th element of s
$\mathcal{P}(node(t))$ is 0 (i.e., chance move)	$\sum_{t_i \in sub(t)} \pi(node(t_i))esm(t_i, s)$

Fig. 4. The exhaustive strategy minimisation algorithm.

(player 2) now has perfect information. He therefore always knows his current position in the game tree, and for any choice of x_1 by MAX he can select an optimal x_2. MAX, on the other hand, is not party to the same information as MIN and will not be able to directly identify whether any x_2 is optimal for MIN. Specifically, since MAX does not have perfect information, there may be moves in the game for which MIN knows the outcome but MAX does not. Selection of an optimal x_2 will require reasoning about the outcome of these moves.

The algorithm we formulate deals with this problem by identifying (possibly different) choices of x_2 that are optimal under *each* of the possible outcomes of these moves. Essentially, the algorithm directly computes (5) with respect to the best defence model: all strategies x_1 for player 1 (MAX) are enumerated and for each of them $\min_{x_2} K_1(x_1, x_2)$ is separately and exhaustively evaluated by examining each possible x_2 under each possible outcome of the chance moves. Since this carries out a minimisation operation for each MAX strategy, we call the algorithm *exhaustive strategy minimisation.*

More concretely, assume that for t a tree, $node(t)$ returns its root node and $sub(t)$ computes the set of immediate subtrees of $node(t)$. Fig. 4 then defines exhaustive strategy minimisation.

The assumptions involved in modifying a game into its best defence form are required to establish that exhaustive strategy minimisation correctly computes the optimal MAX strategy given by (5). First, to be able to actually form a finite set of MAX strategies, S, assumption A-III restricting consideration to pure strategies is needed. Second, in fixing a particular $s_j \in \mathcal{S}$ and calling $esm(t, s_j)$, assumption A-II that MIN selects his strategy after MAX is required. Finally, assumption A-I (that MIN has perfect information) is used at chance and MIN nodes, since it is only valid to assume that the evaluation of a particular node depends just on the subtree of that node if there is no ambiguity over the position in the tree. Under these assumptions, it follows by induction on the height of game trees t, that for any strategy s for MAX, $esm(t, s)$ computes the minimal payoff to which MAX can be restricted by MIN. Therefore, given the top-level maximisation loop, the algorithm returns the optimal strategy for MAX and thus correctly computes (5).

4.2. An example

To illustrate exhaustive strategy minimisation, we will consider again the example of Fig. 1. Let us interpret this diagram under the common convention that nodes where it is MAX's turn to move are represented as squares, and those where it is MIN's turn are represented as circles. MAX therefore has two information sets, with two possible moves in each, allowing him four strategies. These strategies can be identified by the tuples $(1, 1)$, $(1, 2)$, $(2, 1)$, and $(2, 2)$. Let us assume that the "lower" of the information sets in Fig. 1 is the set "1", and the "upper" is the set "2": the tuple $(1,1)$ corresponds to selecting the left-hand branch at every MAX node; $(1, 2)$ corresponds to choosing the left-hand branch at all the MAX nodes in the lower set and the right-hand branch in the "upper" set, and so on.

In (5), the result of each of MAX's possible strategies is found by minimising over all the possible responses by MIN. The $esm(\Gamma)$ algorithm models this by calculating $esm(t, s)$ for each possible MAX strategy, s. Let us consider how this will function for the strategy $(1, 1)$. The original call to the $esm(t, s)$ algorithm will examine the root of the tree, find it to be a chance node, and make recursive calls on each of the subtrees. Each of these subtrees has a MIN node at the root, so further recursive calls will then be made on each of their subtrees. The roots of these trees are now MAX nodes, so the branch to select is recovered from the strategy under consideration, and found to be branch 1. Further recursive calls on the subtrees along these branches encounter leaf nodes, at which point the payoffs are returned. These payoffs are then passed back up the tree. At the MIN nodes, the minimum of the subtree evaluations is returned. This process is depicted in Fig. 5, which shows that the strategy $(1, 1)$ will lead to a payoff of 0 under each outcome of the chance move except the third. If the outcomes of the chance move are all equally likely, then, the evaluation of the strategy $(1, 1)$ is $1/5$.

Examining the remaining strategies in the same way shows that the strategy $(1,2)$ leads to a payoff of 1 in just the first two of the outcomes of the chance move, strategy $(2, 1)$ leads to a payoff of 1 in just the final two outcomes, and strategy $(2,2)$ always produces a payoff of 0. If we assume equally likely outcomes for the chance move, the two strategies which maximise (5) are therefore $(1,2)$ and $(2, 1)$.

4.3. Comparison with standard minimaxing

The exhaustive strategy minimisation algorithm should not be confused with minimaxing, which correctly produces the game-theoretic value of any finite tree in which

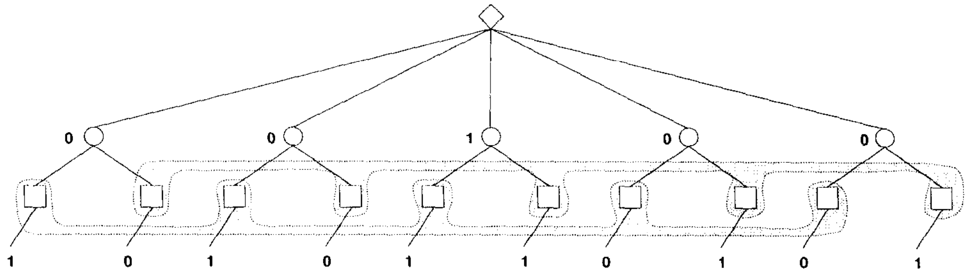


Fig. 5. MAX's expected payoffs when selecting strategy (1, 1).

Algorithm $mm(\Gamma)$: Take the following actions, depending on t.

Condition	Result
t is leaf node	$K_1(t)$
$\mathcal{P}(node(t))$ is 2 (i.e., MIN to move)	$\min_{t_i \in sub(t)} mm(t_i)$
$\mathcal{P}(node(t))$ is 1 (i.e., MAX to move)	$\max_{t_i \in sub(t)} mm(t_i)$
$\mathcal{P}(node(t))$ is 0 (i.e., chance move)	$\sum_{t_i \in sub(t)} \pi(node(t_i))mm(t_i)$

Fig. 6. Simple adaptation of the minimax algorithm to best defence, two-player, extensive form games.

the players have perfect information. For comparison, in Fig. 6 we give a formalisation of the minimax algorithm on games in extensive form.

Notice that, where the $esm(\Gamma)$ algorithm explicitly manipulates strategies by passing them as arguments and analysing them, the $mm(\Gamma)$ algorithm builds a MAX strategy for each of the possible outcomes of the chance moves by determining a course of action for each MAX node on the basis of the subtree with the largest minimax value. It should be clear that this approach will always produce an evaluation that is greater than or equal to the maximum E_j produced by the $esm(\Gamma)$ algorithm. To see this, consider any MAX node (the only node at which the actions of the $mm(\Gamma)$ and $esm(\Gamma)$ algorithms essentially differ). At these nodes, the $mm(\Gamma)$ algorithm selects the maximum subtree value to back up through the tree, whereas the $esm(t, s_j)$ algorithm must select the branch determined by s_j. The evaluation produced by the $mm(\Gamma)$ algorithm can therefore never be less than that produced by $esm(\Gamma)$.

The differences between the two algorithms can be summarised as follows:

(1) $mm(\Gamma)$ does *not* respect the constraint imposed by information sets. That is, it does *not* always choose the same branch at nodes that belong to the same information set.

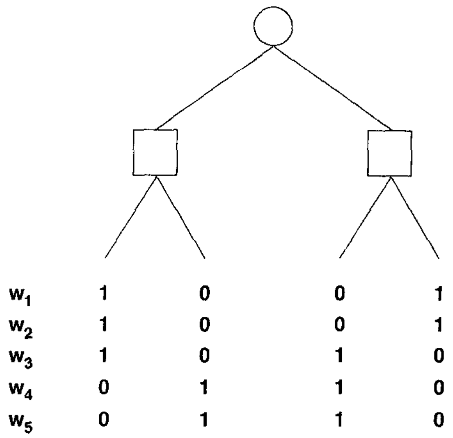


Fig. 7. Flattened tree of MIN and MAX moves in a domain with five worlds.

