Opponent Modeling in Scrabble

Mark Richards and Eyal Amir
Computer Science Department
University of Illinois at Urbana-Champaign
{ mdrichar,eyal } @cs.uiuc.edu

Abstract

Computers have already eclipsed the level of human play in competitive Scrabble, but there remains room for improvement. In particular, there is much to be gained by incorporating information about the opponent's tiles into the decision-making process. In this work, we quantify the value of knowing what letters the opponent has. We use observations from previous plays to predict what tiles our opponent may hold and then use this information to guide our play. Our model of the opponent, based on Bayes' theorem, sacrifices accuracy for simplicity and ease of computation. But even with this simplified model, we show significant improvement in play over an existing Scrabble program. These empirical results suggest that this simple approximation may serve as a suitable substitute for the intractable partially observable Markov decision process. Although this work focuses on computer-vs-computer Scrabble play, the tools developed can be of great use in training humans to play against other humans.

1 Introduction

Scrabble is a popular crosswords game played by millions of people worldwide. Competitors make plays by forming words on a 15 x 15 grid (see Figure 1), abiding by constraints similar to those found in crossword puzzles. Each player has a rack of seven letter tiles that are randomly drawn from a bag that initially contains 100 tiles. Achieving a high score requires a delicate balance between maximizing one's score on the present turn and managing one's rack in order to achieve high-scoring plays in the future.

Because opponents' tiles are hidden and because tiles are drawn randomly from the bag on each turn, Scrabble is a stochastic partially observable game [Russell and Norvig, 2003]. This feature distinguishes Scrabble from games like chess and go, where both players can make decisions based on full knowledge of the state of the game. Stochastic games of imperfect information can be modeled formally by partially observable Markov decision processes (POMDPs) [Littman, 1996]. While POMDPs are expressive and theoretically powerful, solving them is intractable for problems containing more than a few states.

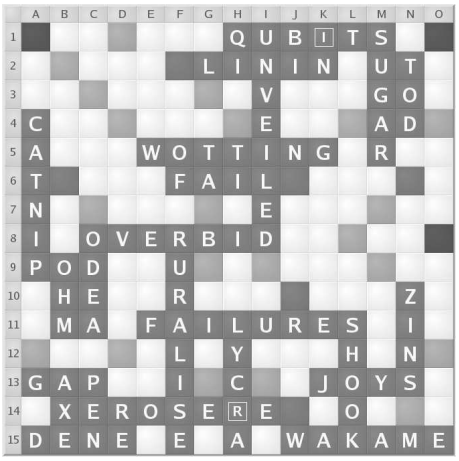

Figure 1: A sample Scrabble game. The shaded premium squares on the board double or triple the value of single letter or a whole word. Note the frequent use of obscure words.

At the beginning of a Scrabble game, an opposing player can hold any of more than four million different racks. Although the number of possibilities decreases as letters are drawn from the bag, solving Scrabble directly with a formal model like POMDPs does not seem to be a viable option.

Scrabble's inherent partial observability invites comparison with games like poker and bridge. Significant progress has been made in managing the hidden information in those games and in creating computer agents that can compete with intermediate-level human players [Billings *et al.*, 2002] [Ginsberg, 1999]. In Scrabble, championship-level play is already dominated by computer agents [Sheppard, 2002]. Although computers can already play better than humans, Scrabble is not a solved game. Even the best existing computer Scrabble agents can improve their play by incorporating knowledge about the unseen letters on the opponent's rack into their decision-making processes. Improvements in the handling of hidden information in Scrabble could shed insight into more strategically complex partially observable games such as poker. Furthermore, advanced computer Scrabble agents are of great benefit to expert human Scrabble

players. Humans rely on computer Scrabble programs to improve their play by analyzing previous games and identifying where suboptimal decisions were made.

One of the strategies that has been successfully used in poker-playing programs is opponent modeling—trying to identify what cards the opponents have and how they might play, based on observations of previous plays. [Billings *et al.*, 2002] In this work, we propose an opponent modeling strategy for Scrabble.

