Project Github: **https://github.com/ivyaccount/pcap_project**

Make sure your project schedule on your main project page is up to date with work completed so far, and well as with a revised plan of work for the coming weeks. As by this time you should have a good understanding of what is required to complete your project, I want to see a very detailed schedule for the coming weeks. I suggest breaking time down into half-week increments. Each increment should have at least one task, and for each task put a person's name on it.

Week 4:

1. Finish implementing parallelism of CDCL algorithm to reduce time on backtracking (Raashi)
2. Run CDCL implementations (without specific load balancing methods) on test suite and capture metrics (Raashi)
3. Finish implementing parallelism on unit propagation (Ivy)
4. Finish implementing DPLL guesses on variable values (Ivy)

Week 4.5:

1. Finish implementing load balancing methods for CDCL algorithm (Raashi)
2. Run CDCL implementations on test suite and capture metrics (Raashi)
3. Finish implementing DPLL multiple guesses at once on variable values (Ivy)
4. Run DPLL implementations on test suite and capture metrics on both platforms

Week 5:

1. Begin working on poster presentation (Raashi and Ivy)
2. Perform an in-depth comparison of all of the methods created (Raashi and Ivy)
3. Create final project report with collected statistics (Raashi and Ivy)

Week 5.5:

1. Finish creating poster presentation (Raashi and Ivy)
2. Present poster at poster session (Raashi and Ivy)

One to two paragraphs, summarize the work that you have completed so far. (This should be easy if you have been maintaining this information on your project page.)

So far, we have set up our environments and built a suite of diverse tests which we've chosen to use to compare all of our implementations. Additionally, we've begun to collect metrics on our sequential implementations to use as baselines in the context of our parallel implementations.

So far, for both algorithms, we've identified possible points of parallelism, and are working towards implementing these ideas. More specifically, for the CDCL algorithm, we're attempting to divide possible branches of variable assignments or unit propagation among processors to reduce time lost to backtracking. Additionally, to enable load balancing, we are considering distribution of clauses between threads (in a manner that reduces thread communication and the need for conflict resolution), as well as dynamic work allocation methods. As for the DPLL algorithm, we're trying different approaches including parallelizing the variables false/true guess and guessing multiple combinations of variables at a time. We will also be considering preprocessing the income clauses trying to get the load balanced.

Describe how you are doing with respect to the goals and deliverables stated in your proposal. Do you still believe you will be able to produce all your deliverables? If not, why? What about the ”nice to haves”? In your milestone writeup we want a new list of goals that you plan to hit for the poster session.

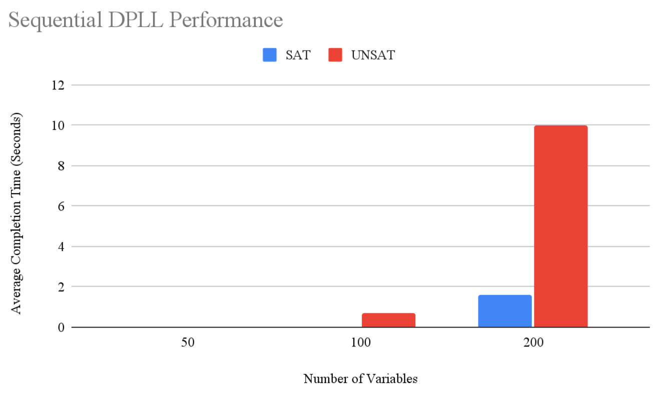
With respect to the original timeline proposed, we are on track. We believe that we will still be able to produce at least one CPU-based parallelized version of the DPLL and CDCL algorithms, as well as analyze the performance of these implementations in depth.

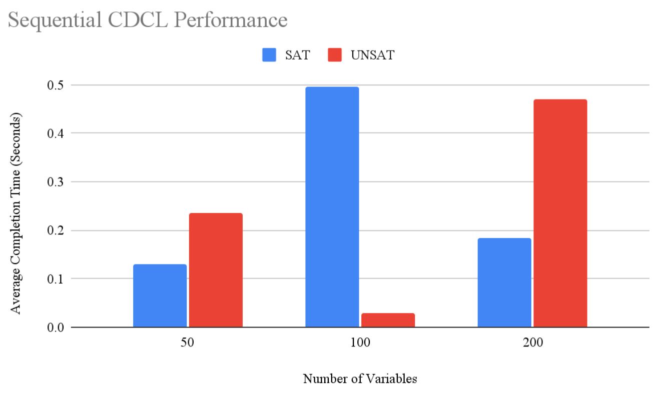
In our initial proposal, we mentioned that it would be "nice to have" sequential and CPU-based parallelized versions of the WalkSAT algorithm - however, we are adjusting this goal, by choosing to focus on differing versions of the chosen algorithms with possibly differing resources, which we have more familiarity with.

What do you plan to show at the poster session? Will it be a demo? Will it be a graph?

At the poster session, we plan to display a multitude of graphs showcasing the performance improvements achieved through parallelization, as well as other metrics compiled (such as scalability and proportions of different types of computational cost). Through our poster, we aim to present a comparative analysis report summarizing the strengths and weaknesses of each parallel implementation.

Do you have preliminary results at this time? If so, it would be great to include them in your milestone write-up.





The above bar graphs show the average completion times for the two algorithms - the test suite included tests with 50, 100, and 200 variables, with both Satisfiable and Unsatisfiable problems. (Note the scale here of both charts) The DPLL algorithm seems to be slower than the CDCL algorithm on the 200 variable cases, but overall both algorithms seem to perform relatively quickly. Note that two algorithms are running on GHC machines with Intel Core i7-9700 processor (3.0 GHz , Eight cores, 8 threads).

List the issues that concern you the most. Are there any remaining unknowns (things you simply don't know how to solve, or resources you don't know how to get) or is it just a matter of coding and doing the work?

At the moment, we are still looking for tools that we can use to measure cache hits and misses, to serve as a metric for determining the efficiency of our implementations.

We're also thinking and searching for other creative ways possible to parallelize our SAT solvers.