(2) *mm*(Γ) commits to *one* branch selection at each MAX node, whereas *esm*(Γ) examines the result of each possible strategy. *mm*(Γ) therefore risks incompleteness.

In Sections 6 and 7 we show that these differences cause difficulties in games with incomplete information. Specifically, we will show that the first leads to the problem of *strategy fusion*, and the second leads to the phenomenon of *non-locality*.

4.4. Possible worlds

Let us say that in a tree that contains chance nodes, each possible pure strategy for player 0 (chance) defines a *world state*, or more simply a *world*, in which the play takes place. For example, the tree of Fig. 1 has just one chance move, which has five possible outcomes. Therefore, there are five possible chance strategies and five corresponding possible worlds.

We will use the notion of possible worlds to visualise the extensive form of a game in a more compact manner. In particular, we will consider cases, such as Fig. 1, where the one chance node occurs at the root of the game tree, and where the subtrees in each possible world have the same shape (i.e., the same node and branching patterns). In such situations the possible worlds can be represented in the vertical dimension as differing payoffs at the leaf nodes, rather than in the horizontal dimension using different subtrees for each world. Fig. 7 shows how the tree of Fig. 1 can be represented if we refer to the possible outcomes (left to right) of the initial chance move as the worlds w_1, w_2, w_3, w_4, and w_5.

One part of the game definition that is obscured in this modified form of presentation is the information sets. However, this information is easily recovered by allowing the players to make their branch selections at each node conditional on the world state. For example, in Fig. 7, the MIN node is the result of identifying together five nodes in *distinct* information sets, whereas the two MAX nodes each identify together five nodes from the *same* information set. A MIN strategy must therefore allow *different* branch choices at each MIN node under each world and a MAX strategy must specify the branches to be chosen at each MAX node under *all* worlds.

Allowing a different choice of branch at each MIN node in every world effectively gives MIN perfect information, as in our assumption A-I about the best defence model. MAX, on the other hand, can only assign probabilities to the possible outcomes of the chance move, and must make the same move at each MAX node under every world. We will call the extensive form trees presented and interpreted in this way *flattened* trees. We will use such simplified trees in the remainder of this paper to compactly and simply present the extensive form of games that start with a single chance move.

Note that in general it will not be possible to represent game trees in Bridge in a flattened form, even though they will always have just one chance move at the top of the tree. This chance move will determine a world, but the possible plays under each of these worlds will not in general be the same. For example, MAX will only be able to play a given card, such as the A♠, in worlds where he was actually dealt the card to begin with. Thus, the subtrees in each world are different and cannot be "flattened" as described above. The introduction of the flattened form is simply a presentational device to allow simple game trees in extensive form to be presented more easily.

4.5. *The complexity of exhaustive strategy minimisation*

We have stated that the purpose of defining exhaustive strategy minimisation is to form a basis for investigating the characteristics of other architectures. That it is probably not a practical algorithm itself can be seen by examining its complexity.

Exhaustive strategy minimisation incorporates a top-level loop that examines the set of all possible MAX strategies, $\mathcal{S}$. We will therefore investigate the algorithm's complexity by considering the size of this set. For simplicity, we do this by looking at the type of flattened trees we introduced in the previous section, using the complete binary tree of Fig. 8 as an example. Recall that this tree represents a game where MIN has perfect information, but where MAX never knows the outcome of the one chance move that occurs at the start. Each MAX node corresponds to a set of nodes that come from a single information set. Here, we have numbered the MAX nodes in the order in which they would be encountered during a pre-order traversal of the tree. We will use this numbering to construct 5-tuples that correspond to the possible strategies in the game by virtue of the ith element representing the choice to be made at the node numbered i (effectively, the ith information set).

Let us consider MAX's possible branch selections. In the strategies where MAX initially selects the left-hand branch at the root of the tree, play is directed to the left-hand MIN node. Since we do not know which branch will be selected at this node, we must now specify branch selections throughout *both* its subtrees. Continuing to examine the left-most branch first, we encounter MAX node 2 and then MAX node 3, where we will assume that we initially again select the left-hand branch. It should be clear that by selecting the left-hand branches at these two nodes, we will in fact complete a strategy for the game; once the left-hand branch has been selected at the root of the tree, it is no longer necessary to specify branch selections for nodes 4 and 5. We will indicate

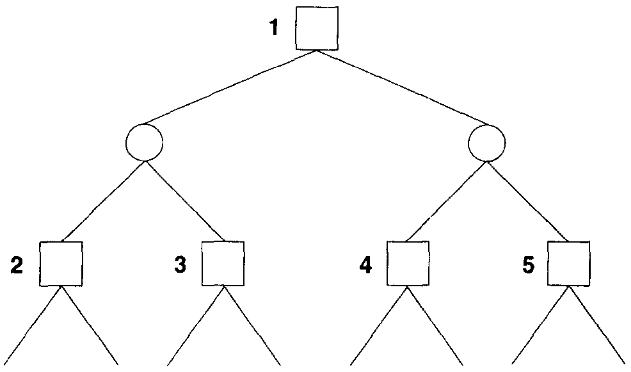


Fig. 8. Complete binary tree of MIN and MAX moves.

such superfluous branch selections by an underscore. The strategy we have generated by always selecting the left-most branch, then, is (1, 1, 1, _, _); overall, the set of possible strategies is:

Left-hand branch at root	Right-hand branch at root
(1, 1, 1, _, _)	(2, _, _, 1, 1)
(1, 2, 1, _, _)	(2, _, _, 2, 1)
(1, 1, 2, _, _)	(2, _, _, 1, 2)
(1, 2, 2, _, _)	(2, _, _, 2, 2)

In general trees, the number of actual strategies in a game is bounded by the number of possible n-tuples. As we have seen above, when encountering a MAX node, any one of the branches may be selected. However, at nodes that are moves of another player, MAX will have to cater for all of the possible branches that may be chosen. Thus, for a given tree, t, the total number of strategies $g(t)$ is given by the following:

$$g(t) = \begin{cases} \sum\limits_{t_i \in sub(t)} g(t_i) & \text{if MAX is to move at the root of } t, \\ \prod\limits_{t_i \in sub(t)} g(t_i) & \text{if another player is to move at the root node of } t, \\ 1 & \text{if } t \text{ is a leaf node.} \end{cases}$$

For complete b-ary trees that alternate between the moves of MAX and MIN this formula can be written as a standard recurrence relation. As the addition of an extra layer of MIN nodes to the leaves of a tree does not alter the number of MAX strategies present, we will write this recurrence as a function of the number of MAX levels, n, in a tree. For a b-ary tree with a MAX node at the root, then, we can write the number of MAX strategies, g_n, as

$$g_n = \begin{cases} b & \text{if } n = 1, \\ b(g_{n-1})^b & \text{if } n \geqslant 2, \end{cases}$$

which has the solution

$$g_n = b^{(b^n-1)/(b-1)}.$$

For the example above where $n = 2$, and $b = 2$ this formula gives 8 strategies, as expected. For trees with a MIN node at the root, we can solve a similar recurrence to produce the formula

$$g_n = b^{b(b^n-1)/(b-1)}.$$

For b-ary trees in general, then, the number of strategies that must be examined is doubly exponential in the number of MAX levels that the tree contains. Further, all of these strategies are examined in *each* of the possible worlds. Thus, the exhaustive strategy minimisation algorithm will require too much computation to be applied to all but the smallest of game trees. However, note that the algorithm given provides only an upper bound to the complexity of this problem. We do not know if there is a more efficient way of finding the optimal strategy; however, as we will show in the following sections, minimaxing, and algorithms based on it, are not a substitute.