First, we run simulations in which one of the players is given full knowledge of his opponent's rack. These results show how much potential benefit there could be in attempting to make such inferences. Then we attempt to achieve some fraction of that potential improvement by creating a simple model of our opponent based on Bayes' theorem [Bolstad, 2004].

Our computer agent makes inferences about what tiles the opponent may have based on observations from previous plays. While our agent relies heavily on multi-ply simulations—as other popular computer Scrabble programs do—we use these inferences to bias the contents of our opponent's rack during simulation towards those letters which we believe he is more likely to have. Our model sacrifices accuracy for simplicity. But even with this simple model, we show empirical results that suggest this strategy is significantly better than other common approaches for computer play that make no attempt to deduce the opponent's letters. This strategy may be a suitable substitute for the computationally intractable partially observable Markov decision process.

The structure of the paper is as follows. Section 2 gives an overview of Scrabble, including a review of the rules and basic strategy. Section 3 discusses previous work on Scrabble-playing artificial intelligence. In Section 4, we present an algorithm which makes inferences about the opponent's tiles. Experimental results are discussed in Section 5. Finally, possibilities for future work are presented in Section 6.

2 Scrabble Overview

Alfred M. Butts invented Scrabble during the 1930s. He used letter frequency counts from newspaper crossword puzzles to help determine the distribution of tiles and their relative point values. Of the 100 letters in the standard Scrabble game, there are 9-12 tiles for common vowels like A, E, and I, but only one tile each for less common letters like Q, X, and Z[1].

Point values for the individual letters range from one point for the vowels to 10 points for the Q and Z. There are two blank tiles that act as "wild cards"; they can be substituted for any other letter. The blanks do not have an intrinsic point value but are extremely valuable because of the flexibility they add to a player's rack.

The first player combines two or more of his letters into a word and places it on the board with one letter touching the center square. Thereafter, players alternate placing words on the board, and each new word must have at least one letter that is adjacent to an existing word. New tiles placed on a single turn must all be played in one row or one column.

Players score points for all new words formed on each turn. The score for each word is determined by adding up the total points for the individual tiles; premium squares distributed throughout the board can double or triple the value of an individual tile or the whole word.

As long as letters remain in the bag, players replenish their rack to seven tiles after each turn. A Scrabble game normally ends when the bag is empty and one player has used all of his tiles. The game can also end if neither player can make a legal move, but in practice this rarely happens.

When a player manages to use all seven of his letters in a single turn, the play is called a *bingo* and scores a 50-point bonus. While novice players rarely, if ever, play a bingo, experts might average two or more per game. Since experts usually score in the 400-500 point range, the bonus for a bingo is highly significant.

2.1 Basic Strategy

Human Scrabble players must exert considerable effort to develop an extensive vocabulary—including knowledge of many obscure words. Since computer agents can easily be programmed to "know" all the legal words and can quickly generate all possible plays for any rack and board configuration, they already have a significant advantage over human players.

A large vocabulary and the ability to recognize high-scoring opportunities are necessary but not sufficient for high-level play. Simply making the highest-scoring legal play on each turn is not an optimal strategy. Using this greedy approach frequently causes a player to retain tiles that are naturally more difficult to play, eventually leading to awkward racks like <UHHWVVY>. With such a challenging rack, even the highest-scoring move is not likely to be very good.

More experienced players are willing to sacrifice a few points on the current turn in order to play off an awkward letter or to retain for future turns sets of letters that combine well with each other. Maintaining a good mix of consonants and vowels and avoiding duplicate letters (which reduce flexibility) are common goals of *rack balance* strategies. Perhaps most importantly, expert Scrabble players try to manage their racks so as to maximize the potential for bingo opportunities. [Edley and Williams, 2001]. In general, the concept of rack balance causes a player to evaluate the merits of a move based on how many points it scores on the current turn and on the estimated value of the letters that remain on the rack (called the *leave*).

