Game-Tree Search with Combinatorially Large Belief States

Austin Parker, Dana Nau, V.S. Subrahmanian
Department of Computer Science, University of Maryland
College Park, MD, 20742, USA
Email: {austinjp,nau,vs}@cs.umd.edu

Abstract

In games such as kriegspiel chess (a chess variant where players have no direct knowledge of the opponent's pieces' locations) the belief state's sizes dwarf those of other partial information games like bridge, scrabble, and poker–and there is no easy way to generate states satisfying the given observations. We show that statistical sampling approaches can be developed to do well in such games.

We show that it is not necessary for the random sample to consist only of game boards that satisfy each and every one of a player's observations. In fact, we win 24% more often by beginning with such completely consistent boards and gradually switching (as the game progressed) to boards that are merely consistent with the latest observation.

This surprising result is explained by noting that as the game progresses, a board that is consistent with the last move becomes more and more likely to be consistent with the entire set of observations, even if we have no idea what sequence of moves might have actually generated this board.

1 Introduction

One of the biggest difficulties in using game-tree search algorithms in imperfect-information games is that the game trees have huge branching factors. Usually the *belief-state size* (the number of current states that are consistent with everything a player knows about the game) is combinatorially large. Hence the number of moves that the opponent *might* be able to make, and the number of states that these moves might lead to, are also quite large.

The best known way to circumvent this problem is to do statistical sampling: to generate a set of random game boards consistent with the current belief state, do a perfect-information game-tree search on each of those game boards, and average the minimax values. This has worked well in games such as bridge [Smith *et al.*, 1998; Ginsberg, 1999], scrabble [Sheppard, 2002], and poker [Billings *et al.*, 2003].

The success of statistical sampling in the above games relates, at least in part, to several conditions that do not hold in various other imperfect-information games, such as kriegspiel chess, Civilization, StarCraft, Quake3, Counter-Strike, and FreeCiv. More specifically:

- *Belief-state size.* Bridge, poker, and scrabble have relatively small belief states. At the start of a bridge hand, the size of each player's belief state is about 10,000,000 after the dummy's cards have been exposed; and this number decreases exponentially as the game proceeds. Although 10,000,000 may seem like a large number, it is dwarfed by the belief-state sizes in several other games. For example, in kriegspiel chess,[1] the size of each belief state at midgame is well above 10^{13}; and in the *small* world map for FreeCiv, the belief-state size is about 10^{30}.
- *Uncertainty model.* The uncertainty in bridge, poker, and scrabble is due to external factors that can easily be modeled stochastically: the random deal of the cards in bridge or poker, and the random choice of a tile in Scrabble. Thus it is easy to tell whether or not a state s is consistent with a belief state b, and to assign a probability to s given b.

  In games such as kriegspiel chess, the uncertainty arises from lack of observability of the opponent's actions. This uncertainty has no simple stochastic model. Instead, telling whether a state s is consistent with the current belief state b means checking whether there is a *history* (i.e., a sequence of moves) that is consistent with b and leads to s. In the average case, this takes exponential time.
- *Simplifying approximations.* In bridge and poker, usually the statistical sampling is not done on the game itself, but on a simplified approximation in which the state space and belief states are much smaller. In bridge, the approximation is done by treating various sets of states as if they were equivalent, a technique that was first used in the game of sprouts [Applegate *et al.*, 1991]. In poker, a linear-programming approximation has been used [Billings *et al.*, 2003]. In kriegspiel chess, it is unclear how or whether such an approximation could be constructed.

To summarize, our objective is to determine how to make game-tree search work successfully in imperfect-information games that have huge belief states, no simple stochastic model of uncertainty, and no obvious way to construct a simplified approximation of the game. Our results are:

[1] Kriegspiel chess is like ordinary chess except that neither player can see the other's pieces. Section 3 gives a fuller description.

```
procedure Choose-move(S)
  M ← {moves applicable to states in S}
  for every s ∈ S and every m ∈ M do
    v_{s,m} ← Evaluate-board(γ(s, m))
  return argmax_{m∈M} Σ_{s∈S} v_{s,m} P(s)
```

Figure 1: Abstract statistical-sampling algorithm for move evaluation. Evaluate-board is a perfect-information search algorithm such as alpha-beta.

