Search in Games with Incomplete Information: A Case Study using Bridge Card Play

Ian Frank David Basin

DAI Research Paper No. 780

January 11, 1996

Submitted to *Artificial Intelligence*

Department of Artificial Intelligence
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
Scotland

© Ian Frank, David Basin

Search in Games with Incomplete Information: A Case Study using Bridge Card Play*

Ian Frank David Basin

January 11, 1996

Abstract

We examine search algorithms in games with incomplete information. We proceed by adopting the standard game theoretic framework of zero-sum two-player games, introducing a reduced form of incomplete information games in which equilibrium point strategies can be identified, and giving an algorithm that finds such strategies. We apply our formalisation to the game of Bridge, for which the reduced form corresponds to the typical model analysed in expert texts. In particular, we examine search algorithms for Bridge that have been proposed by other researchers, countering the optimistic suggestion that the incomplete information in the game can be tackled by an algorithm which considers some statistically significant subset of the possible worlds (card distributions). We use our model to pinpoint why, both in theory and practice, such approaches must lead to suboptimal performance and we generalise the reasons for this failure into two specific and distinct problems. We illustrate these problems and our solution with simple game trees and with actual play situations from Bridge itself.

1 Introduction

We examine the problem of choosing a strategy in two-player games with incomplete information. Our formalisation is based on the introduction of incomplete information into the standard framework of strategy optimisation for zero-sum two-player games [Nash 51]. To aid analysis of actual games, we formalise a particular *reduced* model of a game which corresponds to the often-made assumption of playing against a rational, informed opponent. We show for such games that an *equilibrium point* for the two players' strategies is well-defined, and we describe an algorithm which we call *exhaustive strategy minimisation* that identifies such equilibrium points. This algorithm formalises the idea of selecting a strategy directly, by explicitly enumerating the possibilities, computing the minimum payoff of each, and then selecting the strategy with the best payoff. By explicitly manipulating strategies, our algorithm has different properties from standard minimaxing and produces more accurate results than adaptations of minimaxing (discussed below) to domains with incomplete information.

Our interest in this problem arose from our own work in designing a system to play Bridge [Frank *et al* 92] and the realisation that standard search algorithms, such as minimaxing, could not be employed to find optimal strategies in this domain. Search problems, both theoretical and practical, arise in Bridge because information is incomplete. That is, the players lack knowledge about the actual 'state of the world' — for example, which opponents have which cards. Yet the effects of incomplete knowledge on search in this domain are typically either not addressed or are mis-analysed in the literature on computer Bridge. Perhaps because of this, Bridge card play itself has proven to be a surprisingly difficult task to automate: a history of academic research [Carley 62, Napjus 69,

*We thank 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.

Stanier 76, Lustig 85, Gamback *et al* 91, Frank *et al* 92, Kibler & Schwamb 92, Nygate & Sterling 93, Smith & Nau 93] and a proliferation of more than two dozen commercial software packages [Throop 83, Lee 94], has failed to produce solutions or systems capable of competing with even good novice human players [Geake 92, Manley 94].

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 [Berlin 85]), or by removing the uncertainty over incomplete information altogether and considering instead the problem of finding the best strategy when all the players reveal their cards to each other [Berlekamp 63, Wheen 89]. The latter of these approaches has prompted some researchers, for example Levy [Levy 89] and Ginsberg [Ginsberg 95], 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) would enable the minimax value of each possible action to be established in each of these randomly generated sub-problems, and the best overall action to be predicted by suitably combining these values.

We use our formalisation to show that, even in very simple incomplete information games, algorithms like those suggested by Levy and Ginsberg can yield suboptimal results. In particular, we identify two serious problems which afflict such algorithms, independently of how many worlds are considered. The first, which we name *strategy mixing*, affects any algorithm which attempts to combine together 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 the play may not be known to a player. This fact imposes a constraint on a player's strategy that he must behave the *same* way in *all* possible worlds at such points; a constraint which is typically broken when combining together strategies designed for individual worlds.

The second problem, which we name *non-locality*, arises when the players of a game have differing information on the possible worlds. An opponent with knowledge of the likely possible worlds can use this to his advantage. In particular, he can direct play towards the portions of the tree of possible moves which 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 which nodes in the search space will be reached under which 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 that node. As in strategy mixing, the problem here is one of handling the notion of strategy incorrectly. By just evaluating subtrees, such algorithms consider 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 the subtree.

We show that in incomplete information games, both strategy mixing and non-locality can lead to sub-optimal performance, illustrating our arguments using simple game trees and with actual play situations from Bridge itself. In the context of Bridge, we also show that our model corresponds closely to the form of the game typically analysed in expert texts, thus raising serious questions about the possibility of achieving true expert-level Bridge with the type of architecture suggested by Levy and Ginsberg.

We proceed as follows. In §2 we introduce preliminary concepts from game theory and apply these in §3 to games with incomplete information; in particular we show how Bridge can be analysed within this framework by introducing a reduced form of such games. In §4 we present a general algorithm which finds optimal strategies in reduced games. In the second half of the paper we consider Bridge in some detail: in §5 we present proposed architectures for Bridge card play based on the minimax algorithm, and in §6 and §7 we use our game theoretic framework to identify why such architectures must yield suboptimal results. Afterwards, we draw 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 & Morgenstern 44] and [Luce & Raiffa 57].

2.1 The Extensive and the Normal Forms of 2-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, and a *play*, α, of the game involves starting at this 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 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 of nodes between which a player will not be able to distinguish. We will follow [Luce & Raiffa 57] in referring to these sets as *information sets*. We will also model our actual definition on that of [Luce & Raiffa 57, pages 39–51], requiring the specification of a 2-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 $1, 2, \cdots$) 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.

Figure 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 payoff he will receive (represented by the numbers at the leaf nodes) therefore depends on information which is not known to him.

