SOLVING 7×7 HEX: VIRTUAL CONNECTIONS AND GAME-STATE REDUCTION

R. Hayward, Y. Björnsson, M. Johanson, M. Kan, N. Po, J. van Rijswijck
Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada
{hayward,yngvi,johanson,mkan,nathan,javhar}@cs.ualberta.ca, http://www.cs.ualberta.ca/

Abstract We present an algorithm which determines the outcome of an arbitrary Hex game-state by finding a winning virtual connection for the winning player. Our algorithm performs a recursive descent search of the game-tree, combining fixed and dynamic game-state virtual connection composition rules with some new Hex game-state reduction results based on move domination. The algorithm is powerful enough to solve arbitrary 7×7 game-states; in particular, we use it to determine the outcome of a 7×7 Hex game after each of the 49 possible opening moves, in each case finding an explicit proof-tree for the winning player.

Keywords: Hex, virtual connection, pattern set, move ordering, move domination, game-state reduction

1. Introduction

Hex is the classic two-player board game invented by Piet Hein in 1942 and independently by John Nash around 1948 (Gardner, 1959; Nasar, 1998). The board consists of a rhombus-shaped $n \times n$ array of hexagons, also called cells. Each player is assigned a set of stones and two opposing board sides, all with the same colour; say Black gets black stones and sides, while White gets white stones and sides. Players alternately place a stone on an unoccupied cell. The first player to form a path connecting his/her two sides with his/her stones wins the game. See Figure 1. For more on Hex, see Browne (2000) and Hayward and Van Rijswijck (200x).

In Hex, an unrestricted opening allows the first player to gain a considerable advantage: it is known that there exists a winning strategy for the first player (Gardner, 1959), and while no explicit strategy which holds for arbitrary sized boards is known, most players believe that opening in the centermost cell in particular is a winning move. In order to offset this opening move advantage, the game is often started according to the following "swap rule": colours are assigned to the four sides of the board, but not to the players; one player then places a stone on any cell; the other player then chooses which colour stones

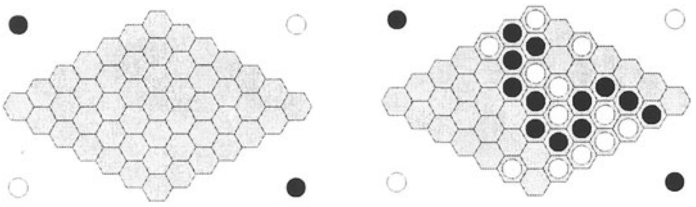


Figure 1. An empty 7×7 board, and . . . a finished game; Black wins.

to play with. The second move is played by the player whose stones are the opposite colour of the first stone. From then on, the game continues in normal fashion, namely with players alternating moves.

With respect to Hex, a *board-state* describes a particular placement of some number of black stones and some number of white stones, such that each cell has at most one stone. We assume no constraint on the relative number of stones of each colour, as the game may have started with a handicap advantage for one of the players. The *empty board-state* has no stones on the board. A *k-opening* is a board-state with exactly k stones on the board. A *turn-state* describes which player has the next move. A *game-state*, or simply a *state*, consists of a board-state and a turn-state. We denote by $G = [P, B]$ the game-state with turn-state P and board-state B; for this game-state, we say that P *wins* G if P has a winning strategy for G. For a board-state B, we say that P *wins* B if P wins $G = [P, B]$.

A state is *solved* if the winning player is known, and *explicitly solved* if a winning strategy is known. As we have already remarked, for arbitrarily large boards, Hex has been solved for the empty board-state, but not explicitly solved.

In this paper we consider the problem of solving arbitrary Hex states, and present an algorithm which solves this problem. The worst-case running time of our algorithm is exponential in the number of cells in the board, which is not surprising given that solving arbitrary Hex states is PSPACE-complete (Reisch, 1981). As a benchmark for the efficiency of our algorithm, we solve all 7×7 1-openings. Previously known 1-opening results are summarized in Figure 2.

Our results yield the first computer solution of any Hex state on a 7×7 or larger board. Solving Hex states on 5×5 or smaller boards is a computationally routine task. To solve arbitrary 6×6 Hex states, Van Rijswijck (2000, 1999-2003) used an alpha-beta search guided by a Hex-specific evaluation function; his algorithm solved all 1-openings and many longer openings. As this method was not strong enough to solve 7×7 states, he further described but did not implement an alternative recursive-descent algorithm (Van Rijswijck, 2002). Recently Yang et al. solved by hand several 7×7 1-openings (Yang et al., 2001, 2002a), one 8×8 1-opening (Yang et al., 2002b), and one 9×9 1-opening (Yang, 2003).

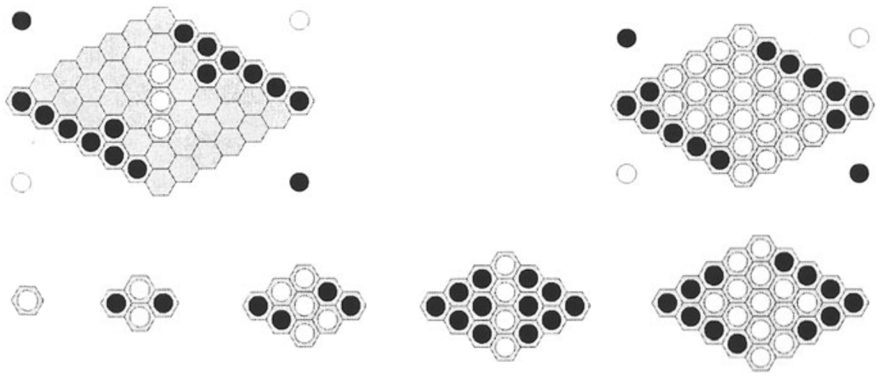


Figure 2. Previously known 1-opening results. The stone on each cell indicates the winner with perfect play if White's first move is to that cell. For cells with no stone, the winner was not previously known. The 6×6 results were obtained by Van Rijswijck by computer (Van Rijswijck, 2002). The 7×7 results were obtained by Yang et al. (2001, 2002) by hand.

Our algorithm solves an arbitrary Hex state by computing a winning virtual connection according to dynamic-state composition rules. Following the recursive descent game-tree search proposed by Van Rijswijck, our algorithm is enhanced by the computation of fixed-state virtual connections; additionally, some new Hex move domination and state reduction results allow significant pruning of the game-tree.

Before presenting our algorithm in Section 4 and our 7×7 results in Section 5, we provide necessary background information on virtual connections in Section 2 and state reductions in Section 3.

2. Connection Sets

Roughly, a *connection set* in Hex is a subgame in which one of the players can form a connection between two specified sets of cells. If the player can connect the two sets even if the opponent moves first, the connection set is called a *virtual connection* or *link*; if the player must have the first move in order to guarantee the connection, the connection set is called a *weak connection* or *prelink.*

More formally, with respect to a fixed Hex state, a player P, sets of cells[1] X, Y, and a set of cells S, $(P{:}X, S, Y)$ is a *virtual connection* or *link* if there exists a strategy whereby, in the game restricted to the set of cells $X \cup S \cup Y$, P can form a chain connecting at least one cell of X with at least one cell of Y, even if P's opponent moves first; in other words, $(P{:}X, S, Y)$ is a virtual connection if there exists a second-player-win strategy for P to connect X and

[1]Here each of the four sides is also be considered as a cell.