5. Bridge architectures based on standard minimaxing

In the remainder of this paper we use our model of best defence in Bridge to examine other algorithms for strategy selection, in particular the technique, mentioned in the Introduction, of simplifying the task of card play by solving instead the easier problem where all the players reveal their cards to each other. Since the perfect knowledge situation created by this act is akin to the opponents placing their cards on the table in the same manner as the dummy, this scenario is often described as *double-dummy* Bridge. Below, we present the double-dummy architectures proposed by both Levy and Ginsberg [14, 15, 20]. In the following sections, we then use our framework to demonstrate why such approaches produce suboptimal results and to formalise two general problems that can afflict search algorithms in games with incomplete information. Thus, the question that motivates us is not "How well will such algorithms play in practice?" but the stiffer "How will the solutions produced by such algorithms compare against the solutions found by experts?"

Repeated minimaxing

The first person to explicitly propose the use of a double-dummy solver as the basis for a program to play Bridge appears to have been Levy [20]. In the paper "The Million Pound Bridge Program", he is confident that this kind of program could win the 1 million pound prize offered by former Bridge World Champion, Zia Mahmood, to

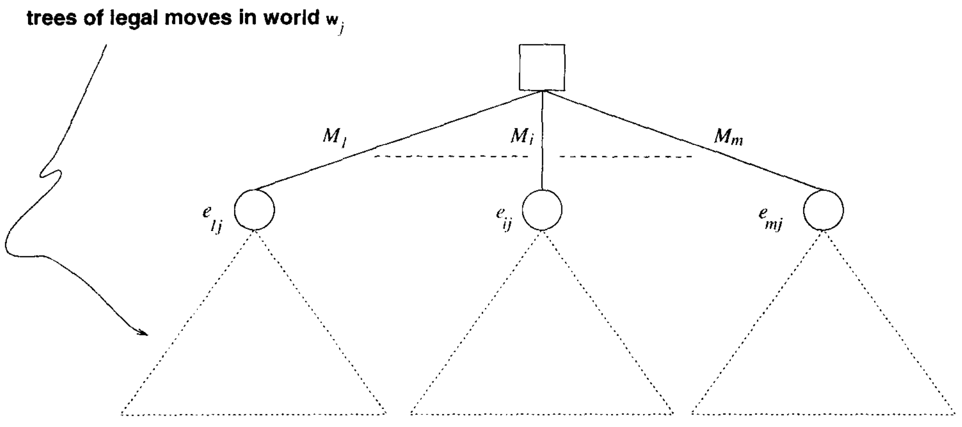


Fig. 9. The minimax values, e_{ij}, of each move M_i under world w_j.

the designers of a computer player that could defeat him. More recently, Matt Ginsberg has produced what he claims to be a "whole new standard" of Bridge-playing program with an architecture based on this principle [35].

Let us describe Levy's algorithm by considering the general problem of selecting MAX's next move in an arbitrary incomplete information game. Suppose that, for some move under consideration, the set of worlds that are consistent with the outcomes of the previous (anterior) moves is given by W. Let us also say that it is possible to choose an n such that when we randomly generate n members, $w_1, \ldots, w_n$ of W, we have sufficient computing resources to find the minimax value of the current (complete information) game tree under each of these worlds. If there are m possible moves, $M_1, \ldots, M_m$, to choose between and we use e_{ij} to denote the minimax value of the ith possible move under world w_j, the situation in this world will be as depicted in Fig. 9.

Levy's proposal was that each legal move, M_i, could be given a score based on its expected payoff. In the context of Fig. 9, Levy's score can be expressed as the scoring function, f:

$$f(M_i) = \sum_{j=1}^{n} e_{ij} prob(w_j).$$

Selecting a move is achieved by actually using the minimax algorithm to generate the values of the e_{ij}s, and determining the M_i for which the value of $f(M_i)$ is greatest. Since this technique relies on repeated applications of the minimax algorithm to problems with perfect information, we will refer to it as *repeated minimaxing*, and to the limiting case where *every* possible world is examined as *exhaustive minimaxing*.

Of course, other possible definitions for the scoring function f can be envisaged, and indeed Ginsberg's notion of selecting "the play that works best on average" [14] suggests the following alternative:

$$f(M_i) = \sum_{j=1}^{n} \left(e_{ij} \overset{?}{=} \max_k e_{kj} \right),$$

where k ranges from 1 to m, and $\overset{?}{=}$ is an infix function that equals 1 if both sides are equal, and 0 otherwise (that is, a move is given a score of 1 for each world in which it is the best, or equal best, alternative). Another possibility (particularly in Bridge, where the objective is usually to win at least a certain number of tricks) is to only consider moves that guarantee a result at least as large as some minimum value, e, say. That is,

$$f(M_i) = \sum_{j=1}^{n} \left(e_{ij} \overset{?}{>} e \right) e_{ij} prob(w_j).$$

However, more important than the choice of the scoring function in our current context is the answer to the more fundamental question of whether Levy is justified in speculating that this kind of architecture is the key to playing the cards "perfectly" (Levy's quotes). Below we show that no possible scoring function can work optimally.

6. How repeated minimaxing fails: strategy fusion

The repeated minimaxing architecture described above is based on looking at a representative sample of possible distributions of outstanding cards and using them to evaluate the best strategy. We use our framework to show that against best defence this may fail, i.e., suboptimal strategies may be returned. Indeed, we show that even an exhaustive minimaxing algorithm, examining *all* the possible distributions, may fail to select the correct strategy. Although our discussion is driven by the problems experienced by repeated minimaxing, we also show that any algorithm which shares specific characteristics with minimaxing will experience the same difficulties. We use examples from Bridge to show how such sub-optimal algorithms lead to improper play in real games.

Let us return to the flattened version of the game tree we first introduced in Section 2, this time labelling the nodes and adding one extra possible path, as shown by the nodes d, e, and f in Fig. 10. If MAX picks the left-hand branch at node d, he will encounter the tree with four possible strategies that we have already examined in Section 4. Alternatively, he may select the right-hand branch, which leads to a subtree in which no further MAX choices are necessary. Thus, the extra branch increases the number of possible MAX strategies by one. Since we have already seen that none of the strategies in the left-hand subtree give a payoff of 1 in more than two worlds, MAX's best option at the root of the tree, guaranteeing a return of 1 under every world, is to select the right-hand branch.

However, using the standard minimax algorithm under any single world, the two MIN nodes a and e will always have a minimax value of 1 (since in any single world MAX can always choose a branch with a payoff of 1 at nodes b, c and f). Applying repeated minimaxing to the root of the tree then, irrespective of the scoring function used, must assign both the left-hand branch and the right-hand branch the

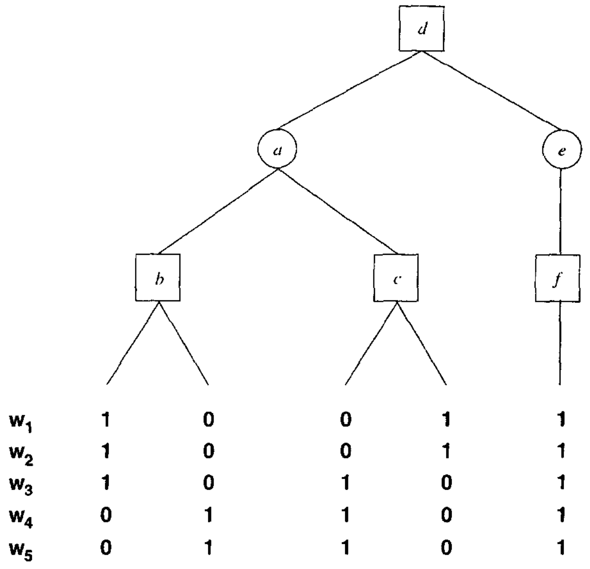


Fig. 10. A tree of MIN and MAX choices that is mis-analysed by exhaustive minimaxing.

