Computers and Games, 7th Int'l Conf, CG2010 (Kanazawa), H.J. van den Herik and H. Iida and A. Plaat (eds.), Springer LNCS 6515, 2011, pp 1-10

Solving Hex: Beyond Humans

Broderick Arneson, Ryan B. Hayward, Philip Henderson

Dept. of Computing Science, University of Alberta
{broderic,hayward,ph}@cs.ualberta.ca

Abstract. For the first time, automated Hex solvers have surpassed humans in their ability to solve Hex positions: they can now solve many 9×9 Hex openings. We summarize the methods that attained this milestone, and examine the future of Hex solvers.

1 Introduction

Hex has simple rules: Black and White alternate turns; on each turn, a player places a single stone of their colour on any unoccupied cell. The winner is the player who forms a chain of their stones connecting their two opposing board sides. See Figure 1.

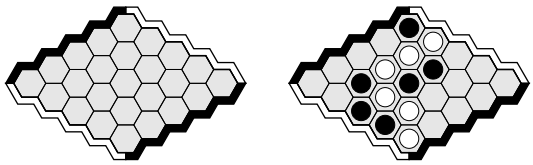

Fig. 1. An empty 5×5 Hex board and a game won by White.

Since Hex was invented, humans have been interested in solving Hex positions, either by hand or by computer.

Several results on winning strategies are known. Nash [20] used the now well-known strategy-stealing argument to prove the existence of a first-player winning strategy for Hex on $n \times n$ boards. Shannon (see [9]) found a winning pairing strategy for Hex on $m \times n$ boards with $m \neq n$, and Alpern and Beck [2] found a winning strategy for Hex on cylindrical boards. Henderson and Hayward [16] found a winning handicap strategy for Hex on an $n \times n$ board where the first player is allowed to place $\lceil (n+1)/6 \rceil$ stones on their first turn.

Some results on losing strategies are also known. For $n \times n$ boards, Beck et al. [5, 6] showed that opening at the acute corner cell loses for $n \geq 2$, and that opening at the cell adjacent to the acute corner cell and the first-player's edge loses for $n \geq 3$.

However, for arbitrary positions on the $n \times n$ Hex board it seems unlikely that there is any polynomial time algorithm to find (even the first move of) a winning strategy, as Reisch [24] showed that solving such positions is PSPACE-complete.

Until recently computers were incapable of matching human ability, but that has changed: computers can now solve all Hex openings (and more) that humans have solved.

In this paper we review the algorithmic methods that attained this milestone. In §2 we review previous Hex solving benchmarks, summarizing the vital methods. In §3 we review our new algorithmic and proof techniques, analyze their respective contributions, and present the current state of solved Hex openings. In §4 we analyze the scaling of our Hex solver to larger board sizes, and extrapolate to predict future milestones and the effects on Hex players.

2 Previous Hex Solving Benchmarks

Due to the first-player advantage, Hex is often played with the swap rule: the first-player makes Black's first move, and then the second-player chooses whether to play as Black or White. White makes the next move, and the players alternate turns thereafter. This variation is a second-player win, but in practice produces closer matches than without the swap rule.

As mentioned, the empty $n \times n$ Hex board is ultra-weakly solved, i.e. the outcome (assuming perfect play) is known. The goal of current Hex research is to achieve the following on successively larger Hex boards:

- to weakly solve the root state, i.e. to find a winning strategy for the first-player;
- to ultra-weakly solve all opening moves, i.e. to determine the second-player's correct choice of colour when playing with the swap rule;
- to weakly solve all opening moves, i.e. to find a winning strategy for the second-player when playing Hex with the swap rule;
- to strongly solve the board, i.e. to find an algorithm capable of solving any position with the given board size in a reasonable amount of time.

Solving Hex on board sizes up to 4×4 is not difficult. Hein [12] commented that interesting positions start to arise on 5×5 boards. Gardner [9] briefly discussed winning opening moves on board sizes up to 5×5, and added that he was unaware of any complete 6×6 analysis. In 1995 Enderton [8] weakly solved all 6×6 openings, reporting the solutions but no algorithmic details. In 2000 Van Rijswijck [25] strongly solved the 6×6 board.

