IS MINIMAX REALLY AN OPTIMAL STRATEGY IN GAMES?

Katarzyna Kościuk[1]

[1]Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: In theory, the optimal strategy for all kinds of games against an intelligent opponent is the Minimax strategy. Minimax assumes a perfectly rational opponent, who also takes optimal actions. However, in practice, most human opponents depart from rationality. In this case, the best move at any given step may not be one that is indicated by Minimax and an algorithm that takes into consideration human imperfections will perform better.
In this paper, we show how modeling an opponent and subsequent modification of the Minimax strategy that takes into account that the opponent is not perfect, can improve a variant of the Tic-Tac-Toe game and and the game of Bridge. In Bridge we propose a simple model, in which we divide players into two classes: conservative and risk-seeking. We show that knowing which class the opponent belongs to improves the performance of the algorithm.

Keywords: Minimax, optimality, player modeling, bridge

1. Introduction

Typically, programs for game playing use the Minimax strategy [5], which assumes that the opponent is a perfectly rational agent, who always performs optimal actions. However, most humans depart from rationality [7]. In this case, at any given step, a move that is practically the best may not be one indicated by Minimax. If we know that the opponent plays defensively, for example, we can count on her not noticing or not performing moves that seem to be too daring. In order to consider a player's departures from rationality, we need to create her model and learn over time her strategies.

In this paper, we describe our attempt to model players in a variant of the Tic-Tac-Toe game and in the card game Bridge. The underlying assumption, supported by both anecdotal and empirical evidence, is that games can benefit from adapting to a particular opponent, rather than using the same general strategy against all players. We implement the programs with conventional Minimax strategy and the Minimax with Alpha-beta pruning. These algorithms search the game tree in order to choose the best move for the computer. We add an algorithms that makes decisions based on a model of opponent's weaknesses. The algorithm includes a learning module that observes the opponent carefully and learns his/her strategies. We test it against the Minimax strategy against human players.

This paper is organized as follows. Section 2. presents the Minimax algorithm. Section 3. describes our improvements in a variant of the Tic-Tac-Toe game. Section 4. describes the card game Bridge. Section 4.2 describes the classes of Bridge players and our experiments. Finally, Section 5. proposes some ideas of future work.

2. The Minimax Algorithm

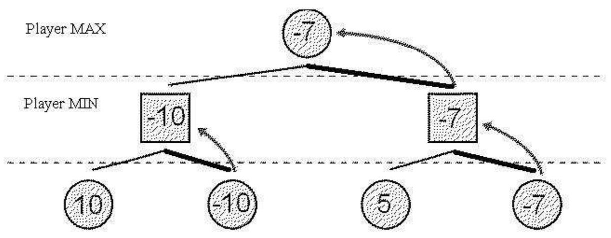


Fig. 1. Minimax algorithm

One way to pick the best move in a game is to search the game tree using the Minimax algorithm or its variants (Fig. 1). The game tree is a the directed graph in which nodes are positions in a game and edges are the moves. Minimax is used in zero-sum games (when we have two players A and B, zero-sum means that in any outcome of the game, player A's gains equal player B's losses). Each position or state in the game is evaluated using an evaluation function, that defines how good it would be for a player to achieve this position.

Minimax is the method that minimizes the maximum possible loss. At each step of the game the assumption is made that player A is trying to maximize the chances of A's winning, while player B is trying to minimize the chances of A's winning. We call the player A — MAX, the player B -– MIN, and we assume that MAX starts the game. The player MAX makes the move that maximizes the minimum value of the position resulting from the possible next moves of the opponent and assigns a value to each of his or her legal moves. Minimax assumes that opponent always chooses the best move, but opponents are human and may depart from rationality – they can choose an inferior move (e.g., not know of or not notice a better move).

2.1 Bayesian Network

Because we use Bayesian networks for modeling the opponent in games, this section introduces them briefly.