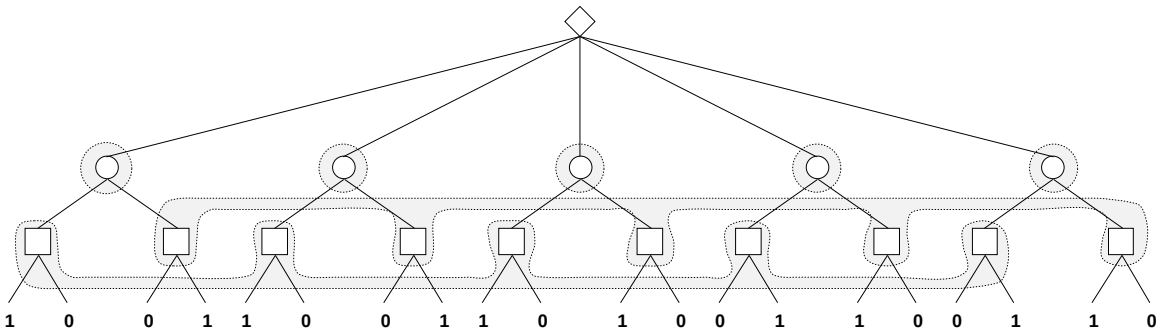


Figure 1: Information sets in the extensive form of a two player game with one chance move

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 $1, \cdots, r$, 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 2-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 $K_2(x_1, x_2)$ which 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 $(\hat{x}_1, \hat{x}_2)$ determines a unique play of the game, α, and we can define $K_i(\hat{x}_1, \hat{x}_2) = K_i(\alpha)$. If the game contains chance moves then $(\hat{x}_1, \hat{x}_2)$ instead impose a probability distribution over all the possible plays. If we use $prob(\alpha)$ to denote the probability of play α occurring when $(\hat{x}_1, \hat{x}_2)$ are chosen, the payoff for each player is expressed in terms of the mathematical expectation $K_i(\hat{x}_1, \hat{x}_2) = \sum_\alpha prob(\alpha) K_i(\alpha)$.

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 of the concept of an *equilibrium point*, due to J. Nash [Nash 51]. 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 $(\hat{x}_1, \hat{x}_2)$ is an equilibrium point if the following hold.

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

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

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

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* 2 chooses x_2, so that 2 makes his choice in full knowledge of the particular x_1 decided on by 1. Since 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 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 1 is the first to select a strategy. For any particular selection, $\hat{x}_1$, 2 will choose an x_2 to minimise the value of $K_1(\hat{x}_1, x_2)$. Thus, when 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 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 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 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 logically 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 [Shannon 50].

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 $\vec{p} = (p_1, \cdots, p_m)$ and $\vec{q} = (q_1, \cdots, q_n)$ $(p_i, q_i \geq 0, \sum p_i = 1, \sum q_i = 1)$ where p_i gives the *probability* that 1 will select the ith member of X_1 as his strategy and q_i is the probability that 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

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

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* (a term which should not be confused with the phenomenon of *strategy mixing*, which we introduce later). 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 $M_1, M_2, M_3, \cdots$, we can define the moves which are *anterior* to some personal move M_k as being the sequence $M_1, \cdots, M_{k-1}$. 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 $M_1, \cdots, M_{k-1}$, 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 $i \in \{1, \cdots, k-1\}$, 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 §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 [von Neumann & Morgenstern 44, page 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 [von Neumann & Morgenstern 44, page 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 Equilibrium Points in Bridge

With the definitions of the previous section behind us, we can now formalise the particular problem that we address in this paper. Since our motivation is the study of Bridge as a game with incomplete information, we first describe the game and its characteristics in more detail, and then present some simplifying assumptions that will aid analysis of the problem.

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 four of these to A, K, Q, and J). 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 > ... > 2) of the suit led. The only exception to this 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.

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 which are not known to the other [Korf 89]. His description 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 §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, since the absence of perfect information will entail solutions which are mixed strategies; there is an advantage to preventing the opponent from 'finding out' your strategy.

3.2 The Reduced Form 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 [Frank & Basin 95]). Thus, rather than tackling the actual game directly, we will make the following simplifying assumptions.

A–I MIN has perfect information (*i.e.*, preliminarity and anteriority coincide for MIN). This cuts the cycle of reasoning about beliefs 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, this assumption removes the need for MAX and MIN to signal information about their holdings.

A–II MIN chooses his strategy after MAX. This is the same assumption as that made by Von Neumann and Morgenstern in reducing a game to its minorant form. As indicated in §2.2, the outcome of the game can then be specified as a function of MAX's strategy only, thus aiding formal analysis.

A–III The strategy adopted by MAX is a pure strategy (as defined in §2.3). This enables us to search for the optimal strategy from among a finite (although possibly very large) set, in contrast to the set of possible mixed strategies.

We will refer to the result of transforming any game by making these modifications as its *reduced* form. It may seem that the assumptions are so severe that analysis of the reduced version will tell us very little about the original game. However, we can justify the transformation by noting that, at least in the context of Bridge, it is very common to see this reduced form of the game implicitly used in expert texts, which often analyse play problems under exactly the assumptions given formally above. For any given example, such texts typically start by selecting a small number of promising strategies. For each of these strategies, its chance of success is then found by enumerating the possible card distributions and analysing the return produced when the opponents are allowed to choose their best strategy in each perfect information situation. The Bridge Encyclopaedia [ACBL 94], for example, devotes 56 pages to presenting the best strategies (with percentage chances of success) for different card combinations, all of which are analysed in this way. Notice that such an analysis is precisely the situation modelled by the assumptions A-I and A-II: first MAX picks a strategy, and then a MIN with perfect information about the outcome of all prior chance moves is allowed to select the best response. This kind of MIN opponent is one which knows the way in which MAX plays, and will always use this knowledge to be ready with the best possible response to MAX's actions in any world.

