Are Bees Better than Fruitflies?
Experiments with a Hex Playing Program

Jack van Rijswijck

Department of Computing Science
University of Alberta
Edmonton, Alberta
Canada T6G 2H1
javhar@cs.ualberta.ca

Abstract. Traditionally, chess has been called the "fruitfly of Artificial Intelligence". In recent years, however, the focus of game playing research has gradually shifted away from chess towards games that offer new challenges. These challenges include large branching factors and imperfect information. The game of Hex offers some interesting properties that make it an attractive research subject. This paper presents the key ideas behind Queenbee, the first Hex playing program to play at the level of strong human players.
Keywords: Heuristic search, game playing, alpha-beta, Hex.

1 Introduction

Game playing has often been described as an ideal test bed for Artificial Intelligence research. Traditionally, game playing research has focussed mainly on chess. Several decades of research have produced some powerful techniques, mostly geared at the efficient traversal of large game trees. It has also produced some notable triumphs; humans have been surpassed by programs in games such as checkers, Scrabble, and Othello, and other games such as Go-Moku, Connect-4, and Nine Men's Morris have even been solved.

With the advent of the checkers world champion program *Chinook* and the chess program *Deep Blue*, researchers started to realize that the techniques that drive these programs had been all but stretched to their limits. Yet there are other classes of games for which these methods would be of little use in constructing a program that can play on par with the strongest humans. One such class includes games in which not all of the information is available to each player, such as in card games like bridge and poker. Another class is the one containing games whose *branching factor*, defined as the typical number of available options for a player when it is time to make a move, is too large to make brute force tree search algorithms feasible. The most well-known of these games is the Oriental board game Go, for which no strong programs exist despite considerable effort and expertise that has been devoted to it.

Another member of the class of high branching factor games is Hex. It is similar to Go, but may be a better choice for study due to the simplicity of the goal and the rules. The game has several interesting properties. It can be played on a board of any size, thus becoming arbitrarily complex in terms of the branching factor. The rules are simple, yet they give rise to elaborate strategic ideas. Hex has a rich underlying mathematical structure that ties in to advanced concepts in graph theory and topology. It has the interesting properties that games cannot end in a draw[1] and that it can be proved that the game is a first player win with perfect play. However, the winning strategy is not known.

Queenbee is a Hex playing program that is based on a novel idea for an evaluation function. It is the first Hex program to surpass "novice" level in human terms. Indeed, it now plays at the level of very strong human players, if not quite yet at the level of the top players. Queenbee has also carried out the first complete analysis of all opening lines on a 6×6 board. The program has its own web page, `http://www.cs.ualberta.ca/~queenbee`, which includes the 6×6 opening analysis.

This paper is organized as follows. Section 2 introduces the game of Hex and its properties. The next two sections describe Queenbee's evaluation function and its search algorithms, respectively. An assessment of Queenbee's playing strength is given in Section 5. Section 6 mentions current work in progress and future work, followed by a summary and conclusions.

2 Hex

Hex was invented by Danish engineer, poet, and mathematician Piet Hein (1905–1996) in 1942, and independently rediscovered by John Nash (1928–) in 1948. It is a board game with simple rules, but a complex strategy. Indeed, winning strategies are only known for board sizes up to 7×7, whereas the game is commonly played on sizes of 10×10 or larger. The game is a special case of a more general graph colouring game known as the *Shannon switching game*, which was proved to be PSPACE-complete [ET76]. This section describes the rules of Hex, as well as some special properties of the game.

2.1 Rules

Hex is played on a rhombic hexagonal pattern, as in Figure 1. This particular Hex board has 5×5 cells, but the game can be played on boards of any size. The board has two *white borders* and two *black borders*, indicated in the figure by rows of discs placed next to the borders. It is often helpful to imagine the presence of these "ghost edge pieces". Note that the four corner cells each belong to two borders.

