The Game of Hex: An Automatic Theorem Proving Approach to Game Programming

Vadim V. Anshelevich

Vanshel Consulting
1200 Navaho Trail
Richardson, Texas 75080
vanshel@earthlink.net

Abstract

The game of Hex is a two-player game with simple rules, a deep underlying mathematical beauty, and a strategic complexity comparable to that of Chess and Go. The massive game-tree search techniques developed mostly for Chess, and successfully used for Checkers, Othello, and a number of other games, become less useful for games with large branching factors like Go and Hex. We offer a new approach, which results in superior playing strength. This approach emphasizes deep analysis of relatively few game positions. In order to reach this goal, we develop an automatic theorem proving technique for topological analysis of Hex positions. We also discuss in detail an idea of modeling Hex positions with electrical resistor circuits. We explain how this approach is implemented in Hexy - the strongest known Hex-playing computer program, able to compete with best human players.

1. Introduction

The rules of Hex are extremely simple. Nevertheless, experienced players recognize that Hex requires both deep strategic understanding and sharp tactical skills. Multiple attempts to build a strong Hex-playing program show that it is a difficult task. One of the major reasons is the large branching factor. For a classic 11×11 Hex board the average number of legal moves is about 100 (compare with 40 for Chess and 8 for Checkers). The massive game-tree search techniques developed over the last 30-40 years mostly for Chess (Marsland, 1986), and successfully used for Checkers (Schaeffer et al. 1996), and a number of other games, become less useful for games with large branching factors like Hex. On the other hand, many experienced game players believe that in most positions, intelligent decisions can be made without a massive game-tree search. Instead, the emphasis can be on a deep strategic analysis of a relatively small number of game positions.

In this paper we concentrate on building a far-sighted evaluation function, which is capable of foreseeing future development of Hex positions. We believe that if such a function is built, it could be used for highly selective game-tree search. In order to reach this goal, we identify topological objects, called ***virtual connections***, which contain information about the potential of Hex positions, and develop an automatic theorem proving technique for their calculations. The construction of virtual connections can be thought of as a very narrow search, which focuses on relevant paths. As a result, this approach is much more efficient than a brute-force search.

There are also other games (Chess and Sokoban), where important game-specific properties were identified, and real time proofs of necessary conditions were successfully used (Adelson-Velskiy, Arlazarov, and Donskoy 1975; Junghanns and Schaeffer 1998). In both cases it has resulted in significant reduction in the size of the search tree.

In section 2 we introduce the game of Hex and its history. In section 3 we present a model for evaluating Hex positions based on electrical resistor circuits. In section 4 we discuss the concept of a virtual connection, and define contributions of virtual connections to the evaluation function. In section 5 we introduce the algebra of virtual connections as a tool for their calculation. In section 6 we explain how this approach is implemented in Hexy - the strongest known Hex-playing computer program, able to compete with best human players. A Windows 95/98/NT version of the program (165KB) is publicly available at the website: http://home.earthlink.net/~vanshel.

2. Hex and Its History

The game of Hex was presented to the general public in *Scientific American* by Martin Gardner (Gardner 1959).

Hex is a two-player game played on a rhombic board with hexagonal cells (see Figure 1). The classic board is 11×11, but it can be any size. The 10×10 and 14×14 board sizes are also popular. The players, Black and White, take turns putting pieces of their color on empty cells of the board. Black's objective is to connect the two opposite black sides of the board with a chain of black pieces. White's objective is to connect the two opposite white sides of the board with a chain of white pieces (see Figure 1). In practice, players often employ "one move equalization",

where the second player has the option of taking the first player's opening move (also known as the swap rule).

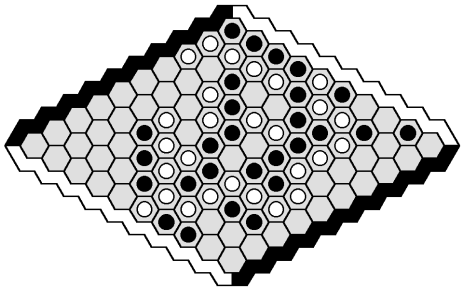

Fig. 1. The chain of white pieces connects white boundaries. White has won the game.

