Opponent Modeling in Stratego

Jan A. Stankiewicz Maarten P.D. Schadd

Departement of Knowledge Engineering, Maastricht University, The Netherlands

Abstract

Stratego[1] is a game of imperfect information, where observations of the opponent's behaviour are crucial for determining the best move. This article describes how one can model the opponent in the game of Stratego, using a Bayesian approach. By observing the moves of the opponent, a probability distribution can be derived to help determine the identity of unknown pieces of the opponent. Experiments show that there is a significant increase in the percentage of correctly guessed unknown pieces. Moreover, the average probability assigned to the real identity of an unknown piece, shows an increase as well. These results eventually translate into an improved win rate.

1 Introduction

Stratego is a turn-based two-player game of imperfect information, played on a 10×10 grid. Each player starts with 40 pieces, each piece having a certain rank. The goal of the game is to either capture the opponent's flag or to capture enough pieces such that the opponent cannot make any moves.

The player has to deal with pieces of the opponent, some of which the ranks are unknown. In fact, at the start of the game, none of the opponent's pieces are known. In this case, one can use heuristics to determine an initial probability distribution for every piece, based on setup statistics from a database of games [8]. Predicting the probability on each rank for every unknown piece can greatly help in determining the best move to make. In order to make an accurate prediction, it is necessary to model the opponent's playing style, based on the game's history. The process of creating such a model is called opponent modeling.

Each time the opponent makes a move, it may provide new information on the rank of an unknown piece. For instance, if the opponent moves an unknown piece towards a player's piece of which he knows the rank, then the player might become suspicious that the opponent's piece is stronger than his own.

However, it might be the case that the opponent is bluffing, pretending the piece is stronger or weaker than it really is. Keeping track of the opponent's bluffing behaviour in past situations gives valuable information for the future. If an opponent is known to bluff in a certain situation, one can take advantage of this by making a different move than one normally would.

The focus of this article is on how to model the opponent in Stratego. Certain techniques are translated from or inspired by Poker, a game in which some research regarding opponent modeling has been done [4, 7, 15]. The goal is to create an opponent model which improves the performance of a Stratego player.

The structure of this article is the following. Section 2 gives some background on the game Stratego and its basics. Section 3 then gives a short summary of previous research done on opponent modeling. Section 4 explains the methods used for creating a model of the opponent in Stratego. The experiments and results using these methods are discussed in Section 5. Finally Section 6 draws the conclusions and suggests areas for future research to improve the opponent model.

2 Stratego

Stratego is an imperfect-information game. It was developed at least as early as 1942 by Mogendorff. The game was sold by the Dutch publisher *Smeets and Schippers* between 1946 and 1951 [6]. The following rules are an edited version of the Stratego rules published by the Milton Bradley Company in 1986 [1].

[1]Stratego is a registered trademark of Hauseman & Hotte N.V., Amsterdam, Netherlands. All Rights Reserved. ©2002 Hasbro, Pawtucket, RI 02862.

Stratego is played on a 10×10 board. The players, Red and Blue, place each of their 40 pieces in such a way that the back of the piece faces the opponent in a 4×10 area. The movable pieces are divided into ranks (from the lowest to the highest): Spy, Scout, Miner, Sergeant, Lieutenant, Captain, Colonel, Major, General, Marshal. Each player has also two types of unmovable pieces, the Flag and the Bomb.

Players move alternately, starting with Red. Passing is not allowed. Pieces are moved to orthogonally-adjacent vacant squares. The Scout is an exception to this rule, and may be moved like a rook in chess. The *Two-Squares Rule* and the *More-Squares Rule* prohibit moves which result in repetition.[2] The lakes in the center of the board contain no squares; therefore a piece can neither move into nor cross the lakes.

A piece, other than a Bomb or a Flag, may attempt to capture an orthogonally adjacent opponent's piece; a Scout may attempt to capture from any distance. When attempting a capture, the ranks are revealed and the weaker piece is removed from the board. The stronger piece will be positioned on the square of the defending piece. If both pieces are of equal rank, both are removed. The following special rules apply to capturing. The Spy defeats the Marshal if it attacks the Marshal. Each piece, except the Miner, will be captured when attempting to capture the bomb.

The player whose Flag is captured loses the game. A player also loses the game if there is no possibility to move. The game is drawn if both players cannot move.

