Project 2 - James Tappe

To obtain the results I did, I created 10 graphs with n nodes for n up
to 800. For AC3 and MIN_CONFLICTS I created graphs up to 2000. For
BACKTRACKING I only ran the app on graphs with 180 nodes or less
because BACKTRACKING was consistently running out of memory. This
indicates to me inferences reduce the search tree tremendously. I've
only included the graphs of 4-colorings because my searches for
3-colorings generally failed for n greater than 12 nodes.

What I found was that not only execution time increases with the
number of nodes but the number completed searches dropped. In other
words, I tended to run out of memory the higher the node count.

As for calculating the average search times, for a given number of
nodes n, I ran a search strategy on the 10 different graphs (each with
n nodes). If the search completed, I included it's runtime in the
average, which was what I graphed.

This project was very time consuming. Obviously I'm already late,
however if I had time I would not run the search for 3-colorings. If
you look at the individual logs (report.log report2.log) you'll the
3-colorings were the most responsible for the number of crashes I
had. The only 3-color search strategy that worked for me consistently
n>12 was MIN_CONFLICTS, however I think this was an error because
MIN_CONFLICTS initially assigns a color for every node, and the way I
checked completion was if the domain size of every node was equal to
one. This leads me to believe the MIN_CONFLICTS data for 4-colorings
is erroneous as well.