Under the 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. However, many Bridge books, including the Bridge Encyclopaedia, restrict their consideration to pure strategies as solutions to these situations. This restriction may either be motivated by a desire to make the game easier to analyse (as in our case above), or may simply be an error of oversight. Whatever the reason, such books typically refer to this as choosing the best strategy 'against best opposition' or 'against best defence', and the formulation is presumably popular because the payoff that can be obtained against the strongest opposition can be treated as a kind of lower bound on the payoff which can be expected when the opponents are less accomplished. Moreover, we are not aware of any way of analysing the actual game, with all the inherent difficulties identified in the previous sections. We will therefore follow the tradition of using an analysis of the reduced version to produce useful insights into the actual game itself.

3.3 Solving the Reduced Form

What does it mean to solve the reduced form of an incomplete information game? One of the main differences between the reduced form and the actual game is that MIN chooses his strategy after MAX. This makes the reduced form similar to the minorant form of a perfect information game, for which, as we saw in §2.2, the value to be expected by MAX (player 1) is:

$$v_1 = \max_{x_1} \min_{x_2} K_1(x_1, x_2)\,. \tag{4}$$

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 our reduced form, however, there is a complication. MIN (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, however, is not party to the same information as MIN, so he will be unable to directly identify whether any x_2 will be optimal for MIN. Specifically, since he 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 knowledge of the outcome of these moves. In the following section we will introduce an algorithm which deals with this problem by identifying (possibly different) choices of x_2 which are optimal under *each* of the possible outcomes of these moves. After, we discuss the implications of this for games with incomplete information in general, and also more specifically for some algorithms which have been proposed for the game of Bridge.

4 Exhaustive Strategy Minimisation

The algorithm we propose comes from directly computing (4): we enumerate all strategies x_1 for player 1 (MAX) and for each of them we separately and exhaustively evaluate $\min_{x_2} K_1(x_1, x_2)$ by examining each possible x_2 under each possible outcomes of the chance moves. Since this carries out a minimisation operation for each MAX strategy, we call the algorithm *exhaustive strategy minimisation*.

4.1 The Algorithm

Assume that for t a tree, $node(t)$ returns its root node and $sub(t)$ computes the set of immediate subtrees of $node(t)$. Figure 2 then defines exhaustive strategy minimisation. Note that this algorithm is very simply defined to directly compute (4). It is therefore not necessarily the best implementation in terms of efficiency — an issue we will discuss in more detail in §4.5. The assumptions involved in modifying a game into its reduced form are central to the correctness of this algorithm. Firstly, to be able to actually form the set of MAX strategies, $\mathcal{S}$, assumption A–III restricting consideration to pure strategies is needed. Secondly, 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

Algorithm $esm(\Gamma)$: Returns optimal strategies for player 1 (MAX) in the reduced, extensive form, 2-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 $I_1(n) = i$.
- 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.

where, $esm(t, s)$ takes 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)	$\min_{t_i \in sub(t)} esm(t_i, s)$
$\mathcal{P}(node(t))$ is 1 (*i.e.*, MAX to move)	$esm(t_i, s)$, where i is the $I_1(node(t))$th element of s.
$\mathcal{P}(node(t))$ is 0 (*i.e.*, chance move)	$\sum_{t_i \in sub(t)} \pi(node(t_i))\, esm(t_i, s)$

Figure 2: The exhaustive strategy minimisation algorithm

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 (4).

4.2 An Example

To illustrate this algorithm, we will consider again the example of Figure 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 Figure 1 is the set '1', and the 'upper' is the set '2': the first of the tuples above then corresponds to selecting the left-hand branch at every MAX node; the second 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 (4), 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 Figure 3, 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 this move are all equally likely, then, the evaluation of this strategy is 1/5.

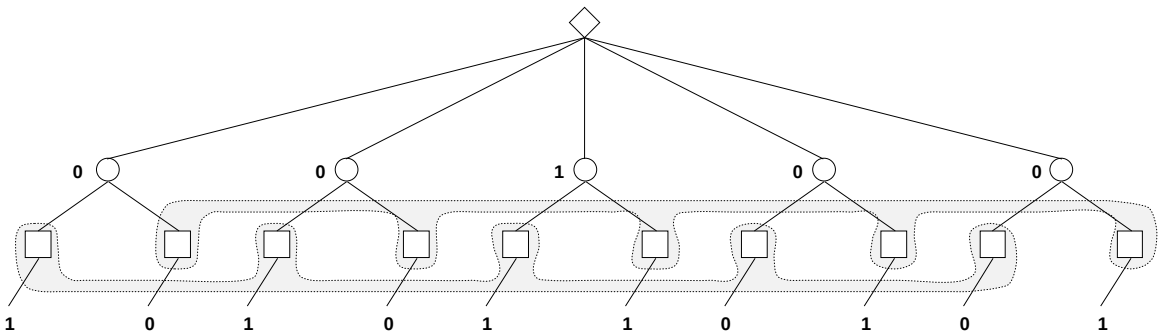


Figure 3: MAX's expected payoffs when selecting strategy (1,1)

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. The two strategies which maximise (4) are therefore (1,2) and (2,1), if we assume equally likely outcomes for the chance move.

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 the players have perfect information. For comparison, in Figure 4 we give a formalisation of the minimax algorithm on games in extensive form.

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)	$\sum_{t_i \in sub(t)} \pi(t_i)\, mm(t_i)$

Figure 4: Simple adaptation of the minimax algorithm to reduced, 2-player, extensive form games

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

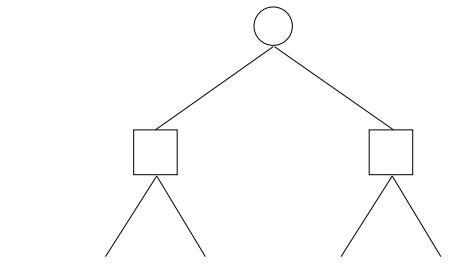
outcomes of the chance moves by determining a course of action for each MAX node on the basis of the subtree with the largest minimax value. It should be clear that this approach will always produce an evaluation which is greater than or equal to the maximum E_j produced by the $esm(\Gamma)$ algorithm. To see this, consider any MAX node (the only node at which the actions of the $mm(\Gamma)$ and $esm(\Gamma)$ algorithms differ substantially). 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:

- $mm(\Gamma)$ does *not* respect the constraint imposed by information sets. That is, it does *not* always choose the same branch at the nodes which belong to the same information set.
- Since $mm(\Gamma)$ commits to *one* branch selection at each MAX node, it risks incompleteness.