A Bayesian network [6] is an acyclic directed graph, whose nodes represent random variables, such as observable quantities, latent variables, unknown parameters or hypotheses. The edges represent direct influences. Nodes that are not connected represent variables that are conditionally independent of each other. Each node is associated with a probability function that takes as input a particular set of values for the node's parent variables and gives the probability distribution over the variable represented by the node. An example of simple Bayesian network is shown in Fig. 2

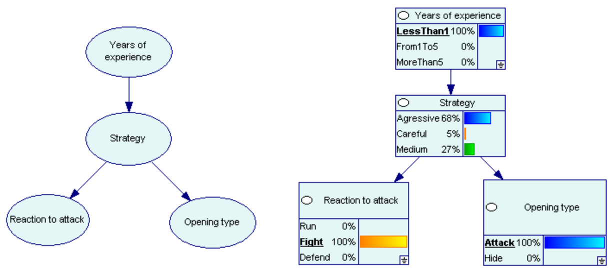


Fig. 2. Example Bayesian Network

The network has four nodes:

- The node Years_of_ experience represents the player's experience, it has three states: More_than_5, 1_5, Less_than_1.
- The node Strategy corresponds with the player's strategy, it has three states: agressive, careful, medium.
- The node Opening_type has two states: attack, hide.
- The node Reaction_to_attack has three states: run, fight, defend.

When the network is learned (for example, from the data collected during the games) we can set the evidence and check the player's strategy. For example, when the player opens the game attacking, his reaction to attack is fight and he has less than 1 year experience, the probability that he is aggressive is 68%.

3. A simple game: Tic-Tac-Toe

3.1 Rules and Algorithm

We implement and compare several algorithms, such as Minimax with and without Alpha-beta pruning, in a variant of the Tic-Tac-Toe game [3].

We choose the board dimension 4x4. The winning position are three ‘X’ or three ‘O’ in the same line. For simplification we assume that computer plays ‘O’ and human player plays ‘X’. We establish the following evaluation function F:

- F=0 for a draw,
- F=1 when computer wins,
- F=-1 when computer looses.

In such variant of the Tic-Tac-Toe the player who starts the game always wins, at least when she/he chooses optimal moves. We notice, as we assumed before, that people sometimes loose because they do not choose the best move.

Conventional Minimax evaluates every possible move, and when it calculates that ‘O’ looses (always when ‘X’ starts the game) it returns no move — surrenders, or it returns the first empty place on the board (depending on the implementation). It does not take into account that people often do not see the winning move.

We modify the algorithm. We count on opponent’s weaknesses and continue the game even if it looks lost (evaluation function returns -1 for every possible move). For every possible move we calculate how many moves remains to the end of the game – end-depth value. Our Minimax choose the move that lets the computer play longer in order to wait for the opponent’s mistake. In that case a lost game often turn out to be a winning game. When there are several moves with the evaluation function equal 1, our Minimax choose a move that lets the computer win fastest. When there are more than one move with the same F value and the same end-depth value, our algorithm returns random move. Conventional Minimax, depending on the implementation, always returns the first or the last move in such situations.

Experiments were carried out with two persons. Human player ‘X’ always started the game, because there were no chance of winning in another situations. Here are the average results:

- After 20 games our Minimax won in the 13 cases in 20 games, conventional Minimax won only in 1 case.

- After 60 games our Minimax won in the 27 games, conventional Minimax won in 2 games.

Experiments show that waiting for the opponent's mistake turns out to be a better strategy than surrendering. After about 30 games human player learnt where she/he should put the first move to win the game (for example in the middle of the board).

3.2 Modeling

In order to consider a player's weaknesses, it is necessary to model the player – learn and know his/her strategies. Our next step was adding an algorithm that makes decisions based on a model of opponent's weaknesses [4]. The algorithm includes a learning module that observes the opponent. We study the improvements of the algorithm over time and test it against the Minimax strategy. We build a Bayesian network to model the player. We learn the conditional probability tables in the network from data collected in the course of the game.