That being said, from the rest of the data I would say that the MAC
strategy was the best. Although it had some "crazy" outliers for
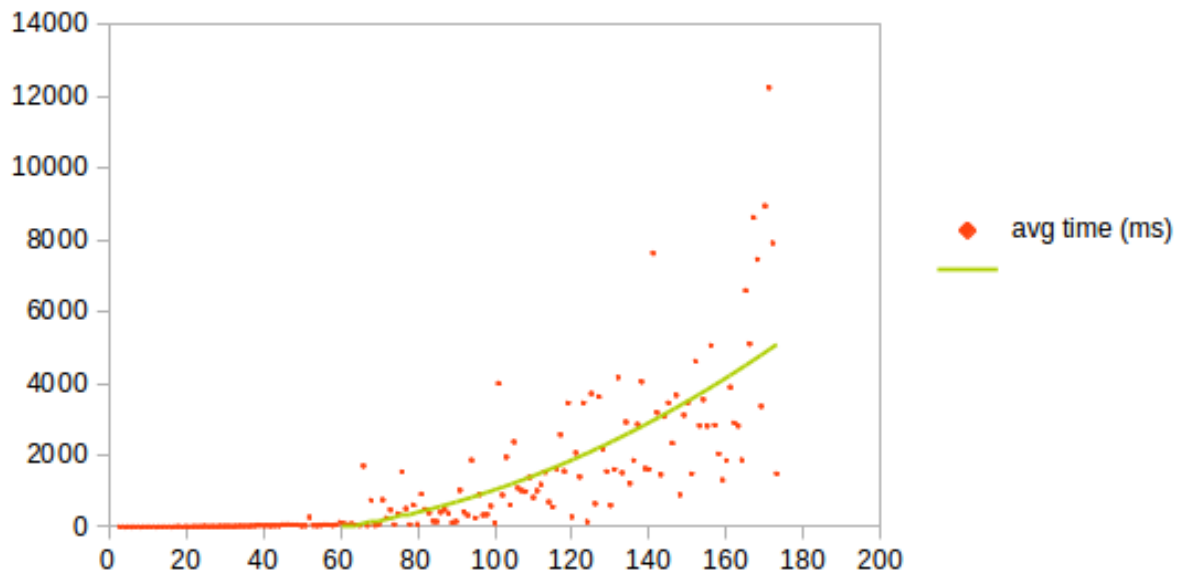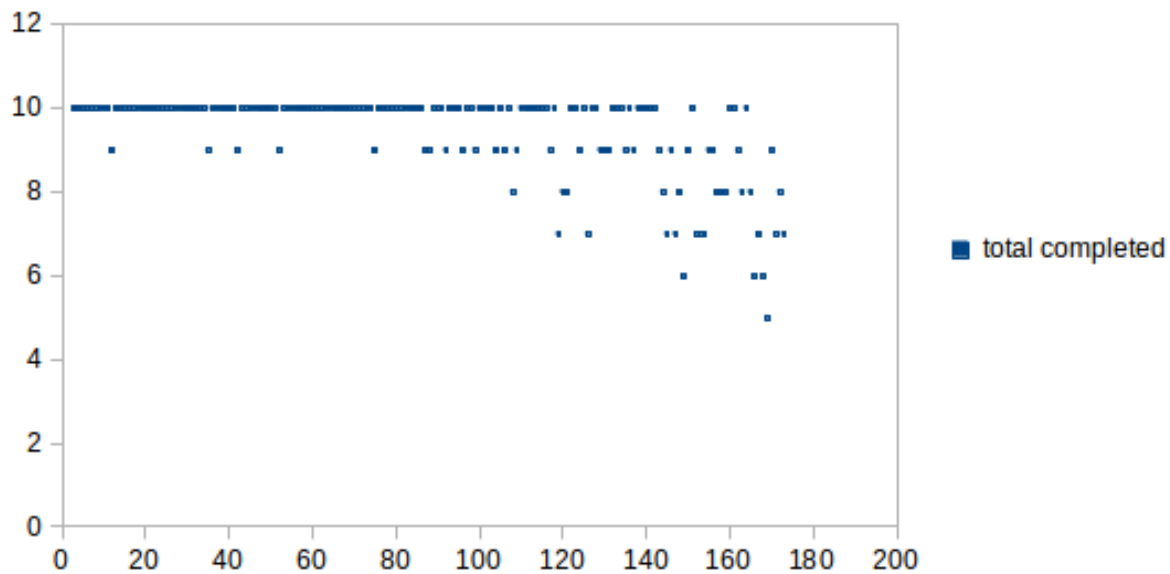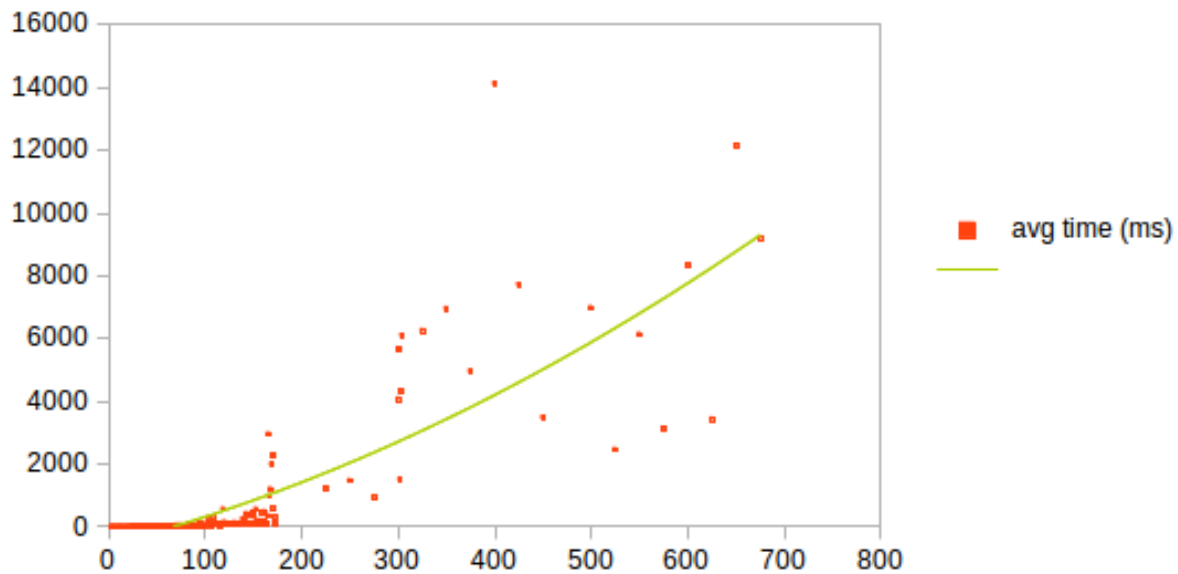completions times, it had the most completions.