In §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 mixing*, and the second leads to the phenomenon of *non-locality*.

4.4 Possible Worlds

The tree of Figure 1 contains just one chance node, and this node is located at the root. Let us say that in a tree which 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. Consider the special case of a tree which contains just one chance node, and where that chance node occurs at the root of the tree. If, with the exception of the payoffs and the MIN information sets, the subtrees in each possible world are identical, we will say that the tree can be 'flattened' by representing the possible worlds in the vertical dimension as differing payoffs at the leaf nodes, rather than in the horizontal dimension using information sets. For example, Figure 5 shows how the tree of Figure 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.



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

Figure 5: Flattened tree of MIN and MAX moves in a domain with five worlds

We will use such trees in the remainder of this paper to compactly and simply present game trees. Notice that the MIN nodes are the result of identifying together nodes with *distinct* information sets, and that a MIN strategy must therefore allow *different* branch choices under each world. A

MAX strategy, on the other hand, must specify the branches to be chosen at each MAX node under *all* worlds.

Note that game trees in Bridge cannot in fact be represented in this way, 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 applicability of the $esm(\Gamma)$ algorithm, of course, is unaffected.

4.5 The Complexity of Exhaustive Strategy Minimisation

Exhaustive strategy minimisation, while identifying optimal strategies, is not cheap: its complexity is significantly worse than conventional minimaxing. To investigate this, we first examine how the set of possible MAX strategies, $\mathcal{S}$, may be generated. We do this by looking at the type of flattened trees we introduced in the previous section, using the complete binary tree of Figure 6 as an example. 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 which 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).

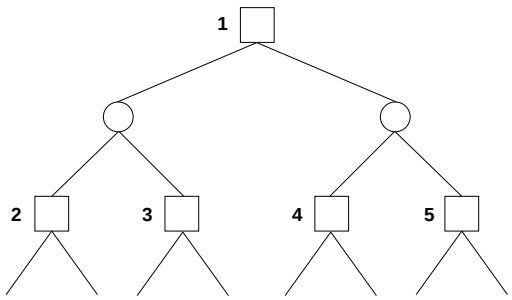


Figure 6: Complete binary tree of MIN and MAX moves

To examine how this set of 5-tuples may be generated, 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. This detail has been glossed over until now, but 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 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 which 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:

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

For complete b-ary trees which 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

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

which has the solution

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

For the example above where $n = 1$, 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

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

For b-ary trees in general, then, the number of strategies that must be examined is doubly exponential in the number of MAX levels which 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 this is 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 the above framework to analyse the game of Bridge and in particular to examine 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 dummy, this scenario is often described as *double-dummy* Bridge. Below, we present the double-dummy architectures proposed by both Levy and Ginsberg [Levy 89, Ginsberg 95]. After, we will use our framework to suggest that this approach cannot scale-up to produce expert play in actual Bridge, and to formalise two general problems that can afflict search algorithms in games with incomplete information.

A Million Pound Bridge Program?

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 [Levy 89]. He was confident that this kind of program could win the prize of 1 million pounds offered by former Bridge World Champion, Zia Mahmood,

to the designers of a computer player that could defeat him. Here, we describe Levy's algorithm by considering the general problem of selecting MAX's next move in an arbitrary incomplete information game.

Let us say that, for some move under consideration, the set of worlds which are consistent with the outcomes of the previous (preliminary) 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, $\mathsf{w}_1, \cdots, \mathsf{w}_n$ 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, $M_1, \cdots, M_m$, 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 Figure 7.

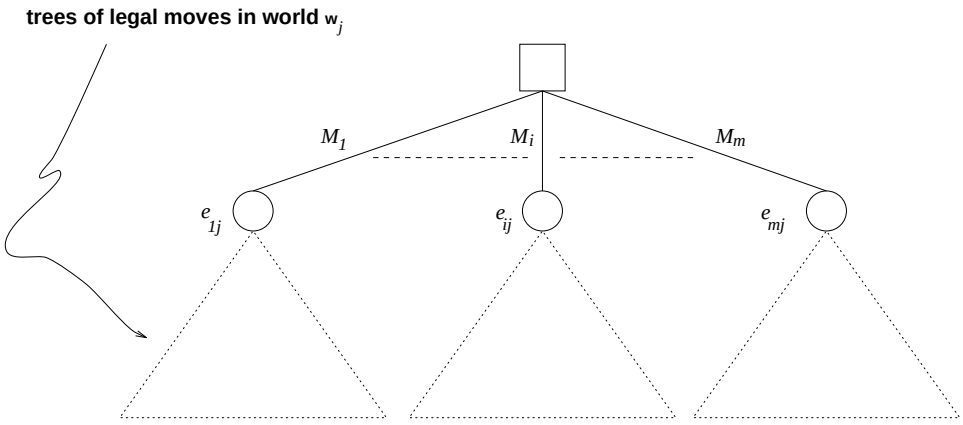


Figure 7: The minimax values, e_{ij}, of each move M_i under world w_j

Levy's proposal was that each legal move, M_i, could be given a score based on its expected payoff. In the context of Figure 7, Levy's score can be expressed as the scoring function, f:

$$f(M_i) = \sum_{j=1}^{n} e_{ij}\, prob(\mathsf{w}_j)\,.$$

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' [Ginsberg 95] suggests the following alternative:

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

where k ranges from 1 to m, and $\stackrel{?}{=}$ is an infix function which 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 which guarantee a result at least as large as some minimum value, e, say. That is:

$$f(M_i) = \sum_{j=1}^{n} \left(e_{ij} \stackrel{?}{>} e \right) e_{ij} prob(\mathsf{w}_j) \, .$$

