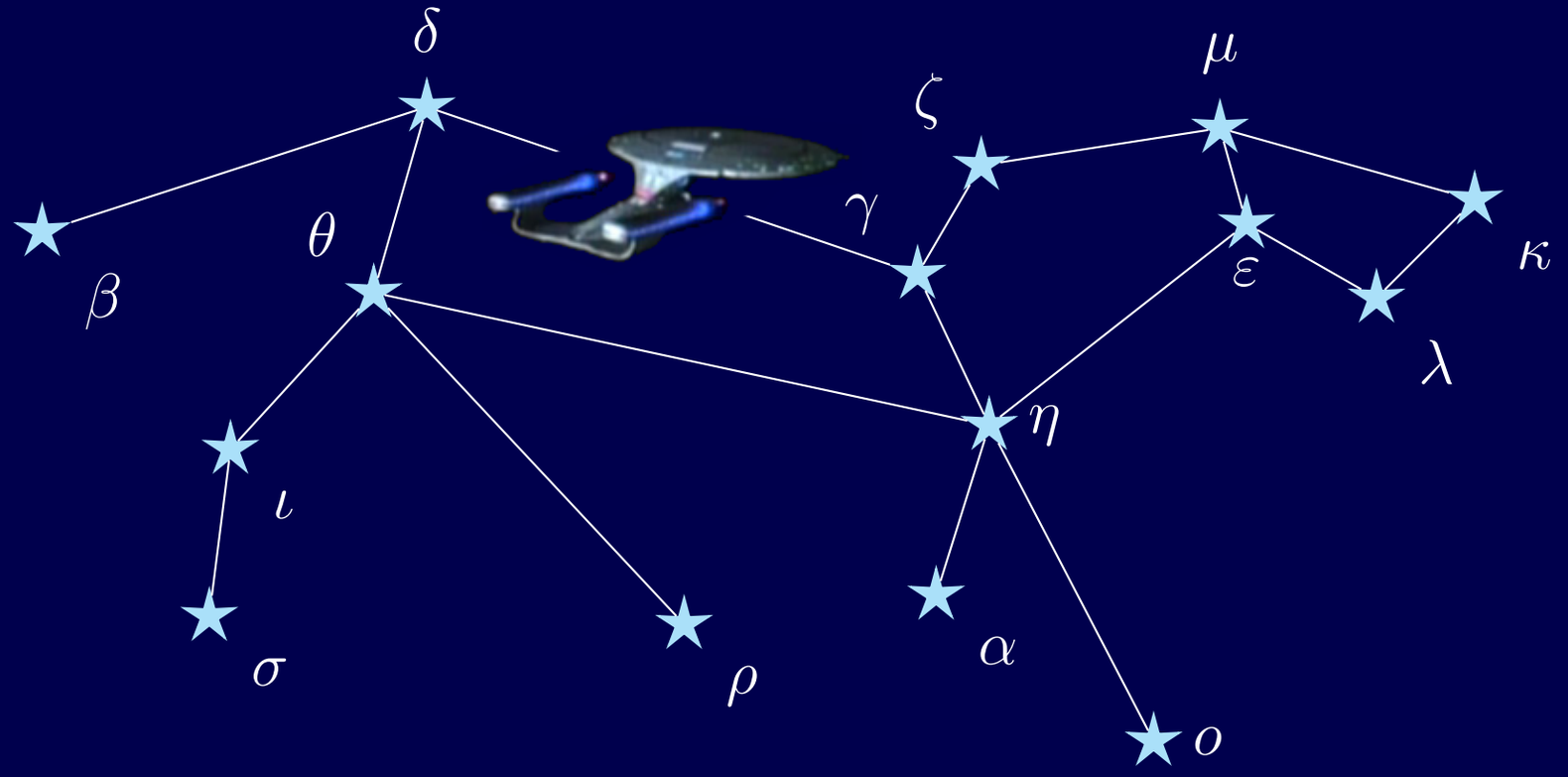
Cycle counting: the next generation



Matthew Skala

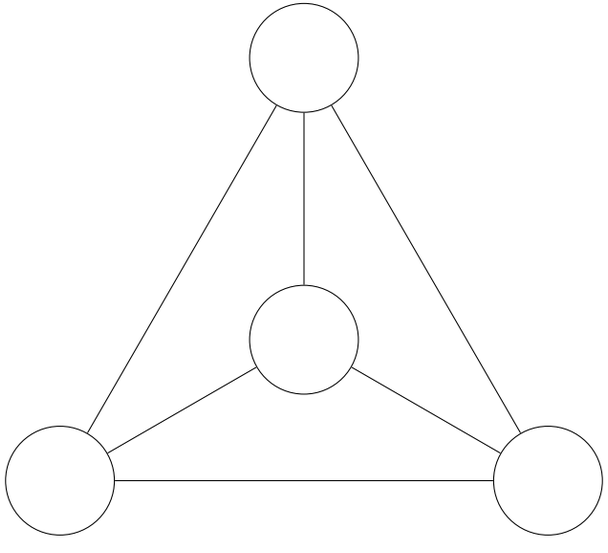
30 January 2013

Outline

- Cycle counting
- ECCHI
- Knight's Tours
- Equivalent circuits
- The next generation

Cycle counting

Let G be a labelled graph. How many distinct H exist such that H is a cycle and is a subgraph of G? (Decision version: accept iff the answer is at least k.)



This is a $\#\mathcal{P}$-complete problem, equivalent in hardness to counting the certificates of an $\mathcal{NP}$-complete problem, even though "is there a cycle?" is not itself $\mathcal{NP}$-complete. We could ask much the same question about Hamiltonian cycles.

Applications for cycle counting

- Reliability.
- Extremal graphs.
- Prototypical $\#\mathcal{P}$-complete graph problem.
- Combinatorics for its own sake.

Backtracking