3 Related Research

Opponent modeling in Stratego is a fairly new topic. Most research regarding opponent modeling has been focused on Poker, which is a non-deterministic game of imperfect information. Different approaches have been used to model the opponent. In 1998, Billings *et al.* [4, 5] introduced a poker playing agent, which used weights to determine the likeliness of its opponents holding a certain pair of cards. These weights were modified according to the observations of the community cards and the opponent's actions. In 2000, Davidson *et al.* [7] proposed an opponent modeling method using artificial neural networks. The network used a set of inputs to determine the opponent's next action given those inputs [7]. Korb *et al.*[9] proposed a poker playing agent which uses a Bayesian network to model its opponent. Other Bayesian opponent modeling methods were proposed by Southey *et al.* [15] in 2005 and Ponsen *et al.* [11] in 2008.

Stratego has not received much scientific attention in the past. De Boer [8] describes the development of an evaluation function using an extensive amount of domain knowledge in a 1-ply search. Treijtel [17] created a player based on multi-agent negotiations. Stengård [16] investigates different search techniques for this game. Schadd *et al.* developed a forward pruning technique for chance nodes and tested it in Stratego [13]. At this moment, computers play Stratego at an amateur level [12]. An annual Stratego Computer Tournament[3] is held on Metaforge.[4] Finally, we remark that some research has been done in Siguo, a four-player variant of Stratego [10, 18].

4 Modeling Opponents in Stratego

This section discusses the methods used to model the opponent in Stratego. First, Section 4.1 describes how each move may give information about a piece. Section 4.2 shows how the probability distribution is updated after each opponent's move. Finally, Section 4.3 discusses how bluffing behaviour can be modeled.

4.1 Observing Moves

Each move in the game might give the opponent a clue of which rank a certain piece is. The most obvious observation one can make is that any piece that has been moved in past turns, cannot be a Flag or Bomb, as those ranks are immovable. Also, any piece that moves multiple squares in one turn, is definitely a Scout.

The moves described in the examples mentioned above, give a clear indication which ranks can be excluded from consideration for a given piece. In most cases however, the player can merely guess which rank a certain piece could have. It is fair to assume that players make their moves based on the opponent's pieces of which they know the rank. For instance, if a player knows that the opponent's Colonel is occupying a certain square and his Major is standing nearby that square, the player will move the Major away from that

[2]For details of these rules we refer to the International Stratego Federation(ISF), www.isfstratego.com.

[3]http://www.strategousa.org/wiki/index.php/Main_Page

[4]http://www.metaforge.net/

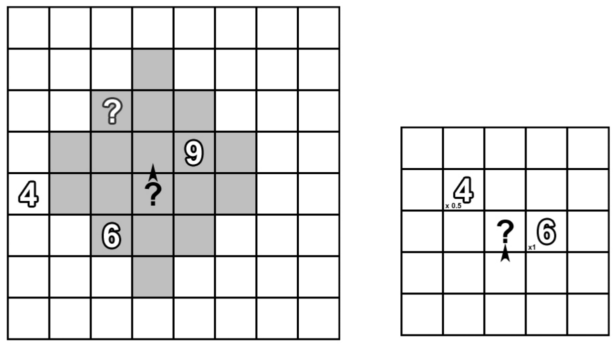
square to avoid being captured. However, if a General is standing nearby the opponent's Colonel, the player might choose to move the General closer to that square.

This kind of reasoning, if done from the other player's perspective, can be used to determine a more accurate probability distribution for the rank of an unknown piece. If the player knows that his opponent has knowledge about his Colonel and the player observes that the opponent moves his unknown piece closer to this Colonel, the player can increase the probability on a General for that unknown piece. The Stratego playing agent uses such an approach to update the probability distribution of unknown pieces after each opponent's move. The method is inspired by De Boer [8].

4.2 Updating Probability Distributions

The probability distribution of a moved unknown opponent's piece is determined by the own pieces surrounding that piece that are known to the opponent.