AC3 4-Color

- avg time (ms)
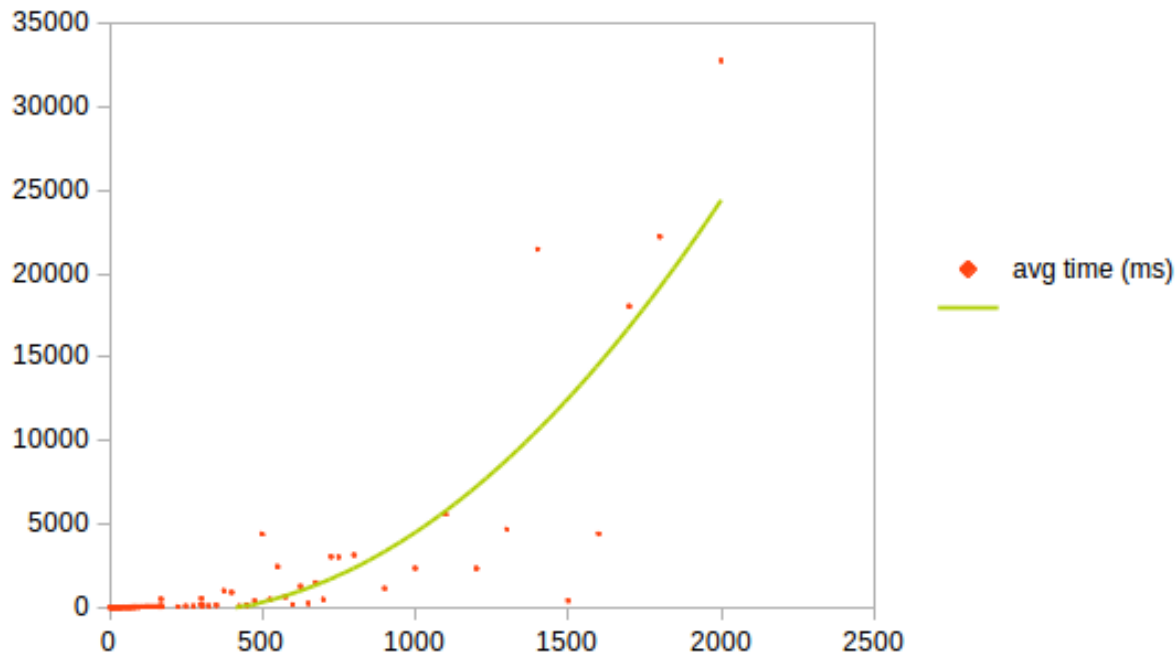
AC3 4-Color
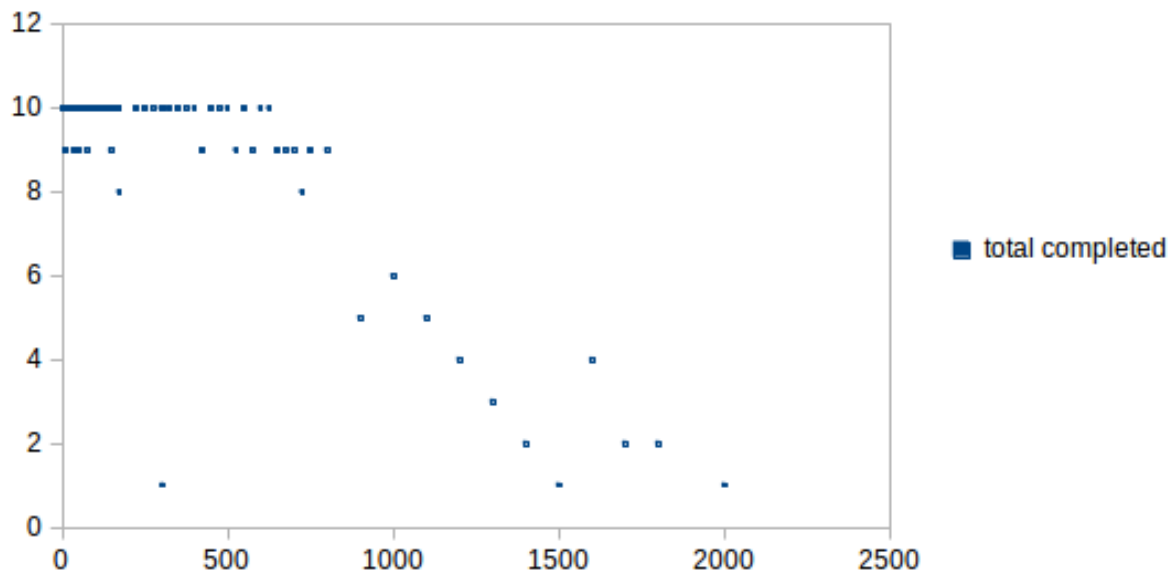