Despite the simplicity of the rules, the game has a complexity comparable to Chess, in both strategy and tactics.

Hex was invented by a Danish poet and mathematician Piet Hein in 1942 at the Niels Bohr Institute for Theoretical Physics, and became popular under the name of Polygon. It was re-discovered in 1948 by John Nash, when he was a graduate student at Princeton. At that time, this game was commonly called John, referring mainly to the fact that it was often played on the hexagonal tiles of bathroom floors. Parker Brothers marketed a version of the game in 1952 under the name Hex.

The game of Hex can never end in a draw. This follows from the fact that if all cells of the board are occupied then a winning chain for Black or White must necessarily exist. While this two-dimensional topological fact may seem obvious, it is not at all trivial. In fact, David Gale demonstrated that this result is equivalent to the Brouwer fixed-point theorem for 2-dimensional squares (Gale 1979). It follows that there exists a winning strategy either for the first or second player. Using a "strategy stealing" argument (Berlekamp, Conway, and Guy 1982), John Nash showed that a winning strategy exists for the first player. However, this is only a proof of existence, and the winning strategy is not known for boards larger than 7×7.

A Hex-playing machine was built by Claude Shannon and E. F. Moore (Shannon 1953). Shannon associated a two-dimensional electrical charge distribution with any given Hex position. His machine made decisions based on properties of the corresponding potential field. In the next section we introduce another way to model Hex positions. Nevertheless, we gratefully acknowledge that our work was inspired by Shannon's original idea.

3. Using Electrical Circuits and Resistance for a Hex Evaluation Function

A reasonable evaluation function for Hex should estimate how much closer Black is to building a winning black chain than White is to building a winning white chain. A popular way to measure how close a player is to building his chain is to calculate the minimal number of pieces he needs to add to connect his two sides of the board. Unfortunately, this type of approach does not take into account the number of potential chains. We attempt to fix this flaw with an ***electrical circuit*** representation of Hex positions.

Consider the four polygonal boundary bands as additional cells (see Figure 1). We assume that black boundary cells are permanently occupied by black pieces, and white boundary cells are permanently occupied by white pieces.

With every Hex position we associate two electrical circuits. The first one characterizes the position from Black's point of view (Black's circuit), and the second one from White's point of view (White's circuit). To every cell c of the board we assign a resistance r in the following way:

$r_B(c) = 1$, if c is empty,
$r_B(c) = 0$, if c is occupied by a black piece,
$r_B(c) = +\infty$, if c is occupied by a white piece,
for Black's circuit, and
$r_W(c) = 1$, if c is empty,
$r_W(c) = 0$, if c is occupied by a white piece,
$r_W(c) = +\infty$, if c is occupied by a black piece,
for White's circuit.

For each pair of neighboring cells, (c_1, c_2), we associate an electrical link with resistance:

$r_B(c_1, c_2) = r_B(c_1) + r_B(c_2)$, for Black's circuit,
$r_W(c_1, c_2) = r_W(c_1) + r_W(c_2)$, for White's circuit.

We now apply an electrical voltage to the opposite boundary cells and measure the total resistance between them, R_B for Black's circuit, and R_W for White's circuit (see Figure 2).

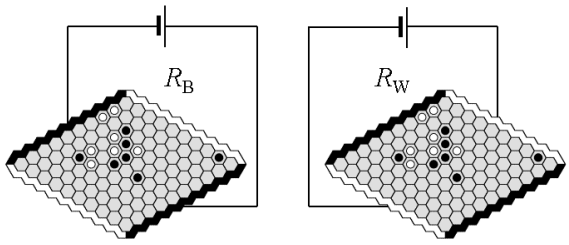

Fig. 2. Black's and White's circuits.