First, the direction of the move of the opponent is determined. This information is then used to decide which known pieces of the player the opponent is moving away from and which pieces he is moving towards. Only pieces within a radius of two squares from the moved opponent's piece are taken into account, in order not to do unnecessary computations. A piece is within the two-square radius if its Manhattan distance to the opponent's piece is at most two squares. Pieces that are further away than two squares usually have little or no influence on one's decision making process. This concept is shown in Figure 1(a), where a part of the board is depicted. The player's pieces are shown in white, the opponent's piece is shown in black. A number indicates the rank of a piece which is known to the other player. The higher the number, the stronger the piece. A question mark means that the piece is unknown to the other player. The opponent moves his piece in the direction of the arrow. The shaded area is taken into consideration for determining the rank of the opponent's unknown piece. In this case, the '9' and '6' are taken into account, where the opponent's piece is moving to and away from, respectively. The other pieces are not considered because the '4' is outside the two-square radius and the '?' is unknown to the opponent.



(a) The two-square radius surrounding a moved opponent's unknown piece, depicted as a black question mark. The pieces of the player are shown in white.

(b) Computing the weighted average

Figure 1: Stratego situations

After determining the direction, the updated probability distribution for the opponent's unknown piece can be calculated. The basic principle behind this update process is to 'move' the old probability distribution towards a new statistical probability distribution by a factor γ. This distribution is based on statistics retrieved from 70,000 games, played by humans. These statistics were gathered by analyzing the type of moves players made and which pieces they used in which situations.

The update process consists of two steps. These steps are shown in Equation (1a) and (1b).

$$\bar{W_r} = \frac{\sum_{a \in A}[P(r|r_a, m_a) \cdot w_a]}{\sum_{a \in A} w_a} \tag{1a}$$

$$\hat{P}_r = P_r + \left[(\bar{W}_r - P_r) \cdot \frac{\max_{a \in A} w_a}{w_{max}} \cdot \gamma \right] \tag{1b}$$

The first step is to calculate the weighted average statistical probability $\bar{W}_r$ for each rank r, shown in Equation (1a). Here, A is the set of all player's pieces a within the two-square radius of the moved opponent's piece. $P(r|r_a, m_a)$ denotes the probability on rank r given the rank r_a of the player's piece a. The parameter m_a represents the type of move, in this case, whether the opponent's piece moved towards or away from piece a. These probabilities were calculated beforehand from the statistics. The weight w_a reflects how much influence the player's piece has on the moved opponent's piece.

For example, an unknown piece has moved to the situation shown in Figure 1(b). The piece is surrounded by two own pieces, '4' and '6', located at a distance of 2 and 1 squares, respectively. The weights for those pieces are set as $\frac{1}{d}$, where d is the distance to the opponent's piece. These weights are shown in the corner of the square. Now suppose that $P(Major|r_a, m_a)$ with respect to piece '4' is 0.3 and 0.4 with respect to piece '6'. Then the weighted average statistical probability on a Major will be $\frac{(0.4 \cdot 1)+(0.3 \cdot 0.5)}{1+0.5} \approx 0.37$. By computing $\bar{W}_r$ for all ranks r, one can determine the weighted average statistical probability distribution.

The new probability on rank r is then calculated according to Equation (1b). The factor γ denotes how much the probability should change from the old probability to the average statistical probability, where $0 < \gamma \leq 1$. A value of $\gamma = 1$ means that the old probability is completely disregarded and the new probability simply becomes the average statistical probability.

Finally, a correction factor needs to be applied which depends on the largest weight assigned to a player's piece and the largest possible weight that can be assigned, denoted by w_{max}. This correction is needed because the weighted average statistical probability distribution is calculated as if the player's piece closest to the opponent's piece has the largest possible weight. This is incorrect if the closest piece is located at for instance two squares from the opponent's piece. In the experiments described in Section 5, the value of w_{max} is 1 and therefore the correction factor simply becomes the weight of the closest piece.

It should be noted that each time that the player moves a piece that is known to the opponent, the opponent can decide not to move his unknown pieces that are nearby that player's piece. In fact, this will almost always happen, as players may only move one piece at a time. For those unknown opponent's pieces that do not move and are not blocked by other pieces, the probability distribution should be modified as well, because by not moving the pieces, the opponent claims that those pieces are stronger than the player's piece. For instance, if the player moved his known Captain towards an unknown opponent's piece and the opponent decided not to move that piece away, the probability that the unknown piece is *e.g.* a Major should increase. This case is treated as a third type of move, next to 'move away' and 'move towards'.