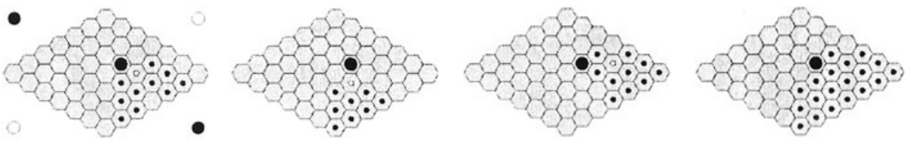


Figure 3. A virtual connection formed by weak connections. Each of the three leftmost figures shows a Black weak connection, indicated by the dotted cells, from the black stone to the bottom right side; the white dot indicates a cell whose occupation would transform the weak connection into a virtual connection. The common intersection of these weak connections is empty, so their union forms a Black virtual connection from the black stone to the bottom right side, shown in the rightmost figure.

Y in the game restricted to $X \cup S \cup Y$. Analogously, $(P{:}X, S, Y)$ is a *weak connection* or *prelink* if there exists a strategy whereby, in the game restricted to the set of cells $X \cup S \cup Y$, P can form a chain connecting X and Y if P moves first; in other words, $(P{:}X, S, Y)$ is a weak connection if there exists a first-player-win strategy for P to connect X and Y in the game restricted to $X \cup S \cup Y$. A P-link (respectively P-prelink) is a link (prelink) for player P. See Figure 3.

In this paper, all virtual and weak connections have the form $(P{:}X, S, Y)$ where X and Y each consist of a single cell; we denote such connections $(P{:}x, S, y)$ where now x and y represent single cells instead of sets of cells.

Although defined slightly differently by different authors, virtual connections have long been recognized as being central to Hex strategy. References to virtual connections permeate the Hex literature, where they are also referred to as "connections" or "safe groups". For example, virtual connections are discussed by Berge (1977)[2] and Browne (2000).

Virtual connections are useful in solving states since, when accompanied by an explicit strategy, a virtual connection serves as a proof or certificate that a pair of cells can be connected.

In particular, if P has a virtual connection $(P{:}x, S, y)$ where x and y are the two sides belonging to P, then this virtual connection certifies that P wins the game. For this reason, we call $(P{:}x, S, y)$, a *win-link* (respectively *win-prelink*) if it is a link (prelink) and x and y are P's two sides. Since the sides of each player are fixed, we will sometimes abbreviate $(P{:}x, S, y)$ by $P{:}S$ whenever x, y are the sides of P.

Connection sets are particularly effective in Hex end-game analysis. For example, the following is a restatement in our terminology of an observation made by Berge.

[2]A translated version of appears in Hayward (2003a).

THEOREM 1 *(Berge, 1977; Hayward, 2003a) Consider a state in which a player P has the next turn and P's opponent Q has one or more win-prelinks. Then Q wins unless P's next move is to a cell which intersects all Q-win-prelinks, for otherwise Q can on the next move convert a win-prelink to a win-link.*

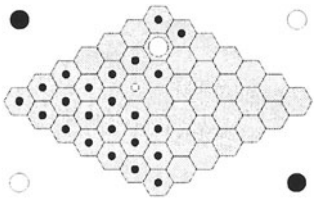
In light of this result, for any fixed state and a player P with opponent Q, we refer to the set of unoccupied cells in the intersection of all Q-win-prelinks as P's *mustplay region.* Notice that the computation of a mustplay region is a form of null move analysis, as it involves the consideration of what can occur if a player skips a turn.

A useful feature of virtual connections is that smaller ones can be combined in various ways to form larger ones. The knowledge of this fact is as old as Hex itself; for example, it is discussed in detail by Berge (1977) and Hayward (2003a). Recently, Anshelevich (2002) used the following set of combining rules to compute connection sets in an inductive or "bottom-up" fashion. A P-stone is a stone belonging to P; ϕ denotes the empty set.

THEOREM 2 *(Anshelevic, 2002) $(P{:}x, \phi, y)$ is a virtual connection if x and y are adjacent. Also, if $(P{:}x, S, y)$ and $(P{:}y, T, z)$ are virtual connections and $\{x\} \cup S$ and $T \cup \{z\}$ do not intersect, then $(P{:}x, S \cup \{y\} \cup T, z)$ is a virtual connection if y is occupied by a P-stone and a weak connection if y is unoccupied. Also, if $(P{:}x, S_1, y)$, $(P{:}x, S_2, y)$, $\ldots$, $(P{:}x, S_k, y)$ are weak connections and the common intersection of the sets S_j is empty, then $(P{:}x, S, y)$ is a virtual connection, where S is the union of the sets S_j.*

Notice that this set of rules is static, in that it yields a class of connection sets for a fixed state. This set of rules is not sufficient to establish all virtual connections of a state, and is thus not strong enough to solve all Hex states. However, the rules do yield a sufficiently large class of virtual connections to provide an effective subroutine of a strong Hex-playing program (Anshelevich, 2002).

As Van Rijswijck observed, an alternative method of computing connection sets is to proceed through the game-tree dynamically. Let $G = [P, B]$ be a state and let Q be the opponent of P. For each unoccupied cell x of B, let $B + x$ be the board-state obtained by adding to B a P-stone at x, and let $G + x$ be the



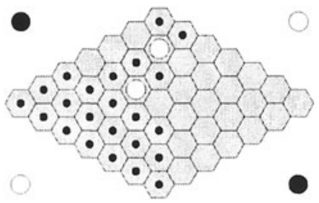


Figure 4. A white win-prelink ... and a corresponding win-link.

associated state, namely $G+x = [Q, B+x]$. Call a collection of sets *mutually exclusive* if the intersection of all the sets is empty. Van Rijswijck's comments suggest the following rules for solving a state.

THEOREM 3 *(Van Rijswijck, 2002) If P (respectively Q) has a winning chain in B then $P{:}\phi$ (respectively $Q{:}\phi$) is a win-link. If neither player has a winning chain in B, then*

- *P wins G if and only if P wins $G + x$ for some move x; in this case, $G + x$ has some win-link $P{:}S$ and G has a win-prelink $P{:}S + x$,*
- *Q wins G if and only if Q wins $G + x$ for all moves x; in this case, for each x, $G + x$ has a win-prelink $Q{:}S_x$ and any collection C consisting of one such S_x for each x is mutually exclusive; also, for each $C' \subset C$, if C' is mutually exclusive then the union U of the elements of C' is a win-link $Q{:}U$ for G.*

Figure 5 illustrates this theorem. The root state G is a loss for White. Three of White's possible moves are explored. In each state $G + x_i$, the move y_i yields a black win; the resulting state $G + x_i + y_i$ has a black win-link S_i, so $G + x_i$ has a black win-prelink $S_i \cup y_i$; this win-prelink implies that x_i loses in G, and moreover that *any* white move outside of $S_i \cup y_i$ loses. The set of these three win-prelinks is mutually exclusive. Indeed, the set containing just the win-prelinks $S_2 \cup y_2$ and $S_3 \cup y_3$ is already mutually exclusive, which means that the union of these two prelinks is a black win-link in G. It also means that the exploration of these two branches of the game-tree is sufficient to determine that White loses G; the consideration of any other move is unnecessary.