1. We describe several algorithms for generating the random sample of game boards used in the tree search. AOSP generates game boards that are consistent with the entire sequence O of observations that a player has made during the game. LOS only requires consistency with the last observation o_i. HS behaves like AOSP at the beginning of the game, but as the game progresses it gradually switches over to behaving like LOS.
2. We analyze the performance of our algorithms theoretically. Surprisingly, our analysis suggests that there are cases in which LOS will play well and HS will do better than AOSP.
3. Our experimental tests in the game of kriegspiel chess confirm our analysis' hypotheses. In our experiments, Timed LOS (LOS with a time limit) did much better than random play, Timed AOSP did much better than Timed LOS; and Timed HS did much better than Timed AOSP.

2 Game-Tree Search with Statistical Sampling

To describe how statistical sampling is used for game-tree search in imperfect-information games, we need an abstract model for the kinds of games where it is used. For simplicity, we assume the game is zero-sum and there are two players p_1, p_2 who move in alternation.

As the game progresses, the players' moves will generate a sequence of states $S_i = \langle s_0, s_1, \ldots \rangle$ called the *game history*. At each state s_i, each player p_j will be able to make an observation o_{ij} of s_i; usually o_{ij} will include complete information about p_j's position and partial information about the other player's position. At s_i, player p_j's *observation history* is $O_{ij} = \langle o_{1j}, o_{2j}, \ldots, o_{ij} \rangle$, and p_j's *belief state* is $b_{ij} = \{$all states that satisfy $O_{ij}\}$.

Our sampling algorithms will be based on the following properties of a state s: s is *last-observation consistent* if it is consistent with o_{ij}, and *all-observation consistent* if it is consistent with O_{ij}.

Figure 1 shows an abstract version of statistical game-tree search. S is the sample set of states, $\gamma(s, m)$ is the state produced by performing move m in state s, Evaluate-board is a perfect-information game-tree-search algorithm such as alpha-beta, and P is a probability distribution over the states in S. Some additional code must be added to handle the case where a move m is applicable to some states but not others; this code tends to be game-specific and we discuss it further in Section 4.

We now can define four different sampling algorithms that provide input for Choose-move. In each case, k is the desired number of states in the statistical sample, and i is how many moves the players have played so far.

- **LOS** (**L**ast **O**bservation **S**ampling) If there are fewer than k last-observation consistent states, then let S contain all of them; otherwise let S contain k such states chosen at random. Return Choose-move(S).
- **AOS** (**A**ll **O**bservation **S**ampling): Let $S = \emptyset$. Generate a random history $\langle s_1, \ldots, s_i \rangle$. If the history satisfies O_{ij}, then add s_i to S. Do this repeatedly until $|S| = k$. Return Choose-move(S).
- **AOSP** (**A**ll **O**bservation **S**ampling with **P**ool): AOSP returns both a move and a set of states (a pool) for p_j to use as input to AOSP next move. Every state in the pool is to be consistent with O_{ij}, though we do not assume that all such states are in the pool. Let S_0 be the pool AOSP returned last time, and $M = \{$all of the other player's possible responses to p_j's last move$\}$. Let $S_1 = \{\gamma(s, m) \mid s \in S_0, m \in M,\ m$ is applicable to s, and $\gamma(s, m)$ satisfies $O_{ij}\}$. If $|S_1| < k$, then let $S_2 = S_1$; otherwise let S_2 contain k states chosen at random from S_1. Let $m =$ Choose-move(S_2). Return $(m, \{\gamma(s, m) \mid s \in S_1\})$.
- **HS** (**H**ybrid **S**ampling): Like AOSP, HS returns a move and a set of states. Compute S_1 and S_2 same as in AOSP. If $|S_2| < k$ then let S_3 be a set of $k - |S_2|$ random last-observation consistent states; otherwise $S_3 = \emptyset$. Let $m =$ Choose-move$(S_2 \cup S_3)$. Return $(m, \{\gamma(s, m) \mid s \in S_1\})$.

Analyzing the performance of these algorithms is impossible without making simplifying assumptions, but there is more than one set of assumptions one might make. Below we do two analyses, based on two different sets of assumptions. The differing assumptions lead to differing conclusions about which algorithm will do better.