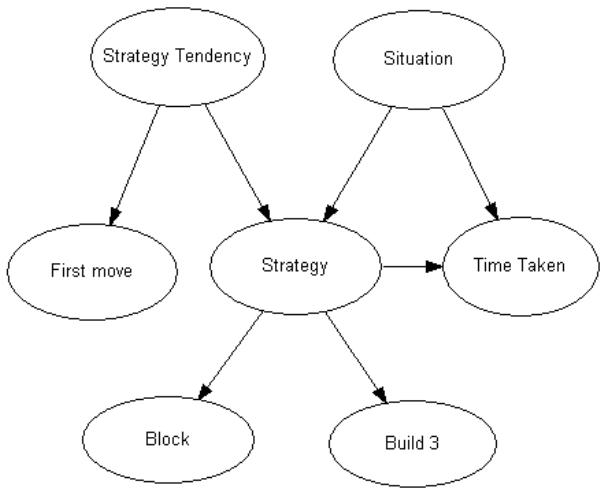


Fig. 3. Bayesian Network

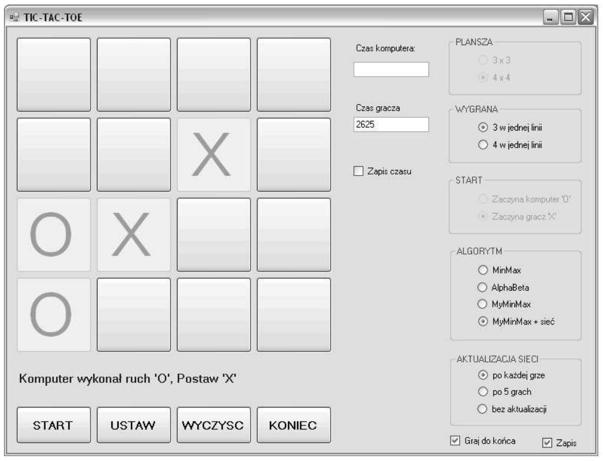
Our program stores the following information about the opponent:

- the place where the opponent puts the first move,
- the opponent's long-term strategy,
- types of moves,

- situation on the board,
- how long the player thinks before a move.

We capture this information in the Bayesian network shown in Fig. 3.

- The node First_move represents the place where the player puts the first mark. It has three possible states: corner, side, and center. It might seem that there are 16 possible positions (corresponding to the 16 free places on the board). However, because of the symetry of the board, many of them are equivalent (e.g. placing the mark in any corner).
- The node Block has two states: yes and no. It assumes the state yes when the player puts a mark 'X' in order to block two 'O's in row, otherwise it assumes the state no.
- The node Build_3 has two states: yes where the player puts two 'X' in row, and has a opportunity to put the third 'X' in row, otherwise the state is no.
- The node Time_taken represents how long the player thinks before his/her move.
- The node Strategy corresponds with the player's strategy in the particular move:
  - Attack — the player attacks and sometimes does not see the necessity of defense.
  - Defense — the player blocks 'O's (even unnecessary, when he/she has a opportunity to win).
  - Attack_Defense — the player chooses the move that block 'O's and build three 'X' in row, or the move that does not change the situation on the board.
- The node Strategy_Tendency represents the player's general strategy:
  - Attack — the player has a tendency to attack more often than to defend.
  - Defense — the player has a tendency to block more often than to attack.
  - Attack_Defense — the player sometimes blocks, sometimes attacks
- The node Situation corresponds with the situation on the board. We distinguish seven situations:
  - UnderAttack -— two unblocked 'O's in row,
  - BeforeWin -— two unblocked 'X's in row,
  - Trap (fork) – an opportunity where 'O' can win in two ways,
  - Initial — before the first move,
  - Scarce — a lot of free places n the board,
  - Dense — a few free places on the board,
  - AttackedBeforeWin — two unblocked 'O's in row and two unblocked 'X's in row (player has an opportunity to win, but sometimes does not see it and blocks 'O's instead).



Czas komputera:

Czas gracza: 2625

☐ Zapis czasu

PLANSZA: 3 x 3 [✓ 4 x 4]

WYGRANA: [✓ 3 w jednej linii] 4 w jednej linii

START: Zaczyna komputer 'O' [✓ Zaczyna gracz 'X']

ALGORYTM: MinMax AlphaBeta MyMinMax [✓ MyMinMax + sieć]

AKTUALIZACJA SIECI: [✓ po każdej grze] po 5 grach bez aktualizacji

☑ Graj do końca ☑ Zapis

Komputer wykonał ruch 'O', Postaw 'X'

Fig. 4. A screen shot of the Tic-Tac-Toe program

When there are several moves with the same F value and the same end-depth value, our program (Fig. 4) consults the Bayesian network about the opponent's strategy.