Do a DFS, backtrack when you detect a cycle, do a little bit of extra handling to deal with the cycle starting point. Run it to the end, you will see every cycle.

This is pretty much optimal if you want to list (not just count) the cycles.

The difficulty: you are doing at least a constant amount of work for every cycle you count.

How many cycles are there?

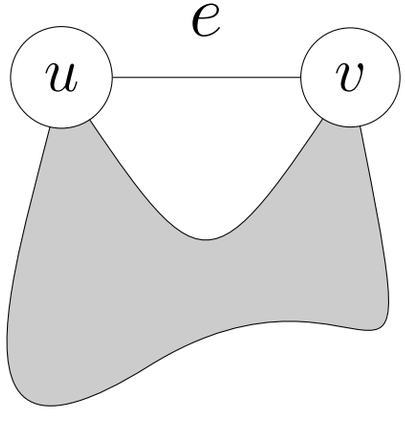
Lots.

For a family of dense graphs with n vertices, often something like $n!$ cycles. "Something like $n!$" means "$\Omega(n!c^{-n})$ for some constant c." Exponentials are *small* compared to factorials!

You can't afford to do anything once per cycle.

A simple induction

Choose an edge $e = (u, v)$. Every cycle must either pass through that edge or not.

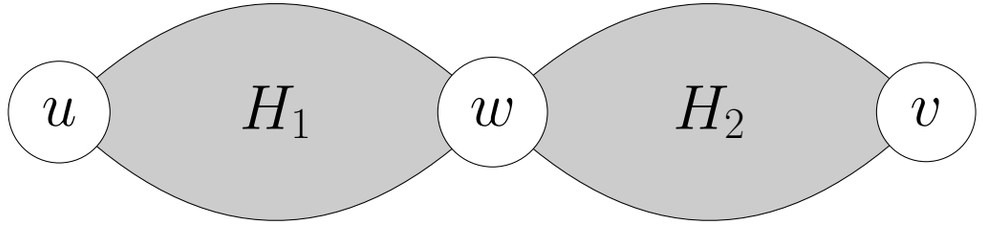


$$c(G) = p(u, v, G \setminus e) + c(G \setminus e)$$

For the "path" subproblem a similar induction applies. But this is just backtracking in disguise.