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 Repeated Minimaxing Fails: Strategy Mixing

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 the framework we developed in §3 to show that this is not possible, *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 §2, this time labelling the nodes and adding one extra possible path, as shown by the nodes d, e, and f in Figure 8. If MAX

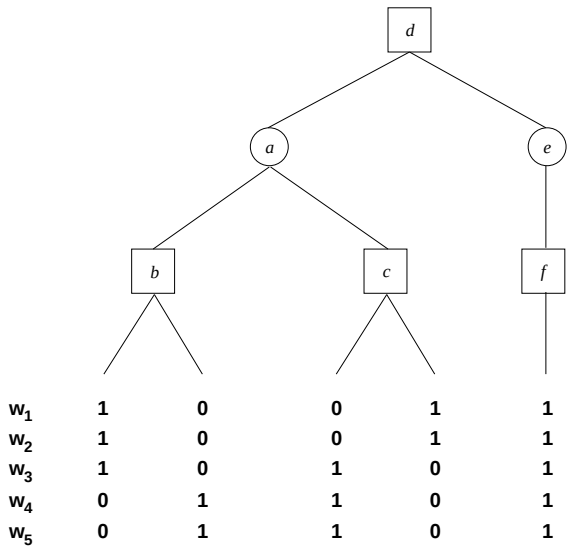


Figure 8: A tree of MIN and MAX choices which is mis-analysed by exhaustive minimaxing

picks the left-hand branch at node d, he will encounter the tree with four possible strategies that we have already examined in §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 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 Figure 9(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 §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 the others. To see that no *possible* scoring function can cope with all such situations, consider Figure 9(b). To an exhaustive minimaxing algorithm, this tree will be indistinguishable from that of Figure 9(a), as the minimax values of the MIN nodes in each are the same under every world. However, in Figure 9(a) the best move is the right-hand branch, and in Figure 9(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.

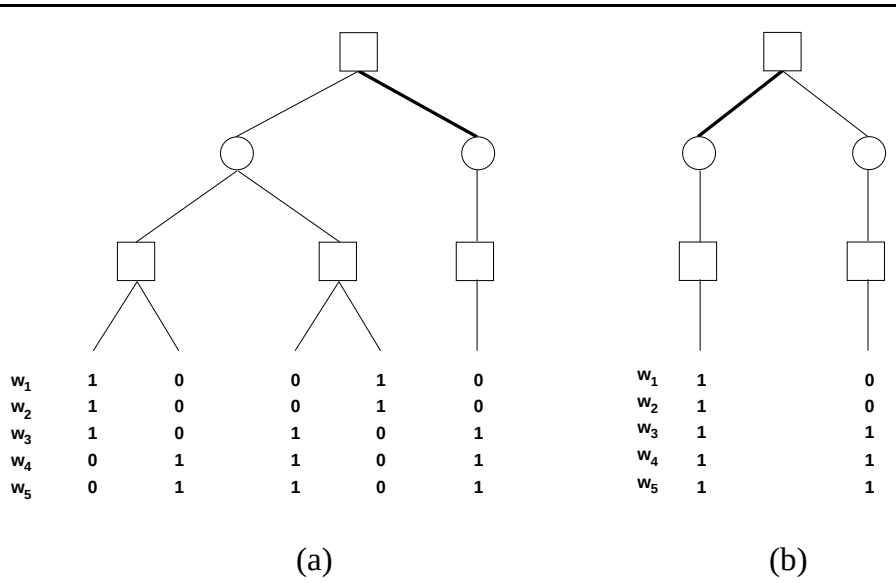


Figure 9: Two trees with the best initial move marked in bold

The source of the difficulty which repeated minimaxing experiences on these trees lies in a crucial deviation it makes from the exhaustive strategy minimisation algorithm we presented in §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 §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 mixing*. As we saw in §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 mixing 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 mixing problem is to note that the repeated minimaxing architecture actually models the task of selecting between some number of perfect information *games* each starting with the same chance move. For example, imagine that the subtrees rooted on nodes a and e in Figure 8 represent the MIN and MAX moves in a game starting with a chance move that selects one of the possible worlds $\mathsf{w}_1, \cdots, \mathsf{w}_5$. Which of these games would we rather play given that they have perfect information (for both players)? This is the question which 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 a reduced game, since it involves assumptions about MAX, as well as MIN, having perfect information.

6.1 A Bridge Example

To see an example of strategy mixing in the game of Bridge, consider the situation of Figure 10, 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.

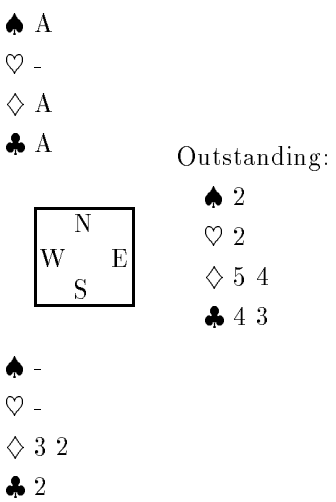


Figure 10: A Bridge situation requiring a guess over the best strategy

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 which starts a trick, it is therefore possible to win all the remaining tricks by leading the suit which 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 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 mixing can therefore be constructed by adding four cards to the above situation, as in Figure 11.

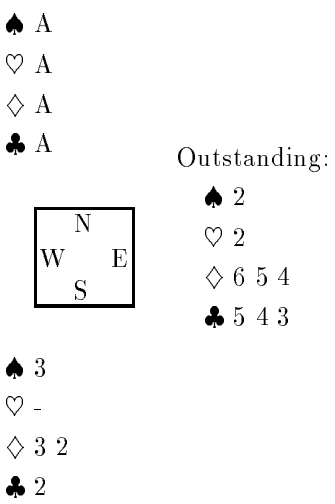


Figure 11: A Bridge example where strategy mixing obscures the best strategy

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 Some Practical Consequences

To our knowledge, the concept of strategy mixing has not been formalised before. However, some of the effects of strategy mixing have been discussed in the literature on computer Bridge, although they have typically been mis-analysed. Ginsberg, for instance, presents the card combination of Figure 12 (where we control both the North and the South hands, an 'x' represents an arbitrary low card, and we are only concerned with the cards in this suit).

