Hex: Passing on the Torch

Philip Henderson

November 3, 2010

1 Introduction

This document is intended to help those contributing to the University of Alberta's Computing Science Hex research group. Whether you are developing new code, trying to understand the theory, or simply applying the software to analyze Hex positions, this document will direct you to the important resources and help you on your way. Best of luck!

2 Installing the Software

There are two major units of the Hex project: the underlying engine (Benzene) and the graphical user interface (HexGui).

2.1 Benzene

The Hex engine builds on Fuego, the code base for the Monte Carlo Tree Search (MCTS) Go player built by the University of Alberta's computer Go research group headed by Martin Müller. Thus you must download and compile both Fuego and Benzene in order to use Benzene. Fuego and Benzene require that the following software be installed:

- svn (1.6.6 or greater)
- git (1.7.0.4 or greater)
- Boost (1.33 or greater)
- Berkeley DB (4.8 or greater)
  - libdb4.6
  - db4.6-util
  - db4.7-doc
  - db4.7-util
  - libdb4.8-dev
  - libdb4.8-doc

Checkout Fuego via "svn co https://fuego.svn.sourceforge.net/svnroot/fuego/trunk fuegoDir", where fuegoDir is the directory you want it checked out to.

Checkout Benzene via "git clone git://benzene.git.sourceforge.net/gitroot/benzene/benzene benzeneDir", where benzeneDir is the directory you want it checked out to. If you're going to be committing code, use "git clone ssh://username@benzene.git.sourceforge.net/gitroot/benzene/benzene benzeneDir" instead (where username is your sourceforge user name), but for this you need permission to alter the sourceforge code (right now Broderick holds the keys to this).

Once you have both checked out, there is a README file in benzeneDir that gives instructions on how to compile both. Please pay attention to the following:

1. The Fuego and Benzene versions have to be compatible.
2. Assertions must be on in Fuego if you want them on Benzene. Assertions reduce speed and break multithreaded MoHex, but I recommend them if you will be doing significant development and/or testing.
3. If you want to work on Hex boards larger than 11x11, you have to use the appropriate compiler options.
4. Compilation of both programs is quite slow (especially Fuego), so I recommend using the multithreaded make with as many threads as you have available (e.g. "make -j4").

2.2 HexGui

The interface can be used with or without the Benzene engine. However, without the engine, one is limited to rather simple tasks: setting up positions, playing games against a human, opening up Hex .sgf files, etc; basically, using it as an electronic Hex board. By connecting Benzene, one can play against the computer, solve positions, perform inferior cell analysis, identify connection strategies, and so on. HexGui requires that the following software be installed:

- java (1.6.0_18 or greater)
- svn (1.6.6 or greater) if you will be editing the HexGui code

If you are only going to be using HexGui (i.e. not coding new features), you can download it from http://webdocs.cs.ualberta.ca/~broderic/hex/. This should be transferred to a more permanent site in the near future; probably sourceforge, the UofA Hex site, or Ryan Hayward's personal webpages.

Once you have downloaded hexgui-0.9.0.zip, move it to where you want your HexGui directory (e.g. your home directory) and just do "unzip hexgui-0.9.0.zip". The README file will tell you how to compile and run; it's quite simple (do not worry about the warning messages).

If instead you want to commit changes to the HexGui code, then you need to checkout HexGui via "svn co file:///usr/tees1/cshome/broderic/svnroot/hexgui/trunk/ hexguiDir".

2.3 Six

Six is not part of our code base, but it was the gold medallist for several years, and it is still a decent player. It is open source, and can be downloaded from its website: http://six.retes.hu/. However, since Six's command interface does not match ours, we modified its interface slightly, and you can download this modified version using "git clone ~broderic/git/six.git". As always, instructions on how to build it are in the README file; note that this modified version requires Fuego and Benzene in order to compile.

3 Applying the Software

3.1 Benzene Executables

Compiling benzene produces two executables (within your benzeneDir): src/wolve/wolve and src/mohex/mohex. Note that the solvers and analytic tools are available in both executables, so you should only choose between them based on which player you wish to use, alpha-beta Wolve or Monte Carlo tree search MoHex.

3.2 Benzene Commands

When running these executables in the terminal, the following Hex Text Protocol (htp) commands are available:

- quit: Ends the program.
- boardsize N M: Changes the board size to N x M, where N and M are positive integers (recall that these integers are bounded according to Benzene's compiler options). If M is omitted, changes the board size to N x N.
- showboard: Displays current board position.
- clear_board: Clears the board of all stones and starts a new game.
- play COLOUR CELL: Plays a COLOUR stone at the given cell, where COLOR is either "black" or "white" (shorthand "b" and "w" also works) and CELL is of the form "a1" (i.e. column letter followed by row number, with no space between the two).
- genmove COLOUR: Generates a move (using whichever player corresponds to the program you are running) for the COLOUR specified.
- dfpn-solve-state COLOUR: solves the current position, with COLOUR indicating the player to move.
- vc-build COLOUR: Computes inferior cells and connection strategies for the current state, and indicates the mustplay for the COLOUR specified.
- param_mohex/param_wolve: Only one of these commands will be available, depending on which executable you are running. This command lists the settings of the corresponding player. You can use these parameters to adjust the player.
- param_player_vc: Lists the player's H-search parameters.
- param_player_ice: Lists the player's inferior cell analysis parameters.
- param_dfpn, param_dfs, param_solver_vc, param_solver_ice: Lists the parameters for the DFPN/DFS solvers and their H-search and inferior cell analysis engines.