A different inductive step

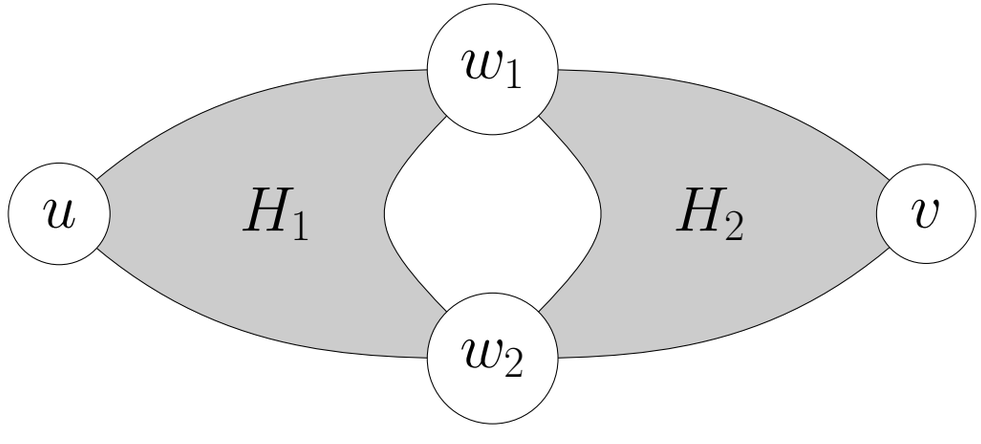
Suppose we are looking for a path from u to v and there is a cut-vertex w between them. Then a path from u to v is any path from u to w followed by any path from w to v, and we can multiply.



$$p(u, v, G) = p(u, w, H_1) \cdot p(w, v, H_2)$$

Yet another inductive step

With a 2-vertex cut between u and v, the cases are more complicated, but we can still break it up into adding, multiplying, and subtracting smaller cases.



ECCHI (the Enhanced Cycle Counter and Hamiltonian Integrator)

- Canonically label subproblems using nauty [McKay, 1981].
- Cache the answers to subproblems in a hash table.
- Forget cached subproblems if we run out of memory.
- Looking at a subproblem: just write the answer if it's trivial, get the answer from the cache if it's to be found there, else split it with whichever is the strongest heuristic that applies and add the resulting subproblems to the to-do pile.
- Implemented in C with multithreading, to run on a single multicore PC.

Application to extremal graph theory

Joint work with Durocher and Li: Among all graphs with n vertices and girth (length of smallest cycle) g, which ones have the most cycles?

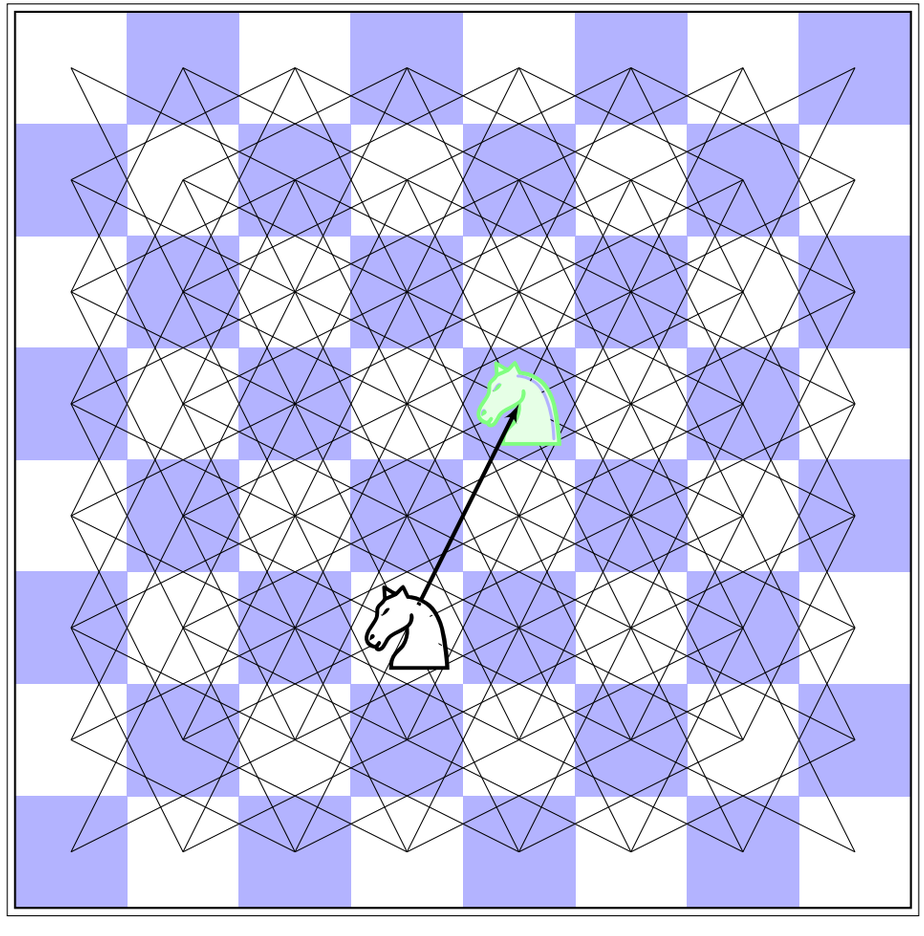
For small n, answerable by simply testing them all. Works up to $n \approx 15$.