We omit the proof of correctness of the preceding theorem, which follows by elementary game-theory arguments from the fact that any Hex state has exactly one winner.[3] Notice that these rules are by their definition complete: they can be used to solve any arbitrary Hex state.

From a computational point of view, the difficulty with both of these sets of rules is that the number of possible connection sets that can be computed in this way is exponential in the number of cells. For this reason, an exhaustive approach to computing connection sets based on either rule set will be forced to limit the number of intermediate connection sets computed. For example, Anshelevich's (2002) game-playing program has maximum effectiveness when the number of x-to-y connection sets stored is limited to about 40 per pair of cells x, y .

For both the static and dynamic computational processes, what is needed is some way of distinguishing those intermediate connection sets which are

[3]This fact in turn requires some care to prove; see for example Beck (1969), Gale (1979), and Hayward and Van Rijswijck (200x).

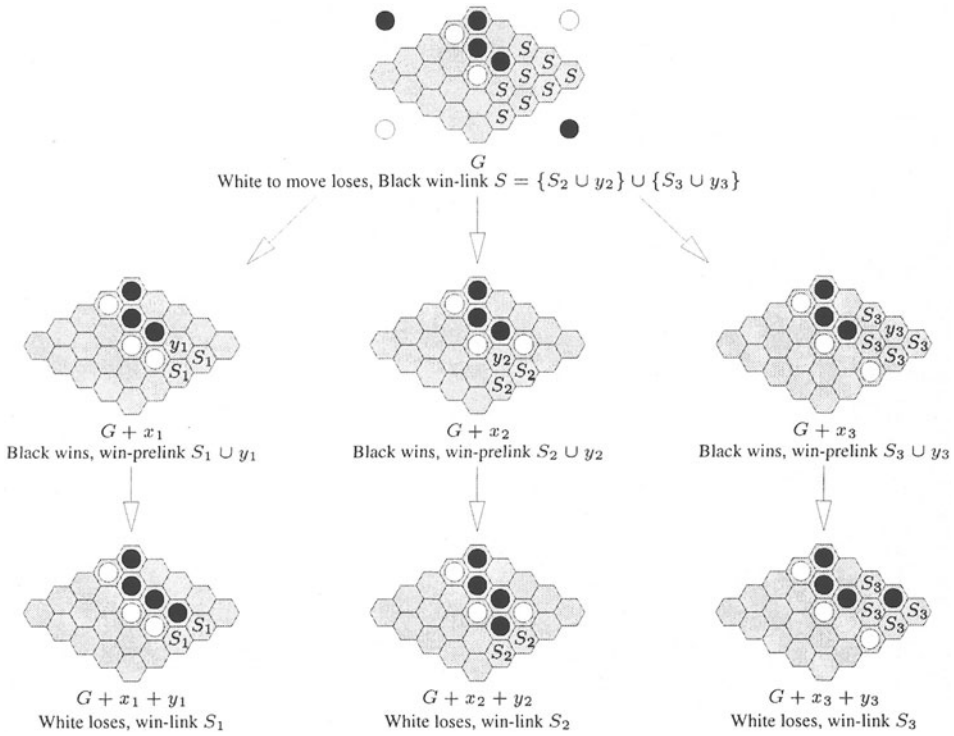

G
White to move loses, Black win-link $S = \{S_2 \cup y_2\} \cup \{S_3 \cup y_3\}$
$G + x_1$
Black wins, win-prelink $S_1 \cup y_1$
$G + x_2$
Black wins, win-prelink $S_2 \cup y_2$
$G + x_3$
Black wins, win-prelink $S_3 \cup y_3$
$G + x_1 + y_1$
White loses, win-link S_1
$G + x_2 + y_2$
White loses, win-link S_2
$G + x_3 + y_3$
White loses, win-link S_3

Figure 5. An example illustrating Theorem 3.

critical to solving the particular state from those which are not. We close this section by giving evidence that this is likely to be a difficult problem.

Assume that at some point in a computation involving the dynamic rules it is discovered that player P has no winning move in a state G. It follows that P's opponent Q has a win-prelink S_x after each possible move x by P and that the union of any collection of these win-prelinks which have an empty intersection establishes a win-link for Q. If G is an intermediate state in the process of solving some earlier state, then P needs to compute such a win-link to pass back to the state which gave rise to G. It is reasonable to expect that a useful win-link to pass back would be one that has the smallest number of cells, among all such possible win-links. However, it is also reasonable to expect this problem to be computationally difficult, since it seems to be intimately related to determining the outcome of a Hex game, which we have already noted is PSPACE-complete. Saks (2003) observed that this problem is indeed computationally difficult, as we now explain.

Formally, the *Min-Union Empty Intersection Problem (MUEIP)* is the decision problem which takes as input an integer k together with a set $S = \{S_1, \ldots, S_t\}$ of subsets of a finite set V and asks whether there is a subset T

of S whose element-intersection is empty and whose element-union has size at most k. *Min Cover* is the decision problem which takes as input an integer k together with a set $A = \{A_1, \ldots, A_t\}$ of subsets of a finite set V and asks whether there is a subset of at most k elements of A whose union is V.

THEOREM 4 *(Saks, 2003) MUEIP is NP-complete.*

Proof. Consider an instance of Min Cover, where k, A, and V are as defined above and $n = |V|$. This instance can be transformed in polynomial time into an instance of MUEIP, as follows.

For each index j, let B_j be the set complement (with respect to V) of A_j, and let $B = \{B_1, \ldots, B_t\}$. Observe that the union of k elements of A is equal to V if and only if the intersection of the corresponding k elements of B is empty. Let V' be the set obtained by adding $t(n+1)$ new elements to V. For each index j, let B'_j be the set obtained by adding $n+1$ of the new elements to B_j in such a way that each B_j gets expanded by a set of new elements disjoint from all other new elements. Let $B' = \{B'_1, \ldots, B'_t\}$. Observe that a set of k elements of B has empty intersection if and only if the corresponding set of k element of B' has empty intersection, and this occurs if and only if the same set of k elements of B' has empty intersection and union with size at most $k(n+1)+n$.

Since MUEIP is clearly in NP, the theorem follows from the preceding transformation and the fact that Min Cover is NP-complete (Karp, 1972). □

Since using virtual connections alone to solve arbitrary Hex states is likely to be computationally difficult, some extra game knowledge must be used to reduce the complexity of searching through the game-tree. We discuss some such reductions in the next section.

3. Move Domination and Game-State Reduction

One reason that Hex is a challenge for computers to play or solve is the high branching factor; especially in the early stages of the game, the number of possible moves is high. In this section we describe some move ordering information which considerably strengthens the algorithmic approach implicitly described by the virtual connection composition rules of the previous section.