Analysis 1: Suppose each state has exactly b children, for some constant b. Suppose that we know all of p_j's moves but not the other player's moves. If the number of states is very large (e.g., 10^{13} or 10^{30} as described earlier), then during the early stages of the game, the number of states grows exponentially, with roughly $b^{i/2}$ possible states at the i'th move. Suppose that for each state s where it is the other player's move, the observation history O_{ij} eliminates, on the average, some fraction $1/c$ of that player's possible moves, where $c > 1$. Then the number of possible states at the i'th move given O_{ij} is $(b/c)^{i/2}$. Thus the probability of any individual state at depth i being consistent with O_{ij} is $(1/c)^{i/2}$, which approaches 0 at an exponential rate as i increases.

Thus, if the game continues to grow as a tree with a branching factor of b, then our analysis suggests the following:

- AOS will be computationally intractable. The probability of a random history being consistent with b_i is likely to be something like $1/c^{i/2}$. Thus, in order to produce a sample of size k, AOS will probably need to generate $k \cdot c^{i/2}$ histories, taking exponential time.
- AOSP is much more efficient computationally than AOS, but its sample set S_2 will decrease in size as the game progresses. The probability of a state s's successors being

consistent with b_i is $1/c$, since s is already known to be consistent with b_{i-1}. Hence the amount of time needed to generate k states for S_2 does not increase as quickly as it would if we generated the states from random histories like AOS does. It is still the case, however, that as the game progresses, S_2 will soon become too small for the results to have much statistical significance and AOSP's play will begin to resemble random play.

- Each board generated by LOS is unlikely to be consistent with the current belief state; thus the values computed by LOS are likely to be close to random.
- At the beginning of the game, HS will behave identically to AOSP. As the game proceeds and the size of S_2 decreases, HS will put more and more randomly generated boards into S_3, thus making the results more noisy. Thus HS's quality of play is likely to be worse than AOSP's.

Analysis 2: If there are n possible states in the game, then the number of moves at each level cannot continue to grow exponentially, but will eventually flatten out. The game "tree" will be a graph rather than a tree, with n nodes (at most) at each depth, one for each possible state. There will be b edges from each node at depth i to nodes at depth $i+1$, $1/c$ of which are consistent with any given observation; suppose these edges go to a random set of nodes. Then for each state s, the probability (under certain independence assumptions) that it is reachable in i moves is about $min(1, (n-1)^{i-3}((b/c)/(n-1))^{i-1})$. In other words, the probability that a randomly chosen state s has a history consistent with O_{ij} approaches 1 at exponentially. This suggests the following:

- As the game proceeds, the amount of time needed by AOS will eventually level off; its upper bound will be n times the amount of time needed by each call to the classical game-tree search algorithm.
- Rather than degrading to random play as in Analysis 1, AOSP's quality of play will eventually level off at some level above that, depending on the number of states available in the pool.
- As the game proceeds, the probability of a randomly generated board being consistent with the current belief state will increase toward 1; thus LOS will produce increasingly good quality of play. However, its play will be limited by the fact that it has no good way to assign relative probabilities to its randomly generated boards.
- At the beginning of the game, HS will behave identically to AOSP. As the game proceeds and AOSP's sample size decreases, HS will fill up the rest of the sample with randomly generated boards—but as the game proceeds, it will become increasingly likely that these randomly chosen boards are consistent with the current belief state. Thus HS's quality of play is likely to be better than AOSP's.

Summary. With the first set of assumptions, AOSP is likely to perform much better than LOS, and somewhat better than HS. With the second set of assumptions, it is unclear which of LOS and AOSP will be better, but HS is likely to perform better than AOS, AOSP, and LOS.

Since both sets of assumptions represent extremal cases, our hypotheses are that in many games the actual performance is likely to be somewhere in between, such that HS will perform somewhat better than AOSP. In other words, we hypothesize that in many games, if last-observation consistent boards are included in the statistical sample later in the game, this will help rather than hurt the evaluations. Section 5 describes our experimental test of that hypothesis.

3 Kriegspiel Chess

As our test domain, we chose the game of *kriegspiel chess* [Li, 1994; 1995]. This game is like chess except that neither player can see the other's pieces. When player x captures one of player y's pieces, a referee announces that he/she has made a capture but not what piece was captured, and the referee removes the captured piece from y's board but does not say what piece captured it. When x tries to make a move that is illegal (an attempted pawn take, or moving into check, or attempting to jump an opponent's piece), the referee announces that the move is illegal but not why. When x puts y's king in check, the referee tells both players that y's king is in check, and gives them partial information about how the check occurred (namely by rank, file, long diagonal, short diagonal, knight, or some combination of the above). Both players hear all referee announcements.