Play proceeds as follows. The two players, henceforth to be called *White* and *Black*, take turns placing a piece of their colour on an empty cell. There is no standard convention on which colour gets the first move. White wins the game by connecting the two white borders with a chain of white pieces, while Black

[1] This is directly equivalent to a fundamental theorem of topology; see [Gal86].

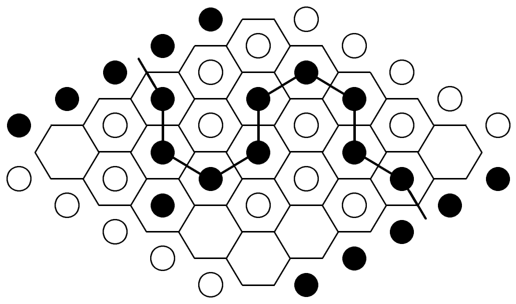

Fig. 1. A 5×4 Hex board with a winning chain for Black

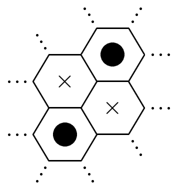

Fig. 2. A two-bridge

wins by establishing a chain of black pieces connecting the two black borders. In Figure 1, Black has completed a winning chain.

A detailed discussion of Hex strategy is beyond the scope of this paper. However, it is necessary at this point to introduce its most basic element: the "two-bridge". Figure 2 depicts two black pieces; the two cells marked '×' are still empty. Black can connect the two pieces even if White plays first; whenever White plays in one of the cells marked '×', Black occupies the other. This ensures what is known as a "virtual connection" between the two black pieces. It is the simplest example of a virtual connection guaranteed by two disjoint threats.

2.2 Properties

One of the properties of Hex is that the game can never end in a draw. The proof of this intuitively obvious fact is based on the observation that a Hex board that is completely filled with pieces must necessarily contain a winning chain for one of the two players.[2] Another interesting property, first noted by John Nash, is that Hex can be proved to be a theoretical win for the first player. This proof is based on the "strategy stealing argument": if there were a winning strategy **S** for the second player, then the first player could play an arbitrary opening move and subsequently apply **S** to win. The arbitrary opening move cannot spoil this

[2] See [Gal86].

strategy because an extra piece can never be a disadvantage in Hex.[3] Since we cannot have both players winning the game, there cannot be such a strategy, and since draws are impossible, it follows that there is a winning strategy for the first player.

3 Evaluation Function

A game playing program needs a good evaluation function to help guide the search. It is not immediately obvious how to construct a meaningful evaluation function for Hex. For example, unlike in many other board games, the concepts of material balance and mobility are not useful in Hex. This section contains new ideas for a Hex evaluation function, which are implemented in Queenbee. The function calculates the distance to each edge of all the unoccupied cells on the board, according to an unconventional metric called "two-distance". The resulting distances are referred to as "potentials".

3.1 Two-Distance

Given a graph Γ with an adjacency function $n(p)$ that maps a vertex p onto the set of the vertices that are adjacent to it, the conventional distance metric is actually a special case of a more general distance metric d:

$$d(p,q) = \begin{cases} 0 & \text{if } q = p, \\ 1 & \text{if } q \in n(p), \\ \min_k c_k(p) \geq z & \text{otherwise,} \end{cases}$$

where

$$c_k(p) = |\{r \in n(p) | d(r,q) < k\}|.$$

The conventional distance metric corresponds to $z = 1$, in which case the distance of a cell to an edge on the Hex board represents the number of "free moves" that it would take for a player to connect the cell to the given edge. Unfortunately this distance function is not very useful for building an evaluation function for Hex, as will be shown later. Rather, the concept of *two-distance* is used, where $z = 2$. The two-distance is one more than the *second lowest* distance of p's neighbours to q, with the proviso that the two-distance equals 1 if p and q are directly adjacent. The intuition behind the two-distance idea is that, when playing a adversary game, one can always choose to force the opponent to take the second best alternative by blocking the best one. The two-distance captures this concept of "the best second-best alternative".