According to the Kirchhoff electrical current laws, the total resistance estimates both the number of pieces that need to be added to the board in order to connect the opposite sides of the board, and the number of ways it can be done.

Now we define an evaluation function:

$$E = R_B / R_W,$$

It is clear that:

- $E = 0$, iff there exists a winning black chain,
- $E = +\infty$, iff there exists a winning white chain,
- The smaller E is, the better this position is for Black, and the worse it is for White.

4. Virtual Connections

In this section we work with Black's circuits only. White's circuits can be dealt with in a similar way.

Consider the two positions in Figure 3. In both positions White cannot prevent Black from connecting the two groups of black pieces, *x* and *y*, even if White moves first, because there are two empty cells *a* and *b* adjacent to both *x* and *y*. If White occupies one of those empty cells, then Black can move to the other. As a result, White should resign immediately in the right position. Note that the black connection between groups *x* and *y* is secured while two cells *a* and *b* stay empty. Black can postpone moving to either *a* or *b* and can use his precious moves for other purposes. In this type of situation we say that the groups of black pieces *x* and y form a ***two-bridge***.

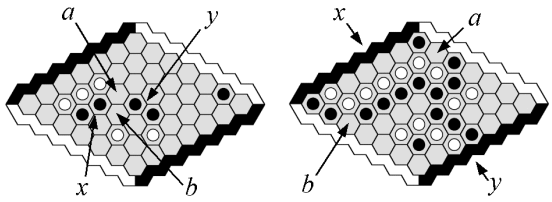

Fig. 3. Groups of black pieces, *x* and *y*, form two-bridges. In the position on the right, those groups are connected to the black boundaries.

We are now going to enhance our evaluation function to reflect this local Black advantage. One natural way of doing this is to add a link with zero resistance between groups *x* and y to Black's circuit. Then the virtual connection would be treated as an actual connection. However, virtual connections are weaker than actual ones, so our evaluation function should reflect this. Instead of connecting black groups *x* and *y* with a shortcut, we add other links to Black's circuit in the following way. If an empty cell *c* is a neighbor of the black group *x*, then we also treat this cell as a neighbor of group *y* (and vice versa). This means that we connect cell *c* and group *y* with an additional link in the same way as we do with actual neighbors.

Black's new circuit serves as a better model for the Hex position than the original one, and the enhanced evaluation function $E = R_B / R_W$ provides a better estimation of the value of the position. This estimate now includes information about the potential of the position two moves ahead.

Our intention is to further modify the electrical circuits in order to build a more far-sighted evaluation function, which takes into account the distant potential of positions. The following definition generalizes the two-bridge concept. First we need to clarify some terms. We say that a ***cell is black*** iff it is occupied by a black piece, and we say that a set of black cells forms a ***group*** iff it is connected.

Definition. *Two groups of black cells x and y, or a group of black cells x and an empty cell y, or two empty cells x and y, form* ***virtual connection*** *iff White cannot prevent Black from connecting them, even if White moves first.*

Any set *A* of empty cells that guarantees that the given pair *x* and *y* form a virtual connection is called a ***carrier*** of that virtual connection. We will describe a virtual connection *V* as a triplet (*x*, *A*, *y*), where groups or cells *x* and *y* are ***ends*** of the virtual connection *V*, and *A* is its carrier.

We represent virtual connections with diagrams as in Figure 4.

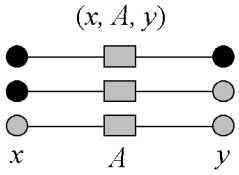

Fig. 4. Diagrams of three types of virtual connections: black-black, black-empty, and empty-empty.

Let us assume that in a given position with a virtual connection, White moves first. The number of moves which must be made in order to ***realize*** this virtual connection (i.e. to convert the virtual connection into an actual one, under the condition that Black does his best to minimize this number, and White does his best to maximize it) characterizes the ***depth of the virtual connection***. In other words, the depth of the virtual connection is the depth of a game-tree search required to discover this virtual connection.

