The University of Sydney

HexGraph
Applying Graph Drawing to the Game of Hex

Technical Report Number 540

September, 2003

Colin Murray, Carsten Friedrich, and Peter Eades

School of Information Technologies
University of Sydney NSW 2006
ISBN 1 86487 586 0

HexGraph
Applying Graph Drawing Algorithms to the Game of Hex

Colin Murray, Carsten Friedrich, and Peter Eades

School of Information Technologies
The University of Sydney
Australia
{cmurray,carsten,peter}@it.usyd.edu.au

Abstract Hex is a classic board game for two players. There exists an intuitive mapping of the state of a hex game onto a graph. The motivation of the work presented in this paper was to create a new user interface paradigm for playing Hex, which is based on the graph representation of the state of the game, and which increases the player's ability to find good moves/strategies. The paradigm uses graph reduction, graph drawing, and graph animation techniques. The HexGraph paradigm has been implemented and is available from http://www.it.usyd.edu.au/~carsten/hex.

1 Introduction

Hex is a two player board game which is traditionally played on a rhombic hexagonal pattern, most commonly consisting of 11x11 fields (See Figure 1). Players are assigned a colour (eg red or blue) and make moves by putting a token of their colour onto an empty field on the board. The first player to connect the two borders of the board in his colour by a path of his tokens on the board wins the game. Hex has the further property that there is always one winner; the game cannot end in a draw.

There exists an intuitive mapping of the state of a Hex game onto an undirected graph [7]:

- The fields of the board are represented by nodes. Nodes can be empty, red, or blue.
- Adjacent fields on the board are connected by an edge.
- The four borders of the board are represented by one node of equivalent colour each. Each of the four nodes is connected to all nodes which represent the fields adjacent to the equivalent border.

The game plays similarly with the graph representation. The players take turns colouring one of the nodes with their colour. The Red player attempts to build a path of red coloured nodes, trying connecting the two red border nodes, s and t (See Figure 2); while the Blue player tries to connect the blue border nodes with blue coloured nodes (not shown in Figure 2).

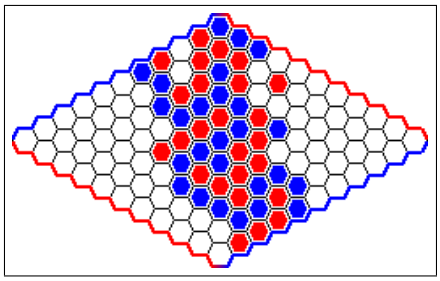


Figure 1. Traditional Hex Board. The blue player wins as he was able to connect the two blue borders with a blue path on the board. Source http://www.cs.ualberta.ca/~javhar/hex/.

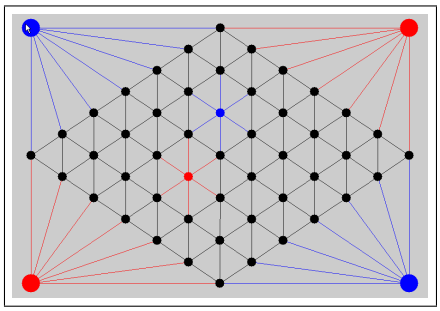


Figure 2. Graph representation of a Hex game.

Making a move can have several implications to the graph. We make the following observations:

- Edges to nodes held by the opponent are obsolete and can be removed from the graph.
- Adjacent nodes of the same colour can be contracted to a single node.
- One-connected components are obsolete, as they can not be part of a shortest winning path, and can be removed from the graph.
- Bi-connected components are not obsolete, but can easily be blocked by the opponent at the end points of the components.

We can use these observations to reduce the graph and thus the complexity of the board, making it easier to play a winning strategy. By simply merging nodes, the problem arises that edges can cross, making it hard to see which nodes are connected. The underlying graph, however, is still planar. This is illustrated in Figure 3.

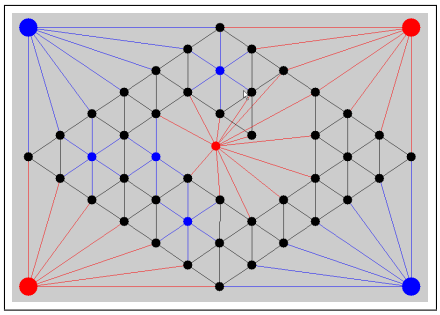


Figure 3. Merging nodes can lead to edge crossings.

2 Graph Drawing Algorithms and Features

In the HexGraph paradigm, we use planar and embedding-preserving graph drawing algorithms, in combination with the proposed graph reductions, to create a user interface for the game. We briefly introduce the used algorithms in the following.