The experiments were carried out with the 7-old boy. The Human player 'X' always started the game. We did not know before which strategy player prefers - offensive or defensive. The network was learnt during the games. Here are the average results:

- After 20 games Minimax + Bayesian Network won in the 14 cases, conventional Minimax won in 1 case.
- After 60 games Minimax + Bayesian Network won in the 25 games, conventional Minimax won in 1 game.

Experiments show that the gain from modeling the opponent in Tic-Tac-Toe is not large – our previous program, that waits for opponent's mistake, performs just as good. One of the reasons for this modest gain may be simplicity of the game therefore we start the experiments with more complex games, such as Bridge.

4. A Complex Game: Bridge

4.1 An Introduction to Bridge

Bridge is a card game played by four players with a standard deck of 52 cards. The players form two teams: North (N) — South (S) and East (E) — West (W).

The partners sit opposite each other. Bridge consists of two stages, (1) the auction (bidding), and (2) the play. During the auction, each player declares how many tricks she can probably take and what her best suit(s) is(are). The bidding ends with a contract — a declaration by one team that they take at least a stated number of tricks. They could agree on a specified suits as trump (clubs, diamonds, hearts, spades) or the game without trumps (no trump – NT). The partners that get the contract are called Declarer and Dummy, the other players are called the defenders. The first lead is made by the defender seated to the left of the Declarer. After the opening lead is played, the Dummy lays her hand face up on the table, i.e., visible to every player, including the opponents, and she does not take part in the game. Her cards are played by the Declarer. The play proceeds clockwise around the table.

If the winning team takes at least as many tricks as they have declared, they get the score [9]. Otherwise, the defenders get points for each trick that the Declarer lacks for making the contract. The score depends on:

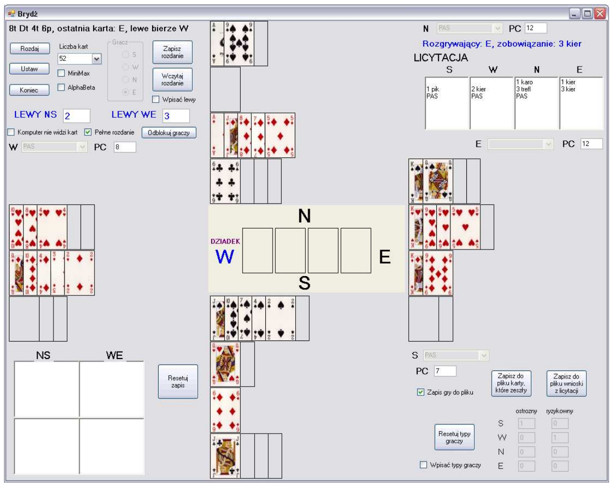
- the trump ranked from the lowest to highest: clubs, diamonds, hearts, spades, no trumps;
- the number of tricks taken.

Bridge is an imperfect information game, as the players do not see each other hands, with the exception of the cards of the Dummy, visible by everybody. It is possible to predict the probability distribution of the unseen cards from the information gathered during the bidding.

The auction is an opportunity for all players to gather information about unseen cards. Very often, auction is based on the so called *basic natural system*, based on the High Card Points (this system is also called the Milton Work Point Count: Ace has 4 HCP, King 3 HCP, Queen 2 HCP and Jack 1 HCP). A hand that holds 12 HCP is considered sufficient to open. Opening, for example hearts, usually promises at least 4 or 5 cards in that suit. When the partners agree on hearts, we can assume that they have at least 8 cards in that suit. When a player passed, we can assume that she does not have more that 6 points (if her partner opened before) or 11 points (if she starts the bidding).