This is by no means a complete list, but will definitely give you enough tools to start off with.

I should also note that the DFS solver has not been maintained, and is broken at this time (due to changes in inferior cell analysis, I believe). On average the DFS solver is significantly slower than the DFPN solver, which is why it hasn't been a high priority to keep it maintained.

3.3 Benzene Tournaments

As you develop new versions of the players, you will likely want to test them against each other to see whether you have improved their playing strength. Of course Wolve and MoHex are available for testing and, if you downloaded our modified version of Six (which uses our htp interface), then you can use that program in the tournaments as well.

First of all, for each program you should prepare a .htp file which determines its (non-default) settings. For instance, here is the Wolve-CG2010.htp file:
param_game allow_swap 1
param_wolve panic_time 0
param_wolve ply_width "15 15 15 15"
param_wolve use_parallel_solver 1

Once you have a .htp file for each player you wish to test, follow the instructions found in the HOWTO file within your benzeneDir/tournament directory.

3.4 Benzene Regression Tests

In your benzeneDir/regression directory, there are several regression tests that can be used to test the solvers, players, and analytic tools (such as inferior cell analysis and connection strategy computation). The instructions are in that directory's README file, but here are a couple examples to clarify:

- time ./run.sh -l dfpn-solve-7x7.tst: Solves all 7x7 openings with the DFPN solver, indicating when each opening is solved and reporting the total time at the end.
- ./run.sh -p "../src/wolve/wolve –config settings-vc.htp" vc.tst: Tests the strength of the connection strategy computation algorithms when using the settings specified in settings-vc.htp, only indicating unexpected passes and failures.

Note that the detailed output of such tests can be viewed in the html/ subdirectory (the latest results for a particular .tst file overwrite any previous such results).

3.5 HexGui

The interface is reasonably self-explanatory (although the code is not in a clean, finalized state). Menu items such as printing, time control, edit preferences and help are pretty useless at this time, so you will probably use the following functionality:

1. Open Hex sgf files.
2. Save a game or position (the latter only contains one state - the current stones - rather than a sequence of moves).
3. Create and connect programs (details below).
4. Edit the board size. Recall that benzene must be compiled with certain options if you want to apply its tools on boards larger than 11x11.
5. Change the view's board type/orientation (standard is flat and Black on top).

To connect Benzene's executables to the GUI, do the following:

1. Program, New Program: Enter the name you wish to use for this program, and the relative path from the hexgui executable to the wolve/mohex executables in your benzene directory. For instance, the command "../benzene/src/wolve/wolve" with a name of "Default Wolve" and no entry for the working directory. You should probably have at least one entry for Wolve and one entry for MoHex. This process only needs to be done once.

2. Program, Edit Program: Edit one of the above entries that you have defined. For instance, you can add " –config wolveSettings.htp" in the command if you do not want to use default settings. At the very least, I recommend turning on the GUI effects for both players and the DFPN solver via "param_wolve use_guifx 1", "param_mohex use_livegfx 1", and "param_dfpn use_guifx 1". Note that any such .htp files must be in your hexguiDir directory.

3. Program, Connect Local Program: Connects to the benzene executable that you specified. You can then do everything you could in the terminal, including generating moves and using its analytic tools (DFPN solver, inferior cell analysis, connection strategies, etc). Note that all commands are listed in the AnalyzeDialog.

4 Background Reading

I am not going to explain any Hex theory or algorithms here, but simply direct you to the resources I found to be most useful. Resources are organized by topic, and within each topic are ordered from basic to more advanced. A complete bibliography of all Hex resources I know of (as of November 2010) is available from Ryan Hayward and/or the UofA Hex website.

4.1 Overview

The basics of Hex, including rules, monotonicity, the no-draw property, strategy-stealing argument, PSPACE-completeness, and so on. Note that I do not list Reisch's proof of PSPACE-completeness, not because this isn't an important result, but because I have not yet found a decent translation.

1. Martin Gardner. Mathematical games. Scientific American articles on Hex, 1957.

2. Bert Enderton. Answers to infrequently asked questions about the game of Hex. http://www.cs.cmu.edu/∼hde/hex/hexfaq/, 1995.

3. Anatole Beck, Michael N. Bleicher, and Donald W. Crowe. Excursions into Mathematics: the Millennium Edition, chapter 5 (pages 327-339) and Appendix 2000. A.K. Peters, Natick, Massachusetts, 2000.

4. Craige Schensted and Charles Titus. Mudcrack Y and Poly-Y. Neo Press, Peaks Island, Maine, 1975.

5. Shimon Even and R. Endre Tarjan. A combinatorial problem which is complete in polynomial space. Journal of the Association for Computing Machinery, 23(4):710-719, 1976.