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

Figure 12: An example card combination used by Ginsberg

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 KJ10x 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 Figure 13.

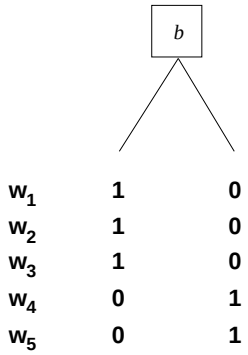


Figure 13: A simple tree with just one MAX choice

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 mixing. However, consider this new situation where 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 which win under different worlds has to be made. If such a decision point occurs within a game tree, as in Figure 8, a repeated minimaxing algorithm will deceive itself into thinking that it can always make the correct choice. If the decision point occurs at the root of the tree as in Figure 13, 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 mixing.

To see how this is relevant in Ginsberg's example, observe that in order to be able to always win the finesse against *either* East or West for the outstanding Queen 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 a 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, 10, 9, 8, x. Indeed, an exhaustive minimaxing 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 Figure 14, where the aim is to win *all* the tricks and there is no trump suit.

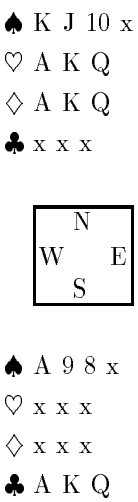


Figure 14: An example deal in Bridge. South is declarer, North is dummy and there are no trumps

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 [Frank 89, Page 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.

7 Non-locality (Repeated Minimaxing Fails Again)

The failure of minimax-like algorithms we address here is more subtle than that of strategy mixing 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 §2, which we repeat here 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 of the

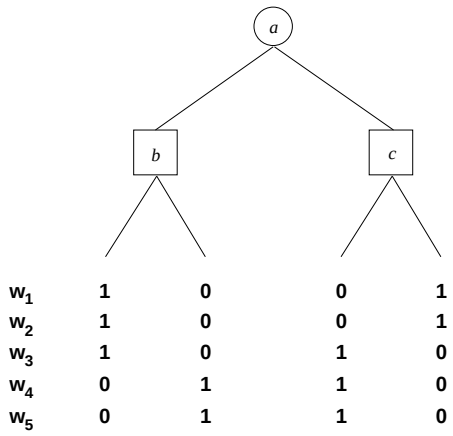


Figure 15: Simple tree of MIN and MAX choices in a domain with five possible worlds

information sets 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 §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 §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 mixing, 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 for such games. With more than one possible world, however, this assumption is no longer valid. The reason for this is that 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 §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 which leads to a payoff of 1 in most worlds.

In general, the choice of branch at a given MAX node is not simply a function of the payoffs of the paths which contain the MAX node, but of the payoffs along *any* path in the tree. If MIN can choose a play at a node's ancestor which reduces the payoff (in any world) from that which MAX would expect from examining the MAX node's direct subtree, the best choice of branch at that node may change. For example, in the case of Figure 16, 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.

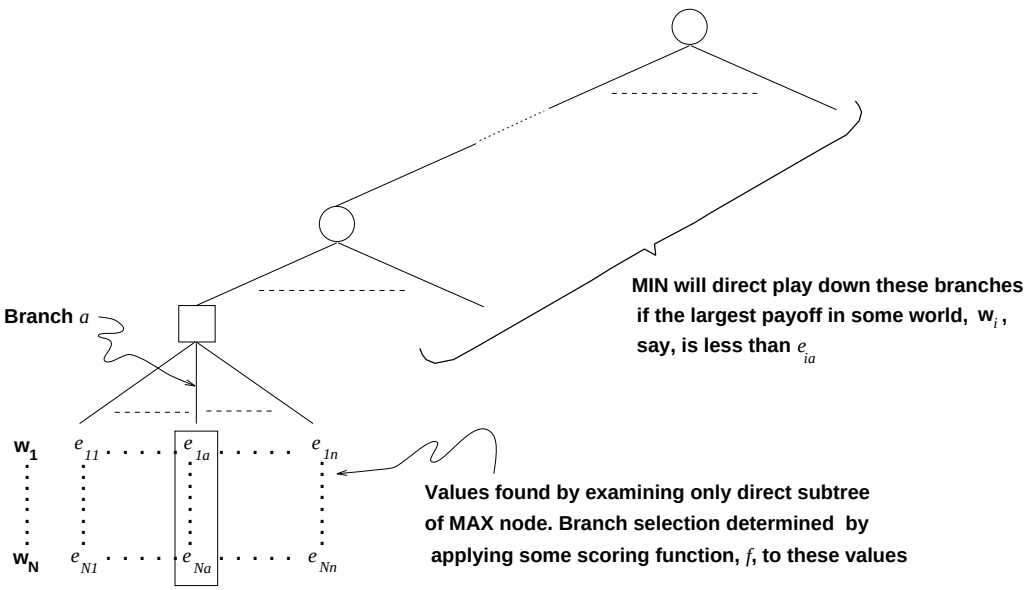


Figure 16: A search process in which the selection of branch a may be rendered incorrect by non-locality

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, consider the examples of Figure 17. The left-hand MAX node is the same in both trees, and leads to a 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.