After the probability distribution of the unknown opponent's piece is altered, the probabilities of all unknown pieces have to be normalized, according to two constraints [8]. The first constraint is that for every piece, the sum of the probabilities should be equal to 1. The first step of the normalization algorithm ensures that this constraint is met. For every unknown piece on the board, the sum of its probabilities is calculated and each of its probabilities is divided by this sum, as shown in Equation (2). Here $P_{x,y}$ denotes the probability of piece y having rank x and R is the set of all ranks.

$$\hat{P}_{x,y} = \frac{P_{x,y}}{\sum_{r \in R} P_{r,y}} \tag{2}$$

The second constraint is that for every rank, the sum of probabilities on that rank over all unknown opponent's pieces, should be equal to the number of missing pieces of that rank. This is the second step of the algorithm, shown in Equation (3). Here n_x denotes the number of unknown pieces of the opponent having rank x left on the board and S is the set of the unknown opponent's pieces left on the board.

$$\hat{P}_{x,y} = \frac{P_{x,y}}{\sum_{s \in S} P_{x,s}} \cdot n_x \tag{3}$$

Each time that the probabilities are normalized to fulfill one constraint, the other one is violated because of the way the probability mass is redistributed. However, by iterating this algorithm these violations become smaller every time. After several iterations, the probabilities have been normalized properly.

4.3 Bluffing

Although the model described in Section 4.1 is enough for the player to make a good move in most cases, it was created under the assumption that the opponent never bluffs. This is usually not the case, especially if the opponent is a professional Stratego player. Detecting these bluffs might be crucial, for instance if the opponent bluffs that a certain piece is stronger than it really is.

From the statistics, the probability of bluffing, $P(B)$, was determined for the average Stratego player. This is used as the initial probability that the opponent is bluffing. During the game, this bluffing probability is updated after each encounter where a new opponent's piece is revealed, according to Equation (4).

$$P_i(B) = \frac{1}{1+\beta} \cdot \left(P_{i-1}(B) + \frac{n}{i} \cdot \beta\right) \tag{4}$$

Here $P_i(B)$ is the bluffing probability after the *i*-th encounter in the game where a new opponent's piece was revealed. $P_{i-1}(B)$ is the bluffing probability before the *i*-th encounter. The factor β, where $0 < \beta \leq 1$, determines by how much the bluffing probability should change and n denotes the number of bluffs counted so far. By keeping track of the claims made by each opponent's piece during the game, one can determine whether these claims were bluffs the moment the piece is revealed. For instance, the opponent can claim that a certain piece is stronger than a Major if he moves that piece towards a player's known Major. If upon revealing it turns out that the rank is weaker than a Major, the opponent bluffed in the past.

To include the bluffing probability when calculating new probability distributions for the opponent's pieces, the model from Section 4.1 has to be altered. This can be done by changing Equation (1a). This modification is shown in Equation (5a) and (5b).

$$\bar{W_r} = \frac{\sum_{a \in A} P_{r,a} \cdot w_a}{\sum_{a \in A} w_a} \tag{5a}$$

$$P_{r,a} = \sum_{B} P(r|r_a, m_a, B) \cdot P(B) \tag{5b}$$

The probabilities $P(r|r_a, m_a, B)$ and $P(r|r_a, m_a, \neg B)$ were calculated beforehand from the statistics. The probabilities only have to be looked up in the corresponding conditional probability table. By combining Equation (5a) and (1b), one can calculate the new probability distribution for an unknown opponent's piece, while bearing in mind that the opponent might be bluffing.

5 Experiments and Results

This section describes the experiments done to measure the influence of opponent modeling. First, Section 5.1 discusses an experiment carried out to determine the influence of different values of the factor γ (see Section 4.2) on the correctness of the opponent model. Section 5.2 shows a similar experiment, except for different values of the factor β (see Section 4.3). Then, Section 5.3 discusses the influence of different values of both these factors on the average probability assigned to the real rank of an unknown piece. Finally, Section 5.4 discusses the win rates achieved by opponent modeling.

5.1 Influence of γ on Correctness

The first experiment was carried out to determine how different values of γ influence the correctness of the guesses of the opponent model and how this compares to the case where no opponent modeling is used.