Defensive tactics are also important. A player does not want to make moves that will create high-scoring opportunities for the opponent. Furthermore, if the board configuration and opponent's rack are such that the opponent could make a high-scoring move on his next turn, a player might want to consider moves that will block that opportunity, even if the blocking play does not score as well on the current turn as some other available alternatives. Also, if one player manages to establish a large lead early in the game, it may be in his best interest to keep the board as closed as possible. For

[1]Variants of Scrabble have been developed for many different languages. Here we restrict our focus to the original English version.

example, he might try to cut off areas of the board where a lot of bingos could be played.

It should be noted that luck plays a significant role in the outcome of a Scrabble game. Sometimes the random drawing of letters overwhelmingly favors one player, and not even the best strategy could compensate for the imbalance. In a game like poker, a timely bluff can lead to a big win for a player with a lousy hand. But in Scrabble, a bad rack can be downright crippling. Of course, over the course of many games, the luck of the draw evens out and the most skilled player can expect to win more games.

2.2 Scope of Study

The primary goal of this work is to improve upon championship-caliber Scrabble computer programs by addressing the elements of uncertainty inherent in the game. We assume that our opponent is also a computer. Not all aspects of the game are relevant to the present work. We are restricting our focus in the following ways.

Number of Players

We assume a two-player game. Official rules allow for up to four players, but tournament matches (and even most casual games) involve only two competitors. As mentioned previously, there is already a great deal of luck involved in drawing "good" tiles; having more than two players only exacerbates the problem because each competitor takes even fewer turns and therefore has fewer opportunities to overcome bad luck with skill.

Timing Constraints

Tournament Scrabble is played under time constraints, often 25 minutes per player per game. Point deductions are assessed for time taken in excess of the limit. Since our Scrabble-playing code is developmental in nature, we do not impose rigid timing restrictions. However, for practical purposes, the computation allowed to each agent is limited in a way that keeps the running time for a game to about what it would be in a tournament setting: 40–60 minutes.

Endgame

Once the bag of letters has been exhausted, a player may deduce exactly the opponent's rack, simply by observing the letters on the board and on his own rack. Expert Scrabble players do this kind of tile counting routinely. When the bag is empty, Scrabble becomes a game of perfect information, and strategy changes. The focus of this work is decision-making under uncertainty, so we ignore the facets of endgame strategy, which have been studied elsewhere [Sheppard, 2002].

Lexicon

The set of permissible words has a significant impact on how Scrabble is played. Words which are hyphenated or which occur exclusively as proper nouns or abbreviations are always illegal. But inclusion of certain slang, colloquial, archaic, and/or obscure words varies from one "official" word list to another. These differences can change how some letters are evaluated. For example, if the word QI is allowed, the Q is likely considered an asset; otherwise it is a significant liability. In this study, we use the TWL06 (Tournament Word List) lexicon exclusively. Our belief is that while specific parameters may need to be tuned for the various lists, the general models and strategies are applicable to any of the commonly used word sets.

Challenges and Bluffing

Finally, we ignore the "challenge" rule. Normally, when a competitor makes a play, his opponent has the option of challenging the legality of the newly played word(s). In this case, both players look up the word in whatever dictionary has been agreed upon for the game, and the loser of the challenge forfeits a turn. Some players will intentionally make a "phoney," hoping that their opponents will be unfamiliar with the word and will challenge it. This kind of bluffing tactic must be reckoned with in human tournament play. In this work, we focus on computer-vs-computer play. Our agent's knowledge base contains all of the legal words, and we assume that our opponent has the same information.

3 Previous Work

While computers are indisputably the most proficient Scrabble players, it is not generally known which Scrabble-playing program is the best. Brian Sheppard's Maven program decisively defeated human World Champion Ben Logan in a 1998 exhibition match. Since that time, the National Scrabble Association has used Maven to annotate championship games. Maven's architecture is outlined in [Sheppard, 2002]. The program divides the game into three phases: the endgame, the pre-endgame, and the midgame. The endgame starts when the last tile is drawn. Maven uses B*-search [Berliner, 1979] to tackle this phase and is supposedly nearly optimal. Little information is available about the tactics used in the pre-endgame phase, but the goal of that module is to achieve a favorable endgame situation.