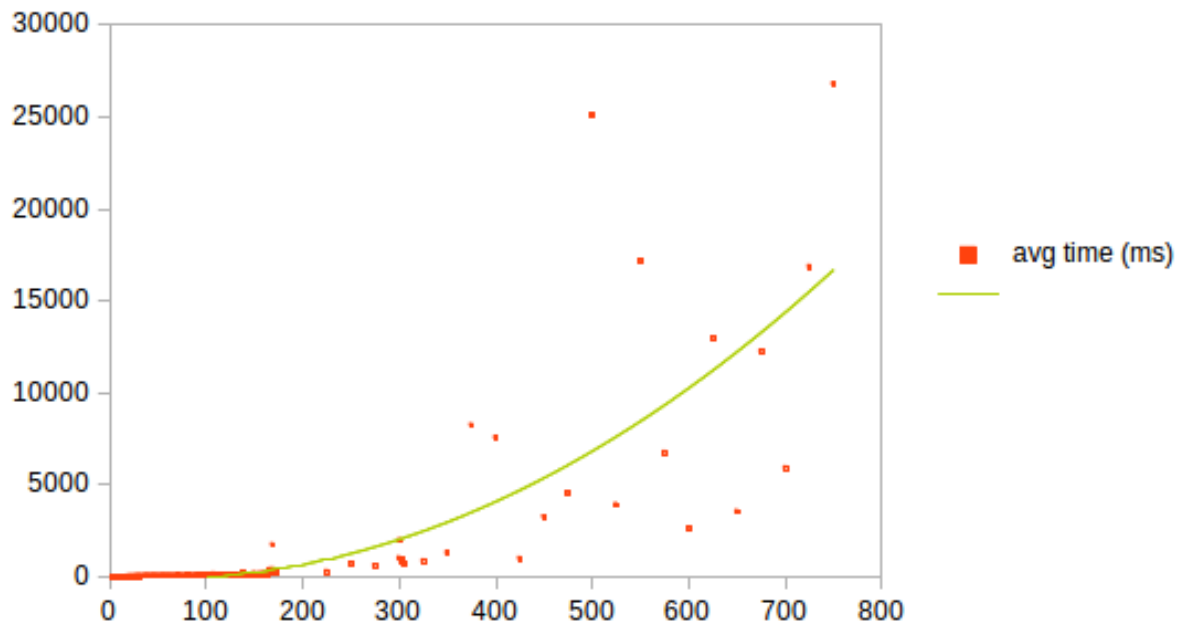
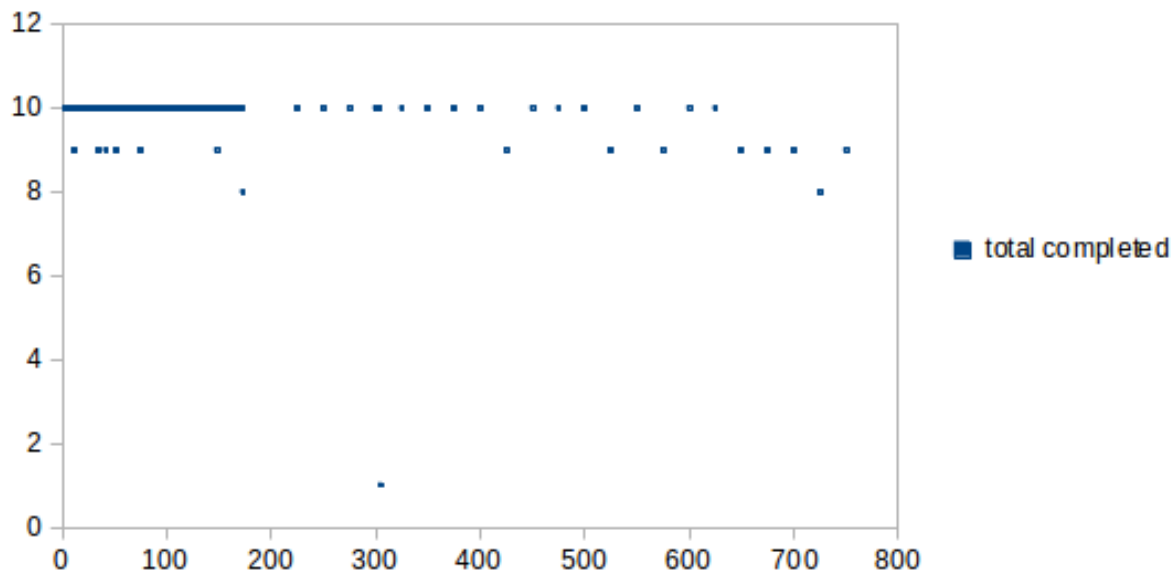BACKTRACKING 4-Color

