Miai and VCs

Now we add in miai and (limited) virtual connection detection. Instead of using connection graphs to determine if one of the players has won, we now use a UnionFind datastructure. There is a new function called vc_search. It initializes all adjacent empty cells to be virtually connected, and then iteratively builds semiconnections from two virtual connections with one end in common and the common end as the key. Additionally, two semiconnections with the same endpoints can then be combined into a virtual connection if the intersection of their carriers is empty. More generally, more than two semiconnections can be combined into a virtual connection as long as the intersection of their carriers is empty, but this program only combines pairs of semiconnections into virtual connections for simplicity. When two semiconnections with the same endpoints are combined into a virtual connection and both ends of the vc are occupied with ptm stones, their keys are miai. So now if our opponent plays in one of our miai, there is an easy response to restore the vc. Now we can also add a better move ordering, using the virtual connections computed with vc_search. This gives rise to a new function, rank_moves_by_vc. There are also some new commands to visualize miai info and the move ordering generated by rank_moves_by_vc. The function win_move has also been updated. Now we can return if there is a vc between the two sides after a move, but we also have to include the winset associated with that vc in the win threat set we return. This is a significant improvement, and now this solver can handle any 5x5 position except for a4 and e2 in less than 5 minutes on a pre-2015 Dell Latitude with an Intel i5.

Further reading:

https://www.cs.auckland.ac.nz/courses/compsci767s2c/resources/VAnshelevich-ARTINT.pdf