There is an important distinction between adjacency and neighbourhood. Adjacency implies neighbourhood, but not vice versa. Two cells are adjacent if they share a common edge on the board. The notion of neighbourhood takes into account any black and white pieces that are already on the board. Two

[3] For this reason, the argument will not work for games such as chess.

unoccupied cells[4] are neighbours from White's point of view if either they are adjacent or there is a string of white stones connecting them. Note that a cell's neighbourhood can therefore be different from White's point of view than it is from Black's point of view. These two neighbourhoods will be referred to as the *W-neighbourhood* and the *B-neighbourhood.* Correspondingly, there will be a distinction between W-distance and B-distance.

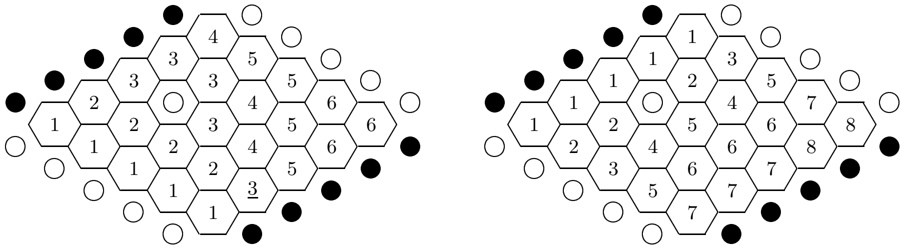
I. W-distance to lower white edge II. B-distance to upper black edge

Fig. 3. Two-distances on a non-empty board

The two-distances can be computed using the standard Dijkstra algorithm for calculating distances in graphs, modified appropriately for $z = 2$. An example of two-distances is shown in Figure 3. Consider the cell containing the underlined 3. Its two-distance to the lower white edge is not 2, because it only has one neighbour that is at two-distance less than 2. Note also that the ghost edge pieces must be taken into account when calculating the distances. This explains why the rightmost cell in Figure 3-I is at two-distance 6. Due to the ghost edge pieces, all the cells along the upper white edge are its W-neighbours, and at least two of those are at two-distance less than 6.

3.2 Potentials

The goal in Hex is to connect two sides of the board. To help achieve this, one might look for an unoccupied cell that is as close as possible to being connected to both sides, as this would be a promising candidate for being part of a winning chain. The evaluation function calculates *potentials* that capture this concept. Each unoccupied cell is assigned two potentials, based on the two-distance metric. A cell's W-potential is defined as the sum of its W-distance to both white edges; its B-potential is the sum of its B-distance to both black edges.

Cells with low W-potentials are the ones that are closest to being connected to both white borders by White. If White can connect a cell to both white borders, this would establish a winning chain. White will therefore focus on those cells that have the lowest W-potentials. The white *board potential* is defined as

[4] Neighbourhood is only ever used for empty cells.

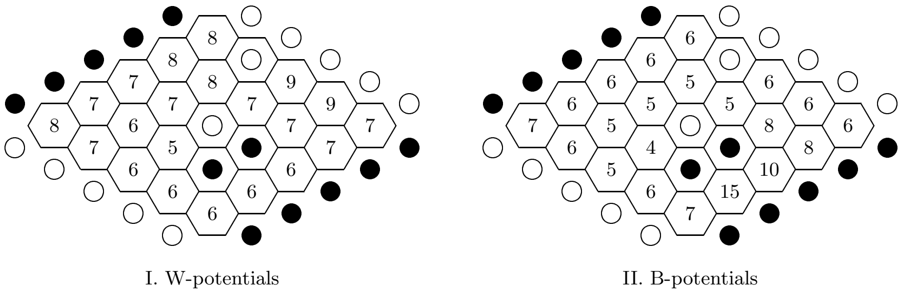

Fig. 4. Cell potentials

the lowest W-potential that occurs on the board. In the example of Figure 4 the white board potential is 4, and the black board potential is 5. As lower potentials are better, it appears that White is ahead.