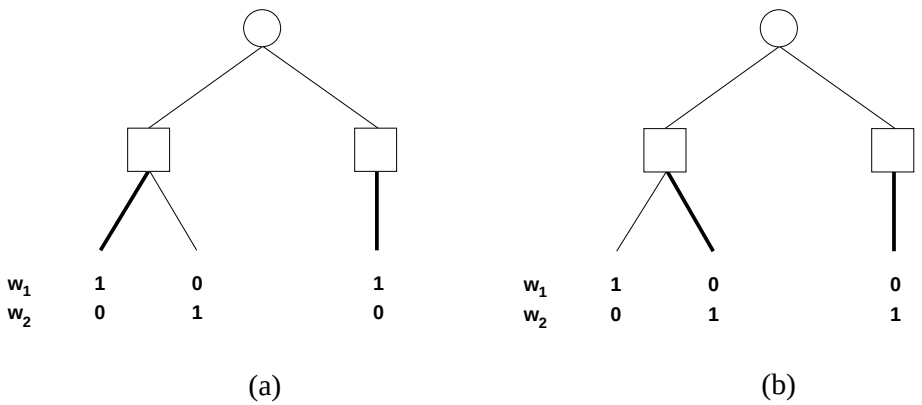


Figure 17: Two trees with the best MAX strategy marked in bold

Unlike strategy mixing, which is linked with incorrect assumptions about MAX having perfect

knowledge, it should be clear that 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 reduced form), and knowledge of the actual state of the world (which in reduced games is complete knowledge, from assumption A-I). Under these conditions, it will generally benefit MAX to select mixed strategies rather than pure ones, since this will help prevent his actual moves from being known by MIN. The main limitation of our model is therefore that it restricts consideration to pure strategies. In our case, this restriction was imposed to aid formal analysis, allowing the $esm(\Gamma)$ algorithm to be specified using a simple loop. However, as we noted in §3.2, pure strategies are also often given as solutions in standard Bridge texts, and we can further justify this restriction by pointing out that the repeated minimaxing model, in its present form, is also incapable of producing mixed strategies.

We should emphasise that since non-locality occurs as a consequence of (the interaction between the assumptions of the reduced form and) the implicit assumption that the best choice at a node can be determined by analysing just the node's subtree, the phenomenon will be independent of the accuracy of the evaluations on which the branch selection is based. For example, in Figure 16, non-locality would still be possible whether the e's were produced using repeated minimaxing (risking incorrect values caused by strategy mixing) or by using exhaustive strategy minimisation to find the results of the optimal strategy for the subtree (therefore producing exact values). Thus, the solution to improving the double-dummy architectures of Levy and Ginsberg does not lie with the simple replacement of the minimax algorithm with modified versions, such as average propagation [Nau *et al* 88] or product propagation [Pearl 81, Pearl 83]. 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 algorithm may therefore reduce the effects of strategy mixing, but non-locality will still be present. Also, the very property 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.

Currently, we are aware of no correct algorithm for identifying optimal strategies in reduced games other than the exhaustive minimaxing algorithm of §4. This algorithm overcomes both the phenomena of non-locality and strategy mixing by the simple expedient of examining the possible outcomes of each complete strategy separately.

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 Figure 18, where one trick must be lost as 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* [Frank *et al* 92, Frank 91], which are operators 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.[1]

We will look at the two tactics which succeed under most worlds. The first of these is a play which begins by leading the 7 from the North hand. If East plays the Queen straight away, declarer

[1] Note that Ginsberg suggests a similar representation when discussing the example of Figure 12. 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 mixing for one level of search, since instead of choosing between individual moves — which are really just partial strategies with only the first step determined — we are now choosing between options which are partial strategies with the first *two* steps determined. However, the strategy mixing which 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 mixing 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.

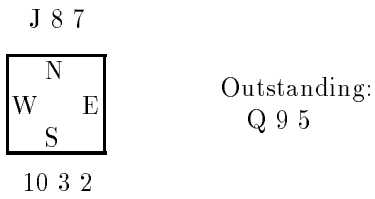


Figure 18: A Bridge card combination where one trick must be lost

plays a low card from the South hand and wins the next two tricks with the Jack and the 10. If East plays the 9, declarer covers with the 10 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 holds 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 Table 19.

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

Figure 19: Possible worlds under which each tactic will produce 2 tricks

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 Figure 18 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 Figure 20.

If MAX 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 suit. Similarly, if East starts with Kx or Qx, the best option will again be to play low. Thus, if the situation of Figure 18 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.

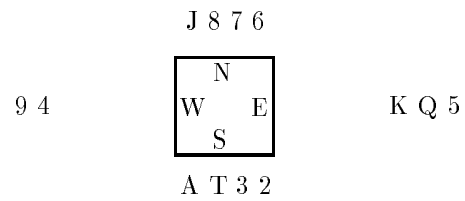


Figure 20: A Bridge example giving rise to non-locality

This extra information would have a crucial effect on the situation in Figure 18. 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 Figure 19 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.

8 Conclusions

We have looked at the problem of selecting a strategy in zero-sum two-player games with incomplete information. We formalised the notion of a reduced form of such games in which equilibrium point strategies could be more easily identified, and presented exhaustive strategy minimisation as an algorithm capable of finding such strategies.

Although the problems we address seem central to the design of programs for playing games in incomplete information domains, it appears that they have not been adequately studied or addressed in the literature. We were therefore able to newly formalise two general problems which can result from the incorrect handling of the technical notion of a strategy in such domains. The first of these, strategy mixing, results from combining different MAX strategies in different possible worlds. The second, non-locality, results from examining only partial strategies at internal nodes of a game tree.

We demonstrated that both strategy mixing and non-locality are present when the standard minimax algorithm is applied to games with incomplete information, as in the repeated minimaxing architecture. In the domain of Bridge, where we showed that the reduced form of the game corresponds to the typical model analysed in expert texts, our results must therefore call into question some of the claims about the prospects of computer players based on fast double-dummy programs. For instance, it seems at best unproven that such programs will inevitably mean that, as humans, 'our time as world Bridge champions is limited' [Ginsberg 95].

We pointed out that the main limitation of our model is the restriction to pure, rather than the general case of mixed, strategies. Also, our solution of exhaustive strategy minimisation, although capable of producing optimal strategies, is highly intractable. This suggests that there is a real cost involved in overcoming the effects of incomplete information. However, we have not yet established a lower bound for the complexity of this problem, and this remains an open — and apparently important — problem for the design of programs which play games like Bridge.