The experiment was done in a self-play setup of 1,000 games. The games were played between two players, both running the same Stratego engine. This was an EXPECTIMAX engine, enhanced with the STAR1 and STAR2 pruning algorithms [3]. Additionally, the History Heuristic [14], Killer Moves [2] and Transposition Tables [19] were used. Player A used opponent modeling to determine the ranks of unknown pieces. Player B used a simple method, based on the number of unknown pieces left of a particular rank. A selection of 10 board setups was made. Both players iterated through all 10 setups such that every combination of board setups was played. This was done 10 times such that 1,000 games were played.

The correctness of the opponent model was determined by measuring how often the rank of an unknown piece was guessed correctly. This check was carried out every time that an unknown opponent's piece was involved in an encounter, revealing its rank. A strict scoring system was applied. Only if the highest probability in the probability distribution of the unknown piece was assigned to the real rank, the guess was counted as correct. In all other cases, it was counted as incorrect. Furthermore, if two ranks had the same highest probability, where one of the ranks was the real rank, then the guess was counted as incorrect.

The values of γ were varied between values of 0.25 and 1.0, while the factor β was kept at a constant value of $\beta = 0.01$. The higher the value of γ, the more the probability distribution of an unknown piece is altered after that piece makes a move. The results are shown in Table 1.

γ	Correct	Incorrect
0.25	12,467 (40.0%)	18,673 (60.0%)
0.50	12,456 (40.3%)	18,418 (59.7%)
0.75	12,541 (40.3%)	18,568 (59.7%)
1.0	12,126 (39.3%)	18,743 (60.7%)
No OM	5,384 (17.7%)	25,108 (82.3%)

Table 1: Number of correct guesses for different values of γ over 1,000 games

As Table 1 shows, by using opponent modeling there is a significant increase in the percentage of correctly guessed ranks. Without the use of opponent modeling, only 17.7% of the time the rank was guessed correctly. With opponent modeling however, this value was increased to approximately 40%. The values of $\gamma = 0.50$ and $\gamma = 0.75$ show the best result of 40.3%. If γ is too low, the probability distributions will not change fast enough, because new observations during the game will have little influence on the probability distributions. On the other hand, if γ is too high, the probability distributions will change too fast, as only the most recent observations are taken into account.

5.2 Influence of β on Correctness

The second experiment was similar to the one described in Section 5.1, except that this time the influence of different values of β was tested. The values of β were varied between 0.01 and 0.2, while the factor γ was kept at a constant value of $\gamma = 0.75$. The reason why β was varied between such small values, is that higher values will produce unrealistically high bluffing probabilities. If at an early stage in the game it turns out that so far the opponent bluffed half of the time, it would mean that for high values of β, the bluffing factor would quickly go towards 0.5. However, no player bluffs that often. For example, it could be that only 4 opponent's pieces were revealed so far, of which 2 bluffs. Considering that only 10% of the pieces are revealed at that point, it does not mean that this is the bluffing probability for the rest of the game. It is merely an indication what the real bluffing probability could be. Therefore, the bluffing probability is only changed slightly in the direction of 0.5. The results for different values of β are shown in Table 2.

β	Correct	Incorrect
0.01	12,541 (40.3%)	18,568 (59.7%)
0.05	12,282 (39.9%)	18,469 (60.1%)
0.1	12,517 (40.1%)	18,713 (59.9%)
0.2	12,309 (39.9 %)	18,539 (60.1%)
No OM	5,384 (17.7%)	25,108 (82.3%)

Table 2: Number of correct guesses for different values of β over 1,000 games

A value of $\beta = 0.01$ gives the best result. Although the differences shown in Table 1 and Table 2 seem marginal, they translate into larger differences when comparing the win rates, as shown in Section 5.4.

5.3 Influence of γ and β on Probabilities

The third experiment was carried out to determine the average probability on the real rank of an unknown piece. This was done by checking what the probability on the real rank of an unknown piece was, before it was revealed. Additionally, a second, similar calculation was done, but only for the case where the rank was guessed correctly, thus giving the average probability on the real rank in the case it was guessed correctly.

Like for the previous experiments, the tests were done for different values of γ and β. The results are shown in Table 3(a) and Table 3(b), for γ and β, respectively.

As the tables show, the probabilities that are assigned to the real rank of the piece are doubled, when compared to the case where no opponent modeling is used. The higher these values are, the better the engine will determine the best moves to make.