The $n \times n$ hex board has rotational symmetry, so only $\lceil n \times n/2 \rceil$ openings need to be solved. In 2001 Yang [27] weakly solved 7×7 by hand, and in 2002 Yang et al. [28] weakly solved 9 of the 25 asymmetric opening moves. Yang's main technique is the decomposition method: build up larger connection templates from basic ones, so that a common substrategy can be used to respond to large sets of moves, thus moderating the combinatorial explosion. Yang's solution uses over 40 templates, and its correctness proof has 12 pages of case analysis. In 2004

Noshita [21] gave a similar 7×7 strategy with a simpler proof, by incorporating connections of the form "chain x connects to either chain y or chain z".

In 2003 the first automated 7×7 solution appeared when Hayward et al. [11] weakly solved all 7×7 openings. Two concepts were fundamental to the success of this algorithm: inferior cells and H-search. H-search is an algorithm developed by Anshelevich [3, 4] that simulates Yang's manual decomposition technique: from a base case of adjacent cells, build up connection strategies by combining smaller ones in series and parallel; repeat this until no new connections are found. Hayward's [10] inferior cell analysis proves that when a stone forms a bridge to an edge of its own colour, then both empty cells can be filled in without changing the value of the position. See Figure 2. In 2007 Rasmussen et al. [23] gave a more efficient solution of all 7×7 openings, by having the algorithm store discovered connections (in a generalized form) that cannot be found by H-search.

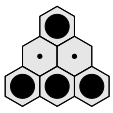

Fig. 2. Edge-captured cells: adding Black stones to the two dotted cells does not change the position's value.

However, before these automated results, in 2002 and 2003 Jing Yang [26] had by hand already weakly solved the centre openings for 8×8 and 9×9. In 2005 and 2006 Noshita [22] and Mishima et al. [17] gave further manual 8×8 opening solutions.

Again, automated Hex solutions were years behind: in 2009 Henderson et al. [14] weakly solved all 8×8 openings (and strongly solved 7×7). Features that led to this improvement included stronger inferior cell analysis (over 250 patterns used to prune moves from consideration), generalized decompositions allowing combinatorial board partitioning, and deduction of equivalent position values.

3 Current Hex Solving Techniques

Until now, the only 9×9 opening solution was the weakly solved centre move by Yang: no automated Hex solver had solved any 9×9 opening. Furthermore, there are orders of magnitude difference in the time required to solve consecutive board sizes: the solver of Henderson et al. [14] weakly solved all 7×7 openings in about 10 minutes, all 8×8 openings in about 300 hours, and no 9×9 openings after several weeks.

By comparison, our current solver ultra-weakly solves all 8×8 openings in about 31 hours, and 28 of the 41 asymmetric 9×9 openings in 1 to 25 days each. See Figure 3. This improvement is due to the following adjustments:

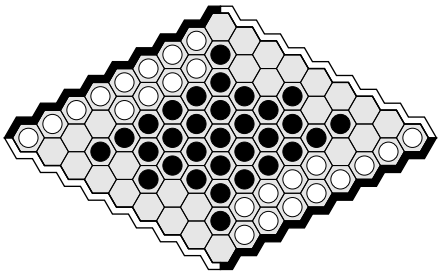

Fig. 3. Solved 9×9 openings.

1. switch from depth-first search to depth-first proof-number search
2. improve H-search by incorporating captured cells
3. improve inferior cell analysis by
   - using permanently inferior cells,
   - using captured-reversible cells,
   - using star-decomposition domination,
4. apply strategy-stealing arguments in the search.

3.1 Depth-first proof-number search

Allis et al. [1] introduced proof-number (PN) search, an algorithm for solving two-player games that uses the branching factor of the search tree to guide it towards a proof tree of small size. Nagai [18] developed a depth-first variant (DFPN) to reduce the large memory requirements of the original algorithm. Until now DFPN has been successfully applied to many games, but not Hex. Two impediments were incremental versus static H-search, and uniform branching factors.