A particularly useful form of move ordering information is move domination. Informally, one move dominates another if the former is at least as good as the latter. Since we are interested here only in solving states, namely in determining which player has a win-strategy, one move is "at least as good as" another if the former yields a win whenever the latter yields a win. Formally, for possible moves u, v from a state $[P, B]$, we say that u *dominates* v if P wins $[Q, B+u]$ whenever P wins $[Q, B+v]$.

Domination results are useful for our purposes since any dominated move can be ignored in searching for a winning move. Unfortunately, few results have been proved to date on domination in Hex. Beck (1969) proved that on an empty board size 2×2 or larger, moving to an acute corner (for example, A1 in Figure 7[4]), is a losing, and so dominated, move. Using similar arguments, Hayward (2003b) recently obtained a move domination result involving certain three-cell configuration, as we now explain.

For a player P, a *side cell* is any cell which borders one of P's two sides, a *side pair* $\{x_1, x_2\}$ consists of two adjacent side cells which border the same side, and a *side triangle* (x_1, x_2, t) consists of a side pair $\{x_1, x_2\}$ together with a third cell, called the *tip*, adjacent to the two side cells. See Figure 7. A *P-triangle* is a side triangle belonging to P.

THEOREM 5 *(Hayward, 2003b) Let P be a player with opponent Q and let B be a board-state with an empty P-triangle (x_1, x_2, t). For each subset S of $T = \{x_1, x_2, t\}$, let $B + S$ be the board-state obtained from B by adding a P-stone at each cell of S.*

Then, for each $j = 1, 2$, P wins $[Q, B + t]$ if P wins $[Q, B + x_j]$. Also, P wins any one of the four states $[Q, B + t]$, $[Q, B + \{t, x_1\}]$, $[Q, B + \{t, x_2\}]$, $[Q, B + \{t, x_1, x_2\}]$ if and only if P wins all of them.

Our algorithm uses the above results in the following two ways. Firstly, for any state $[P, B]$ with an empty P-triangle, P can ignore the two moves to the side of the triangle, since they are dominated by the move to the tip. Secondly, for any state $[Q, B]$ with a P-triangle with a P-stone at the tip and the two side cells empty, P-stones can be added to the two side cells, since this addition does not change the outcome of the game. As can be seen from Figure 12, the second result is particularly useful when combined with our virtual connection computation approach.

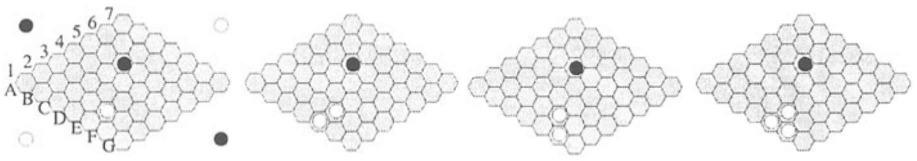


Figure 6. Illustrating the second part of Theorem 5. Applying this result to the white side triangle with tip E2, it follows that a player has a winning strategy for one of these board-states if and only if that player has a winning strategy for all of these board-states.

[4]Throughout this paper, whenever we need to refer to a particular board cell, we assume that the board is oriented as in Figure 7, and use the coordinate system shown there.

4. The Algorithm

Our algorithm SOLVER combines the approaches suggested by Theorems 1, 2, 3, and 5. For a player P with opponent Q, the algorithm solves a state $G = [P, B]$ as follows.

ALGORITHM SOLVER($G = [P, B]$) *For each side triangle for which the second part of Theorem 5 applies, add stones to the appropriate side cells; call the resulting board B^*. Statically compute virtual and weak connections. If a win-prelink for P or a win-link for Q is detected, then return the link (and if the win-link uses the tip of a triangle whose side was filled in, then add the side cells to the link).*

Otherwise, let T be the set consisting of all Q-win-prelinks for G and let R be the P-mustplay region. If T is empty, then initialize R to be all unoccupied cells; otherwise, intialize R to be the intersection of all elements of T. Remove from R any side-cells from any empty P-triangle. While R is not empty, pick a cell x in R, and do the following:

> *Let B^*_x be the state obtained from B^* by adding a P-stone at x and, if x was the tip of an empty P-triangle before this move, filling in the triangle. Recursively solve $G_x = [Q, B^*_x]$.*
> *If P wins G_x, say with win-link X, then add to X the cell x as well as the two associated side-cells if x was the tip of an empty P-triangle, and exit the while loop and return. If Q wins G_x, say with win-prelink X, then add X to T.*

If the while loop terminates without discovering a win-prelink for P, then the union of elements of T forms a win-link for Q.

A sample execution of the algorithm is described in Figures 7 through 9. The correctness of our algorithm follows easily from the previous theorems; we omit the proof.

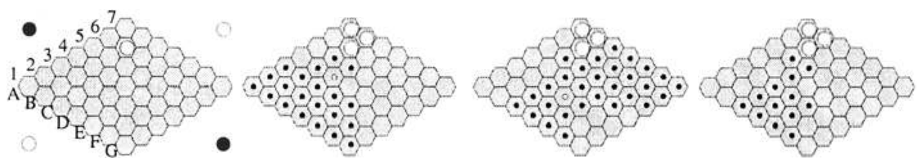


Figure 7. SOLVER solves b6: initialization. After the initial move (left), the game-state is reduced by applying Theorem 5 and adding white stones to the two side-cells of the white side-triangle with tip b6. In the resulting state, White has two win-prelinks (center-left and center-right) whose resulting intersection yields a 13-cell black mustplay region (right). If Black has a winning move, it has a winning move to one of these 13 cells.

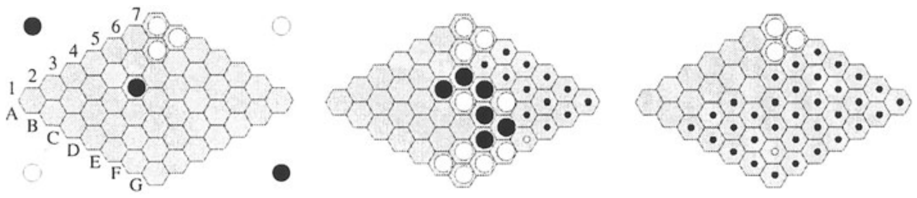


Figure 8. SOLVER solves b6-c4. As shown by the SOLVER b6 recursion tree in Figure 13, c4 is the first black response considered to the white b6 opening (left). Following the topmost path b6-c4-f2-d5-d4-c5-e5-e4-g3-f3-g2-f4 in the recursion tree and applying Theorem 5 after f2 leads to the first solved state (center, with white win-prelink); since f4 is a leaf of the recursion tree, the white win-prelink here was discovered statically. SOLVER continues solving the c4-subtree, eventually determining that c4 is a black loss (right, with white win-prelink). This win-prelink does not contain c4 or b5, so, of the 13 possible b6-responses corresponding to the initial black mustplay region described in Figure 7, 11 moves remain to be checked.