2.1 Tutte Algorithm (Barycentre Method)

Planar drawings make it easy to understand the structure of a given graph by eliminating edge crossings which otherwise can easily be mistaken for additional nodes. Eliminating edge crossings in the hex graph thus helps to ensure that the player understands the state of the game and should improve the player's ability to play a winning strategy.

The Tutte algorithm [5,6] is a well known and relatively simple algorithm for drawing planar graphs. It computes a graph drawing by placing all nodes in the barycentre of their adjacent nodes. This can be easily done by formulating and solving the corresponding linear equations. Given a tri-connected, planar graph and a fixed, convex outer face, this method produces a planar drawing of the graph. Although the theoretic runtime of $O(n^3)$ is comparatively high for a planar graph drawing algorithm, in practice, due to the small size of the graph, runtime is not an issue.

However, there are a number of other problems with this approach:

- In extreme cases the drawing area can become exponential. With a fixed outer face, this means that nodes can end up very close to each other.
- Angular resolution can be very poor.
- In some cases our graph is no longer tri-connected which results in collapsing the bi-connected component onto a line.

Although it can be argued that bi-connected components are not very promising anyway (as they can easily be blocked by the opponent) these deficits render the Tutte algorithm, on its own, inappropriate to display the game.

2.2 Force-Directed PrEd

In 1999, Bertault introduced a force directed graph drawing algorithm [1] which preserves the edge crossings properties of an initial drawing [2]. The algorithm works like a classical force-directed algorithm with one main difference: for each node, the maximum amplitude of a move is restricted to stay within a zone that is calculated such that edge-crossing properties are preserved. The algorithm thus can improve symmetry and uniformity of edge lengths, while eg preserving the planarity of a drawing.

Each round of the force simulation consists of three main steps: Computation of the forces applied to each node, computation of the values of the zone of each node, and the move of each node, with the amplitude of the move bounded by its zone.

The time complexity of the Force-Directed PrEd, $O(|V|^2 + |V||E|)$ per iteration, is rather bad. The efficiency of the algorithm could be improved with a preprocessing step: instead of considering all edges of the graph when defining the zone of a node, we could consider only a restricted set of edges that surround that node. The preprocessing step would determine which edges surround each node. For small graphs, like the one representing the commonly used 11x11 board, however, such an effort seems not to be necessary.

As the algorithm preserves edge crossing properties, it obviously does not remove edge crossings that are introduced by the merging of nodes in our hex graph. However, if we already have a planar drawing of our hex graph, we can use the PrEd Force-Directed algorithm to improve that drawing:

2.3 Tutte combined with Force-Directed PrEd

The best results are achieved by combining the Tutte and the Force-Directed PrEd algorithms. The edge-crossings are removed by the Tutte algorithm and then the force-directed PrEd algorithm improves symmetry and uniformity of edge lengths without introducing any edge crossings.

2.4 Making a move - changing the graph

While playing, the player builds a mental map [4]of the board. Being able to maintain this mental map between moves is essential for developing and executing a specific strategy. This mandates that the graph drawing algorithm must not make drastic, unintuitive changes to the drawing. In HexGame we use graph animation [3] to show the transition from the initial graph drawing to the final graph drawing to help the user maintaining the mental map of the game. As neither the Tutte algorithms nor the force-directed PrEd algorithm makes dramatic changes to the initial graph drawing, with the aid of animation, the user can easily follow the transition from the initial graph drawing to the final graph drawing. This ensures that the mental map can be maintained.

3 Implementation

We implemented the HexGraph paradigm in a prototypical Hex application which is available from http://www.it.usyd.edu.au/~carsten/hex. The implementation contains a classic board interface and the HexGraph interface. The program allows the user to play against the computer or against another human player, either on the same board or over a network.

4 Conclusions

In our experience, the HexGraph paradigm provides great potential to increase a player's ability to find good moves and strategies in a game of hex. Representing a Hex board as a graph is intuitive, and HexGraph provides balanced, planar drawings of Hex-graphs through a combination of the Tutte and the force-directed PrEd algorithm. The mental map of the user is preserved between moves by animating the transitions between graph drawings.

There remain some properties of good graph drawings which could still be improved: Distances between vertices and non-incident edges, and the angular resolution of a graph drawing are important characteristics. Specific rules governing these properties are not imposed by either of the used algorithms.

Another problem results from the special role of the four nodes which represent the border of the board. The positions of the border nodes are fixed and all other nodes should lie within the box defined by these nodes. Therefore, the maximum angular resolution of edges adjacent to the border nodes is only a quarter of the maximum resolution for other nodes. This problem is further heightened by the fact that the border nodes have a particular large amount of adjacent edges, which tends to increases even further during the game. The fact that the fixed nodes gain so many neighbours combined with the constraints provides the biggest challenge in drawing the Hex game graph.

We have introduced a system that provides many of the desirable features of a drawing of Hex game graphs, therefore contributing to a player's ability to play winning moves and strategies. This paper also introduces the hex game graph as a challenging graph drawing problem.

