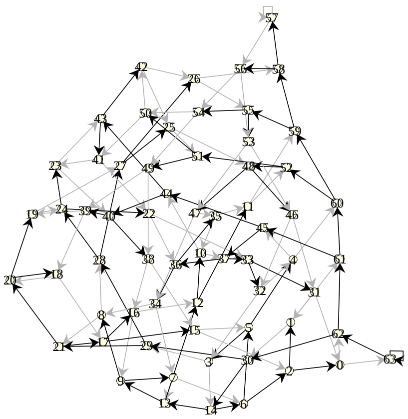
# Hunting Down Graph B[*]

Ulrik Brandes

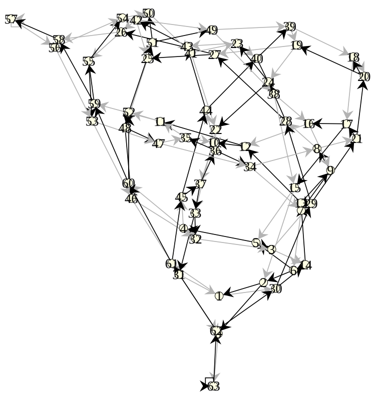
Department of Computer Science, Brown University
Providence, Rhode Island 02912-1910
Ulrik.Brandes@uni-konstanz.de

**Abstract.** Graph B, the "Theory Graph", of the 1999 Graph Drawing Contest is visually explored using available and some hand-written graph drawing software.

This year's "Theory Graph" was given in GML-format.[1] It is a directed graph, given without coordinates, but with a suspicious-looking vertex numbering (non-decreasing order, range 15070–15259, constant increment of 3) and edges colored either in green or in blue.[2] So let's use a simple spring embedder (Fruchterman/Reingold's variant [1] as implemented in LEDA's [2] graph editor GraphWin) to get a first impression of the graph's structural characteristics:



2D spring embedding



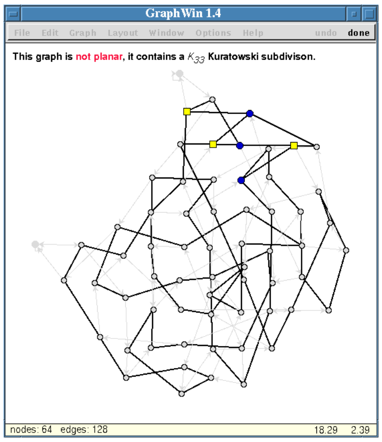
3D spring embedding

No revelation, though fairly symmetric in 3D. Inspection shows that the graph is sparse (64 vertices, 128 edges – hmm, powers of 2), but not planar.

[*] Die Arbeit wurde mit Unterstützung eines Stipendiums im Rahmen des Gemeinsamen Hochschulsonderprogramms III von Bund und Ländern über den DAAD ermöglicht.

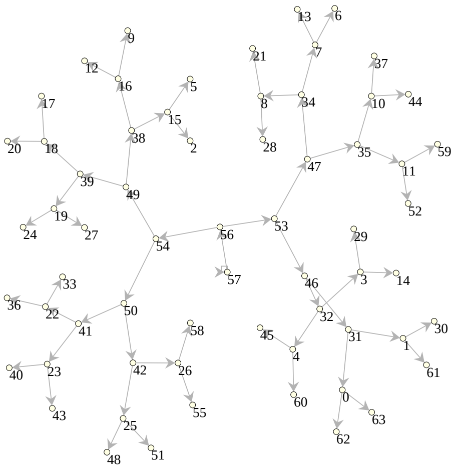
[1] GML homepage http://www.fmi.uni-passau.de/~himsolt/Graphlet/GML/.

[2] Here represented by gray and black, respectively.

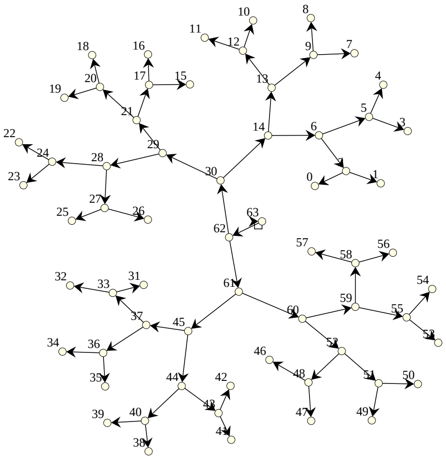



Moreover, it appears to be 4-regular, with each vertex incident to two incoming edges of different color, and two outgoing edges of the same color. There seem to be only two exceptions to this observation (due to a green and a blue loop).