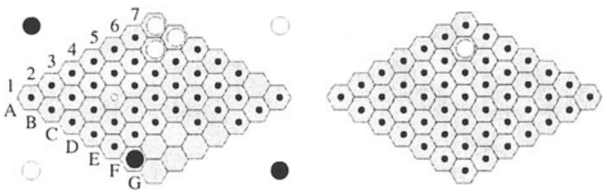


Figure 9. SOLVER solves b6: conclusion. The move to f1 is the last black reply considered in response to the white b6 opening (left, with white win-prelink), since after the discovery of this last white win-prelink, the set of such win-prelinks has empty intersection. The union of these 11 white win-prelinks gives the final win-link for White (right).

5. SOLVER 7×7 1-Opening Solutions

As mentioned earlier, SOLVER is strong enough to solve arbitrary 7×7 states. Figures 10 and 11 summarize the results obtained by running SOLVER on all 49 7×7 1-openings. Figures 13 and 14 show the SOLVER recursion trees from two of these executions, while Figure 15 shows a longest line of play from each of the 49 solutions. Each execution was performed on a single processer machine[5]; in each case, the run time was roughly proportional to the number of nodes in the SOLVER recursion tree, taking about one minute for the five 1-openings with the smallest node-counts, and about 110 hours for the 1-opening with the largest node-count; the total run time for all 49 1-openings was about 615 hours. A listing of all 49 trees (including a tree viewer) is available at `http://www.cs.ualberta.ca/~hayward/hex7trees`.

[5]The program was compiled with gcc 3.1.1 and run on an AMD Athlon 1800+ MHz processor with 512 MB memory running Slackware Linux.

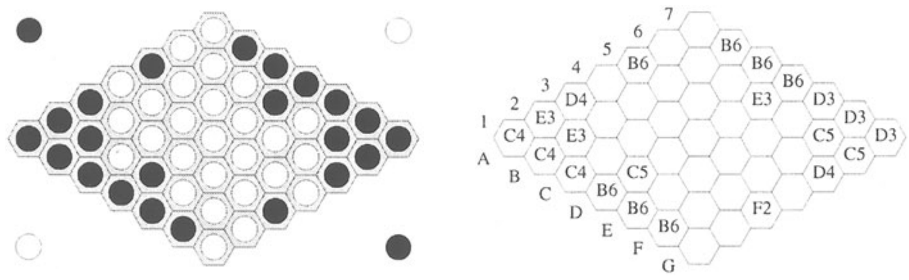


Figure 10. All 7×7 1-opening results, as found by SOLVER. The stone on each cell indicates the winner with perfect play if White's first move is to that cell. The move indicated on each losing cell is the winning countermove discovered.

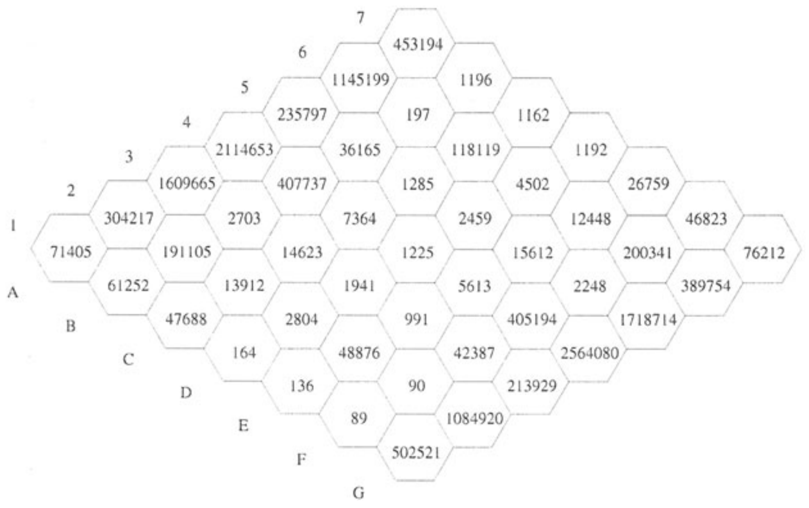


Figure 11. Number of nodes in the SOLVER 7×7 1-opening recursion trees.

For any size Hex board, the set of winning open-move cell locations is symmetric with respect to reflection through the center of the board. Notice that the SOLVER node-counts do not share this symmetry, as neither the order in which SOLVER considers moves nor the static computation of virtual connections is designed to reflect this symmetry.

Figure 12 demonstrates the relative strength of the three key parts of our algorithm, namely virtual connection computation, side-triangle move domination, and side-triangle fill-in, by showing SOLVER node-counts when various of these features are turned off. In particular, notice that adding side-triangle fill-in to virtual connection computation results in a substantial decrease in the number of nodes considered, while further adding side-triangle domination has little effect.

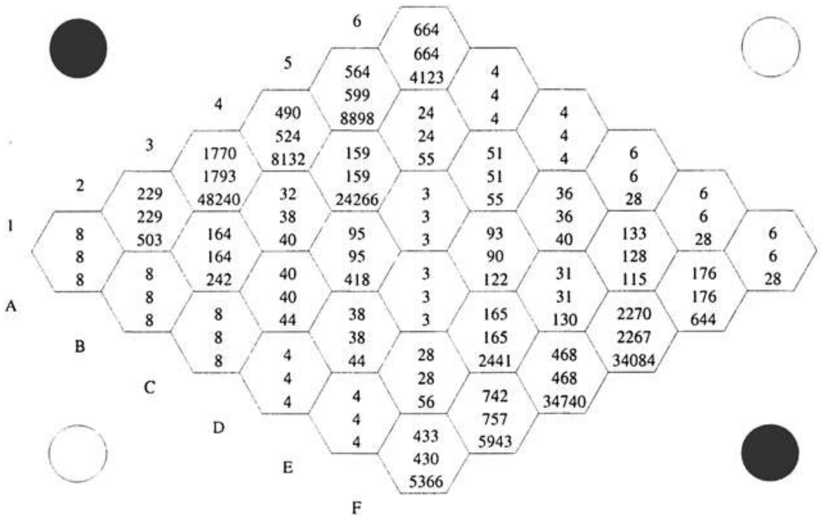


Figure 12. Number of nodes in the 6×6 1-opening recursion trees for SOLVER (top entry), SOLVER−D, namely SOLVER without side-cell domination (middle entry), and SOLVER−FD, namely SOLVER with neither side-cell domination nor fill-in (bottom entry). While corresponding data were obtained for some 7×7 1-openings, SOLVER−FD in particular was too slow to execute for all such openings. For example, the b7 SOLVER−FD tree has 824796 nodes, compared to only 1196 for SOLVER.

In comparing the winning 7×7 opening moves (Figure 10) with winning opening moves on smaller boards (Figure 2), some features common to each of these $n \times n$ boards are worth noting. For example,

- the n cells on the short diagonal (obtuse corner to obtuse corner) are all first-player winning openings,
- the $n - 1$ cells on each of the first-player's sides (except for the cell in the short diagonal) are all first-player losing openings.

It would be of considerable interest to show whether these results hold in general, especially if the proof is positive (as opposed to say a single counterexample), since to date, for arbitrarily large $n \times n$ boards,

- no particular move is known to be a first-player win,
- the only moves which are known to be first-player losses are
  - for $n \geq 2$, the two acute corner cells (Beck, 1969)
  - for $n \geq 3$, the two cells each in the first-player's side and adjacent to the acute corner cell (Beck, 2000).