References

1. Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph drawing: algorithms for the visualization of graphs*. Prentice-Hall Inc., 1999. 2.2
2. F. Bertault. A force-directed algorithm that preserves edge-crossing properties. *Information Processing Letters*, 74(1–2):7–13, 2000. 2.2
3. Carsten Friedrich. *Animation in Relational Information Visualization*. Phd thesis, University of Sydney, 2002. 2.4
4. P. Gould and R. White. *Mental Maps*. Allen and Unwin Inc., Winchester, Mass. 01890, USA, 1986. 2.4
5. W. T. Tutte. Convex representations of graphs. *Proc. London Math. Soc.*, 10:304–320, 1960. 2.1
6. W. T. Tutte. How to draw a graph. *Proc. London Math Soc.*, 13:743–768, 1963. 2.1
7. Jack van Rijswijck. Search and evaluation in Hex. Techreport, University of Alberta. 1

A Picture Gallery

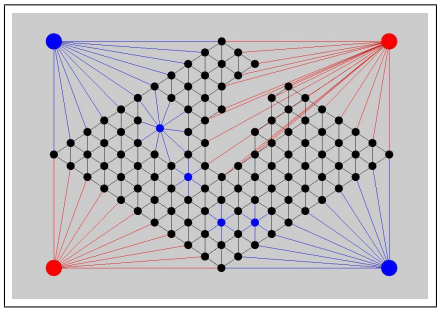
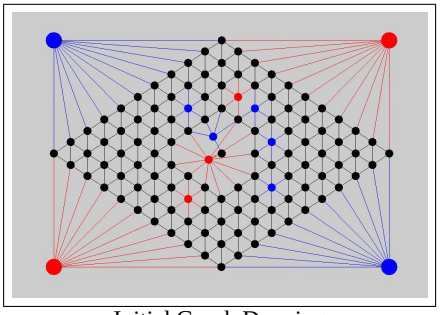
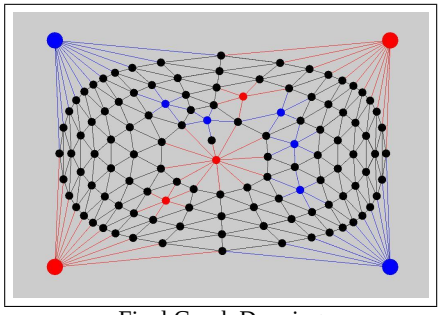


Figure 4. The result of not applying an algorithm - many edge crossings.

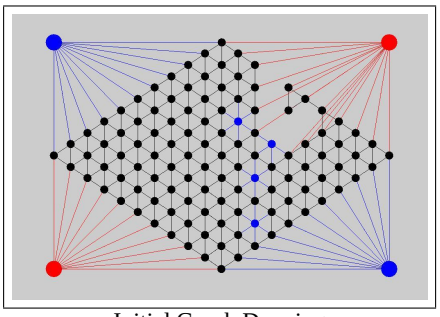


Initial Graph Drawing

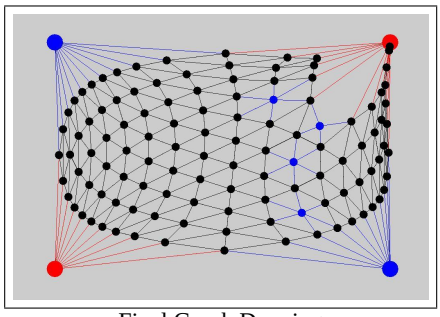


Final Graph Drawing

Figure 5. A case where the Tutte algorithm produces good results.

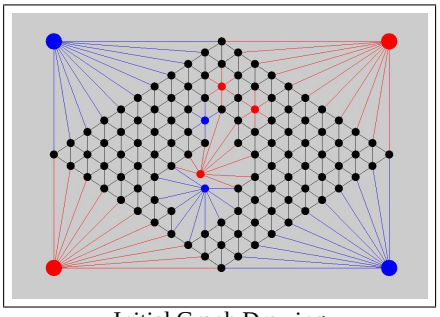


Initial Graph Drawing

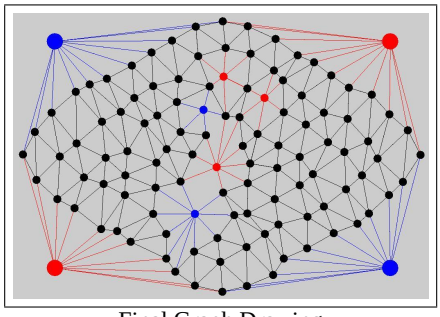


Final Graph Drawing

Figure 6. A case where the Tutte algorithm produces poor results.