A special role is played by a ***winning virtual connection*** formed by the additional boundary cells. If it exists, then there exists a winning strategy for Black, even if White moves first.

Let us now make several remarks:

- Any pair of neighboring cells forms a virtual connection with an empty carrier. The depth of these virtual connections is equal to zero.
- Two-bridges, described previously, form virtual connections with a depth of two.
- The ends *x* and *y* can form a virtual connection with several different carriers. The virtual connection $V = (x, A, y)$ is ***minimal*** iff there does not exist a virtual connection (x, B, y) such that $B \subset A$ and $B \neq A$. We will be primarily interested in minimal virtual connections.

In Figure 5 you can see four samples of virtual connections.

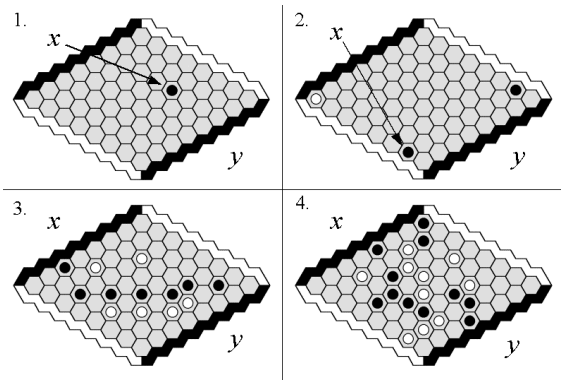

Fig. 5. Black groups *x* and *y* form virtual connections. In each diagram the group *y* is formed by the black pieces connected to the bottom right black boundary.
1. An "edge connection" from the fourth row. Depth = 10.
2. A "ladder". Depth = 14.
3. A chain of two-bridges. Depth = 12.
4. A tactical virtual connection. Depth = 6. This position will be analyzed in the next section.

We can deal with all virtual connections in the same way we did with two-bridges. Let black groups *x* and *y* form a virtual connection. If an empty cell *c* is a neighbor of one of the ends of this virtual connection, say *x*, then we also treat this cell *c* as a neighbor of the other end *y*. This means that we connect cells *c* and *y* with an additional link in the same way as actual neighbors.

Virtual connections with the depth *d* contain information about development of Hex position *d* moves ahead. The more virtual connections we include, and the larger their depths, the more reliable and far-sighted the evaluation function becomes.

5. Algebra of Virtual Connections

In this section we will explain how to detect virtual connections without searching the game-tree. We will define deduction rules (operators), which will allow us to build complex virtual connections starting with simple ones. We will again consider only Black's circuits.

The AND Deduction Rule. *Let* (x, A, u) *and* (u, B, y) *be two virtual connections, where the group u is black, and is a common end for both. If* $x \cap B = \varnothing$, $y \cap A = \varnothing$, *and* $A \cap B = \varnothing$, *then the triplet* $(x, A \cup B, y)$ *also forms a virtual connection.*

Diagram 1 in Figure 6 shows a graphical representation of this deduction rule.

We can explain this deduction rule in the following way. Since $A \cap B = \varnothing$, White cannot attack both virtual connections simultaneously. Let us suppose that White occupies a cell $a \in A$. Since the triplet (x, A, u) forms a virtual connection, then there exists a cell $b \in A$ where Black can play to create a new virtual connection (x, A_1, u). The new carrier A_1 is obtained from A by removing two cells *a* and *b*. In short, if White occupies a cell from A, then Black can restore the first virtual connection by moving to an appropriate cell of A. The same is true for B, and thus every threat can be answered, until an actual connection is obtained.