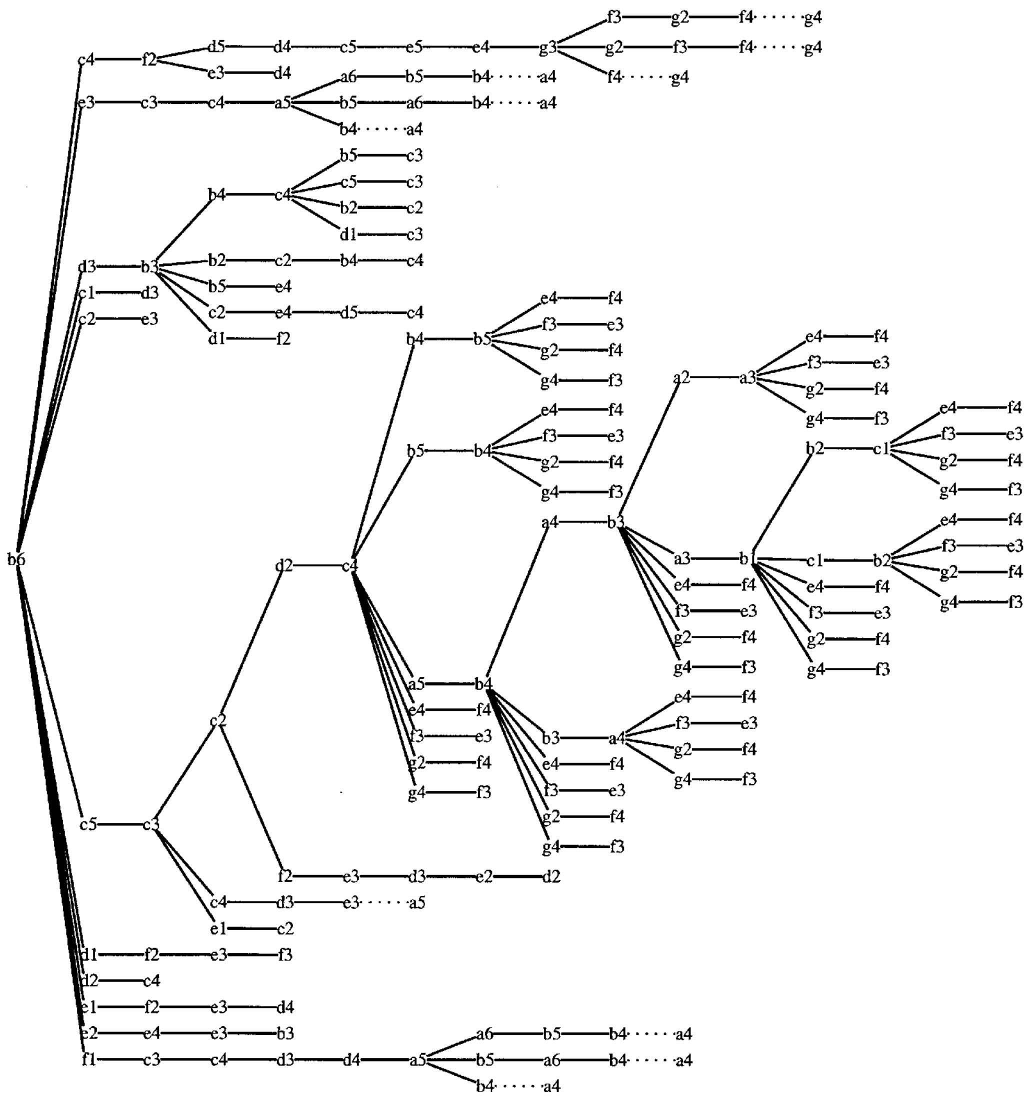


Figure 13. The SOLVER recursion tree for the 7×7 opening White-b6 (with the ten nodes connected by dotted edges added so that every path ends with a winning move). For each node, the order of child generation is top-to-bottom. Each SOLVER recursion tree is a subtree of the complete game-tree, as the only replies to a winning move which appear in the recursion tree are those replies in that state's mustplay region. For example, consider for the tree shown here the state G after White plays b6. As shown in the second diagram in Figure 7, White has a win-prelink created by playing at c4 which does not contain d4; thus d4 is not in the black mustplay region for G, so SOLVER never needs to consider the black move to c4, so c4 does not appear as a child of b6 in this recursion tree. Notice from the tree shown here that in solving the b6 opening the selection of d2 as the first move considered at the b6-c5-c3-c2 subtree was unfortunate, as d2 leads to a white loss whereas f2, the second move considered, leads to a white win. If f2 had been considered first, the d2 subtree would not have been explored, and the resulting recursion tree would have had only 97 nodes instead of 197.