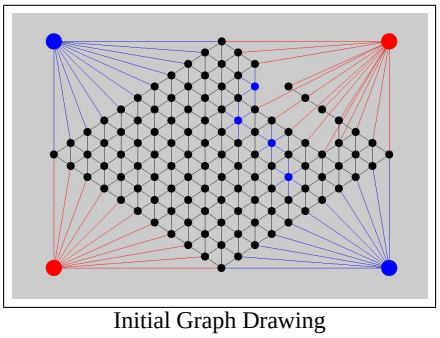


Initial Graph Drawing

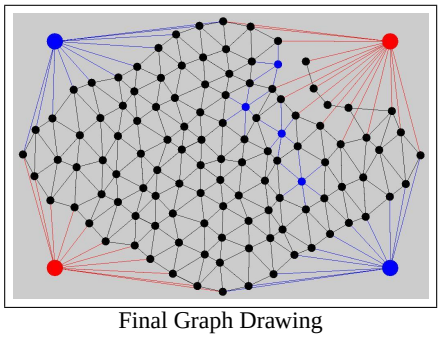


Final Graph Drawing

Figure 7. Force-Directed PrEd improves the drawing by making edge lengths more uniform.

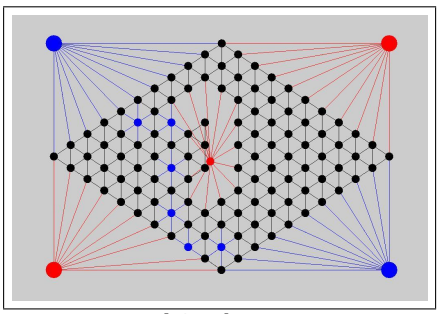


Initial Graph Drawing

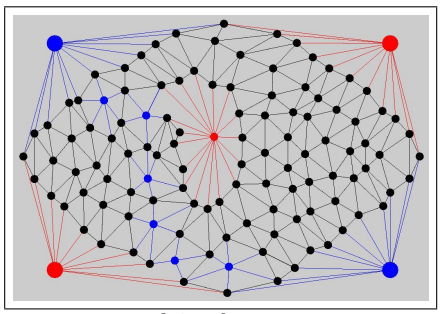


Final Graph Drawing

Figure 8. Force-Directed PrEd doesn't remove edge crossings but makes them less confusing.

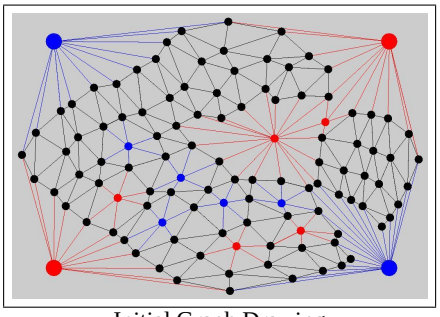


Initial Graph Drawing

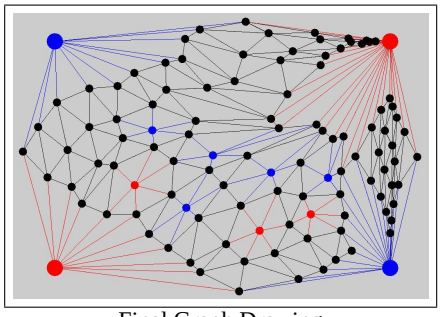


Final Graph Drawing

Figure 9. A case where Tutte + Force-Directed PrEd works well.

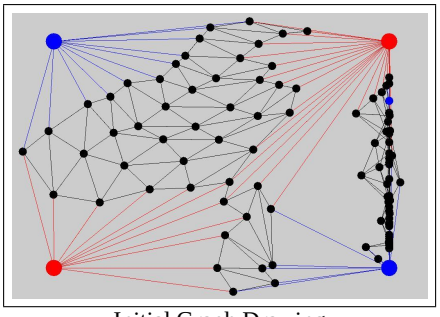


Initial Graph Drawing

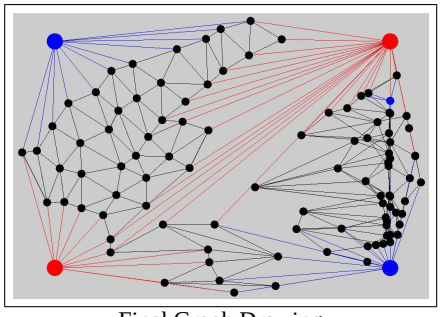


Final Graph Drawing

Figure 10. A case where Tutte + Force-Directed PrEd works poorly. This is largely due to poor performance of the Tutte algorithm.



Initial Graph Drawing



Final Graph Drawing

Figure 11. The initial drawing is poor due to the bunching up of the Tutte algorithm. Applying many more iterations of Force-Directed PrEd makes improvements as seen in the final drawing.