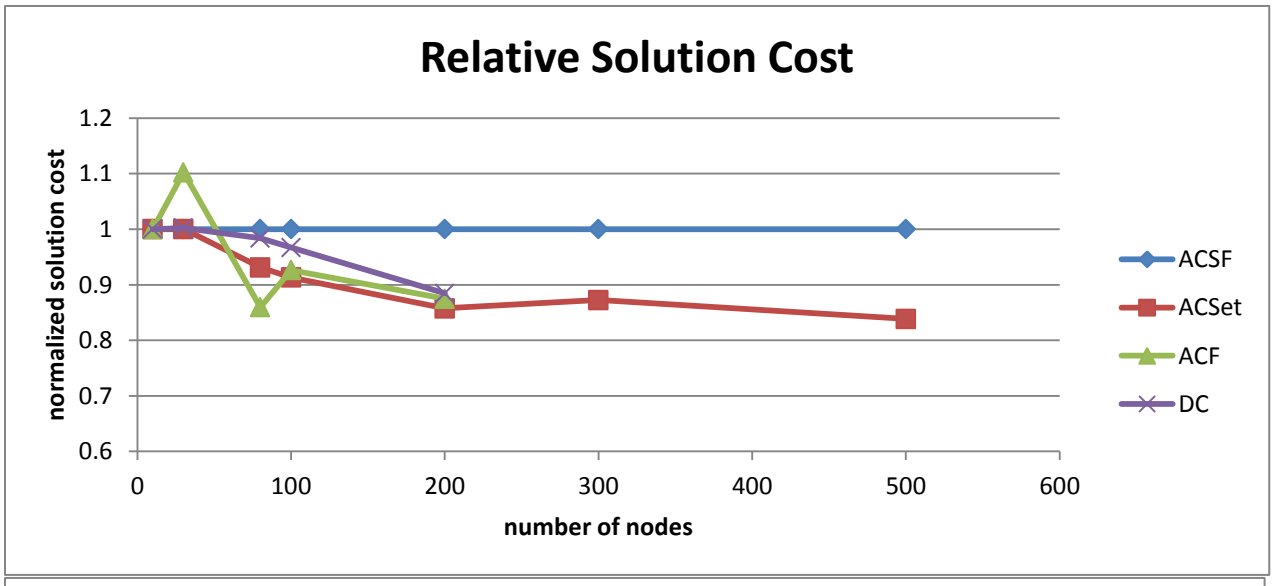
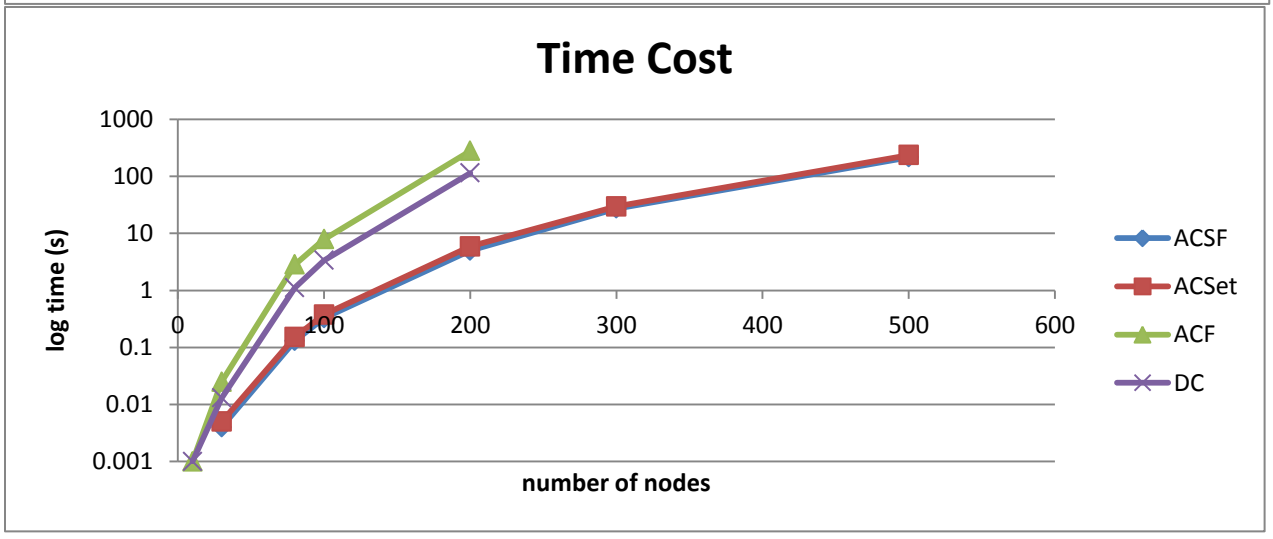
Steven Cabral
6801179
CS130B
F 10AM Section
2/12/14

Project Experimentation and Report

I implemented all the code to test the four algorithms: add closest to source first (ACSF), add closest to set first (ACSet), add cheapest first (ACF), and divide-and-conquer (DC). My specific implementation is submitted on csil. I then ran each algorithm on a set of test files (which can be found named input<number of nodes>_<algorithm ID>). For each test, I obtained the resultant lowest power for each method and the time it took to arrive at that solution. Then I normalized the power result against the power result found from ACSF, representing all of the lowest power solutions as a fraction of the solution generated by ACSF. The following graphs show my results:





From the simple tests I ran, it seems obvious that the ACSet method performed the best. Not only is its minimal power schedule the best on most trials, but it runs in a time nearly equivalent to ACSF (the time difference is negligible). ACSF is a fast procedure, but it is also the least minimal, and it would only be good in situations that require fast estimations, though ACSet is still a strong option in this situation too. ACF and DC seem simply too inefficient for most practical purposes, and their inefficiency doesn't even seem to yield any additional gain in optimality. The only situations that ACF might be useful in are cases with small numbers of nodes, though the drop lowered solution on the graph may also be an artifact of the data. From this data, ACSet is the clear winner among all of these algorithms, at least in the way I implemented them.