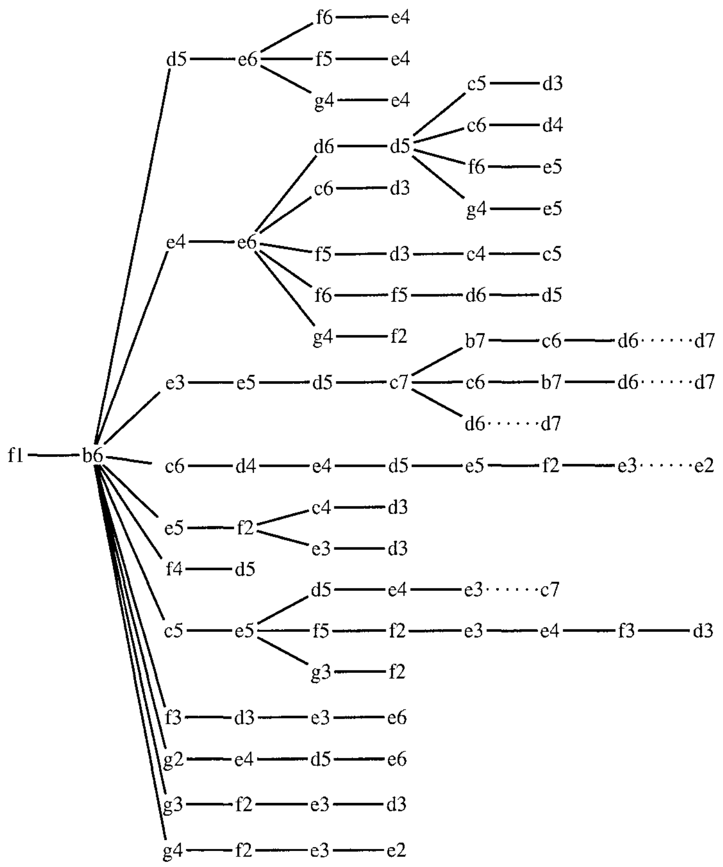


Figure 14. The SOLVER recursion tree for the 7×7 opening White-f1 (with the five nodes connected by dotted edges added so that every path ends with a winning move). For each node, the order of child generation is top-to-bottom. Notice that the f1-b6 subtree, which establishes that b6 is a winning countermove to f1, is paradoxically smaller than the b6 subtree shown in Figure 13, in part because of the move ordering here is more fortunate than there. In this f1-tree, whenever it is White's turn to play, the first move considered turns out to be a winning move; this is not the case in the b6 tree shown in Figure 13.

```
1    2    3    4    5    6    7    8    9    10   11   12   13   14   15   16   17   18   19

a1 c4 a6 f2 d3 b6 e4 d4 e3 e5 d5 e1 e2 f1 a3 b2 c2 c3 a4 d2 b5 a2 b3 a5 b4 a7 c5 c7 b7 c6 d6 d7
a2 e3 f4 d5 c5 d3 g2 e6 e5 c7 d6 d7 g5 f7 e7 f6 b6 d2 c6 g1 f2 f1 e1 e2 a4 b3
a3 d4 c5 b4 c4 d2 c3 c2 b3 b1 b2 c1 e4 f2 e2 d5 d3 b7 a7 b6 a6 b5 e3 e5 c6 c7 d6 d7
a4 b3 d3 d5 c5 d2 c2 b7 a7 b6 a6 b5 e5 c3 a5 b1 a3 b2 e4 f1 e2 e1 g1 f2 g2 f3 g3 f4 g4 f5 g5 f7 f6 e7 e6 d7 c4
a5 b6 c5 e3 f4 d5 c6 c3 d3 e1 e2 f1 g1 f2 d2 d1 g2 f3 g3 e6 d6 e5 g5 f7 e7 f6 d4 e4
a6 f1 c3 b5 c6 d4 c4 c5 a5 b4 a4 b3 a3 a7 b6 b1 b2 c1 c2 d1 e1 d2 e2 d3 e4 d5 e5 e3 g2 f2 g1 f3 g3 f4 g4
a7 d5 c3 d2 c2 c4 a5 b4 a4 a6 e4 d4 e5 e3 b5 b3 a3 b1 b2 b6 c6 c5 g2 f3 g3 g1 f2 f1 e2 d3 e1 f4 g4
b1 c4 a6 f2 d3 b6 e4 d4 e3 e5 d5 e1 e2 f1 a3 b2 c2 c3 a4 d2 b5 a2 b3 a5 b4 a7 c5 c7 b7 c6 d6 d7
b2 e3 g2 g1 a6 b3 c3 d2 f2 f1 e1 e2 c2 b5 d4 c4 d3 d5 c5 b7 a7 b6 a5 b4 e4 e5 c6 c7 d6 d7
b3 b1 c4 c3 b6 f1 b4 c2 b2 c1 d2 d3 e3 d4 e4 e2 g1 d5 e5 f2 g2 f3 g3 f4 g4
b4 f1 c2 b7 a7 c3 b3 b5 c4 c5 e4 d4 e3 e5 d5 c7 b6 c6 d6 d7 e6 e7 g6 f6 g5 f5 g4 f4 g3
b5 f1 c3 b6 c6 d2 c5 c1 a2 b4 c4 b2 a3 b3 c2 d1 e2 d3 e3 d4 e4 e1 g1 d5 e5 f2 g2 f3 g3 f4 g4
b6 c4 f2 d5 d4 c5 e5 e4 g3 f3 g2 f4 g4
b7 b6 e4 d5 e5 e3 g2 f3 g3 e6 f5 f6 c6 c5 d6 f4 d4 d3 g4 g1 c4 c3 f2 f1 e2 e1 d2 d1
c1 c4 a6 f2 d3 b6 e4 d4 e3 e5 d5 e1 d2 c2 a3 d1 b4 a4 b3 b1 e2 b2 f1 c7 b7 c6 a7 b5 a5 c3 d6 d7
c2 d5 c5 b7 a7 b6 a6 b5 a5 b3 c4 f1 b4 c3 e2 d3 e3 d4 e5 e4 g3 f3 g2 g1 f2 d2 e1 f4 g4
c3 d4 b6 b5 c5 c1 c4 d2 a2 b2 a3 b3 a4 b4 c2 d1 e2 d3 e4 d5 e5 e3 g2 f2 g1 f3 g3 f4 g4
c4 c3 b6 d3 e3 f1 d2 e2 b3 b5 b4 c5 d4 d5 e5 e4 g3 f3 g2 g1 f2 b2 c2 f4 g4
c5 d5 c2 b7 a7 b6 a6 b5 a5 b3 c4 f1 b4 c3 e2 d3 e3 d4 e5 e4 g3 f3 g2 g1 f2 d2 e1 f4 g4
c6 d3 c3 c4 e3 d5 c5 d4 e5 e4 a5 b4 a4 b3 a3 b1 b2 b5 a6 a7 b6 c1 c2 d1 d2 e1 e2 f1 g1 f2 g2 f3 g3 f4 g4
c7 b6 e4 d5 e5 e3 g2 f3 g3 e6 f5 f6 c6 c5 d6 f4 d4 d3 g4 g1 c4 c3 f2 f1 e2 e1 d2 d1
d1 b6 c6 d4 e4 d5 e5 f2 e3 e2
d2 c5 e4 d3 f2 e5 d5 f3 e3 c7 b6 c6 b5 c3 c4 d4 a4 b2 b3 c2 d6 d7
d3 d4 b6 c4 a5 a6 b5 b3 c3 b4 e4 d5 e5 e3 g2 f3 g3 g1 f2 f1 e2 e1 c2 f4 g4
d4 d3 c4 c3 e3 f1 c2 d2 e1 e2 g1 f2 f3 g2 a4 b3 a3 b1 b2 b4 a5
d5 d4 f2 e4 g3 f3 g2 f5 e5 f4 c4 c5 a6 a7 b6 b7 c6 d3 c3 b5 a5 c7 e6 b4 a4
d6 e3 c4 d5 b6 c3 d3 b5 c5 e1 f2 e2 f3 e5 d4 e4 g4 f6 e6 f5 d2 d1
d7 b6 e4 d5 e5 e3 g2 f3 g3 e6 f5 f6 c6 c5 d6 f4 d4 d3 g4 g1 c4 c3 f2 f1 e2 e1 d2 d1
e1 b6 c6 d4 e4 d5 e5 f2 e3 e2
e2 d5 e5 e4 c5 d3 e3 d4 c3 c4 g3 f4 g4 f3 g2 f2 g1 f5 g5 f7 f6 e7 e6 d7 c6 d6 a5 b5 a6 a7 b6 b4 a4
e3 d5 b6 c4 c5 d4 b3 b4 e5 e4 g3 f2 f3 e2 d3 d2 c3 c1 a2 a3 b2 f4 g4
e4 e3 d4 d3 b3 b4 c3 c4 g2 g1 f2 f1 e2 e1 d2 b2 c2 d7 c6 f3 g3 f4 g4
e5 d5 e4 e3 d4 d3 b3 c2 b2 c4 a5 a4 b4 b6 c5 b5 g2 g1 f2 f1 e2 e1 d2 c3 d1 f3 g3
e6 f2 e3 d4 e2 e4 f3 f4 c4 c5 a6 a7 b6 b7 c6 d3 c3 b5 a5 c7 d6 b4 a4
e7 d3 a5 b3 e3 d4 e4 f1 e2 e1 d2 d1 c2 c1 a2 b2 g1 f2 g2 f3 g3 f4 g4 f6 f5 e6 e5 d6 c5 b4
f1 b6 e3 e5 d5 c7 b7 c6 d6 d7
f2 e4 b6 d3 d4 e3 c3 c4 a5 a6 b5 b4 a4
f3 d4 e5 f4 e4 f2 e2 e3 c4 c5 a6 a7 b6 b7 c6 d3 c3 b5 a5 c7 e6 d5 d6 b4 a4
f4 c5 f5 d3 e3 f1 g1 f3 d4 e4 c4 d5 a6 a7 b6 b5 a5 b7 c6 c7 d6 d7 f6 e5 e6 b4 a4
f5 d4 f2 f3 e3 e4 c4 c5 a6 a7 b6 b7 c6 d3 c3 b5 a5 c7 d6 d7 f6 e5 e6 b4 a4
f6 c5 a6 a7 g2 f5 e5 d6 b6 b7 c6 d5 c4 c7 e6 f3 e4 e3 d4 c3 d3 e1 e2 f1 g1 f2 f4 g3 d2 d1
f7 d3 a5 b3 e3 d4 e4 f1 e2 e1 d2 d1 c2 c1 a2 b2 g1 f2 g2 f3 g3 f4 g4 f6 f5 e6 e5 d6 c5 b4
g1 b7 e5 e4 f3 d6 g4 f2 e3 f4 g3 e7 g6 g5 f5 f6 d7 e6 c6 d5 c5 c7 a7 g2 e2 d4 c4 b6 a6 d3 c3 b5 a5 b4 a4
g2 b7 e5 f3 e2 d3 e3 e4 d4 d5 a7 c5 g3 f4 g4 f2 g1 f5 g5 f7 f6 e7 e6 d7 d6 c7 b6 c6 a5 b4 a4 b5 a6
g3 f2 e3 c5 d4 c2 a2 b3 e2 d5 e5 e4 f3 f4 g4 f5 g5 f7 f6 e7 e6 d7 d6 c7
g4 e7 c5 d3 c3 b6 c4 c7 c6 c1 b7 d2 a2 b2 a3 b3 a4 b4 a5 b5 c2 d1 e3 d4 e4 d5 e5 f1 e1 e2 g1 f2 g2 f3 g3 d6 e6
g5 d4 c4 d3 e3 e4 c3 c5 a6 b3 b5 a7 b6 b7 c6 c7 d6 d7 e6 e7 g6 f5 f6 d1 c2 c1 a2 b2 d2 e1
g6 c5 g2 f5 c4 c3 e5 f3 f4 g3 b4 d3 a3 d4 a6 a5 b5 a7 b6 b1 b2 a4 b3 b7 c6 c1 c2 d1
g7 d3 c3 c5 c4 e4 d5 d4 a6 a7 b6 b5 a5 b3 b4 b7 c6 c7 d6 d7 e6 e7 f6 f7 g6 d1 c2 c1 a2 b2 d2 e1
```