The first impediment is that our most efficient implementation of H-search does not integrate easily with DFPN. In particular, while depth-first search (DFS) repeatedly moves from one state to a state that differs by one stone, DFPN expands search tree leaf states that differ by many stones. Thus DFS can update its connection strategies incrementally, whereas DFPN cannot. Our solver spends most of its search time computing connection strategies, and shifting from incremental to static connection computation halves our speed. We have no solution to this problem, but fortunately on larger board sizes DFPN usually more than halves the search space, yielding a net improvement.

The second impediment is that Hex begins with near-uniform branching factors, so that initially DFPN is essentially conducting an inefficient breadth-first search. To alleviate this problem, we implement our own variant of DFPN that initially constrains the branching factor using a move ordering heuristic.

3.2 Improved H-search

Henderson et al. [13] presented XH-search, an extension of H-search which, among other things, exploits the observation that connection strategies do not

conflict if they intersect only on captured cells of the strategies. We extend this observation to the series/parallel construction rules of H-search, and thus find more connections.

3.3 Improved Inferior Cell Analysis

We strengthened our inferior cell computation by adding three features: permanently inferior cells, captured-reversible moves, and star decomposition domination.

The first feature is a new kind of inferior Hex cell that comes from combining arguments about Hex dead and captured cells. As observed by Henderson et al. [16], for some new patterns, a cell can be added without altering the value of the position containing the pattern. See Figure 4.

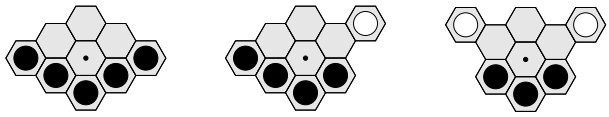

Fig. 4. Permanently inferior patterns. For each pattern, a Black stone can be added to the dotted cell without changing the containing position's value.

The second and third features come from the combinatorial game theory notions of reversible and dominated moves as defined by Berlekamp et al. [7], which allow simplification of the game tree by either bypassing or pruning legal moves. Henderson et al. [15] showed that in Hex these notions lead to captured-reversible moves and star-decomposition domination, and that each allows pruning. See Figure 5.

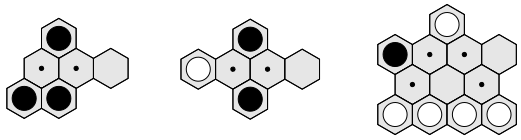

Fig. 5. Two captured-reversible patterns and a star decomposition domination pattern. Black can prune the dotted cells from consideration.

3.4 Application of the Strategy-Stealing Argument

The strategy-stealing argument applies to the empty Hex board, but also to any Hex position where the position for Black mirrors the position for White. Thus,

any such state is a first-player win; we ensure that White avoids any move to such a state. See Figure 6.

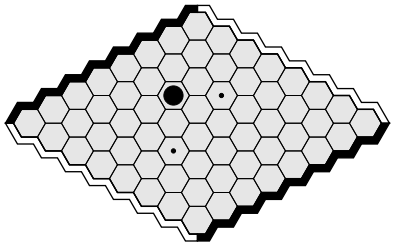

Fig. 6. Strategy-stealing pruning: White can prune each dotted cell from consideration, since each resulting state is Black-White isomorphic and so a Black win.

Our use of this technique implies that we only ultra-weakly solve the opening state, rather than weakly solve it. However, the proof our algorithm finds can easily be extended to a complete one by later computing winning strategies for all pruned Black-White isomorphic states.

3.5 Feature Contributions

The contributions of our new features are shown in Figure 7.

Note that feature importance seems to increase with board size. For instance, our current algorithm is less than two times faster than last year's algorithm on 7×7 openings, almost ten times faster on 8×8 openings, and at least twenty times faster on the 9×9 centre opening (this last ratio is unknown, as last year's algorithm failed to solve this position).

8×8 Hex openings		
Feature f turned off	**Time (s)**	**% slower**
Captured-cell H-search	196,227	75.0
Inferior cell analysis improvements	126,201	12.6
Strategy-stealing pruning	118,010	5.3
None	112,121	0.0

Fig. 7. Feature contributions when ultra-weakly solving all 8×8 Hex openings.