References

[ACBL 94] ACBL. *The Official Encyclopedia of Bridge*. American Contract Bridge League, Inc., 2990 Airways Boulevard, Memphis, Tennessee 38116-3875, fifth edition, 1994.

[Berlekamp 63] E.R. Berlekamp. Program for double-dummy bridge problems — a new strategy for mechanical game playing. *Journal of the ACM*, 10(4):357–364, 1963. Also appears in D. Levy, editor, *Computer Games II*. Springer Verlag, New York, 1988.

[Berlin 85] Daniel L.S. Berlin. Span: Integrating problem solving tactics. In *Proceedings of IJCAI-85*, pages 1047–1051, Los Angeles, California, 1985.

[Carley 62] Gay Carley. A program to play contract bridge. Unpublished M.Sc. thesis, Dept. of Electrical Engineering, Massachussets Institute of Technology, Cambridge, Massachussets, June 1962.

[Frank & Basin 95] I. Frank and D. Basin. An analysis of the bridge search space size. Forthcoming, 1995.

[Frank 89] A. Frank. Brute force search in games of imperfect information. In D.N.L. Levy and D.F. Beal, editors, *Heuristic Programming in Artificial Intelligence 2 – The Second Computer Olympiad*, pages 204–209. Ellis Horwood, 1989.

[Frank 91] I. Frank. Finesse: An adaption of proof-planning techniques to declarer play in the game of Bridge. Unpublished M.Sc. thesis, DAI, 1991.

[Frank *et al* 92] I. Frank, D. Basin, and A. Bundy. An adaptation of proof-planning to declarer play in bridge. In *Proceedings of ECAI-92*, pages 72–76, Vienna, Austria, 1992. Longer Version available from Edinburgh as DAI Research Paper No. 575.

[Gamback *et al* 91] Bjorn Gamback, Manny Rayner, and Barney Pell. An architecture for a sophisticated mechanical bridge player. In D.N.L. Levy and D.F. Beal, editors, *Heuristic Programming in Artificial Intelligence 2 – The Second Computer Olympiad*, pages 87–107. Ellis Horwood, 1991. Later version also available as Cambridge University Computer Laboratory Tech Report 299, May 1993.

[Geake 92] Elizabeth Geake. Playing to win. *New Scientist*, pages 24–25, 19 September 1992.

[Ginsberg 95] M. Ginsberg. How computers will play bridge. *The Bridge World*, 1995. To appear. Also available for anonymous ftp from t.uoregon.edu as the file /papers/bridge.ps.

[Kibler & Schwamb 92] D. Kibler and K. Schwamb. Complete contingency planners. Technical Report UCI TR 92–94, University of California, Irvine, 1992.

[Korf 89] Richard E. Korf. Generalized game trees. In *Proceedings of the 11th IJCAI*, volume 1, pages 328–333, Detroit, Michigan, USA, 1989.

[Lee 94] Ray Lee, editor. *Reviews from Canadian Master Point.* 1992–1994. Available by anonymous ftp as the file /pub/Bridge/FAQ.ReviewsFromCanadianMasterPoint from the site arp.anu.edu.au.

[Levy 89] D.N.L. Levy. The million pound bridge program. In D.N.L. Levy and D.F. Beal, editors, *Heuristic Programming in Artificial Intelligence – The First Computer Olympiad*, pages 95–103. Ellis Horwood, 1989.

[Luce & Raiffa 57] R. Duncan Luce and Howard Raiffa. *Games and Decisions— Introduction and Critical Survey.* Wiley, New York, 1957.

[Lustig 85] R. Lustig. Hibrid: A hierarchically designed bridge playing program. Unpublished M.Sc. thesis, Weizmann Inst. of Science, Rehovot, Israel, 1985.

[Manley 94] B. Manley. Bridge software: Moving forwards or backwards? *The Bulletin, American Contract Bridge League*, October 1994.

[Napjus 69] C.N. Napjus. *Declarer: A Learning Bridge-Playing Program which Generalizes and Infers.* Unpublished PhD thesis, University of Washington, 1969.

[Nash 51] J. F. Nash. Non-cooperative games. *Annals of Mathematics*, 2(54):286–295, 1951.

[Nau *et al* 88] Dana Nau, Paul Purdom, and Chun-Hung Tzeng. An evaluation of two alternatives to minimax. In L. N. Kanal and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pages 505–509. Elsevier Science Publishers B.V., The Netherlands, 1988.

[Nygate & Sterling 93] Y. Nygate and L. Sterling. Aspen — designing complex knowledge based systems. In *10th Israeli Symposium on Artificial Intelligence*, pages 51–60, 1993.

[Pearl 81] Judea Pearl. Heuristic search theory: Survey of recent results. In *Proceedings of IJCAI-81*, volume 1, pages 554–562, Vancouver, Canada, 1981.

[Pearl 83] Judea Pearl. On the nature of pathology in game searching. *Artificial Intelligence*, 20:427–453, 1983.

[Shannon 50] C. E. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41:256–275, 1950.

[Smith & Nau 93] J.J. Smith and D.S. Nau. Strategic planning for imperfect information games. In *Games: Planning and Learning, Papers from the 1993 Fall Symposium*, pages 84–91, AAAI Press, 1993.

[Stanier 76] A. Stanier. Planning to make tricks at bridge. In *Proceedings of the AISB Summer Conference*, pages 256–265, 1976. Also appears in D. Levy, editor, *Computer Games II.* Springer Verlag, New York, 1988.

[Throop 83] T. Throop. *Computer Bridge.* Hayden, 1983.

[von Neumann & Morgenstern 44] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behaviour.* Princeton University Press, 1944.

[Wheen 89] R. Wheen. Solving double dummy bridge problems by exhaustive search. In D.N.L. Levy and D.F. Beal, editors, *Heuristic Programming in Artificial Intelligence – The First Computer Olympiad*, pages 89–94. Ellis Horwood, 1989.