same score. Thus, repeated minimaxing will perform no better than making a random choice between the two moves, even in the limiting case of exhaustive minimaxing. Furthermore, reducing the number of worlds in which the right-hand branch gives a payoff of 1 can produce situations in which repeated minimaxing will usually select the incorrect move, and indeed where exhaustive minimaxing will *always* select incorrectly. For example, consider how exhaustive minimaxing behaves in the situation of Fig. 11(a), where the payoffs on the right-hand branch have been modified so that a payoff of 1 is achieved in only three worlds. Despite this modification, this branch still represents MAX's best move at the root of the tree, under the assumption that all the worlds are equally likely. For exhaustive minimaxing, however, it is the other branch that will be selected by any of the scoring functions of Section 5, as the left-hand MIN node has a minimax value of 1 in all five worlds, whereas the right-hand node has a value of 1 in just three. (Repeated minimaxing *may* pick the correct branch, but only if it does not examine either of the worlds w_1 and w_2 and then makes a fortunate guess). Any *sensible* scoring function will never lead exhaustive minimaxing to select the right-hand branch in this situation, as it clearly cannot be rational behaviour, given a set of alternatives to choose between, to prefer an option whose evaluation is always less than or equal to that of one of the others. To see that no *possible* scoring function can cope with all such situations, consider Fig. 11(b). To an exhaustive minimaxing algorithm, this tree will be indistinguishable from that of Fig. 11(a), as the minimax values of the MIN nodes in each are the same under every world. However, in Fig. 11(a) the best move is the right-hand branch, and in Fig. 11(b) the best move is the left-hand branch. Any given scoring function will therefore either make the correct choice in just one of these situations or will be unable to distinguish the best move in either case.

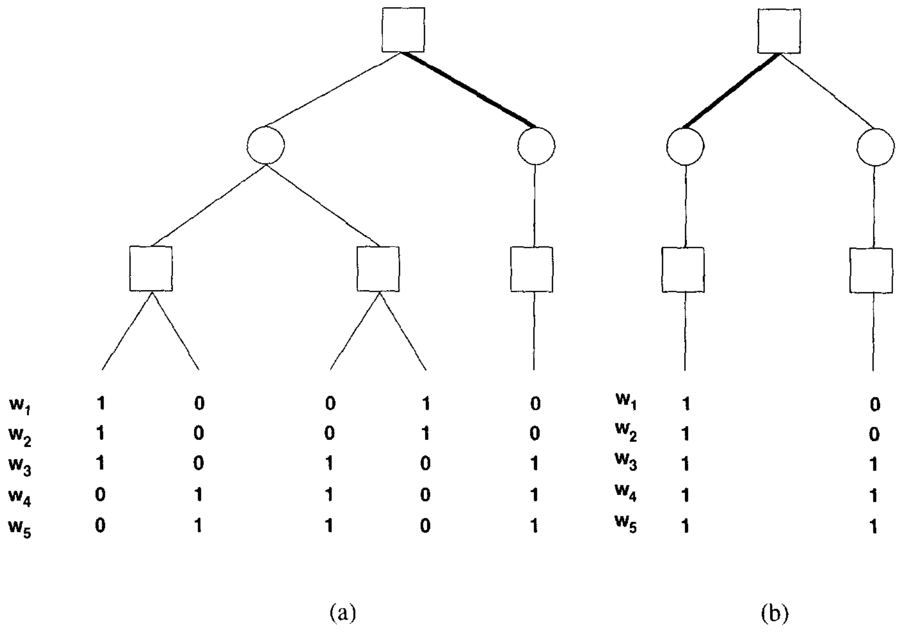


Fig. 11. Two trees with the best initial move marked in bold.

The source of the difficulty that repeated minimaxing experiences on these trees lies in a crucial deviation it makes from the exhaustive strategy minimisation algorithm we presented in Section 4. Recall that for each world that it considers, exhaustive strategy minimisation examines the result of *every* possible strategy. Repeated minimaxing, on the other hand, uses the minimax algorithm to find the *best* strategy in a number of worlds. As we pointed out in Section 4.3, using minimaxing in this way does not respect the constraint imposed by information sets: it allows *different* strategies to be chosen in *different* worlds. Collecting the minimax values of these strategies and assuming that they represent the payoffs that can be expected under each world ignores the fact that a *choice* of a particular strategy has to be made. We therefore call this problem *strategy fusion*. As we saw in Section 4.3, allowing the minimax algorithm to ignore the constraint imposed by information sets results in evaluations greater than or equal to the correct values (e.g., produced by exhaustive strategy minimisation). Repeated minimaxing will therefore have the tendency to over-estimate its scoring of any node; the strategy fusion effect leads it to believe that it has the luxury of choosing a different strategy in each world, instead of the best single strategy across all worlds. For example, given perfect information at node *b* or node *c* it is easy to determine which branch to select to produce a payoff of 1. Node *a* will therefore always have a minimax value of 1 under repeated minimaxing. However, we have already seen that when a *single* strategy selection is enforced, the best that can be achieved is a payoff of 1 in at most two worlds.

Another way to visualise the strategy fusion problem is to note that the repeated minimaxing architecture actually models the task of selecting between some number

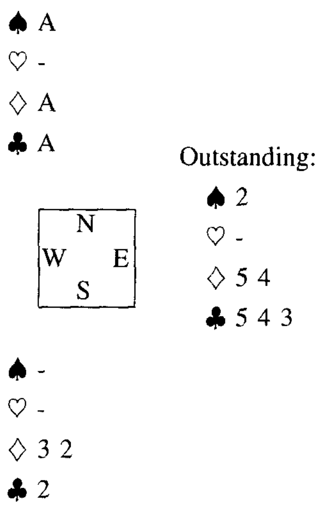


Fig. 12. A Bridge situation requiring a guess over the best strategy.

of perfect information *games* each starting with the same chance move. For example, imagine that the subtrees rooted on nodes *a* and *e* in Fig. 10 represent the MIN and MAX moves in a game starting with a chance move that selects one of the possible worlds $w_1, \ldots, w_5$. Which of these games would we rather play given that they have perfect information (for both players)? This is the question that exhaustive minimaxing answers, and to which repeated minimaxing approximates. It should be clear that such an algorithm will always expect to win the game based on the tree of node *a* and the game based on the tree of node *e*. It should also be clear that the situation modelled by this algorithm is different from the original game as well from the model produced by our assumptions about best defence, since it involves assumptions about MAX, as well as MIN, having perfect information.

6.1. A Bridge example

To see an example of strategy fusion in the game of Bridge, consider the situation of Fig. 12, where we control both the North and the South hands against two opponents who see just their own hand and the North hand, which is the dummy.

Assume that spades are trumps—recall that cards in the trump suit beat cards in every other suit—and that it is South's turn to play. We are worried about the last outstanding trump (the 2♠) but cannot force the opposition to play this card by leading the A♠ because currently it is not North's turn to play, and South has no cards in the suit. However, whichever opponent has the last trump must also have at least one diamond or one club. Since players must always follow the suit of the card that starts a trick, it is therefore possible to win all the remaining tricks by leading the suit that the opponent with the last trump holds, winning in the North hand with the Ace of that suit, and then clearing the trumps with the A♠. In reality, this choice between leading a diamond or a

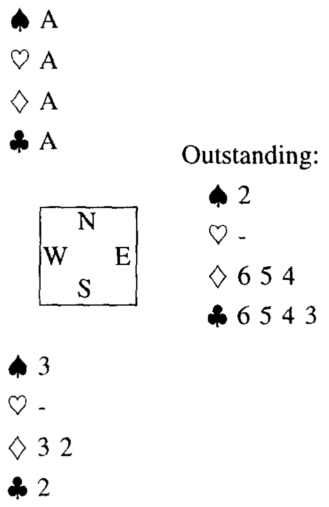


Fig. 13. A Bridge example where strategy fusion obscures the best strategy.

club is a *guess*, but a double-dummy program will find that it is *always* possible because it has perfect knowledge of every world. A Bridge situation where repeated minimaxing is misled by strategy fusion can therefore be constructed by adding four cards to the above situation, as in Fig. 13.