The cost of generating the graphs to test dominates over the cost of counting the cycles.

What about larger graphs?

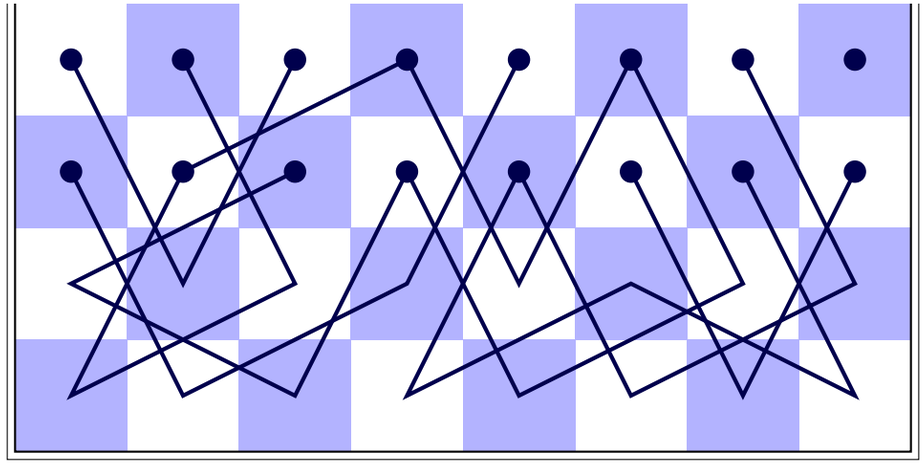
The Knight's Tour

Classic chess puzzle: Hamiltonian cycle in a graph where vertices are the chessboard squares, with an edge between each pair connected by a knight's move.



Counting Knight's Tours [McKay, 1997]

Count, by backtracking, the partial tours that cover half the board:



There are 70433448 of these, associated with 7934470 patterns of connections among the sixteen vertices of the interface. Then somehow stitch these counts together: the total is 13267364410532.

This is an ad hoc technique making use of human intuition regarding the structure of the problem.

How many cycles in a hypercube?

"Isn't there a nice recurrence for it?"

k	general	Hamiltonian
2	1	1
3	28	6
4	14704	1344
5	51109385408	906545760
6	?	358382137225708838707 20

No larger results are known, and no nice recurrence. The current version of ECCHI easily reproduces these up to $k = 5$ (in a minute or two; I use them as test cases), but cannot reasonably be applied to $k \geq 6$.

Important side issue: correctness

At the 16 March 2012 lab meeting I had a slide very similar to the previous one, but reporting a count of 1475466650833443325056O Hamiltonian cycles in the 6-cube due to Deza and Shklyar [arxiv.org posting, 2010]. The new count of 358382137225708838707 20 is due to Haanpaa and Ostergard ["to appear" journal article, 2012] correcting the errors of Deza and Shklyar.

McKay's 1997 count of Knight's Tours on the 8×8 board was also a correction to a previous differing count from Löbbing and Wegener [1996].

So: if a computer produces a number, should we believe it? If we wrote the software ourselves, should we believe it more or less than we might believe someone else's published result?

More philosophy

Appel and Haken [1976] opened this can of worms with the Four-Colour Theorem. Zeilberger lists his computer as co-author, e.g. on a proof of the Cosmological Theorem [Conway 1987; Ekhad and Zeilberger 1997].

Should we let Zeilberger get away with such shennanigans? Does having the name "Ekhad" to blame if the Cosmological Theorem turns out not to be true, solve a problem? Is there a problem to solve?