Twenty moves into a kriegspiel chess game, a conservative estimate is that at each node of the game tree, the current sequence of observations is consistent with more than 10^{13} board positions. Because of this uncertainty, kriegspiel chess is a notoriously difficult game to win [Li, 1994; 1995]; most games end in draws.

4 Implementation of the Algorithms

For Choose-move's Evaluate-board subroutine, we took the GPL'ed chess program provided by GNU and modified it to return a minimax value for a particular board. Note that for each $s \in S$ we call the evaluation function $|M|$ times, where $|M|$ is the number of applicable moves (20-40 in kriegspiel). The only exception to this is the case where a move $m \in M$ is not legal in some state $s \in S$; in this case we omit s when computing the average value for m. As shown in Figure 1, we then return the move with the highest average value.

4.1 Algorithms with Time Limits

In order to make fair comparisons among LOS, AOSP, and HS, they cannot be implemented in the exact way described in Section 2. They must be modified so that one of the inputs to the algorithm is t, the amount of time available to decide on a move, so that the algorithm can do as well as it can in that amount of time. The modified algorithms—Timed LOS, Timed AOSP, and Timed HS—are described below.

We have implemented all three of these algorithms, using a combination of C and C++. In the next few months, we intend to make our implementations publicly available, both as open-source software and as a kriegspiel-chess game server running on our web site.

Timed LOS: Rather than taking the set S as input as shown in Figure 1, Timed LOS generates the members of S one at

a time and evaluates them as they are generated, so that it can generate and evaluate as many boards as it can during the time available. Once the time is up, it returns the move whose average value is highest, as shown in Figure 1.

Timed AOSP: Timed AOSP maintains a pool of states $P = \{s_1, \ldots, s_p\}$ that are known to be consistent with the current belief state b. Using an estimate of how long Evaluate-board will take on each board, it calculates some number of boards k_t that it can evaluate during the available time t. The estimate is deliberately a little low, to try to keep Timed AOSP from running overtime and to ensure that there will be time left over to attempt to generate more consistent boards. There are three cases:

- If $p \geq k_t$ then Timed AOSP calls Choose-move($\{s_1, \ldots, s_{k_t}\}$), and returns the recommended move.
- If $0 < p < k_t$ then Timed AOSP calls Choose-move(P), and returns the recommended move.
- If $p = 0$ then Timed AOSP returns a random move.

During whatever remains of the available time, AOSP tries to generate more histories that are consistent with b; and for every such history, it adds the resultant board to the pool (see section 4.2 below).

Each time the referee makes an announcement, Timed AOSP must update the pool to be consistent with the announcement. This can cause the pool to either shrink (when Timed AOSP is told a move is illegal) or to grow (when Timed AOSP is told that the opponent has moved). This computation occurs at the beginning of AOSP's turn.

If the pool were allowed to grow unchecked, it could potentially get quite large; hence we limit its size to 20,000 boards. If the number of boards in the pool ever goes higher than this, we remove enough boards to get to the number of boards down to 10,000. Because the 30 second time limit allows only enough time to call Choose-move on a set about 350 boards from the pool, this is believed adequate.

Timed HS: Timed HS works the same as Timed AOSP, with one exception. If $0 \leq p < k_t$, then Timed HS generates a set R of $p - k_t$ random boards that are consistent with o_{ij} (called *last-observation* consistent boards), and calls Choose-move($P \cup R$). This rules out the possibility of ever having to make a random move. It also restricts the amount of time that Timed HS can spend generating additional boards to put into the pool.

4.2 Consistent History Generation

The algorithm for generating additional histories consistent with b does a depth first search through the space of game histories. At each node, the known game history (O_{ij}) specifies the possible branches. Whenever the algorithm reaches a node with multiple branches (corresponding to an opponent's hidden move in the game history), it randomly orders the branches and proceeds searching according to that order.

5 Experiments

Our experimental hypotheses (based on the analyses in Section 2) were that (1) Timed LOS would perform better than random play, (2) Timed AOSP would perform better than Timed LOS, and (3) Timed HS would perform somewhat better than Timed AOSP. The third hypothesis caused some controversy because it was based on a notion that not all of the authors believed: that the computation time spent introducing and evaluating last-observation consistent boards would not be better spent trying to find and evaluate more all-observation consistent boards.