This might be because the computational complexity of the runtime of most of our improvements is polynomial in the board size, while the corresponding increase in search space pruning seems to grow exponentially. Furthermore, as the expected game length increases and weak moves are no longer immediately losing, pruning inferior cells becomes more likely to save significant search time.

4 Hex Solver Scaling

To date we have solved 28 of the 41 asymmetric 9×9 openings; however this likely represents only a small fraction of the time needed to solve all 9×9 openings. For example, on smaller $n \times n$ boards, solving all openings adjacent to a White edge takes more than half of the the time to solve all $n \times n$ openings.

The solving time data in Figure 8 suggests some trends. Firstly, solving all openings on a fixed size board takes about 750 times the time needed to solve the easiest opening. If this holds up to 9×9, then solving all positions will take about 750×100,000 seconds, or about 870 days.

Board size	**Fastest opening**	**All openings**
7×7	0.5	384
8×8	155	112,121
9×9	96,168	unknown

Fig. 8. Ultra-weakly solving opening times (seconds) by board size.

Secondly, ultra-weakly solving all $n \times n$ openings takes roughly as long as ultra-weakly solving the easiest $(n+1) \times (n+1)$ opening. If this holds up to 10×10, then (ignoring possible speedups due to improved hardware) solving one position will take about 870 days, and solving all 10×10 openings will take about 750*870 days, or about 1800 years.

In order to gain further information on the expected time to solve larger Hex positions, we collected sets of Hex games, and determined the longest opening from each game that could not be solved in a fixed amount of time. To generate the game sets, we used self-play with MoHex, our Monte Carlo Hex player (and the Hex competition gold medallist from the 2009 Computer Games Olympiad in Pamplona), restricted to 10,000 simulations per move.

For each board size, we played enough rounds (over each possible opening move) to generate a suite of about 1000 games; we then selected the longest 100 games of each suite, under the assumption that these 100 games would represent the most difficult opening sequences to solve. For each set of 100 games, we then determined the longest opening from that game that our solver cannot solve in t seconds; we did this for $t = 10$ and $t = 60$. See Figures 9 and 10.

When competing, our Hex players run the solver on a parallel thread. The results shown in these figures are consistent with our experience in 11×11 tournament play, namely many positions are solved by the 35th game move, and most positions are solved by the 45th game move. Comparing the two figures, a 6-fold increase in solving time corresponds to decrease of about 4 stones in the depth of a solveable position. If we assume a doubling of computing power every 2 years, this suggests that in 10 years this will have led to an increase in solving power of about 32×4/6 stones, or more than 21 stones. This would mean that many 35-21=14-stone 11×11 states, and most 45-21=24-stone states would

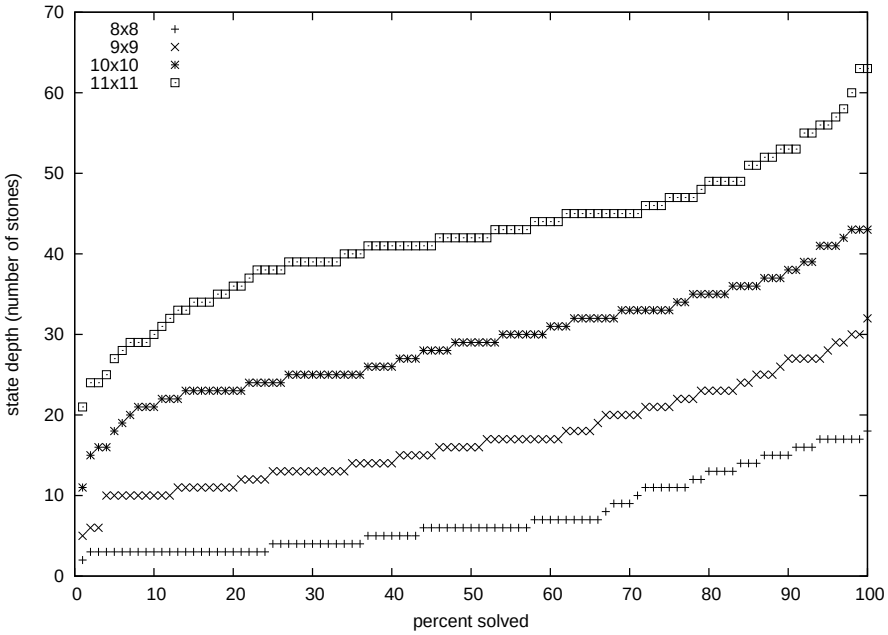

Fig. 9. Percent of states solved with 10s dfpn search versus state depth.