Since it is to be expected that the competition graph is obfuscated, we simply index vertices according the order in which they are given (a default labeling option in GraphWin) and take a look at the green and blue edge-induced subgraphs:



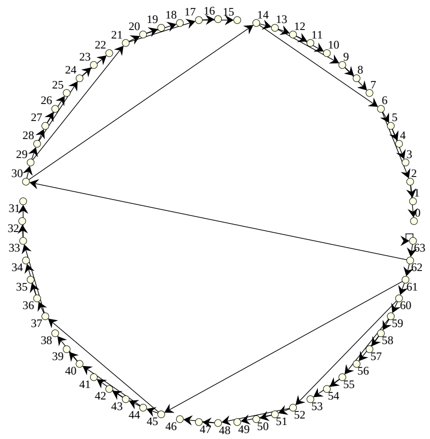
subgraph induced by green edges



subgraph induced by blue edges

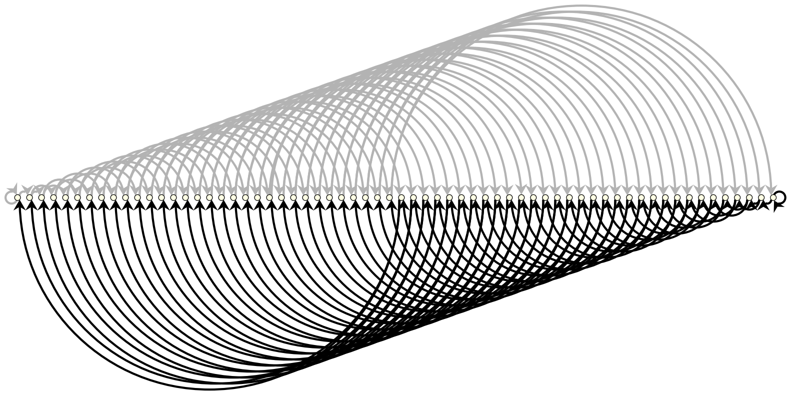
Surprise! The powers of two turn out not to be incidental. These layouts indeed show that the graph consists of two uni-colored complete binary trees with an additional parent of the root. And the order of the vertices was obviously produced by a post-order traversal of the blue tree. But why the additional vertex with a loop?

With a computer science background it is unavoidable to think of binary trees as being labeled with `0`/`1`-strings, where the left and right child of an internal node are labeled by appending `0` and `1`, respectively, to the parent's label. Now it takes only modest intuition to replace the post-order numbering with its binary representation and to hypothesize that climbing down the tree could correspond to appending `0`s and `1`s, while dropping the leading digit. Sound familiar? Note how a circular layout of the blue tree spells




like shift register `;-)`. More serious evidence supporting this hypothesis is that internal nodes of one tree are leafs of the other, and vice versa. The numbering is not quite right (the binary representation of the green root's index should be `000000`, not `111010`), but reordering proves that this is just another obfuscation.[3]

We conclude that Graph B is the transition graph of a 6-bit shift register. Placing vertices equidistantly on a line according to the "appropriate" numbering (i.e. indices corresponding to bit sequences of register states) yields

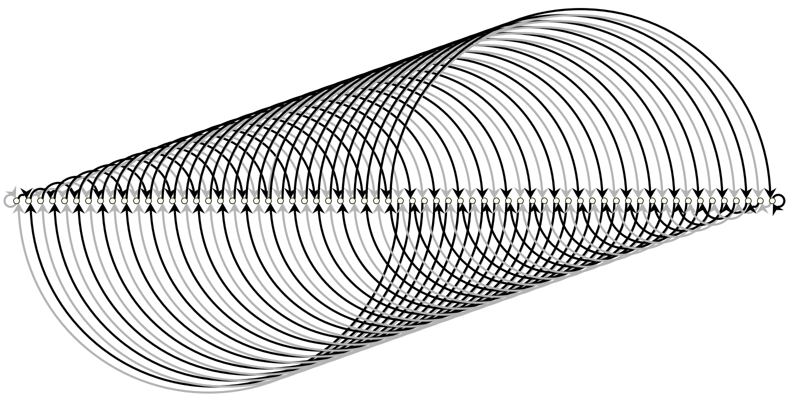


[3] The permutation is $\pi = (31, 30, 47, 29, 28, 46, 55, 27, 26, 45, 25, 24, 44, 54, 59, 23, 22, 43, 21, 20, 42, 53, 19, 18, 41, 17, 16, 40, 52, 58, 61, 15, 14, 39, 13, 12, 38, 51, 11, 10, 37, 9, 8, 36, 50, 57, 7, 6, 35, 5, 4, 34, 49, 3, 2, 33, 1, 0, 32, 48, 56, 60, 62, 63)$.