In this new situation, let us assume that the lead is now in the North hand, so that there are four possible moves: the lead of the Ace of any suit. Choosing either of the diamond or the club suits will be (correctly) evaluated by a repeated minimaxing algorithm as less than 100% plays, provided the algorithm examines at least one world in which the Ace may lose to a trump played by an opponent. Choosing the spade suit, on the other hand, will be (correctly) evaluated as a 100% play, since after the A♠ is played there will be no remaining trumps and North's other Aces will be guaranteed winners. However, exhaustive minimaxing will also assess the Ace of *hearts* to be a 100% play. This is because a trump can be played on this card from the South hand (winning the trick) and then the North hand re-entered by making a guess between diamonds and clubs as described above. This play, of course, is clearly not guaranteed to succeed, as would be revealed by an algorithm such as exhaustive strategy minimisation, which would *separately* evaluate the strategies of re-entering the North hand with a club or with a diamond, finding that they would each lose under some of the possible worlds.

6.2. *Further discussion: a tendency to delay*

To our knowledge, the concept of strategy fusion has not been formalised before. Although others have noted that their algorithms may not "play the same way as humans", the absence of a formal model of the assumptions made by experts when solving such problems has meant that the qualitative differences in performance have

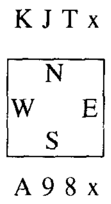


Fig. 14. An example card combination used by Ginsberg (again, North is dummy).

been difficult to describe. Thus, while others have occasionally been aware of some peculiarities in performance, a way to formally describe and analyse such occurrences has been lacking. By providing such a formalisation, we are able not only to pinpoint precise problems such as strategy fusion, but also to make more general characterisations, for example concerning the way in which repeated minimaxing will deal with decisions.

To illustrate, consider the card combination of Fig. 14 introduced by Ginsberg. This is a single suit situation, and an "x" represents an arbitrary low card (as in the example of Section 3.2.1). The essence of this situation is that if the location of the missing Queen is known, it will always be possible to win four tricks. To see this, consider what happens if we know that West holds the Queen. We start by playing a low card from the South hand, and if West plays the Queen we win with the King and cash the remaining top cards. If West chooses *not* to play the Queen, we win with the Jack or the Ten (this play is called a *finesse*), and repeat the same procedure on the following trick. Similarly, we can always win the trick if we know that East holds the Queen, this time by leading low from the North hand. In practice, the location of the Queen is unknown and must be guessed; a wrong guess will allow the Queen to defeat the finesse. However, a double-dummy program will always see the position of the cards and "know" which opponent to play for the card.

Ginsberg correctly realises this problem, pointing out that a double-dummy program will "assume that it can always play KJTx opposite A98x for no losers". However, he also suggests that such an algorithm "won't mind playing this suit by starting with the 9 (for example)", whereas human players "might play the Ace first to cater to a singleton Queen on our right". To see why this is not the whole story, consider a game tree composed of just the node b in Fig. 15.

We have already seen that under any single world the minimax value of node b is always 1, and that such nodes can lead to strategy fusion. However, in this new situation the node is at the *root* of the tree of possibilities. If we use repeated minimaxing to analyse this tree, the best move is found by determining the branch that has the highest score under some function f. This *separate* examination of each branch removes the freedom to select the best strategy for each world, and since neither of these branches alone can guarantee a payoff of 1 under every world, there *must* now be a payoff of 0 in some worlds. In effect, then, node b represents a point where a *choice* between strategies that win under different worlds has to be made. If such a decision point occurs within a game tree, as in Fig. 10, a repeated minimaxing algorithm will deceive itself

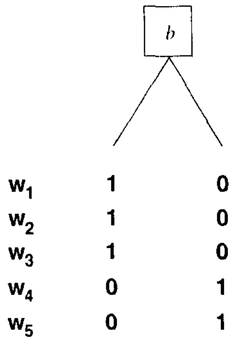


Fig. 15. A simple tree with just one MAX choice.

into thinking that it can always make the correct choice. If the decision point occurs at the root of the tree as in Fig. 15, however, some of the freedom to choose different strategies in different worlds is lost. A repeated minimaxing algorithm will therefore have a tendency to *delay* such decision points wherever possible, as "doing it later" will always appear to be better than "doing it now". Only when it actually reaches a point where a choice *has* to be made will it realise that its previous evaluations were inflated by the effects of strategy fusion.

To see how this is relevant in Ginsberg's example, observe that in order to be able to always win the finesse for the outstanding Queen against *either* East or West, it is necessary to retain both the Ace and the King. Starting a trick by actually playing either of these cards, then, will remove this ability, making it impossible to win all the tricks in the suit under every world. Repeated minimaxing will therefore correctly evaluate the play of the Ace or the King at less than 100% if it selects any of these worlds to examine. The play of *any other* card in the suit, however, leaves the ability to finesse against either opponent intact, always resulting in an (over-estimated) evaluation of 100%. Therefore, it is not that repeated minimaxing is indifferent over the card to lead in the suit, as Ginsberg suggests, but rather that it will *prefer* to play one of the J, T, 9, 8, x. Indeed, an exhaustive minimaxing algorithm will *only* choose to play these cards.

Sometimes the delaying of a crucial decision can be important, as the later an actual choice is made, the more information there is likely to be to inform the decision. For example, consider the situation in Fig. 16, where the aim is to win *all* the tricks and there is no trump suit.

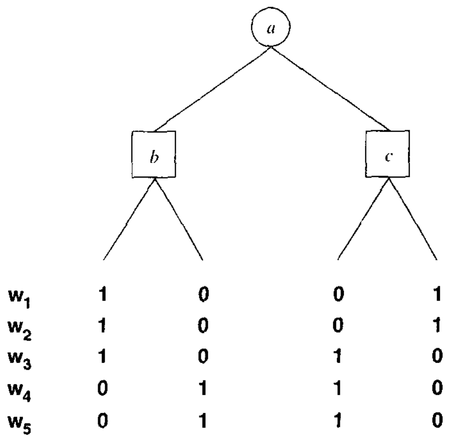
The spade suit is identical to that of Ginsberg's example, so there is again a guess over the position of the Queen. However, this is not necessarily a complete guess. As Frank [7, p. 207] points out in a similar example, "by playing out the winners in the other three suits, information will be gained which will increase one's chance of making the right play, possibly up to certainty". It might be hoped that the tendency of a repeated minimaxing algorithm to delay such decisions would result in this *discovery play* being made. However, as we have already seen, although the Ace and King are less than 100% plays, the lower cards in the suit are not, so a repeated minimaxing algorithm may well choose to play the spade suit early.

♠ K J T x
♡ A K Q
♢ A K Q
♣ x x x

	N	
W		E
	S	

♠ A 9 8 x
♡ x x x
♢ x x x
♣ A K Q

Fig. 16. An example deal in Bridge. South is declarer, North is dummy and there are no trumps.



w_1	1	0	0	1
w_2	1	0	0	1
w_3	1	0	1	0
w_4	0	1	1	0
w_5	0	1	1	0

Fig. 17. Simple tree of MIN and MAX choices in a domain with five possible worlds.

7. Non-locality (how repeated minimaxing fails again)

The failure of minimax-like algorithms we address here is more subtle than that of strategy fusion and, to our knowledge, has not been studied before. To illustrate it, we will refer once more to the flattened version of the tree introduced in Section 2, which we repeat in Fig. 17 for ease of reference.

When actually playing the game represented by this tree, MAX will only ever have to make one decision: the selection of a move at either node *b* or node *c* (effectively, one choice for each information set in the game). Let us again assume equally likely worlds. Irrespective of the MAX node at which the play arrives after MIN's move at the root of the tree, repeated minimaxing, using any of the scoring functions from Section 5, will usually select the left-hand branch (in the limiting case of exhaustive minimaxing, the left-hand branch will always be chosen). Thus, the strategy that will typically be chosen by repeated minimaxing, and indeed the one that will always be selected by exhaustive minimaxing will be the one we have identified as (1, 1). However, we have already seen in Section 4 that under the assumption of equally likely worlds, the best strategy selections are actually (1,2) and (2,1)!