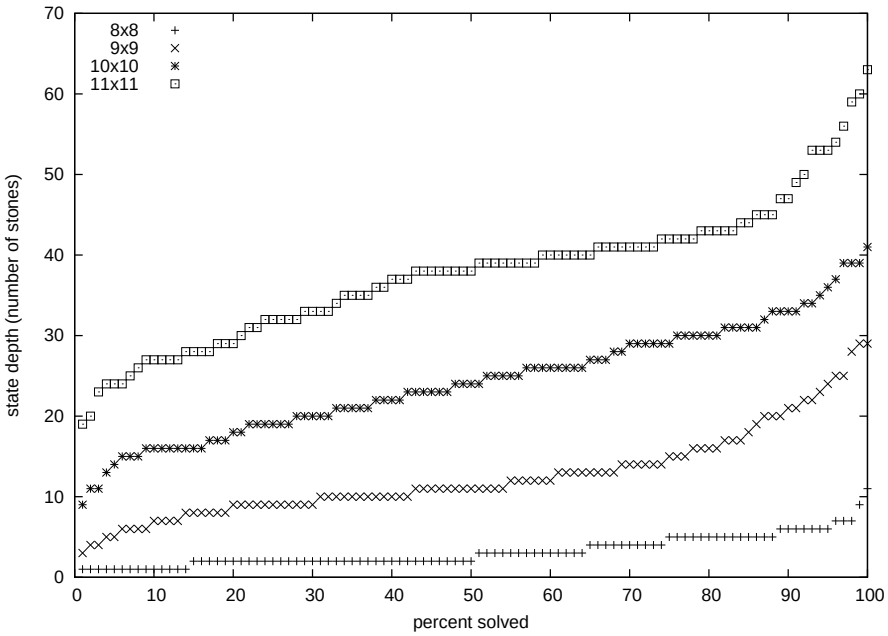

Fig. 10. Percent of states solved with 60s dfpn search versus state depth.

be solveable. It thus seems likely that within the next decade Hex competitions will shift from the original 11×11 board (as introduced by Hein) to playing on a larger board, for example to the 13×13 board used on Little Golem, or the 14×14 recommended by Nash [19].

5 Conclusions

We have surveyed the past and present algorithmic techniques that have led to computers passing human ability in solving Hex: they can now solve more than half of the 81 9×9 openings. Furthermore, experimental results suggest that the next milestone for Hex solvers, namely weakly solving some 10×10 opening, might be reached within two years, but that further progress might be slower unless there are major algorithmic and/or hardware developments.

Acknowledgements

We thank the University of Alberta's Hex and GAMES group members for their feedback on this research, especially Martin Müller.