In the same figure, it can be seen that both Black and White have only one cell that actually realizes their board potential. It would be better to have more than one cell that realizes the board potential, so as to have more attack options and be less vulnerable to the opponent blocking the chain. This is used in Queenbee's evaluation function to break ties between positions with equal board potentials.

3.3 Strategic Relevance

The idea behind the two-distance metric is directly related to the importance of double threats. Indeed the two-distance implicitly takes into account the two-bridges that occur in a Hex position. Consider the position in Figure 5. The Black distance to each edge cannot percolate through the White two-bridges. As White already has a winning connection made up of two-bridges, the result is that the Black board potential is infinite. Thus, the two-distance metric also implicitly recognizes a winning chain that consists of virtual connections through two-bridges, even if the chain is not actually solidified yet.

By contrast, using the conventional $z = 1$ distance metric in the position of Figure 5 would yield board potentials of 4 for both White and Black, suggesting that both players are equally close to establishing a winning connection. It is clear that the two-distance metric is far more suited to Hex than the conventional distance metric is.

3.4 Cell Potentials

As mentioned before, White will want to play in cells that have a low W-potential, as those are the cells that are closest to being connected to both white edges. Simultaneously, White will also want to focus on cells that have low B-potentials. Those are the cells where Black is closest to establishing a winning

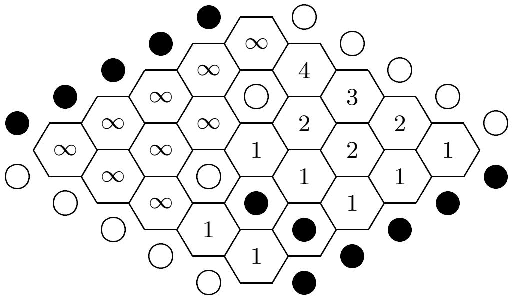

Fig. 5. The two-distance cannot percolate through two-bridges

connection, and therefore White will want to play in those cells to block Black's connection. Combining this, White will prefer to play in cells that have a low *total potential*, where the total potential of a cell is the sum of its W-potential and its B-potential. By symmetry, Black will prefer to play in the same cells. This is analogous to the Go proverb "Your opponent's most important play is your most important play." The total potentials for the position of Figure 4 are shown in Figure 6.

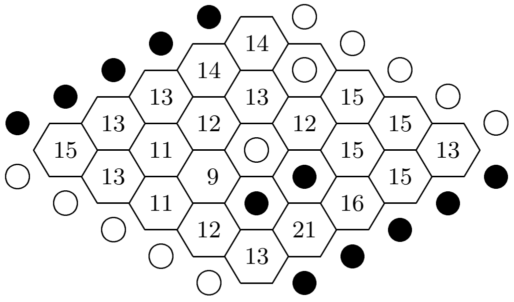

Fig. 6. Total potentials

4 Search

Queenbee uses an iterative deepening α-β search enhanced with Minimal Window / Principal Variation Search and transposition tables. These techniques are used in most state-of-the-art game playing programs [Mar86]. The move ordering is based on cell potentials. Queenbee's search incorporates the fractional ply searching ideas of the "Sex Search algorithm" [LBT89].

4.1 Sex Search

The large branching factor of Hex makes regular full-width searching methods inadequate, even when enhanced with conventional search extensions and reductions. On the other hand, a highly selective search is too unreliable due to its inability to cope with some tactical moves. The Sex Search algorithm, as described by Levy, Broughton, and Taylor [LBT89] is essentially a generalization of search extensions and reductions, whose behaviour can range smoothly over the spectrum between full-width and fully selective. Moreover, it is amenable to automated learning. The name "Sex Search" stands for "search extensions".

Sex Search proposes to assign a weight, or *cost*, to every move in the search tree. Rather than exploring lines until a certain fixed depth is reached, the Sex algorithm explores lines until their moves add up to a fixed cost. This cost limit may be called the *budget*. The idea is that "interesting" moves have low cost, while "uninteresting" moves have high cost. This way, branches with many uninteresting moves are not explored very deeply, which corresponds to search reductions. At the same time, branches that contain many interesting moves will be explored more deeply, corresponding to search extensions.