The majority of the game is played under the guidance of the midgame module. On its turn, Maven generates all possible legal moves and ranks them according to their immediate value (points scored on this turn) and on the potential of the leave. The values used to rank the leaves are computed offline through extensive simulation. For example, the value of the leave QU is determined by measuring the difference in future scoring between a player with that leave and his opponent, and averaging that value over thousands of games in which it is encountered.

Once all legal moves have been generated and ranked according to the static evaluation function, Maven uses simulations to evaluate the merit of those moves with respect to the current board configuration and the remaining unseen tiles. Since it is not uncommon to have several hundred legal plays to choose from on each turn, deep search is not tractable. Sheppard suggests that deep search may not be necessary for excellent play. Since expert players use an average of 3–4 tiles each turn, complete turnover of a rack can be expected every two to four turns. Simulations beyond that level are of questionable value, especially if the bag still contains many letters. Maven generally uses two- to four-ply searches in its simulations.

After the publication of [Sheppard, 2002], rights to Maven were purchased by Hasbro, and it is now distributed with that

company's Scrabble software product. Since its commercialization, additional details about its strategies and algorithms have not been publicly available.

Jim Homan's CrossWise is another commercial software package that can be configured to play Scrabble. In 1990 and 1991, CrossWise won the computer Scrabble competition at the Computer Olympiad. (In subsequent Olympiad competitions, Scrabble has not been contested.) The algorithmic details of CrossWise are not readily accessible. Unfortunately, Maven and Crosswise have not been pitted against each other in an official competition, so it is not known which program is superior. Based on publicly available information, Maven would probably have the edge. Homan claims that CrossWise generated over US $3 million in sales, which shows that there is a great demand for powerful Scrabble computer programs.

In March 2006, Jason Katz-Brown and John O'Laughlin released Quackle, an open source crossword game program[2]. Quackle's computer agent has the same basic architecture as Maven. It uses a static evaluation function to rank the list of candidate moves and then makes a final decision based on the results of simulations using a small subset of the most promising candidate moves. During the simulations, Quackle must select one or more potential moves for the opponent. Since Quackle does not know what letters its opponent holds, it randomly selects a rack of letters from the set of tiles that it has not seen (i.e. all letters that are not currently on its own rack and have not already been placed on the board). Quackle ignores the fact that not all possible racks are equally likely for the opponent.

In the next section we show how we can use the opponent's most recent play to bias our selection of his tiles during simulation towards racks which we believe are more likely to occur. Estimating the probability that our opponent holds certain tiles requires us to create a model of his decision-making process. Opponent modeling has been shown to be profitable in other partially observable games, such as poker [Billings *et al.*, 2002]. We suspect that opponent modeling in Scrabble would be somewhat easier than in poker. Many different styles of play can be played profitably in poker, and expert players are known to change their strategies drastically during a single match. Among expert Scrabble players– computers and humans– there is much less variation in strategy. We expect this fact to lead to simpler opponent models in Scrabble.

4 Modeling the Opponent's Rack and Play Selection

A reasonable question to ask is, "how much would it help me if I could see my opponent's rack?" To help answer this question, we conducted experiments in which we allowed our player to have full knowledge of the opponent's letters. During the simulation phase of the decision-making process, when evaluating the possible responses our opponent could make to one of our available moves, we generate all of the possible moves that the opponent could make using the rack that he actually has (instead of randomly assigning tiles from the set of letters that we have not seen). The opponent in these tests was Quackle's Strong Player, which also uses simulations but makes no assumptions about our letters. The results are summarized in Table 1 and Figure 2. There is high variability in the final scores for the games, with extreme wins and losses for both players. This underscores the role that luck plays in the outcome of a Scrabble game. It is clear, however, that the Full Knowledge Player has a great advantage, scoring 37 more points per game on average. The difference is highly statistically significant ($p < 10^{-5}$ using random permutation tests [Ramsey and Schafer, 2002].)