4.2 Inferior Cell Analysis

Dead cells, captured cells, vulnerable/dead-reversible, captured-reversible, various forms of domination (capture, induced path, neighbour, etc), decompositions (split, captured, star), and so on.

1. Craige Schensted and Charles Titus. Mudcrack Y and Poly-Y. Neo Press, Peaks Island, Maine, 1975.

2. Ryan B. Hayward. A note on domination in Hex. Technical report, University of Alberta, 2003.

3. Yngvi Björnsson, Ryan Hayward, Michael Johanson, and Jack van Rijswijck. Dead cell analysis in Hex and the Shannon game. In Adrian Bondy, Jean Fonlupt, Jean-Luc Fouquet, Jean-Claude Fournier, and Jorge L. Ramirez Alfonsin, editors, Graph Theory in Paris: Proceedings of a Conference in Memory of Claude Berge, pages 45-60. Birkhauser, 2007.

4. Philip Henderson. Playing and Solving the Game of Hex. PhD thesis, University of Alberta, Edmonton, Alberta, Canada, 2010. Chapters 2 and 3.

5. Elwyn Berlekamp, John H. Conway, and Richard K. Guy. Winning Ways for Your Mathematical Plays, volume 1-4. A.K. Peters, 2nd edition, 2000.

6. Jack van Rijswijck. Set Colouring Games. PhD thesis, University of Alberta, Edmonton, Alberta, Canada, 2006.

4.3 Connection Strategies

First and second-player connection strategies, common templates, the H-search algorithm, and augmentations of the H-search algorithm.

1. David King. Hall of hexagons - the game of Hex.
http://www.drking.org.uk/hexagons/hex/templates.html, 2007.

2. Cameron Browne. Hex Strategy: Making the Right Connections. A.K. Peters, Natick, Massachusetts, 2000. Chapters 4 and 5.

3. Jing Yang, Simon Liao, and Mirek Pawlak. A decomposition method for finding solution in game Hex 7x7. In Ning [130], pages 96-111.

4. Vadim V. Anshelevich. The game of Hex: An automatic theorem proving approach to game programming. In AAAI2000 [1], pages 189-194.

5. Rune K. Rasmussen, Frederic D. Maire, and Ross F. Hayward. A template matching table for speeding-up game-tree searches for Hex. In Orgun and Thornton [136], pages 283-292.

6. Philip Henderson. Playing and Solving the Game of Hex. PhD thesis, University of Alberta, Edmonton, Alberta, Canada, 2010. Chapters 2 and 4.

4.4 Solvers

1. Ryan Hayward, Yngvi Björnsson, Michael Johanson, Morgan Kan, Nathan Po, and Jack van Rijswijck. Solving 7×7 Hex with domination, fill-in, and virtual connections. Theoretical Computer Science, 349(2):123-139, 2005.

2. L. Victor Allis. Searching for Solutions in Games and Artificial Intelligence. PhD thesis, University of Limburg, Maastricht, Netherlands, 1994.

3. Philip Henderson. Playing and Solving the Game of Hex. PhD thesis, University of Alberta, Edmonton, Alberta, Canada, 2010. Chapters 2 and 5.

4.5 Players

1. Thomas R. Lincke. Strategies for the automatic construction of opening books. In Marsland and Frank [117], pages 74-86.

2. Claude E. Shannon. Computers and automata. Proceedings of the Institute of Radio Engineers, 41:1234-1241, 1953.

3. Vadim V. Anshelevich. Hexy's home page. home.earthlink.net/∼vanshel/.

4. Gabor Melis. Six. six.retes.hu/, 2006.

5. S. Gelly, Y. Wang, R. Munos, and O. Teytaud. Modification of UCT with patterns in Monte-Carlo Go, 2006. Technical Report RR-6062.

6. Sylvain Gelly and David Silver. Combining online and offline knowledge in UCT. In Ghahramani [74], pages 273-280.

7. Jack van Rijswijck. Search and evaluation in Hex. Technical report, University of Alberta, Edmonton, Canada, 2002.

8. Philip Henderson. Playing and Solving the Game of Hex. PhD thesis, University of Alberta, Edmonton, Alberta, Canada, 2010. Chapters 2 and 6.

9. Levente Kocsis and Csaba Szepesvari. Bandit based monte-carlo planning. In Furnkranz et al. [58], pages 282-293.

5 Conclusion

So, now that you have come this far, maybe you are looking for a summer or thesis project relating to Hex. There is a thorough list of open problems appearing in Appendix D of Philip Henderson's doctoral thesis, so that's a very good place to start. Below I list a couple additional projects that do not appear in that appendix:

1. Implement star decomposition detection, and test its effect on player and solver performance.

2. Implement common miai substrategy in the H-search algorithm, and test its effect on player and solver performance.

3. Use graph-theoretic/topographical equivalence between solved and explored Hex states to improve the DFPN solver.

4. Extend Hex inferior cell analysis to other games, like Y and Havannah.

This should be more than enough material to get started. Enjoy!