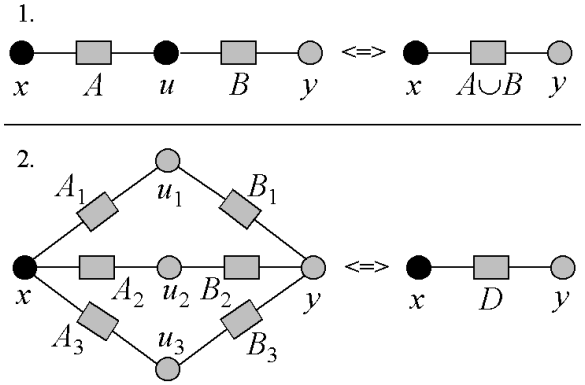

Fig. 6. Deduction rules.
1. The AND deduction rule.
2. The OR(3) deduction rule.

The OR(n) Deduction Rule. *Let* (x, A_k, u_k) *and* (u_k, B_k, y), $(k = 1,2,...,n,$ *for* $n > 1)$ *be virtual connections, where the cells x and y are black or empty, and all cells* u_k *are empty. Let the following conditions be true:*

$$x \cap B_k = \varnothing, \text{ and } y \cap A_k = \varnothing, \qquad \text{for all } k = 1,2,...,n,$$

$$A_k \cap B_k = \varnothing, \qquad \text{for all } k = 1,2,...,n,$$

$$\bigcap_{k=1}^{n} C_k = \varnothing,$$

where $C_k = A_k \cup u_k \cup B_k$, *for all* $k = 1,2,...,n$. *Then the triplet* (x, D, y) *also forms virtual connection with the carrier:*

$$D = \bigcup_{k=1}^{n} C_k.$$

Diagram 2 in Figure 6 graphically represents this deduction rule (for $n = 3$).

The explanation of this rule is as follows: If White occupies a cell $a \in C_i$, then there exists a different carrier C_j, such that $a \notin C_j$. Therefore, Black can move to u_j, to form a new virtual connection $(x, A_j \cup B_j, y)$, since (x, A_j, u_j) and (u_j, B_j, y) satisfy the conditions of the AND deduction rule.

Automatic Theorem Proving

Figure 7 demonstrates how the AND and the OR(n) deduction rules can be used to prove more complex virtual

connections. Diagram 1 in Figure 7 represents the position on the board. The sequence of transformations in diagrams 2 through 4 graphically demonstrates the application of the deduction rules, and proves that Black has a winning position.

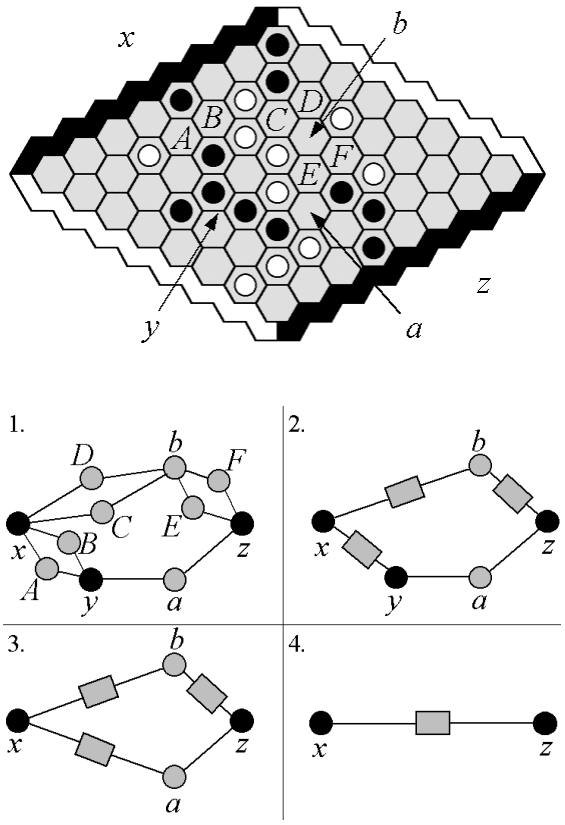

Fig. 7. Automatic theorem proving. Diagram 1 represents the position on the board above. Diagram 2 is obtained from Diagram 1 by applying the OR(2) deduction rule three times. Diagram 3 results from the AND rule. The winning virtual connection (x, z) in Diagram 4 follows from a final application of the OR(2) rule.