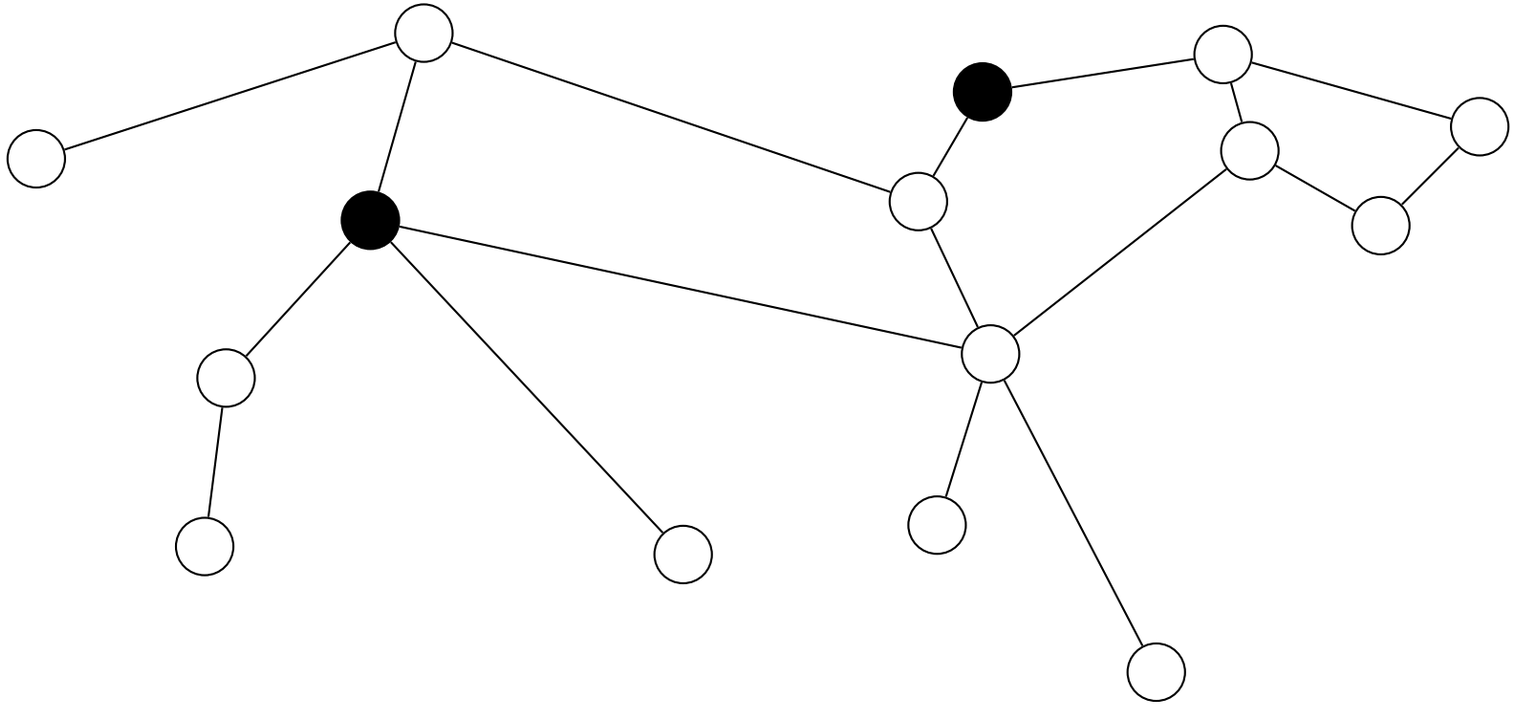
What will I do if ECCHI doesn't agree with either of the existing results for the 6-cube?

If we can't verify whether a statement is true, can it be said to have a truth value at all?

What is truth?

A better subproblem

Count the cycles that include all the black vertices. Using white vertices is allowed but not required.