The problem here is distinct from that of strategy fusion, and can be traced to a different cause: the way in which a branch selection is made at a node on the basis of an evaluation only of its direct subtree. The inherent assumption in making a branch selection in this way is that the correct move is a function only of the possible continuations of the game. In perfect information situations (i.e., where the position in the game tree is known), this assumption is justified and the minimax algorithm, with its compositional evaluation function, finds optimal strategies. With more than one possible world, however, this assumption is no longer valid. This is because a decision procedure making this assumption considers only *partial* strategies at any internal node of a tree: the actual strategies themselves would have to specify further choices at other MAX nodes in the tree. This is the manifestation of the incompleteness we alluded to in Section 4.3.

To illustrate the discussion here, let us return to the task of selecting a strategy in our example tree. If we just analyse the subtree of node *b*, we see that the left-hand branch appears to be the best choice because it produces a payoff of 1 in three out of the five possible worlds. In the context of the entire game, however, selecting the left-hand branch at node *b* affects the analysis of node *c*. Since the left-hand branch at node *b* produces payoffs of 0 in worlds w_4 and w_5, MIN (who chooses his strategy after MAX and has knowledge of the state of the world) will be able to restrict MAX's payoff to 0 in these worlds *irrespective* of MAX's choice at node *c*. Under this circumstance, it is the right-hand branch that is the best choice at node *c*, since it offers a payoff of 1 in two worlds (w_1 and w_2), compared to the single payoff of 1 (in world w_3) offered by the left-hand branch. Similarly, if we consider making a branch selection at node *b* after choosing a branch at node *c*, we find that the best selection is no longer the one that leads to a payoff of 1 in most worlds.

In general, the choice of a branch at a given MAX node *n* is not simply a function of the payoffs of the paths that contain *n*, but of the payoffs along *any* path in the tree. If MIN can choose a move at an ancestor of *n* that reduces the payoff (in any world) from what MAX would expect by examining *n*'s direct subtree then the best branch at *n* may change. For example, in the case of Fig. 18, the initial selection of branch *a* may be rendered incorrect if reducing some of the e_{ia} can result in the maximum value of the function *f* being achieved at a different branch.

We call this problem of having to consider all other nodes in the tree *non-locality*. To see that no possible scoring function can cope with the effects of non-locality,

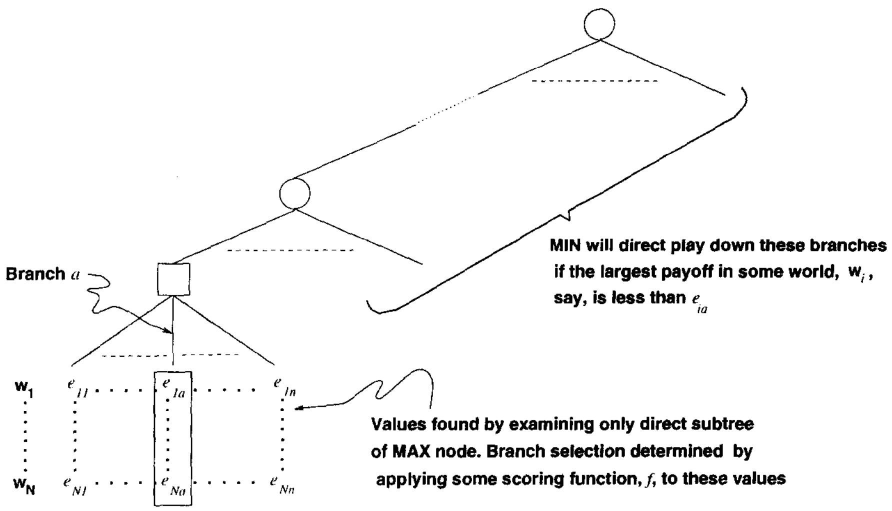

Branch a
MIN will direct play down these branches if the largest payoff in some world, w_i, say, is less than e_{ia}
w_1
w_N
e_{11}
e_{1a}
e_{1n}
e_{N1}
e_{Na}
e_{Nn}
Values found by examining only direct subtree of MAX node. Branch selection determined by applying some scoring function, f, to these values

Fig. 18. A search process in which the selection of branch a may be rendered incorrect by non-locality.

consider the examples of Fig. 19. The left-hand subtree is the same in both trees, and leads to a MAX payoff of 1 in just one of the two worlds. However, in one tree the best choice is the left-hand branch and in the other it is the right (since against a MIN player who knows the actual state of the world, the other strategies in each tree always give a payoff of 0). No algorithm analysing these MAX nodes in isolation will be able to make the correct selection in both situations. Further, non-locality will occur irrespective of the accuracy of the evaluations on which node selection is based. For example, in Fig. 18, non-locality would still be possible whether the e's were produced using repeated minimaxing (risking incorrect values caused by strategy fusion) or by applying exhaustive strategy minimisation to find the results of the optimal strategy for the subtree (therefore producing exact values). Thus, non-locality cannot be eradicated by simply replacing the minimax algorithm with modified versions, such as average propagation [26] or product propagation [28,29]. The back-up rules in such algorithms, when calculating a value to propagate up the tree, take into account the value of *each* subtree at a node. Their use in the repeated minimaxing architecture may therefore reduce the effects of strategy fusion, but non-locality will still be present. Also, the very fact that each node contributes to the result in such algorithms means that search enhancement techniques such as alpha-beta pruning cannot be used to improve efficiency without affecting the values computed.

Non-locality is closely related to the presence of differing levels of information between the players of the game. MIN's ability to lead the play away from a given node under certain worlds by selecting branches higher in the tree relies both on his ability to choose a strategy after MAX (assumption A-II in our best defence model), and knowledge of the actual state of the world (which in our examples is perfect knowledge, from assumption A-I, but need not be so complete). Given these conditions, non-locality will

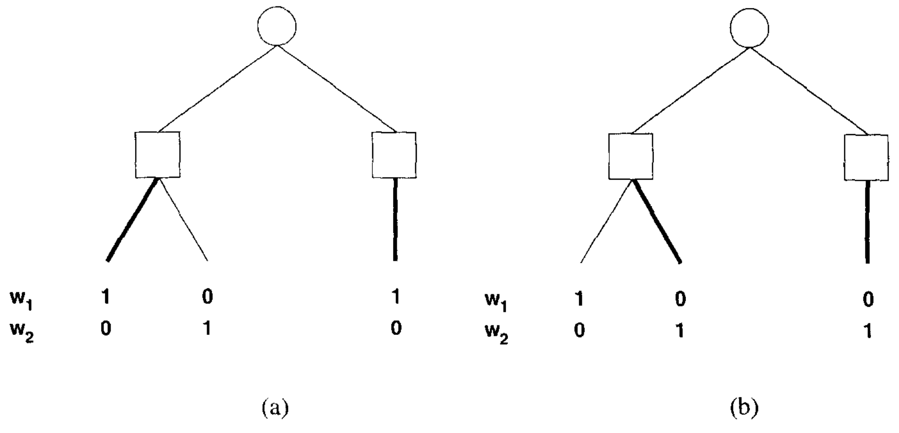


Fig. 19. Two trees with the best MAX strategy marked in bold.

arise in any search algorithm that determines the best choice at a node by analysing just the node's subtree. Note that one consequence of this is that non-locality will only occur at internal nodes of the search space. Thus, it will only be observed with a repeated minimaxing type architecture when a move *other than the first* in a game is analysed. This may explain why the effect has not been formalised before: using partially played game situations as test examples is a far less obvious choice than simply examining initial game configurations.