If all moves are assigned a cost of 1, then a Sex Search with a budget of n is equivalent to a full-width fixed-depth search to n ply. If the moves have varying cost, but the average cost is 1, then the Sex Search is comparable to an n ply search with extensions and reductions. A move cost of k effectively extends the search by $1 - k$ ply if $k < 1$, and reduces the search by $k - 1$ ply if $k > 1$.

If the range of costs of the available moves is large, then the Sex Search algorithm behaves much like a selective search. Consider, for example, a move m with cost 4. This cost ensures that the subtree below m will be explored to a depth of 3 less than the subtrees below m's siblings. Due to the exponential nature of the search tree, the search effort required to explore m becomes insignificant in comparison with the effort required to explore m's siblings. Thus the behaviour is similar like that of a selective search that would discard move m altogether. A selective search suffers from the unavoidable risk of discarding moves that turn out to be critical. Sex Search does not run this risk, as it does not actually discard any moves.

Sex Search is used in some high performance game playing programs, most notably in the well-known chess program Deep Blue [CHH99]. However, in most classes the fractional move costs are only assigned to certain special cases of moves, while the majority of the moves receives weight 1. Queenbee's search is fully fractional, in that each move category has a fractional weight.

4.2 Move Categories

The crux of the Sex Search algorithm is finding a good cost function for moves. Queenbee uses the cell potential as a basis for the cost function. Moves are partitioned into equivalence classes, or *move categories*. Each move category has a weight associated with it. The cost of a particular move is obtained by retrieving the weight of its move category. A move m in a cell with potential p is a member of move category $c(m) = p - p^*$, where p^* is the lowest cell potential of all the *other* cells. If m has the lowest potential then its move category will be 0 if there are other moves with the same potential. If m has the unique lowest potential, the move category will be negative and the static evaluation of m will be $-c(m)$ better than that of the next best looking move.

5 Performance

The best way to obtain information about a game playing program's playing strength is by playing games against opponents of known strength. In Hex this can be done on the online game playing server Playsite.[5] Queenbee played a small number of games on Playsite, in order to get a first impression of its playing strength. To obtain reliable information, however, many games are needed. Unfortunately, it is not yet possible to play online games automatically, as Playsite is accessible only through a Java interface with an unknown communications protocol.

The initial results show that Queenbee achieved a rating of 1876 in six games. Typical ratings for a human players are in the range 1200–1400 for novices, 1800 for advanced players, and 2100–2200 for the top players. Strong human players estimated Queenbee's rating to be about 2100. The program scored two consecutive wins against a player rater 2119. However, these results may not mean much. The number of games played is small, and there likely is some overestimation on the part of the opponents due to their unfamiliarity with Queenbee's style.

5.1 Evaluation Function

Since Queenbee has completely solved many opening positions in 6×6 Hex, it is possible to compare the evaluation function's assessment of these positions with perfect knowledge. In a winning position we distinguish between *good moves* and *bad moves.* A bad move loses against perfect play, while a good move preserves the win. There are two types of good moves: *optimal* moves and *suboptimal* moves. A move is optimal if it maintains the shortest possible win. In a losing position there are no good or bad moves, as every move will lead to a loss against a perfect opponent. Yet there still is a distinction between optimal and suboptimal moves. Optimal moves are those that delay the loss as long as possible, while suboptimal moves do not.

Table 1 lists the number of times each move category occurred, in the "count" rows, as well as the frequency in percentages of each move type. The numbers were obtained from to 27 winning positions and 34 losing positions that Queenbee has analyzed. The table indicates that the lower move categories contain a significantly higher percentage of good moves than average. Note that negative move categories, intuitively corresponding to apparently "forced" moves, appear

[5] `http://www.playsite.com/games/board/hex`.

to be better in losing positions than they are in winning positions. This may be because the winning side typically has several options to choose from, while the losing side is more often forced to reply to a threat.