Let us consider the simplest virtual connections, namely the pairs of neighboring cells, as the set of axioms or in other words the first generation of virtual connections. Applying the AND and OR(n) deduction rules to the appropriate groups of the first generation of virtual connections we build (prove) the second generation of virtual connections. Then we apply the deduction rules to both the first and the second generations of virtual connections to build (prove) the third generation of virtual connections, etc.

The goal is not to prove some specific virtual connection (e.g. a winning one). The goal is to construct a collection of virtual connections, which belong to the given position.

This iterative algorithm can prove all of the virtual connections shown in Figures 3, 5, and 7. We would also like to know whether this system of deduction rules is ***complete***, i.e. whether this process can build ***all*** virtual connections. The answer is negative. The diagram in Figure 8 represents a counter-example of a virtual connection that cannot be proven by this process. The fact that this is a virtual connection can be verified manually. For example, if White plays at a, Black can reply with b, forcing White to occupy c. Then Black plays d securing the connection. A computer program was used to verify that no combination of AND and OR(n) rules can establish the overall connection.

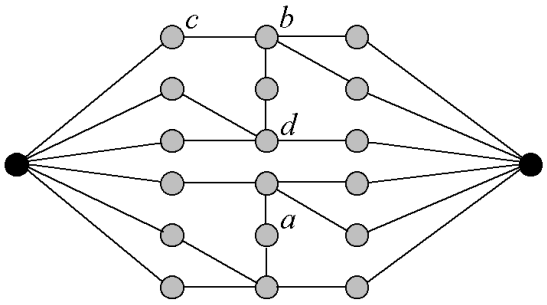

Fig. 8. The two black pieces form a virtual connection, which cannot be proven using AND and OR(n) deduction rules.

The system of deduction rules could be expanded to handle this case, but we are not going to do it in this paper.

6. Hexy Plays Hex

Hexy is a Hex-playing computer program which utilizes the ideas and algorithms presented in this paper. It runs on a standard PC with Windows, and can be downloaded from the website: http://home.earthlink.net/~vanshel. We consider the Advanced level as a standard. It plays a complete 10×10 game in about 10 minutes.

Hexy uses the alpha-beta search algorithm, with the evaluation function described in sections 3 and 4. For calculation of virtual connections, Hexy uses the automatic theorem proving algorithm introduced in section 5. This algorithm must be implemented very carefully. To make the algorithm efficient, we calculate only minimal connections, and enforce some reasonable restrictions.

The program has two thresholds, N and D. N is the maximal number of different virtual connections with the same ends. This threshold indirectly controls the total number of virtual connections calculated. The larger N, the more virtual connections the program builds for every node of the game-tree, and the more time the program spends on their calculation. The second parameter, D, is the depth of the game-tree search. We do not put any limits on the number of iterations, or the depth of virtual connections. There is an obvious trade-off between the parameters N and D, and finding a good compromise is an important task. The best practical results determined experimentally,

were obtained with values of $N = 20$ and $D = 3$ (for a 10×10 board). As a result, Hexy performs a very shallow game-tree search (200-500 nodes per move), but routinely detects virtual connections with depth 20 or more.

Let $E(n)$ be an evaluation function that takes into account only those virtual connections which are built in the first n iterations of the theorem proving algorithm. Then $E(\infty)$ is an evaluation function without limits on the number of iterations (Iterations stop when no new virtual connections are discovered). Let Hexy(n) and Hexy(∞) be corresponding versions of the program. Hexy(0) does not calculate virtual connections at all, and Hexy(1) takes only two-bridges into account. Figure 9 shows the dependence of the ratio $T(n)/T(0)$ on n, where $T(n)$ is the time for the evaluation of a typical Hex position with $E(n)$. The ratio $T(\infty)/T(0)$ varies for different positions, but typical values are in the range 5-15. This means that the additional cost of computing the evaluation function $E(\infty)$, relative to that of the evaluation function $E(0)$, is not greater than the cost of one or two additional plies of game-tree search. Since $E(\infty)$ routinely finds virtual connections with depth 20 or more, it is not surprising that Hexy(∞) with $D = 3$ easily defeats Hexy(0) and Hexy(1) when they use a deeper 5-ply game-tree search.