Knowing the contents of our opponent's rack allows us to play more aggressively in some situations, because we can be certain that our opponent does not have a high-scoring countermove. It allows us to avoid plays that would set him up for a bingo on his next turn. And it gives us an opportunity to block spots on the board that would be lucrative to him on his next turn, given his current rack. In real-world play, we cannot know for certain what letters our opponent holds– unless the bag is empty–but the results of these hypothetical full knowledge simulations give an upper bound on what we can expect to gain by trying to make some inferences about this hidden information.

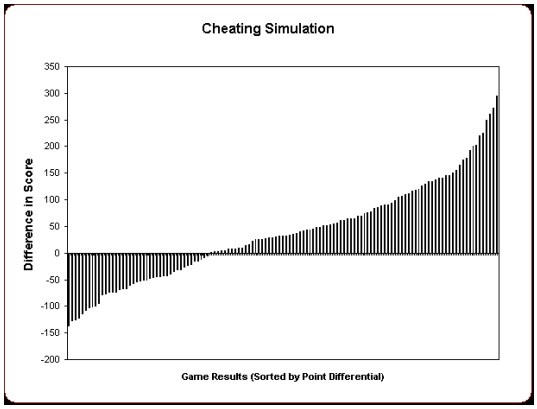

Figure 2: Point differential over 127 games between Quackle's Strong Player and an agent with full knowledge of its opponent's tiles. Each bar shows the result of one game. Negative values indicate games in which the Full Knowledge agent lost. The wide variability in the outcomes is a result of the luck inherent in drawing letters from the bag.

	Full Knowledge	**Quackle**
Wins	87	61
Mean Score	438	401
Biggest Win	295	136

Table 1: Summary of results for 127 games between Quackle's Strong Player and an agent with full knowledge of its opponent's tiles.

During simulation, our model of the opponent consists of two parts. First, we construct a probability distribution over the possible racks that the opponent may have. Second, we

[2]To avoid legal issues, Quackle does not officially have anything to do with Scrabble. References to Quackle in this work denote version 0.91.

must model the decision-making process that our opponent would go through to select a move, given his rack and the configuration of the board. Obviously, these two components are closely related: the letters left on the opponent's rack before replenishing from the bag are a direct result of the move he chose to play on his last turn.

While we do not know exactly what tiles our opponent has, we can make some inferences based on his most recent move. Consider the game situation shown in Figure 3. Our opponent, playing first, held ?IIMNOO and played <8E IMINO (?O) 16>[3]. We can observe only the letters he played—IMINO—and the letters on our own rack GLORRTU, leaving 88 letters which we have not seen: 86 in the bag and two on his rack. When the opponent draws five letters to replenish his rack, each of the tiles in the bag is equally likely to be drawn. But assuming that the two letters left on his rack can also be viewed as being randomly and uniformly drawn from the 88 letters that we have not seen would be a gross oversimplification. Of the 372 possible two-letter pairs for the oppo-

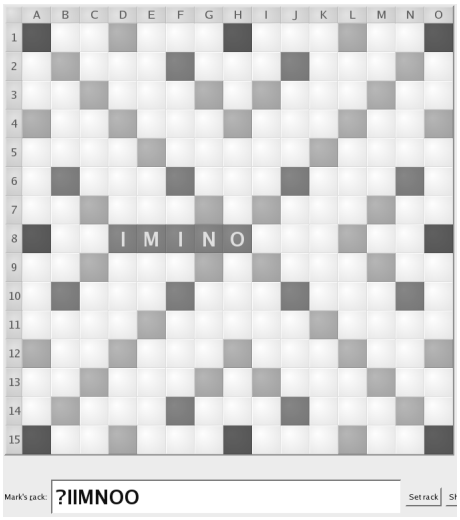

Figure 3: The state of the board after observing the opponent play IMINO with a leave of ?O. The active player holds GLORRTU. Between the two letters left on the opponent's rack and the letters left in the bag, there are 88 unseen tiles.