To test our hypotheses, we played all three algorithms against each other and against a player who moved at random. Each player plays approximately half of the games as white and half of the games as black. All experiments were run on Xeon 2.6GHz chips with 500 MB RAM, running Linux. Each player was allowed to spend 30 seconds deciding each move, including moves which are decided after an attempted illegal move.

Figure 2 shows the results for the algorithms versus the random player, Figure 3 shows the results of head-to-head tests of the three algorithms. The results include the wins/losses/draws, and each player's average material value at each move of the game (using the standard chess value for each piece). We observe the following:

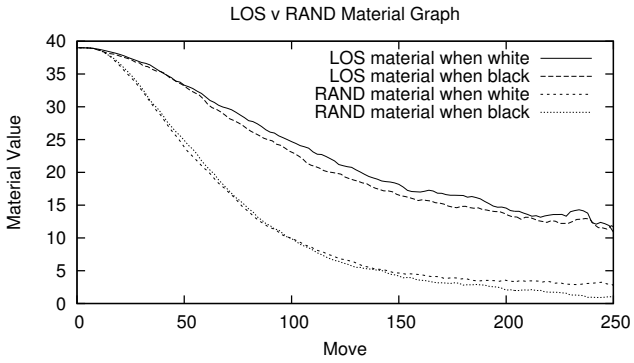
1. All three algorithms played much better than the random player, confirming our first hypothesis. The large number of draws is unsurprising, since kriegspiel chess is a notoriously difficult game to win.
2. Timed AOSP and Timed HS both played much better than Timed LOS, confirming our second hypothesis.
3. Timed AOSP and Timed HS played about equally well versus a random player, but Timed HS did significantly better than Timed AOSP when the two were played head-to-head. Furthermore, as shown in Figure 4, Timed HS did this using a substantial number of last-observation consistent boards as the game progressed.

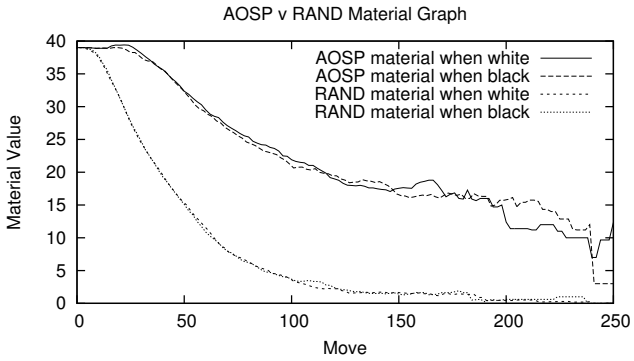
The third observation is very interesting, because it suggests that our hypothesis about last-observation consistent boards is correct: they become more useful as the game progresses, because they are more likely to be consistent with the current belief state. Even though we do not know what probabilities to assign to them in the last line of Choose-move, they still provide useful information.

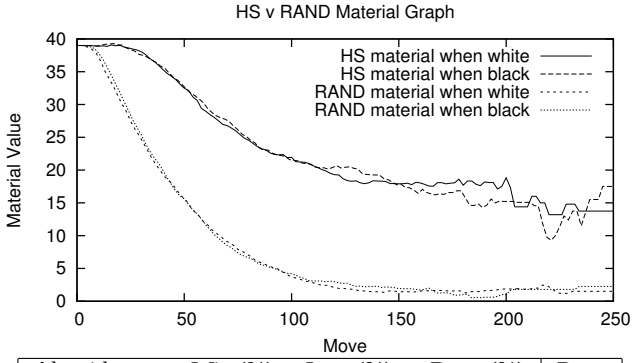
6 Related Work

The best-known work on imperfect-information games includes work on the games of bridge [Smith *et al.*, 1998; Ginsberg, 1999], scrabble [Sheppard, 2002], and poker [Billings *et al.*, 2003]; we discussed this work in Section 1.