4.2 The SBA Program

The SBA (Simple Bridge Assistant) program (Fig. 5) uses the Monte Carlo technique, which is used by Ginsberg's [1] computer Bridge program (GIB). According to Ginsberg himself, GIB is currently the strongest computer Bridge program in the world. The Monte Carlo technique was originally proposed for card play by Levy



8t Dt 4t 6p, ostatnia karta: E, lewe bierze W

Liczba kart: 52

[ ] MiniMax [ ] AlphaBeta

Gracz: S W N [✓ E]

[ ] Wpisać lewy

LEWY NS 2 LEWY WE 3

[ ] Komputer nie widzi kart [✓ Pełne rozdanie]

W PAS PC 8

N PAS PC 12

Rozgrywający: E, zobowiązanie: 3 kier

LICYTACJA

S	W	N	E
1 pik PAS	2 kier PAS	1 karo 3 trefl PAS	1 kier 3 kier

E PC 12

S PAS PC 7

[✓ Zapis gry do pliku]

[ ] Wpisać typy graczy

	ostrozny	ryzykowny
S	1	0
W	0	1
N	0	0
E	0	0

Fig. 5. A screen shot of the SBA

[2]. The unseen cards are dealt at random, using the information gathered during the auction and the cards played thus far. This perfect information variant of the game is called double dummy Bridge. All possible moves are evaluated in all generated deals. Algorithm adds the scores obtained by making every move and returns the move with the maximal sum. GIB uses brute force techniques, called partition search, to solve the double dummy game — we implemented the Minimax algorithm with Alpha-beta pruning. The game tree is very large because the average number of legal moves is four in any position. To deal with the computational complexity of computing all possible moves, we focused in our experiments on the end games consisting of the last seven tricks.

We also implement the player modeling, that recognizes the player's strategy. We divided the players into two classes — conservative and risk-taking. Our program recognizes and counts all risky and cautious steps for each player.

4.3 Classes of Bridge Players

Below we present examples taken from three real games.

The first example (Fig. 6) shows a conservative play. North is the Declarer, South is the Dummy, and the contract is for 4 Clubs. Let West play the Nine of Diamonds. If North is a careful player, she will play the Ace of Clubs (instead of Nine of Clubs, for example) being afraid that East has no diamonds and will beat a smaller trump.

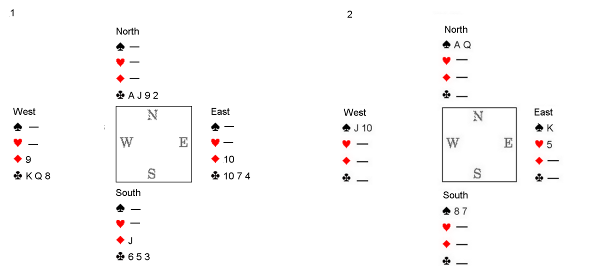


Fig. 6. Game examples: 1 – conservative player, 2 – finesse

A risky player would put a smaller trump, counting on that East has at least one Diamond.

The second example is a finesse (Fig. 6) in which a player can take advantage of the position of the particular cards. Assume that South is the Declarer, North is the Dummy, and the contract is 2 NT. A finesse is a technique that allows a player to promote tricks based on a favorable position of one or more cards in the hands of the opponents. A direct finesse is a finesse that gains a trick without losing one. Let South stars with a Seven of Spade and the Queen of Spade from Dummy's hand. If West had the King of Spade, North-South will win two tricks — one with the queen, second with the ace. The problem is that South does not know who, West or East, has the King. If he puts the Queen of Spades from Dummy, East will take two trumps.

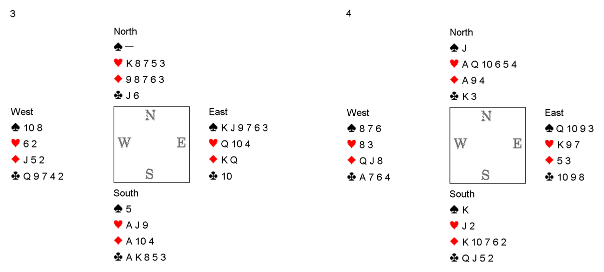


Fig. 7. Game examples: 3 – risky step, 4 – how we can use the knowledge about our opponent

Fig. 7 shows another example of a risky step. South is the Declarer, North is the Dummy, the contract is for 3 Diamonds. The first trick was taken by South. He is a risky player and now he plays the Jack of Hearts. If the Queen of Hearts were in West's hand, South would take the trick (if South makes that assumption, he will play the Five of Hearts from Dummy's hand). West would not put the Queen of Hearts therefore he knows that the Dummy has the King of Hearts. South and North would take two tricks instead of one trick — first with Jack of Hearts, second with the King of Hearts. Unfortunately for South, West does not have the Queen of Hearts, and East will take the trick. Not to put the King of Heart from North's hand it is a risky step.