γ	Avg. Prob.	Avg. Prob. If Correct
0.25	0.315	0.601
0.50	0.320	0.614
0.75	0.328	0.637
1.0	0.330	0.666
No OM	0.152	0.290

(a) Average probabilities on the real rank for different values of γ over 1,000 games

β	Avg. Prob.	Avg. Prob. If Correct
0.01	0.328	0.637
0.05	0.324	0.632
0.1	0.327	0.633
0.2	0.323	0.630
No OM	0.152	0.290

(b) Average probabilities on the real rank for different values of β over 1,000 games

Table 3: Tuning γ and β

Remarkably, the highest probabilities are achieved for the value of $\gamma = 1.0$, while Table 1 shows that $\gamma = 1.0$ achieved the lowest percentage of correctly guessed ranks when using opponent modeling. This result has to do with the way how probability distributions are updated. For lower values of γ, the probability distribution is updated smoothly towards the statistical distribution. This means that if an opponent's move would cause the distribution to move from the correct rank towards an incorrect rank, thus towards a distribution where an incorrect rank has the highest probability, this change would be small and the correct rank would still be the most probable rank in the distribution. However, in general the average probability assigned to the real rank will be slightly lower for the same reason. As the values of γ become larger, the probability distribution is updated less smoothly. This means that the distribution can be updated quickly towards an incorrect rank, resulting in a lower percentage of correctly guessed pieces. At the same time, the average probability assigned to the real rank will in general be higher, because the distribution can move faster towards the correct rank as well.

5.4 Win Rate of Opponent Modeling

This section shows the win rate of player using opponent modeling, for different values of γ and β. The results are shown in Table 4(a) and Table 4(b) for different values of γ and β respectively.

γ	OM	No OM	Draws	Win rate
0.25	521	476	3	52.1%
0.50	553	446	1	55.3%
0.75	559	437	4	55.9%
1.0	545	455	0	54.5%

(a) The win rates over 1,000 games for different values of γ

β	OM	No OM	Draws	Win rate
0.01	559	437	4	55.9%
0.05	541	457	2	54.1%
0.1	546	450	4	54.6 %
0.2	527	471	2	52.7 %

(b) The win rates over 1,000 games for different values of β

Table 4: Tuning γ and β

The tables show the number of games won by each player. The best performance is achieved under $\gamma = 0.75$ and $\beta = 0.01$, with a win rate of 55.9%, which is a significant improvement. Even for other values of γ and β, opponent modeling still wins more than 52% of the games. Moreover, the tables show that the marginal differences that were shown in Section 5.1 and Section 5.2 for different values of γ and β, have been translated into significant differences in the win rates. Also, for a value of $\gamma = 1.0$, the win rate is still quite high, despite that only 39.3% of the pieces was guessed correctly, as shown in Table 1. This is due to how the engine decides the best move to make, based on the probability distributions of unknown pieces. As Table 3(a) showed, the probabilities assigned to the real rank of a piece were the highest for this value of γ. This means that the engine can still make relatively good decisions.

6 Conclusions and Future Research

This article described a method to model the opponent in Stratego. By analyzing the moves of the opponent, probability distributions can be derived in order to determine the most likely rank of an unknown piece.

Experiments reveal that opponent modeling increases the percentage of correctly guessed ranks significantly. If no opponent modeling is done, the percentage of correctly guessed pieces is 17.7%. By modeling the opponent, this percentage is increased to 40.3% for $\gamma = 0.75$ and $\beta = 0.01$.

The average probability that is assigned to the real rank of an unknown piece, shows an increase as well. If no opponent modeling is done, this average probability is 0.152. If one only takes the cases where the piece was guessed correctly, this probability is 0.290. By using opponent modeling, these two probabilities are more than doubled, in the best case to 0.330 and 0.666, respectively.

Finally, the self-play experiments show that opponent modeling results in a win rate of 55.9%, for values of $\gamma = 0.75$ and $\beta = 0.01$, which is a significant improvement. Even for other values of γ and β, opponent modeling still wins more than 52% of the games.