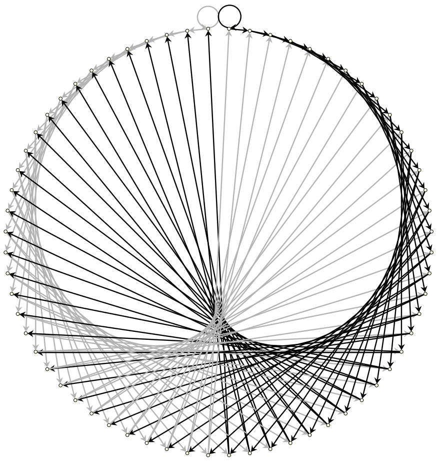
where edges are semicircles with radii half the distance between their incident vertices. Clearly, this drawing is not all that different from the suitably ordered adjacency matrix shown on the right.

```
 0: GG..............................................................
 1: ..GG............................................................
 2: ....GG..........................................................
 3: ......GG........................................................
 4: ........GG......................................................
 5: ..........GG....................................................
 6: ............GG..................................................
 7: ..............GG................................................
 8: ................GG..............................................
 9: ..................GG............................................
10: ....................GG..........................................
11: ......................GG........................................
12: ........................GG......................................
13: ..........................GG....................................
14: ............................GG..................................
15: ..............................GG................................
16: ................................GG..............................
17: ..................................GG............................
18: ....................................GG..........................
19: ......................................GG........................
20: ........................................GG......................
21: ..........................................GG....................
22: ............................................GG..................
23: ..............................................GG................
24: ................................................GG..............
25: ..................................................GG............
26: ....................................................GG..........
27: ......................................................GG........
28: ........................................................GG......
29: ..........................................................GG....
30: ............................................................GG..
31: ..............................................................GG
32: BB..............................................................
33: ..BB............................................................
34: ....BB..........................................................
35: ......BB........................................................
36: ........BB......................................................
37: ..........BB....................................................
38: ............BB..................................................
39: ..............BB................................................
40: ................BB..............................................
41: ..................BB............................................
42: ....................BB..........................................
43: ......................BB........................................
44: ........................BB......................................
45: ..........................BB....................................
46: ............................BB..................................
47: ..............................BB................................
48: ................................BB..............................
49: ..................................BB............................
50: ....................................BB..........................
51: ......................................BB........................
52: ........................................BB......................
53: ..........................................BB....................
54: ............................................BB..................
55: ..............................................BB................
56: ................................................BB..............
57: ..................................................BB............
58: ....................................................BB..........
59: ......................................................BB........
60: ........................................................BB......
61: ..........................................................BB....
62: ............................................................BB..
63: ..............................................................BB
```

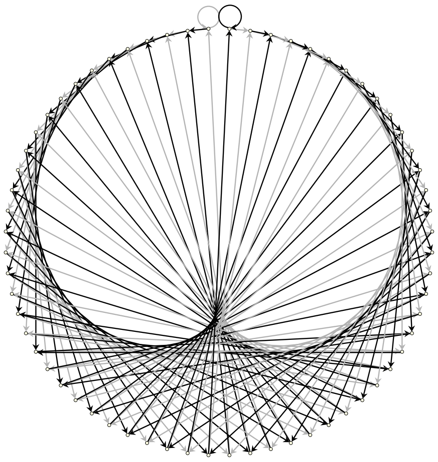
Instead of the coloring edges according to membership in what are essentially the 6-step reachability trees from states `000000` and `111111`, we can use the colors to indicate whether `0` or `1` is inserted as least significant bit:



Circular layouts using this ordering are even nicer,

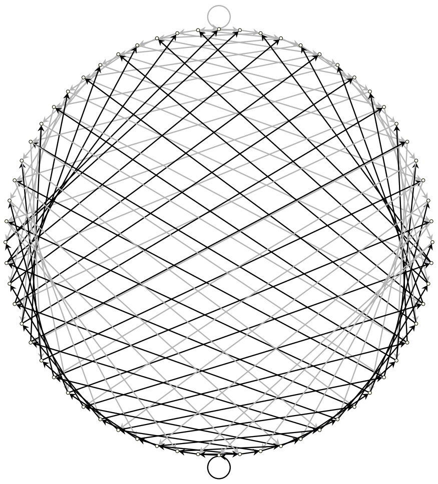


given coloring

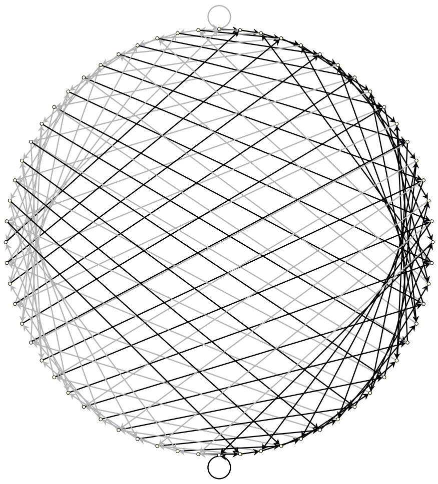


0/1-insertion coloring

but less informative. If, however, we consider states to differ the most, if they differ in every single bit, a more useful convention could be to place 1-complements diametral:

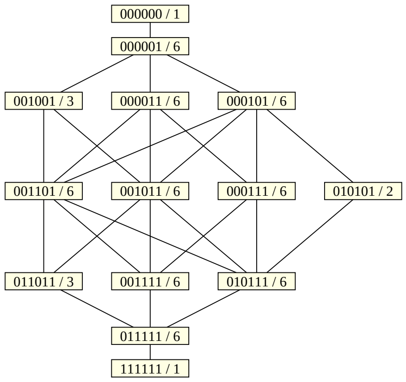


given coloring

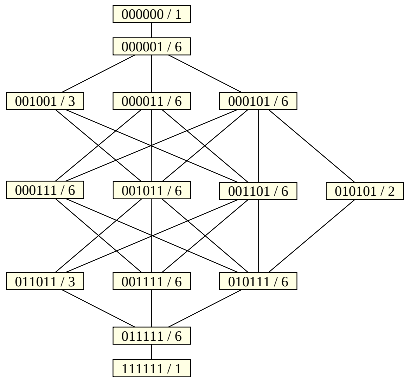


0/1-insertion coloring

The drawing on the right clearly shows that even numbers form the left, while odd numbers constitute the right half of the circle. Finally, it would be interesting to display the remainders under cyclic shifting by placing the vertices of each remainder on a distinct circle. To place the circles, we first consider the graph obtained by contracting the vertices in each remainder and deleting parallel edges. Since the states in each remainder have an invariant number of 0 and 1, there is a natural layering of the remainder states. By applying an instance of the layered layout framework of Sugiyama et al. [4] (as implemented in the AGD library [3]) we obtain




with crossing minimization




manually rearranged to emphasize symmetry

where remainders are labeled with a representative state and the number of states contained. Note the striking resemblance of the initial 3D spring embedding! We leave it to the interested reader to determine a useful ordering of states when remainders are expanded into circles in three dimensions.

## References


1. Thomas M.J. Fruchterman and Edward M. Reingold. Graph-drawing by force-directed placement. *Software—Practice and Experience*, 21(11):1129–1164, 1991.
2. Kurt Mehlhorn and Stefan Näher. *The* LEDA *Platform of Combinatorial and Geometric Computing*. Cambridge University Press, 1999. Project home page at `http://www.mpi-sb.mpg.de/LEDA/`.
3. Petra Mutzel, Carsten Gutwenger, Ralf Brockenauer, Sergej Fialko, Gunnar W. Klau, Michael Krüger, Thomas Ziegler, Stefan Näher, David Alberts, Dirk Ambras, Gunter Koch, Michael Jünger, Christoph Buchheim, and Sebastian Leipert. A library of algorithms for graph drawing. In Sue H. Whitesides, editor, *Proceedings of the 6th International Symposium on Graph Drawing (GD '98)*, volume 1547 of *Lecture Notes in Computer Science*, pages 456–457. Springer, 1998. Project home page at `http://www.mpi-sb.mpg.de/AGD/`.
4. Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man and Cybernetics*, 11(2):109–125, 1981.