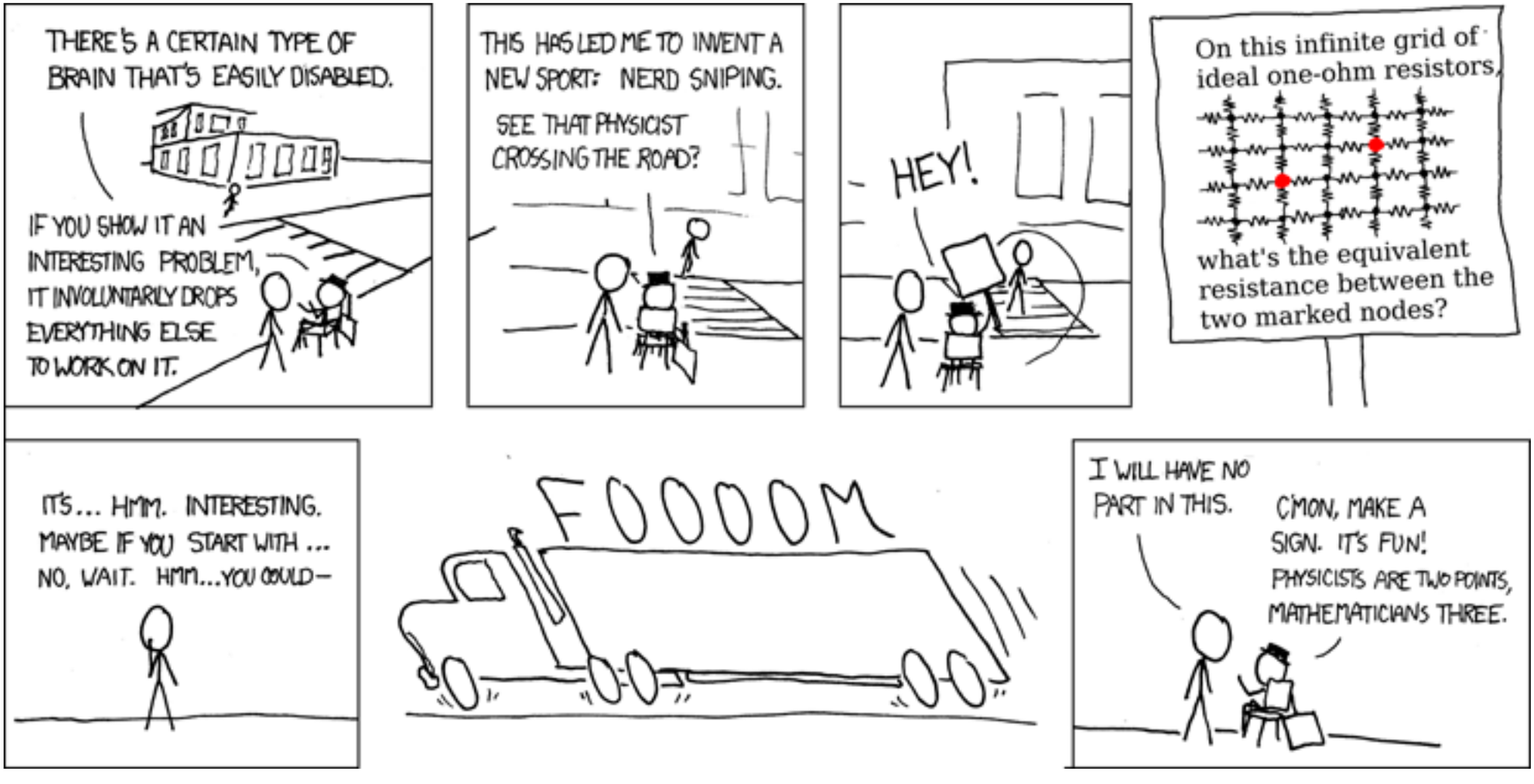
The new subproblem generalizes many others

Before, we had four different subproblems: count cycles or count paths $\times$ Hamiltonian or not. These can only partly share their implementations. Now:

- We can force an edge to be included by inserting a black degree-2 vertex in it.
- To count paths: join the endpoints with a forced edge.
- To count Hamiltonian: make all the vertices black.
- We can also count multi-path patterns, like McKay's half-tours.
- Many new inductive steps are possible.