The existing literature on kriegspiel chess is rather small, and we believe ours is the first attempt to create a kriegspiel chess program capable of reasonable play throughout the entire game. Two books have been written on how to play kriegspiel chess [Li, 1994; 1995]. [Wetherell *et al.*, 1972] have implemented a program to act as the referee who sees the entire kriegspiel-chess game board and tells each player the information described in the second paragraph of Section 3. [Sakuta *et al.*, 2001] have implemented a search strategy for some imperfect-information games that are simpler than

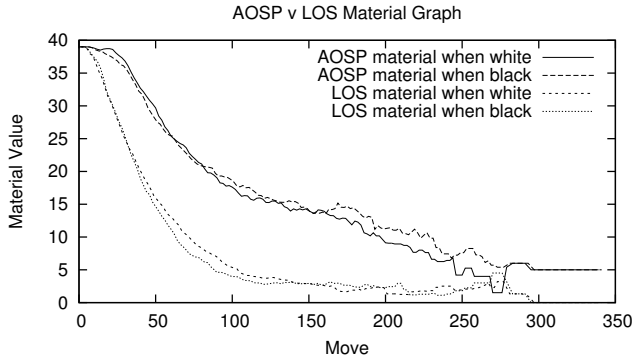


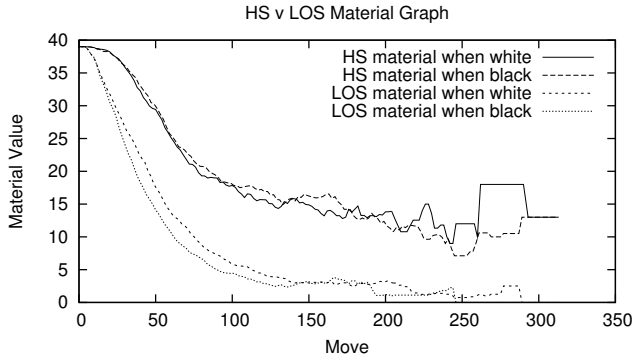


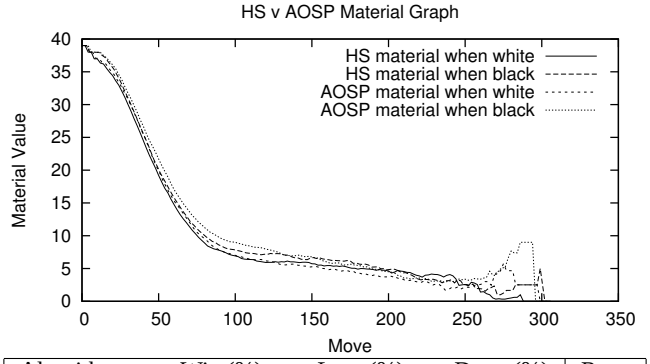


Algorithms	Win (%)	Loss (%)	Draw (%)	Runs
LOS v rand	39 ± 2	0 ± 0.3	61 ± 2	559
AOSP v rand	63 ± 2	0 ± 0.3	37 ± 2	560
HS v rand	65 ± 2	0.5 ± 0.3	35 ± 2	558

Figure 2: Wins/losses/draws percentages plus or minus a 95% confidence interval and average material value at each move, in games between Timed LOS, Timed AOSP, Timed HS, and a random player. Material value is a heuristic measure of a player's ability. We assign material value as follows: queen is worth 9, rook 5, bishop 3, knight 3, and pawn 1. The sum of all a player's pieces material value after a particular move is the player's material value for that move.







Algorithms	Win (%)	Loss (%)	Draw (%)	Runs
AOSP v LOS	31 ± 4.8	0 ± 1	69 ± 4.8	190
HS v LOS	38 ± 5	0.5 ± 1	61 ± 5	190
HS v AOSP	13.3 ± 0.4	10.7 ± 0.4	76 ± 0.5	1669

Figure 3: Wins/losses/draws percentages plus or minus a 95% confidence interval and average material value at each move, in head-to-head comparisons of Timed LOS, Timed AOSP, and Timed HS.

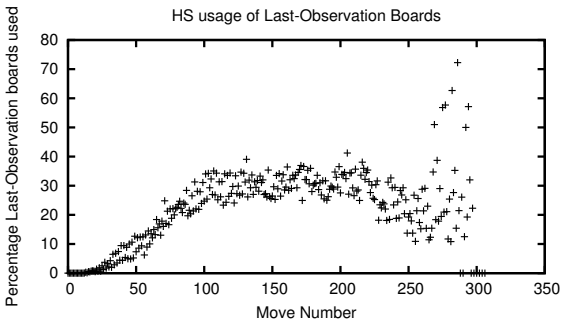


Figure 4: The average percentage of last-observation consistent boards that Timed HS used at each move in its games against Timed AOSP.

kriegspiel, and [Bolognesi and Ciancarini, 2004] and have developed search strategies for kriegspiel-chess endgames.

