

# Re: Graph Theory: Cutting a Graph into Two in an Artificial Chemistry

---

*Source:* <http://sci.tech-archive.net/Archive/sci.math/2008-08/msg00066.html>

---

- *From:* [cbrown@xxxxxxxxxxxxxxxxxxxx](mailto:cbrown@xxxxxxxxxxxxxxxxxxxx)
  - *Date:* Fri, 1 Aug 2008 08:46:48 -0700 (PDT)
- 

On Jul 31, 3:03 pm, Chris Gordon-Smith <use.addr...@xxxxxxxxxxxx> wrote:

riderofgiraffes wrote:

<snip>

It's unclear what you mean by "cut" – there is more than one definition, and for each definition, someone thinks it's "obviously the right one".

Let me put the question a different way. Here is a brute force way to find what I am looking for:–

- 1) I start with a graph that has a single connected component (such that any node can be reached from any other node by traversing a series of edges)
- 2) I then divide the graph's nodes into two mutually exclusive groups and remove all of the edges between nodes in different groups, leaving the other edges in place. If I do this and the result is exactly two connected components, then I count that as a valid way that my original graph (molecule) could split. If the result is more than two components, then I discard the result as invalid. I would count a single node on its own as a connected component.
- 3) I repeat (2) until I have tried all of the possible ways of dividing the nodes of the original graph into two mutually exclusive groups
- 4) The set of all 'valid' splits from (2) is the answer I want.

Is there an efficient way to get this set of 'valid' splits?

## Re: Graph Theory: Cutting a Graph into Two in an Artificial Chemistry

Since no one else has responded, no solution, but some thoughts...

First, it sounds like you don't want to just count these "splits", you want to obtain a list of pairs of disjoint sub-graphs which cover the original set of nodes.

Second, your problem will become quite a bit more difficult if you want only a list of non-isomorphic such pairs. For example, for ethane, one could argue that there are only two "different" such pairings: (CH<sub>3</sub>, CH<sub>3</sub>) and (CH<sub>3</sub>CH<sub>2</sub>, H); even though your algorithm above would naively count 7 distinct such pairings. And determining whether two graphs are isomorphic is (generally) not so easy.

At any rate, there are relatively quick algorithms for determining whether two nodes a and b are members of the same component of a graph (for a starting point see:

[http://en.wikipedia.org/wiki/Disjoint-set\\_data\\_structure](http://en.wikipedia.org/wiki/Disjoint-set_data_structure)

).

This suggests that you can try removing edges  $E_i = (n_a, n_b)$  one at a time, then check for connectivity of  $n_a$  and  $n_b$ . The trick would be to bound your searches so that once you have determined that some set of edges ( $E_1, E_2, \dots, E_k$ ) causes a "split", you don't re-examine sets of edge removals which contain that subset a second time; e.g., by considering the directed graph of all subsets of edges, and then traversing /that/ tree, stopping at any depth that produces a split.

This approach can probably be modified so that if you know that some set of edges are essentially the same (e.g., the C-H bonds in ethane), then you don't revisit removing a single C-H edge 6 times.

It would probably be useful to know the size and complexity of the graphs you are working with. Trivially, if your graphs are trees, then the number of such "splits" is the number of edges. So if they are sufficiently small and generally tree-like except for a few cycles, it may be that this will affect the average expected complexity (as opposed to the worst-case complexity); and perhaps even a brute force search will be sufficient to proceed.

Anyway, post if you find a nice algorithm for this...

Cheers – Chas

.