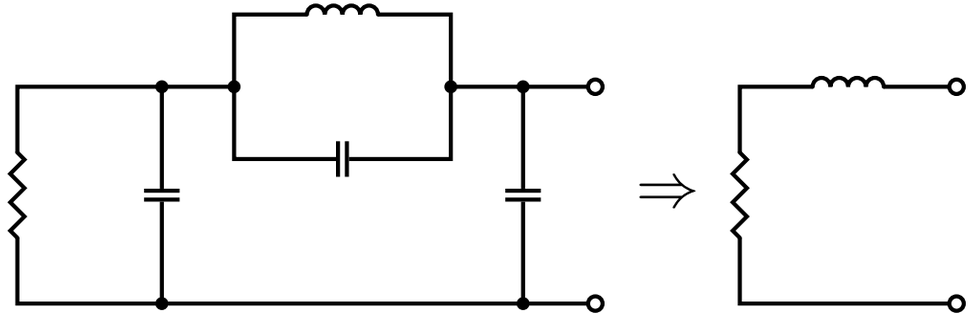
The bad news

- Implementing the new subproblem makes ECCHI slower.
- The new inductive steps are additive instead of multiplicative.
- It's rare to even be able to use the old two-vertex cut rule, let alone any fancier inductive steps.
- Cache hits are 1% or less on Hamiltonian subproblems.
- No improvement for large graphs, and we don't need improvement for small graphs.


THERE'S A CERTAIN TYPE OF BRAIN THAT'S EASILY DISABLED.
IF YOU SHOW IT AN INTERESTING PROBLEM, IT INVOLUNTARILY DROPS EVERYTHING ELSE TO WORK ON IT.
THIS HAS LED ME TO INVENT A NEW SPORT: NERD SNIPING.
SEE THAT PHYSICIST CROSSING THE ROAD?
HEY!
On this infinite grid of ideal one-ohm resistors, what's the equivalent resistance between the two marked nodes?
IT'S... HMM. INTERESTING. MAYBE IF YOU START WITH ... NO, WAIT. HMM...YOU COULD—
FOOOOM
I WILL HAVE NO PART IN THIS.
C'MON, MAKE A SIGN. IT'S FUN! PHYSICISTS ARE TWO POINTS, MATHEMATICIANS THREE.

Equivalent circuits

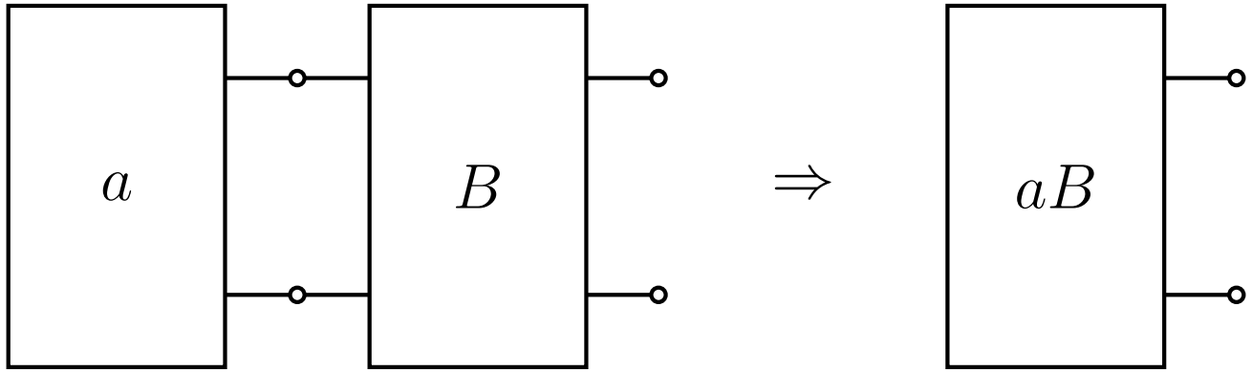
Electronic circuits subject to certain constraints are governed by linear differential equations, and linear functions are *closed under composition*. Therefore any circuit is equivalent to some circuit of a very simple form. (Norton's Theorem, Thévenin's Theorem, etc.)



A vector of constant dimension can describe a circuit of any size.