Currently, we are aware of no correct algorithm for identifying optimal strategies against best defence other than the exhaustive strategy minimisation algorithm of Section 4. This algorithm overcomes both non-locality and strategy fusion by the simple expedient of examining the possible outcomes of each complete strategy separately, but has a complexity that is doubly exponential in the number of a player's moves in the game. [2]

7.1. A Bridge example

We give below an example of non-locality as it can occur in Bridge. Note that it is difficult to construct simple examples for the non-specialist in Bridge. The example below is, however, representative of the kinds of problems that can arise during actual play.

Consider the situation of Fig. 20, where one trick must be lost because the highest remaining card is held by the opposition. For ease of exposition we will assume that the possible options are represented using a slightly higher-level representation: rather than selecting between possible moves, we will examine applicable *tactics* [8, 10]—operators

[2] Recently we have tested a new algorithm, *payoff-reduction minimaxing* [8] that does not suffer from strategy fusion and also reduces the occurrence of non-locality. In [9], we have shown that on simple game trees, payoff-reduction minimaxing significantly out-performs sampling architectures such as repeated minimaxing.

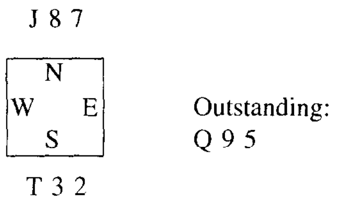


Fig. 20. A Bridge card combination where one trick must be lost.

that specify not just a card to lead on a trick, but also the card to played by the declarer from the third hand, after the first opponent has responded.[3]

We will look at the two tactics that succeed in most worlds. The first of these is a play that begins by leading the 7 from the North hand. If East plays the Queen straight away, declarer plays a low card from the South hand and wins the next two tricks with the Jack and the Ten. If East plays the 9, declarer covers with the Ten and is again guaranteed two tricks. Finally, if East plays low, declarer again plays a low card from the South hand, hoping to find that East also held the 9, thus either forcing out the Queen or winning the trick outright. This play is a particular type of finesse (of the 7) against East. The other possibility we will examine is a different type of finesse, this time of the Jack, against West. This play begins by leading a low card from the South hand, intending to play the Jack unless West plays the Queen. Of the eight possible ways to split the outstanding cards, the distributions under which each of these tactics would produce two tricks are shown in Fig. 21.

With no further information to guide a choice, then, the most promising of the two tactics appears to be the finesse of the Jack. This will fail to win two tricks in only one of the eight possible worlds (which is also the least likely), whereas the finesse of the 7 will fail in two. However, notice that the situation of Fig. 20 can in fact be reached in one round of play (each player contributing one card) from the state where the cards are initially distributed as in Fig. 22.

If the declarer does not know the actual distribution of the outstanding cards, his best play in this situation is to lead the 6 from the North hand and play low from the South hand unless East plays the 9 or one of the King or Queen. Faced with this play, East's best option if he holds the cards shown is to play low with the 5. If he does this, West will win the trick with the 9 and the declarer will be restricted to just two tricks in the

[3] Note that Ginsberg suggests a similar representation when discussing the example of Fig. 14. When leading a card, he proposes that declarer should "decide in advance" the other card he will play on the trick—essentially creating tactics. Within the repeated minimaxing framework, such a representation change has the effect of delaying the onset of strategy fusion for one level of search, since instead of choosing between individual moves—which are really just partial strategies with only the first step determined—the choice is now between options that are partial strategies with the first *two* steps determined. However, the strategy fusion that results from allowing *different* completions of these partial strategies in *different* worlds will still remain beyond this new horizon. Although the use of tactics can indeed "correct" the problems caused by strategy fusion in Ginsberg's example, then, it cannot provide a complete solution unless the "tactics" extend to the end of the play, at which point they become complete strategies.

Possible worlds: East holds	Finesse of the 7	Finesse of the J
Q95	●	●
Q9	●	●
Q5		●
95	●	●
Q	●	●
9	●	●
5		●
void	●	

Fig. 21. Possible worlds under which each tactic will produce 2 tricks.

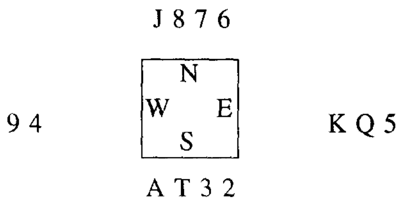


Fig. 22. A Bridge example giving rise to non-locality.

suit. Similarly, if East starts with Kx or Qx, the best option will again be to play low. Thus, if the situation of Fig. 20 is encountered as the result of leading the 6 and having to beat the King with the Ace, there will be extra information about the lie of the cards. Specifically, the cases where East holds the Q5 or the singleton 5 can be ruled out, because under these circumstances another branch, *higher* in the tree, would have been chosen by East to restrict the declarer to just two tricks.

This extra information would have a crucial effect on the situation in Fig. 20. If it was known that this position would *not* be reached under the two distributions where East holds the Q5 or the 5, we can see from Fig. 21 that the finesse of the 7 would then succeed under *all* the remaining worlds, whereas the finesse of the Jack would still fail in one. The probability of the finesse of the 7 producing two tricks in this scenario would therefore be higher than that of the finesse of the J, and the selection made in this position would need to be reversed.

7.2. Estimating the practical significance

As we mentioned in the Introduction, our interest in Bridge arose from our work on a Bridge playing program [8, 10], and in particular from coming to an understanding of why this program could produce analyses different from those found in expert texts. Our program searches for strategies using a set of *tactics* like the ones we used to describe the Bridge example above. These tactics formalise partial strategies for playing one round of cards, and are designed to represent commonly occurring Bridge techniques (such as *playing a winner*, or *finessing*). The use of tactics restricts the search space to the point where, for single-suit problems, all possible tactic combinations can be analysed. However, the tactics do not eliminate any optimal strategies from the search space, as their composition can be used to represent any of the (textually described) strategies found in the Bridge Encyclopedia. Our program solves single-suit problems by conducting a bottom-up pass of the tactic search tree, selecting branches at each node (under the assumption of equally likely worlds) as indicated in Fig. 18, backing up the payoffs in each world of the selected branch.

We tested this program against solutions from the Bridge Encyclopedia, for simplicity restricting attention to the 650 cases where the model strategies were designed to produce the goal of obtaining the maximum possible payoff (ignoring, for example, problems where the Encyclopedia concentrates on *safety plays* that attempt to guarantee a lower than maximum payoff) [8]. The program produced sub-optimal strategies in 218 cases. Hence, non-locality affected 33.5% of the problem set.[4]

We conjecture that one reason for the prevalence of non-locality in these problems is that they are very hard: they were designed as an expert reference and contain no trivial problems (e.g., where all the top cards are held); hence there is always the opportunity for some element of manœuvre. Note, however, that although non-locality appears to be a frequent problem, its consequences need not necessarily spell catastrophe for the use of local algorithms in practice. In an actual playing situation, a more important question than the ability to match expert analysis is the *chance* that a sub-optimal strategy produces a poorer result. For example, in our program the incorrect solutions produce the maximum possible payoff with, on average, a probability of 0.07 less than the model solution. Hence, if we use the program to play the entire Encyclopedia problem set with randomly distributed outstanding cards, the expected number of cases where the maximum possible payoff will be missed is just $0.07 * 218 \approx 15$ cases (about 2.3%).

8. Conclusions

We have looked at the problem of strategy selection in zero-sum two-player games with incomplete information, formalising a best defence model of such games that captures the assumptions implicitly used by Bridge experts in analysing play situations. We showed that equilibrium point strategies exist for this model and gave an algorithm, exhaustive strategy minimisation, capable of computing them.