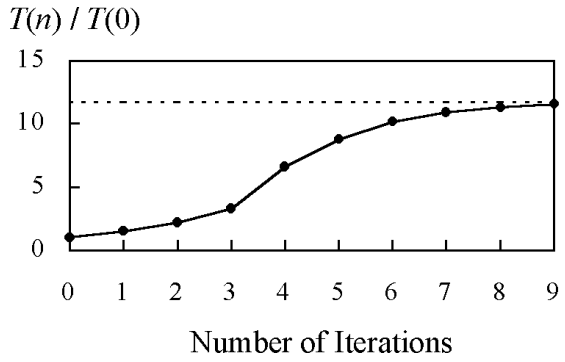

Fig. 9. Ratio $T(n)/T(0)$ versus number of iterations of the theorem proving algorithm. The dotted horizontal line shows the ratio $T(\infty)/T(0)$.

Hexy demonstrates a clear superiority over all known Hex-playing computer programs.[1,2] Hexy was also tested against human players on the popular game website, Playsite (http://www.playsite.com/games/board/hex). After more than 100 games, the program achieved a rating, which is within the highest Playsite *red* rating range.

[1]Chris Lusby Taylor, Hex, 1995.
Zhiping You, Hex, 1995.
A. V. Antonov, D. A. Antonov, Logical Game Hex, 1998.
Bob Kirkland, Hex-7, 1998.
Jack van Rijswijck, Queenbee, 1999.

[2]There is also a program Hex 1.0 by Sven Erik Elfgren (1997). This program plays on a 7×7 board and always moves first. It seemingly demonstrates perfect play.

7. Conclusion

In this paper we have offered the automatic theorem proving approach to Hex programming, and explained how this approach is implemented in Hexy - the strongest known Hex-playing program able to compete with best human players. Unlike conventional game-playing programs, Hexy does not perform massive game-tree search. Instead, this program spends most of its resources on deep analysis of a relatively small number of Hex positions.

Due to the mathematical purity of Hex rules, we have been able to build a far-sighted evaluation function based on virtual connections - topological objects, which contain information about the potential of Hex positions many moves ahead. We have built the automatic theorem proving technique for calculation of virtual connections.

The process of building virtual connections has its own cost. Nevertheless, this approach is much more efficient than brute-force search because it does not consider irrelevant paths. Experiments also show that the foreseeing abilities of this kind of evaluation function greatly outweigh its cost.

Acknowledgements

I would like to express my gratitude to Jonathan Schaeffer. Without his encouragement this paper would never have been written. I wish to thank Ryan Hayward, Jack van Rijswijck, and Sven Erik Elfgren for useful discussions. I owe my special thanks to Darse Billings for invaluable comments on earlier drafts of the paper.

References

Adelson-Velskiy, G.; Arlazarov, V.; and Donskoy, M. 1975. Some Methods of Controlling the Tree Search in Chess Programs. *Artificial Intelligence* 6(4):361-371.

Berlekamp, E. R.; Conway, J. H.; and R. K. Guy, R. K. 1982. *Winning Ways for your Mathematical Plays*. New York: Academic press.

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

Gardner, M. 1959. *The Scientific American Book of Mathematical Puzzles and Diversions*. New York: Simon and Schuster.

Junghanns, A.; Schaeffer J. 1998. Single-Agent Search in the Presence of Deadlock. *AAAI'98* 419-424.

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

Schaeffer, J.; Lake, R.; Lu, P.; and Bryant M. 1996. Chinook: The World man-Machine Checkers Champion. *AI Magazine* 17(1):21-29

Shannon, C. E. 1953. Computers and Automata. *Proceedings of Institute of Radio Engineers* 41: 1234-1241