Fig. 7 also demonstrates how we can use the knowledge of our opponent's inclination to strategy. South is the Declarer, North is the Dummy, the contract is 2 NT. South took the first trick. Assume that he knows from the previous games that West is a very cautious player. South plays the Jack of Hearts. If West puts the Eight of Hearts, North will have to put the Ace of Hearts. If West had the King of Hearts, he would have put it instead. Therefore, with high probability, the Declarer knows that the King of Hearts is in East's hand.

The cautious players often puts the high cards, even it is not needed, because they afraid that someone else could take the trick.

4.4 Experiments

In our experiments we used the examples from the real game [8]. We conducted the experiments with 30 games. We compared:

– player modeling and Minimax with Monte Carlo sampling;
– the real game.

We recognized the player's category (4.3) by counting the risky and the cautious steps in 100 his last games. We chose two players that played about thousand games and strictly belongs to aforementioned categories (if the number of risky steps was greater more than 10 percent than the number of cautious steps we assumed that player is risky, accordingly we did for conservative player).

– the risky player: 123seksity;
– the conservative player: tutoo.

We chose 15 games for each chosen player and our system played the game instead of his opponents. The original moves was changed by the program only if the player's strategy could be used against him and then the rest of the game was playing by the Minimax for every player. The player modeling improved 6 scores: 4 in games with risky player, 2 in games with conservative player.

5. Discussion

Experiments show that games can benefit from adapting to a particular opponent, rather than using the same general strategy against all players. The gain from modeling the opponent in Bridge is more significant than in Tic-Tac-Toe. We plan to make more experiments in Bridge, inter alia add algorithms for other bidding systems.

References

[1] Ginsberg, M., L.: GIB: Imperfect Information in a Computationally Challenging Game. Journal of Artificial Intelligence Research 14, 2001, pp. 303–38.

[2] Levy, D.: The million pound bridge program, Heuristic Programming in Artificial Intelligence: The First Computer Olympiad. Chichester, UK: Ellis Horwood, 1989.

[3] Kościuk, K., Drużdżel, M.: Exploring opponent's weaknesses as an alternative to the Minimax strategy. XV Warsztaty Naukowe PTSK : Symulacja w badaniach i rozwoju, 2008.

[4] Kościuk, K., Drużdżel, M.: Player Modeling Using Bayesian Network. XVI Warsztaty Naukowe PTSK : Symulacja w badaniach i rozwoju, 2009.

[5] von Neumann, J., Morgenstern, O.: Theory of Games and Economic Behavior. Princeton University Press, 1944.

[6] Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. San Mateo, CA: Morgan Kaufmann Publishers, 1988.

[7] Simon, H., A.: A Behavioral Model of Rational Choice. The Quarterly Journal of Economics 69, No. 1, 1955, pp. 99–118.

[8] `[http://www.kurnik.pl]`

[9] `[http://web2.acbl.org/laws/rlaws/lawofcontractbridgecombined_2004.pdf]`

CZY MINIMAX JEST RZECZYWŚCIE OPTYMALNĄ STRATEGIĄ W GRACH?

Streszczenie Algorytmy grające w gry często używają strategii Minimax. Algorytm Minimax zakłada perfekcyjność przeciwnika, który wybiera zawsze najlepsze ruchy. Gracze jednakże mogą nie działać całkiem racjonalnie. Algorytm, który weźmie to pod uwagę może dawać lepsze wyniki niż Minimax. W pracy przedstawiono jak modelowanie gracza i modyfikacje algorytmu Minimax mogą poprawić wyniki w grze kółko-krzyżyk i w brydżu. W brydżu zaproponowany został prosty model, dzielący graczy na dwie kategorie - konserwatywny i ryzykowny. Eksperymenty pokazały, że wiedza, do której klasy graczy należy przeciwnik, poprawia działanie algorytmu.

Słowa kluczowe: Minimax, optymalność, modelowanie gracza, brydż

Artykuł zrealizowano w ramach pracy badawczej W/WI/1/2009.