The formalisation of an explicit model allowed us to rigorously evaluate other search algorithms and to understand their limitations in producing expert analysis of Bridge play. Our model allowed us to precisely demonstrate that algorithms based on mini-

[4] More recently, Ginsberg has reported similar findings on complete Bridge deals using a fast repeated sampling approach [15]. On a hard test set taken from the Bridge tutoring program Bridge Master [16], Ginsberg's program failed to solve 35.6% of the problems [13].

maxing must return suboptimal results, independently of their evaluation function or backup rule. Furthermore, our formalisation of exhaustive strategy minimisation allowed us to pinpoint exactly the sources of sub-optimality in repeated minimaxing: strategy fusion results from combining different MAX strategies in different possible worlds, and non-locality results from examining only partial strategies at internal nodes of a game tree. We gave experimental evidence that non-locality occurs often in actual systems.

While our results provide a clear means of understanding the performance of search algorithms against the commonly used model of expert Bridge play, we caution against using them to judge the practical merits of systems designed for man-machine play. Given the poor complexity of exhaustive strategy minimisation, there may be a real cost involved in overcoming the effects of incomplete information that is unacceptable in practice. However, we have not yet established a lower bound for the complexity of computing equilibrium point strategies for the reduced game model; this remains an open—and apparently important—problem for the design of programs that play games like Bridge.

Acknowledgements

Our thanks go to Alan Bundy and Brian Drabble for their discussions and feedback on this work. The first author was funded by a Science and Engineering Research Council (SERC) Advanced Studentship, award number 92314771. The second author was funded by the German Ministry for Research and Technology (BMFT) under grant ITS 9102. The responsibility for the contents lies with the authors.

References

[1] ACBL, The Official Encyclopedia of Bridge, 5th ed., American Contract Bridge League, Inc., 2990 Airways Boulevard, Memphis, TN 38116-3875, 1994.

[2] E.R. Berlekamp, Program for double-dummy Bridge problems—a new strategy for mechanical game playing, J. ACM 10 (4) (1963) 357–364; also in: D. Levy (Ed.), Computer Games II, Springer, New York, 1988.

[3] D.L.S. Berlin, Span: integrating problem solving tactics, in: Proceedings IJCAI-85, Los Angeles, CA, 1985, pp. 1047–1051.

[4] J.R.S. Blair, D. Mutchler, C. Liu, Games with imperfect information, in: Games: Planning and Learning, Papers from the 1993 Fall Symposium, AAAI Press, 1993, pp. 59–67.

[5] G. Carley, A program to play Contract Bridge, Master's Thesis, Department of Electrical Engineering, Massachussets Institute of Technology, Cambridge, MA, 1962.

[6] E. Crowhurst, Personal communication, May 17, 1996 (Crowhurst made a point of crediting Dorothy Truscott with helping him to verify his results).

[7] A. Frank, Brute force search in games of imperfect information, in: D.N.L. Levy, D.F. Beal (Eds.), Heuristic Programming in Artificial Intelligence 2—The Second Computer Olympiad, Ellis Horwood, Chicester, UK, 1989, pp. 204–209.

[8] I. Frank, Search and planning under incomplete information: A study using Bridge card play, Ph.D. Thesis, Department of Artificial Intelligence, Edinburgh, 1996; also: Distinguished Dissertations Series, Springer, Berlin, to appear.

[9] I. Frank, D. Basin, Payoff-reduction minimaxing, in: Proceedings Fourth Game Programming Workshop in Japan (GPW-97), Hakone, Japan, 1997; also: Technical Report ETL-97-21, Electrotechnical Laboratory, Umezono, Japan.

[10] I. Frank, D. Basin, A. Bundy, An adaptation of proof-planning to declarer play in Bridge, in: Proceedings ECAI-92, Vienna, Austria, 1992, pp. 72–76; longer version: DAI Research Paper No. 575, Edinburgh.

[11] B. Gambäck, M. Rayner, B. Pell, An architecture for a sophisticated mechanical Bridge player, in: D.N.L. Levy, D.F. Beal (Eds.), Heuristic Programming in Artificial Intelligence 2—The Second Computer Olympiad, Ellis Horwood, Chichester, UK, 1991, pp. 87–107; later version: Technical Report 299, Cambridge University Computer Laboratory, 1993.

[12] E. Geake, Playing to win, New Scientist (19 September 1992) 24–25.

[13] M. Ginsberg, GIB vs Bridge Baron: results, Usenet news message posted on the newsgroup rec.games.Bridge, 31 October 1996; Message-id: <56cqmi$914@pith.uoregon.edu>.

[14] M. Ginsberg, How computers will play Bridge, The Bridge World, 1996; also available for anonymous ftp from dt.cirl.uoregon.edu as the file /papers/Bridge.ps.

[15] M. Ginsberg, Partition search, in: Proceedings AAAI-96, Portland, OR, 1996, pp. 228–233.

[16] F. Gitelman, Bridge master, Computer Bridge Program, 15 Lillian Street, Toronto, ON, Canada M4S 2H7.

[17] D. Kibler, K. Schwamb, Complete contingency planners, Technical Report UCI TR 92-94, University of California, Irvine, CA, 1992.

[18] R.E. Korf, Generalized game trees, in: Proceedings IJCAI-89, Vol. 1, Detroit, MI, 1989, pp. 328–333.

[19] R. Lee (Ed.), Reviews from Canadian Master Point, 1992–1994; available as the file /pub/Bridge/FAQ.ReviewsFromCanadianMasterPoint from the site arp.anu.edu.au.

[20] D.N.L. Levy, The million pound Bridge program, in: D.N.L. Levy, D.F. Beal (Eds.), Heuristic Programming in Artificial Intelligence—The First Computer Olympiad, Ellis Horwood, Chichester, UK, 1989, pp. 95–103.

[21] R.D. Luce, H. Raiffa, Games and Decisions—Introduction and Critical Survey, Wiley, New York, 1957.

[22] R. Lustig, Hibrid: a hierarchically designed Bridge playing program, Master's Thesis, Weizmann Institute of Science, Rehovot, Israel, 1985.

[23] B. Manley, Bridge software: moving forwards or backwards?, The Bulletin, American Contract Bridge League (October 1994).

[24] C.N. Napjus, Declarer: a learning Bridge-playing program which generalizes and infers, Ph.D. Thesis, University of Washington, 1969.

[25] J.F. Nash, Non-cooperative games, Ann. of Math. 2 (54) (1951) 286–295.

[26] D. Nau, P. Purdom, C.-H. Tzeng, An evaluation of two alternatives to minimax, in: L.N. Kanal, J.F. Lemmer (Eds.), Uncertainty in Artificial Intelligence, Elsevier, Amsterdam, 1988, pp. 505–509.

[27] Y. Nygate, L. Sterling, Aspen—designing complex knowledge based systems, in: Proceedings 10th Israeli Symposium on Artificial Intelligence, 1993, pp. 51–60.

[28] J. Pearl, Heuristic search theory: survey of recent results, in: Proceedings IJCAI-81, vol. 1, Vancouver, BC, 1981, pp. 554–562.

[29] J. Pearl, On the nature of pathology in game searching, Artificial Intelligence 20 (1983) 427–453.

[30] C.E. Shannon, Programming a computer for playing chess, Philos. Mag. 41 (1950) 256–275.

[31] S.J.J. Smith, D.S. Nau, Strategic planning for imperfect information games, in: Games: Planning and Learning, Papers from the 1993 Fall Symposium, AAAI Press, 1993, pp. 84–91.

[32] A. Stanier, Planning to make tricks at Bridge, in: Proceedings AISB Summer Conference, 1976, pp. 256–265; also in: D. Levy (Ed.), Computer Games II, Springer, New York, 1988.

[33] T. Throop, Computer Bridge, Hayden, 1983.

[34] J. von Neumann, O. Morgenstern, Theory of Games and Economic Behaviour, Princeton University Press, Princeton, NJ, 1944.

[35] G. Walker, Beaten in spades, New Scientist (16 November 1996) 26–27.

[36] R. Wheen, Solving double dummy Bridge problems by exhaustive search, in: D.N.L. Levy, D.F. Beal (Eds.), Heuristic Programming in Artificial Intelligence—The First Computer Olympiad, Ellis Horwood, Chichester, UK, 1989, pp. 89–94.