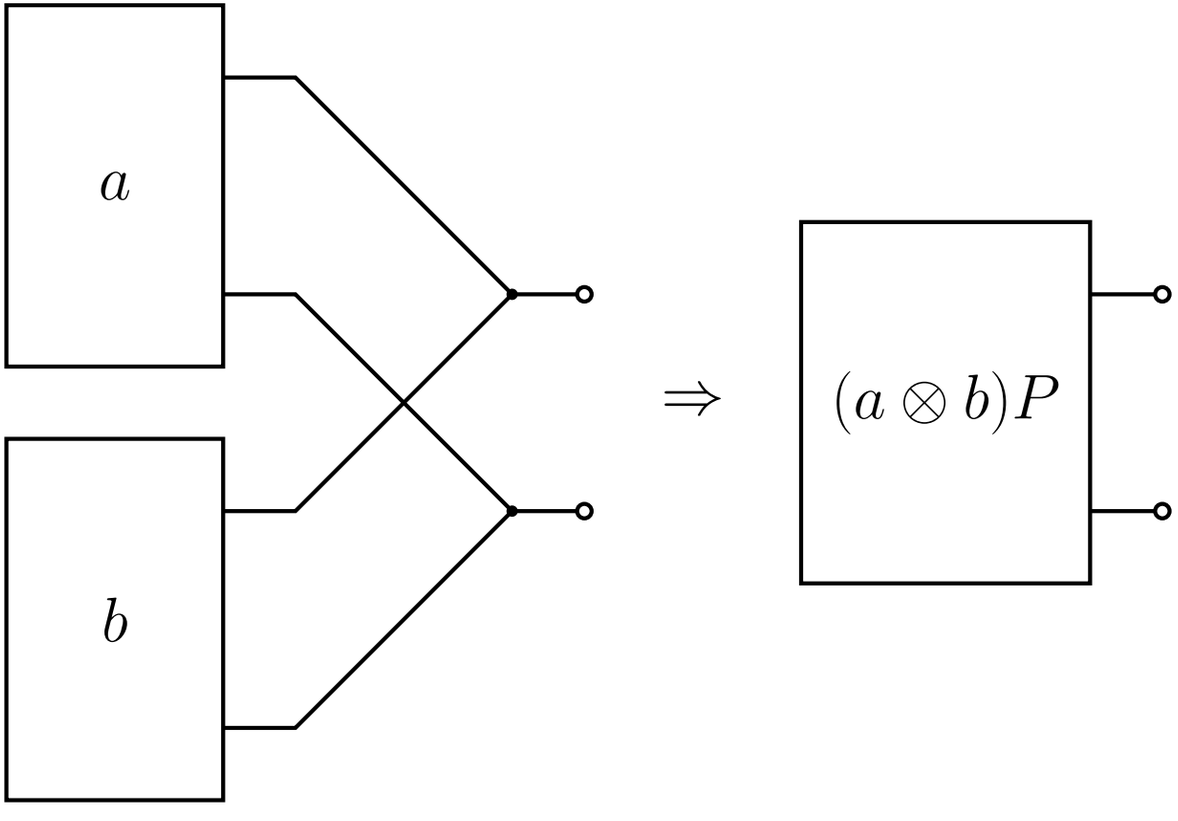
Building up more complicated circuits

A two-port circuit multiplies the vector by a matrix.



Building up more complicated circuits

A constant matrix exists that expresses what happens when you put two circuits in parallel.



Counting with matrices

- Split the graph into chunks with small interfaces between them.
- Solve all patterns within a chunk to get a vector or matrix (or a tensor, if you must) for each chunk.
- Multiply them together to get an overall answer.
- Can be applied recursively.
- Vector dimension is "something like $n!$" in the size of the interface.
- This is why cycle counting and many other $\#\mathcal{P}$-complete graph problems are easy for graphs of constant treewidth.
- Eigenvalues of the matrices allow proving asymptotic bounds for large graphs.

Matrices subsume many previous approaches

- ECCHI's heuristics look like matrix multiplication at the bottom levels, with backtracking above.
- McKay's Knight's Tour count does matrix multiplication $(a \otimes a)P$ at the top level and backtracking to compute the vector a.
- Known results for the 6-cube (both correct and incorrect) use a similar structure, two or more levels of matrix stuff at the top.
- As in any hard problem, expensive heuristics are most worthwhile at the top levels. *That's why current ECCHI doesn't work well on large graphs.*

The next generation

- More abstract, object-oriented design (C⇒C++): move to a *toolkit for building* cycle counters
- Easier to change the subproblem definition (even beyond cycle counting?)
- Easier to choose which heuristics to use, and the meta-heuristic that controls them
- Components for caching, lookup tables, and distribution
- Extensive test suite for components and overall system
- Ultimately: machine intuition