nent's leave (??, ?A,...,?Z,AA,AB,...,AZ,...,YZ), some are considerably more probable, given the most recent play. Suppose our opponent's leave is ?H. That would mean that he held ?HIIMNO before he played. If he had held that rack, he could have played <8D HOMINId 80>, a bingo which would have earned him 64 more points than what he actually played. Were our opponent a human, we would have to account for the possibility that he does not know this word or that he simply failed to recognize the opportunity to play it. But since our opponent is a computer, we feel confident that he would not have made this oversight and conclude that his leave after IMINO was not ?H. Likewise, we can assume that his leave was not ?L (<8D MILIION 72>).

[3]The word IMINO was played on row 8, column E for 16 points, leaving letters ?O on the player's rack.

Suppose his leave was NV. With IIMNNOV, he could have played <8G VINO (IMN) 14>. While this would have scored two fewer points than IMINO, it has a much better leave (IMN instead of NV) and would likely have been preferred. In general, we can consider each of the possible leaves that an opponent may have had (based on the set of tiles we have not seen), reconstruct what his full rack would have been in each case, generate the legal moves he could have made with that rack, and then use that information to estimate the likelihood of that leave. Using Bayes' theorem

$$P(leave \mid play) = \frac{P(play \mid leave)P(leave)}{P(play)}$$

The term $P(leave)$ is the prior probability of a particular leave. It is the probability of a particular combination of letters being randomly drawn from the set of all unseen (by us) letters. This is the implicit assumption that Quackle makes about the opponent's leave. The prior probability for a particular draw D from a bag B is

$$P(leave) = \frac{\prod_{\alpha \in D} \binom{B_\alpha}{D_\alpha}}{\binom{|B|}{|D|}}$$

where α is a distinct letter, B_α is the number of α-tiles in B, D_α is the number of α-tiles in D, and $|B|$ and $|D|$ are respectively the size of the bag and size of the draw.

We will be interested in computing probabilities for all of the possible leaves; we can therefore take advantage of the fact that

$$P(play) = \sum_{leave} P(play \mid leave)P(leave)$$

The $P(play \mid leave)$ term is our model of the opponent's decision-making process. If we are given to know the letters that comprise the $leave$, then we can combine those letters with the tiles that we observed our opponent play to reconstruct the full rack that our opponent had when he played that move. After generating all possible legal plays for that rack on the actual board position, we must estimate the probability that our opponent would have chosen to make that particular play. We are assuming that our opponent is a computer, so it might be reasonable to believe that our opponent also makes his decisions based on the results of some simulations. Unfortunately, simulating our opponent's simulations would not be practical from a computational standpoint. Instead, we naively assume that the opponent chooses the highest-ranked play according to the same static move evaluation function that we use. In other words, we assume that our opponent would make the same move that we would make if we were in his position and did not do any simulations. This model of our opponent's decision process is admittedly overly-simplistic. However, it is likely to capture the opponent's behavior in many important situations. For example, one of the things we are most interested in is whether our opponent can play a bingo (or would be able to play a bingo if we made a particular move). If a bingo move is possible, it is very likely to be the highest-ranking move according to our static move evaluator anyway. A key advantage of modeling the opponent's decision-making process in this way is that the calculation

of $P(play \mid leave)$ is straightforward. If the highest-ranked word for the corresponding whole rack matches the word that was actually played, we assign a probability of 1; otherwise, we assign a probability of 0. Let M be the set of all leaves for which $P(play \mid leave) = 1$. Then the computation simplifies to

$$P(leave \mid play) = \frac{P(leave)}{\sum_{leave \in M} P(leave)}$$

Returning to our earlier example, if the opponent plays IMINO, there are only 27 of 372 possibilities to which we assign non-zero probability. Using only the prior values for $P(leave)$, the actual leave ?O is assigned a probability of 0.003. After conditioning on $play$, that leave is assigned a 0.02 probability. In this example, there were only 372 possible leaves to consider, but in general, there could be hundreds of thousands. It may be too expensive to run simulations for every possible leave, but we would like to consider as much of the probability mass as possible. Using the posterior probabilities, 60% of the probability mass is assigned to about 10 possible leaves. Using only the priors, the 10 most probable leaves do not even account for 10% of the probability mass. However many samples we can afford computationally, we expect to get a much better feel for what our opponent's response to our next move might be if we bias our sampling of leaves for him to those that are most likely to occur.