The effectiveness of the Sex Search relies on the move categories' capacity to distinguish good moves from bad moves. Any category that has a significantly different distribution of good and bad moves compared to the overall distribution is therefore valuable. Categories can be assigned low or high weights according to whether good moves are relatively common or uncommon, respectively. The bottom row of Table 1 indicates that almost all categories do show a large deviation from the average frequency. Moreover, the frequency of optimal moves decreases almost monotonically as the move category increases, indicating that the cell potential is indeed a good estimator of move strength.

position	move	move category												
type	type	≤ -3	-2	-1	0	1	2	3	4	5	6	7	≥ 8	all
winning	optimal	-	25	13	25	11	2	7	2	4	-	-	-	5
	suboptimal	-	-	-	22	6	20	6	6	8	3	-	-	7
	good	-	25	13	47	17	22	14	8	12	3	-	-	11
	bad	-	75	88	53	83	78	86	92	88	97	100	100	89
	count	-	4	8	60	71	134	140	171	131	113	39	19	890
losing	optimal	100	100	100	35	6	1	1	0	-	-	-	-	4
	suboptimal	-	-	-	65	94	99	99	100	100	100	100	100	96
	count	3	3	5	63	62	139	203	213	178	148	56	76	1149
all	optimal	100	57	46	30	9	2	4	1	1	-	-	-	4

Table 1. Move types in each move category

In order for the search to produce reliable results, it is not necessary that all good moves in a position be found. What is important is that at least one optimal move is found. Table 2 lists the lowest move categories in which optimal and suboptimal moves were encountered in the same set of positions. It is clear that in most cases there is an optimal or suboptimal move to be found in categories at most 0 or 1. Positions in which good moves only occur in higher move categories are rare.

position	move	move category							
type	type	-5	-2	-1	0	1	2	3	5
winning	optimal	-	1	1	9	6	2	7	1
	good	-	1	1	12	7	2	3	1
losing	optimal	3	3	5	21	1	-	-	1

Table 2. Category containing the lowest potential move

From these tables it becomes apparent that the evaluation function enables a good partitioning of moves into classes of different move quality. It is also clear that the cell potentials provide a good estimate of the quality of the moves, since good moves are more common in low move categories.

5.2 Search

Queenbee was tested on 25 analyzed winning positions, to find out how many of these positions can be played correctly under actual tournament play conditions. It is not sufficient to merely find a winning move. The program is only considered to have "solved" a position once it settles on a winning move and never changes its mind to a losing move anymore. On a PII/400 machine, Queenbee searches about 12,000 nodes per second. This means that the program is able to search about two million nodes within the time constraint. A position was played correctly if Queenbee settled on a winning move within two million nodes of search.

With all category weights set to 1, corresponding to a full-width fixed-depth search, Queenbee was able to solve 12 out of the 25 positions. With hand tuned weights, the performance increased to 14 out of 25.[6] The median solution length for these positions was 22 ply.

The depth of a win is not necessarily what makes a position hard to solve. If the winning move is obviously better than the other moves, particularly if all other moves lose quickly, the solution is not difficult to see. However, in the positions used in the test, the longest losing lines had a median length of 17 and a maximum length of 23. The correct decision between such deep wins and deep losses is extremely difficult to find. It should also be noted that these are all winning positions at the beginning of the game, and as such they are the most difficult of all 6×6 Hex problems.

6 Work in Progress

The weights for the move categories are hand picked, based on intuitions about the move categories and their relation to the game. These weights form an attractive target for machine learning techniques. Experiments are currently running to apply the "Learning Search Control" algorithm of Yngvi Björnsson [Bjö00] to dynamically establish these weights based on actual game play. Early results seem to indicate that counterintuitive looking weights outperform more plausible looking weights, but the evidence is still very inconclusive.