7 Conclusion

In games such as kriegspiel chess, the belief states are huge compared to those in bridge, scrabble, and poker. Furthermore, there is not a simple uncertainty model from which to generate boards consistent with a belief state. Our results demonstrate that statistical sampling approaches can nevertheless be developed that do well in such games.

Our results also show that to play well, it is not necessary for the random sample to consist only of game boards that satisfy all of a player's observations. In fact, we were able to win 24% more often by starting out with such boards, but gradually switching over (as the game progressed) to boards that merely are consistent with the latest observation.

The reason for our surprising result is that as the game progresses, a board that is consistent with the last move becomes more and more likely to be consistent with the entire set of observations, even if we have no idea what sequence of moves might have actually generated this board.

7.1 Future Work

The biggest limitation of our work is that we have not yet been able to evaluate play against human opponents. To do so will take some effort: we know of no rating system for kriegspiel chess players, hence it is difficult to tell whether a human player is good or bad. To overcome this obstacle, we are currently building a kriegspiel-chess game server which we will run on the web and will make available as open-source software.[2] The server will include a ranking system so that we may compare our algorithms against humans.

We have yet to attempt any opponent modeling. For example, we intend to extend our algorithms to assign a weight to each game board by estimating how likely the opponent was to make all of the moves leading to that board.

A well-known deficiency of statistical-sampling approaches is that they do not take into account the information-gathering value of illegal moves [Frank and Basin, 1998]. We have ideas for how to modify our algorithms to overcome this deficiency, and intend to test these ideas in the near future.

It would be worthwhile to examine different search strategies and evaluation functions. We currently rely on gnuchess for both, but it is not entirely clear whether this is the best approach for a game like kriegspiel chess.

[2]To construct our kriegspiel server, we are using the Generic Game Server that is available as open-source software at http://external.nj.nec.com/homepages/igord/gsa-ggs.htm.

Acknowledgments

This work was supported by the following grants, contracts, and awards: ARO grant DAAD190310202, ARL grants DAAD190320026 and DAAL0197K0135, the ARL CTAs on Telecommunications and Advanced Decision Architectures, NSF grants IIS0329851, 0205489 and IIS0412812, UC Berkeley contract number SA451832441 (subcontract from DARPA's REAL program). The opinions expressed in this paper are those of the authors and do not necessarily reflect the opinions of the funders.

References

[Applegate *et al.*, 1991] David Applegate, Guy Jacobson, and Daniel Sleator. Computer analysis of sprouts. Technical report, Carnegie Mellon University, 1991.

[Billings *et al.*, 2003] Darse Billings, N. Burch, Aaron Davidson, Robert Holte, Jonathan Schaeffer, T. Schauenberg, and Duane Szafron. Approximating game-theoretic optimal strategies for full-scale poker. In *IJCAI-03*, pages 661–668, 2003.

[Bolognesi and Ciancarini, 2004] A. Bolognesi and P. Ciancarini. Searching over metapositions in kriegspiel. In *Computer Games 2004*, 2004.

[Frank and Basin, 1998] Ian Frank and David A. Basin. Search in games with incomplete information: A case study using bridge card play. *Artificial Intelligence*, 100(1-2):87–123, 1998.

[Ginsberg, 1999] Matthew L. Ginsberg. GIB: Steps toward an expert-level bridge-playing program. In *IJCAI-99*, pages 584–589, 1999.

[Li, 1994] David Li. *Kriegspiel: Chess Under Uncertainty*. Premier, 1994.

[Li, 1995] David Li. *Chess Detective: Kriegspiel Strategies, Endgames and Problems*. Premier, 1995.

[Sakuta *et al.*, 2001] M. Sakuta, J. Yoshimura, and H. Iida. A deterministic approach for solving kriegspiel-like problems. In *MSO Computer Olympias Workshop*, 2001.

[Sheppard, 2002] Brian Sheppard. World-championship-caliber scrabble. *Artificial Intelligence*, 134(1-2):241–275, 2002.

[Smith *et al.*, 1998] Stephen J. J. Smith, Dana S. Nau, and Thomas Throop. Computer bridge: A big win for AI planning. *AI Magazine*, 19(2):93–105, 1998.

[Wetherell *et al.*, 1972] C. S. Wetherell, T. J. Buckholtz, and K. S. Booth. A director for kriegspiel, a variant of chess. *Comput. J.*, 15(1):66–70, 1972.