References

1. L. Victor Allis, Maarten van der Meulen, and H. Jaap van den Herik. Proof-number search. *Artificial Intelligence*, 66(1):91–124, 1994.
2. Steve Alpern and Anatole Beck. Hex games and twist maps on the annulus. *Amer. Math. Monthly*, 98(9):803–811, 1991.
3. Vadim V. Anshelevich. The game of Hex: An automatic theorem proving approach to game programming. In *AAAI/IAAI*, pages 189–194, Menlo Park, 2000. AAAI Press / The MIT Press.
4. Vadim V. Anshelevich. A hierarchical approach to computer Hex. *Artificial Intelligence*, 134(1–2):101–120, 2002.
5. Anatole Beck, Michael N. Bleicher, and Donald W. Crowe. *Excursions into Mathematics*, chapter 5, pages 327–339. Worth, New York, 1969.
6. Anatole Beck, Michael N. Bleicher, and Donald W. Crowe. *Excursions into Mathematics: the Millennium Edition*, chapter Appendix 2000. A.K. Peters, Natick, Massachusetts, 2000.
7. Elwyn Berlekamp, John H. Conway, and Richard K. Guy. *Winning Ways for Your Mathematical Plays*, volume 1–4. A.K. Peters, 2nd edition, 2000.
8. Bert Enderton. Answers to infrequently asked questions about the game of Hex. `www.cs.cmu.edu/~hde/hex/hexfaq/`, 1995.
9. Martin Gardner. Mathematical games. *Scientific American*, 197(1):145–150, June 1957.
10. Ryan B. Hayward. A note on domination in Hex. Technical report, University of Alberta, 2003.
11. Ryan B. Hayward, Yngvi Björnsson, Michael Johanson, Morgan Kan, Nathan Po, and Jack van Rijswijck. Solving 7×7 Hex: Virtual connections and game-state reduction. In H. Jaap van den Herik, Hiroyuki Iida, and Ernst A. Heinz, editors, *Advances in Computer Games*, volume 263 of *International Federation for Information Processing*, pages 261–278. Kluwer Academic Publishers, Boston, 2003.

12. Piet Hein. Polygon. *Politiken*, December 27 1942.
13. Philip Henderson, Broderick Arneson, and Ryan Hayward. Hex, braids, the crossing rule, and XH-search. In *12th Advances in Computer Games Conference*, 2009.
14. Philip Henderson, Broderick Arneson, and Ryan B. Hayward. Solving 8x8 Hex. In Craig Boutilier, editor, *IJCAI*, pages 505–510, 2009.
15. Philip Henderson and Ryan B. Hayward. Captured-reversible moves and star decomposition domination in Hex. Submitted to Integers, 2010.
16. Philip Henderson and Ryan B. Hayward. A handicap strategy for Hex. In Richard J. Nowakoswki, editor, *Games of No Chance IV*. Cambridge University Press, 2010 (in press).
17. Ken Mishima, Hidetoshi Sakurai, and Kohei Noshita. New proof techniques and their applications to winning strategies in Hex. *Proceedings of 11th Game Programming Workshop in Japan*, pages 136–142, 2006.
18. A. Nagai. *Df-pn Algorithm for Searching AND/OR Trees and its Applications*. PhD thesis, University of Tokyo, Tokyo, Japan, 2002.
19. Sylvia Nasar. *A Beautiful Mind: A Biography of John Forbes Nash, Jr.* Simon and Schuster, 1998.
20. John Nash. Some games and machines for playing them. Technical Report D-1164, RAND, February 1952.
21. Kohei Noshita. Union-connections and a simple readable winning way in 7×7 Hex. *Proceedings of 9th Game Programming Workshop in Japan*, pages 72–79, 2004.
22. Kohei Noshita. Union-connections and straightforward winning strategies in Hex. *ICGA Journal*, 28(1):3–12, 2005.
23. Rune K. Rasmussen, Frédéric D. Maire, and Ross F. Hayward. A template matching table for speeding-up game-tree searches for Hex. In Mehmet A. Orgun and John Thornton, editors, *Australian Conference on Artificial Intelligence*, volume 4830 of *Lecture Notes in Computer Science*, pages 283–292. Springer, 2007.
24. Stefan Reisch. Hex ist PSPACE-vollständig. *Acta Informatica*, 15:167–191, 1981.
25. Jack van Rijswijck. Computer Hex: Are bees better than fruitflies? Master's thesis, University of Alberta, Edmonton, Alberta, Canada, 2000.
26. Jing Yang. Jing Yang's web site. `www.ee.umanitoba.ca/~jingyang/`, 2003-2008.
27. Jing Yang, Simon Liao, and Mirek Pawlak. A decomposition method for finding solution in game Hex 7x7. In Cyril Tse Ning, editor, *ADCOG*, pages 96–111. City University of Hong Kong, 2001.
28. Jing Yang, Simon Liao, and Mirek Pawlak. New winning and losing positions for 7×7 Hex. In Jonathan Schaeffer, Martin Müller, and Yngvi Björnsson, editors, *Computers and Games*, volume 2883 of *Lecture Notes in Computer Science*, pages 230–248. Springer, 2002.