BACKTRACKING 4-Color

■ total completed

FC 4-Color

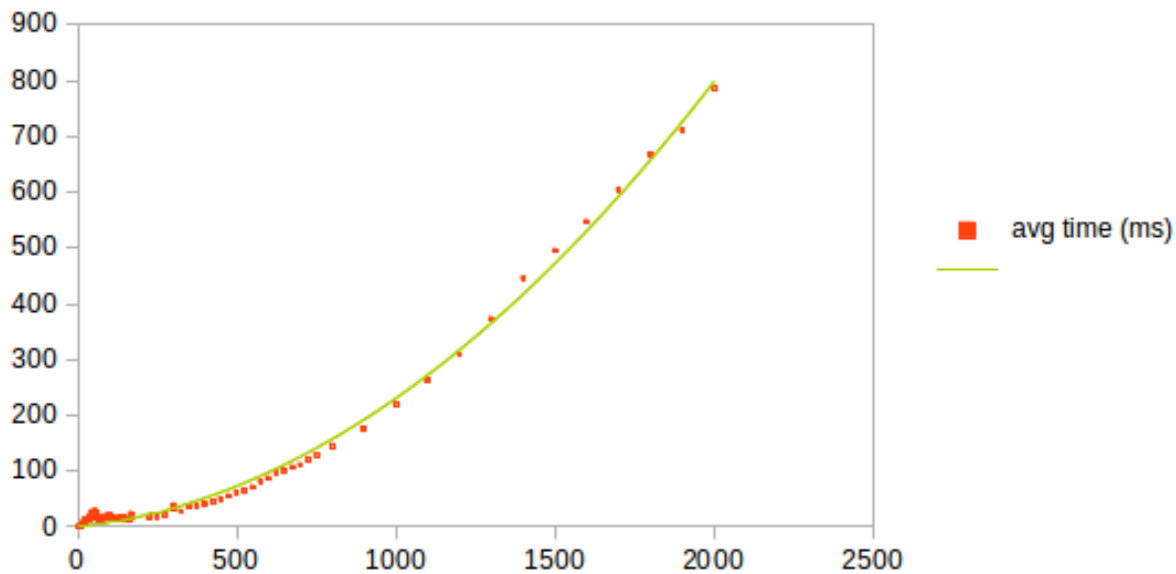FC 4-Color
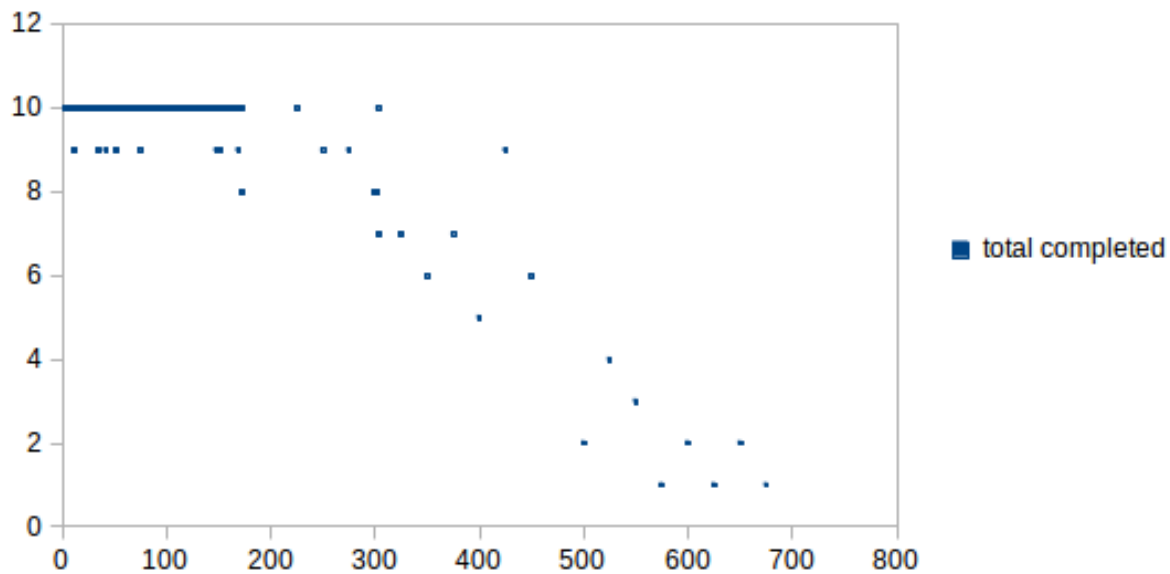
MAC 4-Color

MAC 4-Color

MIN_CONFLICTS 4-Color

Legend:
- avg time (ms)

MIN_CONFLICTS 4-Color