Although the opponent modeling method described in this article shows positive results, there is still plenty of room for improvement. First of all, the values of γ and β may still be tuned to achieve higher win rates. Secondly, the bluffing probability is independent of individual pieces. In other words, the bluffing probability is the same for every unknown piece, regardless of the surroundings of that piece. However, players may be more likely to bluff in certain situations. It would be interesting to see how situation-dependent bluffing probabilities would affect the performance of the opponent model. A third point of improvement is to use statistics of board setups to determine initial probability distributions for each piece. Finally, a last point of improvement is the recognition of patterns on the board, especially Bomb and Flag patterns are useful.

References

[1] *Stratego Instructions*. Milton Bradley Co., 1986. Obtained at http://safemanuals.com/.

[2] S. G. Akl and M. N. Newborn. The Principal Continuation and the Killer Heuristic. In *1977 ACM Annual Conference*, pages 466–473, New York, NY, USA, 1977. ACM Press.

[3] B. W. Ballard. The *-Minimax Search Procedure for Trees Containing Chance Nodes. *Artificial Intelligence*, 21(3):327–350, 1983.

[4] D. Billings, D. Papp, J. Schaeffer, and D. Szafron. Opponent Modeling in Poker. In K. Ford, editor, *AAAI-98*, pages 493–499, Menlo Park, CA, United States, 1998. AAAI Press.

[5] D. Billings, L. Pea, J. Schaeffer, and D. Szafron. Using Probabilistic Knowledge and Simulation to Play Poker. In K. Ford, editor, *AAAI-99*, pages 697–703, Menlo Park, CA, United States, 1999. AAAI Press.

[6] Case No. 04-1344-KI. *Estate of Gunter Sigmund Elkan, vs. Hasbro, INC. et al.*, 2005. District Court of Oregon.

[7] A. Davidson, D. Billings, J. Schaeffer, and D. Szafron. Improved Opponent Modeling in Poker. In R. Perrault and A. G. Cohn, editors, *ICAI'2000*, pages 1467–1473, 2000.

[8] V. de Boer, L. J. M. Rothkranz, and P. Wiggers. Invincible: A Stratego Bot. *International Journal of Intelligent Games & Simulation*, 5(1):22–28, 2008.

[9] K. B. Korb, A. E. Nicholson, and N. Jitnah. Bayesian Poker. In K. Laskey and H. Prade, editors, *UAI-99*, pages 343–350. M. Kaufmann, 1999.

[10] H. Lu and Z. Xia. AWT: Aspiration with Timer Search Algorithm in Siguo. In H. J. van den Herik, X. Xu, Z. Ma, and M. H. M. Winands, editors, *Computers and Games, CG2008*, volume 5131 of *Lecture Notes in Computer Science*, pages 264–274. Springer-Verlag, 2008.

[11] M. Ponsen, J. Ramon, T. Croonenborghs, K. Driessens, and K. Tuyls. Bayes-Relational Learning of Opponent Models from Incomplete Information in No-Limit Poker. In A. Cohn, editor, *AAAI-08*, pages 1485–1487, Menlo Park, CA, United States, 2008. AAAI Press.

[12] I. Satz. The 1st Computer Stratego World Championship. *ICGA Journal*, 31(1):50–51, 2008.

[13] M.P.D. Schadd, M.H.M. Winands, and J.W.H.M. Uiterwijk. CHANCEPROBCUT: Forward Pruning in Chance Nodes. 2009. Submitted to CIG2009.

[14] J. Schaeffer. The History Heuristic. *ICCA Journal*, 6(3):16–19, 1983.

[15] F. Southey, M. Bowling, B. Larson, C. Piccione, N. Burch, D. Billings, and C. Rayner. Bayes' Bluff: Opponent Modelling in Poker. In *UAI-05*, pages 550–558, Arlington, Virginia, United States, 2005. AUAI Press.

[16] K. Stengård. Utveckling av Minimax-Baserad Agent för Strategispelet Stratego. Master's thesis, Lund University, Sweden, 2006. In Swedish.

[17] C. Treijtel and L. J. M. Rothkrantz. Stratego Expert System Shell. In *GAME-ON 2001*, pages 17–21, 2001.

[18] Z. Xia, Y. Zhu, and H. Lu. Using the Loopy Belief Propagation in Siguo. *ICGA Journal*, 30(4):209–220, 2007.

[19] A. L. Zobrist. A new hashing method with application for game playing. *Technical report 88*, Computer Science Department, The University of Wisconsin, Madison, WI, USA, 1970.