Figure 15. Longest 7×7 SOLVER lines of play. For each of the 49 7×7 1-openings, the corresponding line shows a longest line of play from the the associated SOLVER solution. The top row shows the move number of that column.

6. Conclusions and Open Problems

We have shown how combining static and dynamic virtual connection computation methods with some move domination results yields an algorithm strong enough to solve arbitrary 7×7 Hex states. A next step is to design an algorithm strong enough to solve 8×8 states; preliminary results suggest that this is considerably more difficult and that further techniques will be required. Another direction is to use SOLVER to gather 7×7 information which can be used to find better move ordering heuristics for Hex game-tree search on (much) larger boards; for example, such data would be useful in analyzing any local configuration with effective board size at most 7×7.

Acknowledgements

The authors gratefully acknowledge the support of the Natural Sciences and Engineering Research Council of Canada, the University of Alberta Research Excellence Envelope, and the University of Alberta GAMES Research Group. Also, the fourth and fifth authors gratefully acknowledge the support of an NSERC Summer Undergraduate Research Award.

We thank Michael Buro, Maryia Kazekevich, Martin Müller, and Jonathan Schaeffer for their assistance in sustaining the Mongoose Hex project which was the starting point for this work. We also thank the referees for their detailed comments on an earlier version of this article.

References

Anshelevich, V. (2002). A Hierarchical Approach to Computer Hex. *Artificial Intelligence*, 134(1-2):101–120.

Beck, A. (1969). Games. In Beck, A., Bleicher, M. N., and Crowe, D. W., editors, *Excursions into Mathematics*, pages 317–387. Worth Publishers, New York, NY.

Beck, A. (2000). Appendix 2000. In Beck, A., Bleicher, M. N., and Crowe, D. W., editors, *Excursions into Mathematics: The Millennium Edition*. A. K. Peters, Natick, MA.

Berge, C. (1977). L'Art Subtil du Hex. Manuscript.

Browne, C. (2000). *Hex Strategy: Making the Right Connections*. A. K. Peters, Natick, MA.

Gale, D. (1979). The Game of Hex and the Brouwer Fixed Point Theorem. *American Mathematical Monthly*, 86(10):818–827.

Gardner, M. (1959). *The Scientific American Book of Mathematical Puzzles and Diversions*, chapter The game of Hex, pages 73–83. Simon and Schuster, New York.

Hayward, R. (2003a). Berge and the Art of Hex. In Bondy, A. and Chvátal, V., editors, *A Biography of Claude Berge*. Princeton University Press. To appear.

Hayward, R. (2003b). A Note on Domination in Hex. Manuscript.

Hayward, R. and Rijswijck, J. v. (200x). On Hex and Mathematics. Manuscript. Submitted to Discrete Mathematics.

Karp, R. (1972). Reducibility Among Combinatorial Problems. In Miller, R. and Thatcher, J., editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York.

Nasar, S. (1998). *A Beautiful Mind*. Touchstone Books, New York.

Reisch, S. (1981). Hex ist PSPACE-vollständig. *Acta Informatica*, 15:167–191.

Rijswijck, J. v. (1999–2003). Queenbee home page. `http://www.cs.ualberta.ca/~queenbee`.

Rijswijck, J. v. (2000). Computer Hex: Are Bees better than Fruitflies? Master's thesis, University of Alberta, Edmonton, Canada.

Rijswijck, J. v. (2002). Search and Evaluation in Hex. Technical report, University of Alberta.

Saks, M. (2003). Private communication.

Yang, J. (2003). A winning 9x9 Hex Strategy. `http://www.ee.umanitoba.ca/~jingyang`.

Yang, J., Liao, S., and Pawlak, M. (2001). A Decomposition Method for Finding Solution in Game Hex 7x7. In *International Conference on Application and Development of Computer Games in the 21st Century*, pages 96–111, Hong Kong.

Yang, J., Liao, S., and Pawlak, M. (2002a). Another Solution for Hex 7x7. Technical report, University of Manitoba, Winnipeg, Canada. `http://www.ee.umanitoba.ca/~jingyang/TR.pdf`.

Yang, J., Liao, S., and Pawlak, M. (2002b). New Winning and Losing Positions for 7x7 Hex. In *Computers and Games*, Edmonton.