During simulation, after sampling a leave according to the distribution discussed above, we randomly draw tiles from the remaining unseen letters to create a full rack. We have created an Inference Player in the Quackle framework that is very similar to Quackle's Strong Player. It runs simulations to the same depth and for the same number of iterations as the Quackle player. The only difference is in how the opponent's rack is composed during simulation.

5 Experimental Results

Table 2 shows the results from 630 games in which our Inference agent competed against Quackle's Strong Player. While there is still a great deal of variance in the results, including big wins for both players, the Inference Player scores, on average, 5.2 points per game more than the Quackle Strong Player and wins 18 more games. The difference is statistically significant with $p < 0.045$.

	With Inferences	**Quackle**
Wins	324	306
Mean Score	427	422
Biggest Win	279	262

Table 2: Summary of results for 630 games between our Inference Agent and Quackle's Strong Player.

The five-points-per-game advantage against a non-inferencing agent is also significant from a practical standpoint. To give the difference some context, we performed comparisons between a few pairs of strategies. The baseline strategy is the greedy algorithm: always choose the move that scores the most points on the present turn. An agent that incorporates a static leave evaluation into the ranking of each move defeats a greedy player by an average of 47 points per game. When the same Static Player competes against Quackle's Strong Player, the simulating agent wins by an average of about 30 points per game. To be able to average five points more per game against such an elite player is quite a substantial improvement. In a tournament setting, where standings depend not only on wins and losses but also on point spread, the additional five points per game could make a significant difference. The improvement gained by adding opponent modeling to the simulations would seem to justify the additional computational cost. The expense of inference calculations has not been measured exactly, but it is not excessive considering the costs of simulation in general.

6 Conclusions and Future Work

The empirical results discussed above suggest that opponent modeling adds considerable value to simulation. We do not expect that the value of information gained through opponent modeling will be the same in all situations. In particular, we expect the value to vary with the number of unseen tiles and with the number of tiles played by the opponent on his previous moves. Efforts are currently underway to analyze when the opponent modeling is most helpful.

References

[Berliner, 1979] Hans Berliner. The B* tree search algorithm: A best-first proof procedure. *Artif. Intell.*, 12:23–40, 1979.

[Billings *et al.*, 2002] Darse Billings, Aaron Davidson, Jonathan Schaeffer, and Duane Szafron. The challenge of poker. *Artif. Intell.*, 134:201–240, 2002.

[Bolstad, 2004] William Bolstad. *Introduction to Bayesian Statistics*. Wiley, Indianapolis, IN, 2004.

[Edley and Williams, 2001] Joe Edley and John D. Williams. *Everything Scrabble*. Pocket Books, New York, 2001.

[Ginsberg, 1999] M. L. Ginsberg. GIB: Steps toward an expert-level bridge-playing program. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 584–589, 1999.

[Littman, 1996] Michael Lederman Littman. Algorithms for sequential decision making. Technical Report CS-96-09, 1996.

[Merriam-Webster, 2005] Merriam-Webster. *The Official Scrabble Players Dictionary*. Merriam-Webster, 2005.

[Ramsey and Schafer, 2002] Fred L. Ramsey and Daniel W. Schafer. *The Statistical Sleuth: A Course in Methods of Data Analysis*. Duxbury, Pacific Grove, CA, 2002.

[Russell and Norvig, 2003] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 2003.

[Sheppard, 2002] Brian Sheppard. World-championship-caliber Scrabble. *Artif. Intell.*, 134:241–245, 2002.