The search algorithm will also be enhanced with null move searches [Don93]. The null move technique can drastically reduce the size of search trees, and has proven to be very successful in chess. It suffers from the great danger of misjudging "zugzwang" positions, which are positions where the player to move

[6] The weights were set to `[1, 1, 1.5, 3, 5]` for categories 0 through 4, `0.5` for negative categories, and `6` for categories 5 and higher.

would prefer to skip a move. Fortunately, zugzwang does not occur in Hex. *Any* move is always better than no move at all. Null move therefore promises to be a powerful enhancement of the search in Queenbee.

Efforts are underway to construct a Java interface that enables Queenbee to play games online on its own web page. This will not only provide valuable feedback on the program's playing strength, but it will also facilitate the learning of move category weights.

7 Conclusions

The comparison with perfect information on a 6×6 board is a favourable indication of the quality of the evaluation function. It is of course not to be taken for granted that these results will persist on larger board sizes. On the other hand, it should be noted that perfect play on a 6×6 board is far from trivial. One piece of anecdotal evidence is that Queenbee has actually disproved some of the common beliefs of strong Hex players about certain opening moves.

Equipped with the perfect opening book and a powerful search to fill in the blanks where the opening book ends, Queenbee plays stronger than any human player on board sizes up to 6×6. On larger boards, human players' supremacy increases as the board size increases. There is some room for improvement by refining the search methods, such as by incorporating null moves, but it is not clear whether this will be enough to beat the top human players on standard board sizes like 10×10.

What, then, is needed to beat the top players? Human players have the important ability to recognize patterns in Hex. The two-bridge is the simplest example, but there are many common patterns known to human players where a virtual connection of a piece to the edge is guaranteed. It can take many plies of search to uncover these connections implicitly. It seems obvious to humans that knowledge about these patterns is therefore a key ingredient to any strong Hex player. Queenbee is equipped to recognize many common patterns, but it is not yet clear how to integrate this knowledge into the evaluation function. Worse still, the patterns cause significant horizon-type problems[7] that actually lead to a net decrease in playing strength. For this reason, Queenbee currently does not use the patterns at all. Finding out how to correctly use the patterns in the evaluation function and how to avoid destructive interference with the search is therefore an important challenge.

If these problems can be solved adequately, Hex programs may be able to reach the first milestone, beating all human players on a 10×10 board, in the near future. Top human players prefer to play on 14×14 boards, and in some cases even 18×18 boards, where the game is more challenging. As it is progressively more difficult for computers to beat humans on larger board sizes, Hex will

[7] The horizon effect occurs when a program plays suboptimal moves in order to delay an impending threat; if the threat is pushed beyond the search horizon, the program will assume it has "solved" the problem, whereas in reality it has often made it worse.

remain a challenge for game playing programs even after this first milestone is reached.

8 Acknowledgments

I would like to thank Darse Billings, Yngvi Björnsson, and Jonathan Schaeffer for valuable feedback on early drafts of this paper.

References

[Bjö00] Y. Björnsson. Learning Search Control. *Advances in Computer Chess 9*, to appear, 2000.

[CHH99] M. S. Campbell, A. J. Hoane, and F. Hsu. Search Control Methods in Deep Blue. *AAAI Spring Symposium on Search Techniques for Problem Solving under Uncertainty and Incomplete Information*, pages 19–23. AAAI Press, 1999.

[Don93] C. Donninger. Null Move and Deep Search: Selective Search Heuristics for Obtuse Chess Programs. *Journal of the International Computer Chess Association*, 16(3):137–143, 1993.

[ET76] S. Even and R. E. Tarjan. A combinatorial problem which is complete in polynomial space. *Journal of the Association for Computing Machinery*, 23:710–719, 1976.

[Gal86] D. Gale. The Game of Hex and the Brouwer Fixed Point Theorem. *American Mathematical Monthly*, pages 818–827, 1986.

[LBT89] D. Levy, D. Broughton, and M. Taylor. The Sex Algorithm in Computer Chess. *Journal of the International Computer Chess Association*, 12(1):10–21, 1989.

[Mar86] T. Marsland. A Review of Game-Tree Pruning. *Journal of the International Computer Chess Association*, 9(1):3–19, 1986.