I sent this email to Bryan:

This is like an essay update in an email. To summarize, the program slows down as it is storing more things in the computers working memory. There may be a possible solution but I don’t know if you want me to work on it.

---

I was looking at the optimizer function last night, and I thought it would be cool if it told you every now and then how far it had progressed through the alphaCost. So I did some calculations and found a way to say a percent completion for a given cost pair (alphaCost, betaCost). For example it would say it is 30.5% done using NOR to combine circuits that cost 7.0 with circuits that cost 6.0. This way you don’t feel like you are waiting indefinitely. I initially tried to do total percent completion rather than percent of betaCost completed, but it was harder than I thought. Anyway, I started by using a random number generator and said with a probability of x print out the progress, but I thought it seemed to inconsistent, so I changed it to every 5 minutes it will print an update (every time it has been >=300.0 seconds since the last update).

After I implemented this, I noticed the effects of amount of memory stored compared to speed. Like you mentioned, the more things it has stored, the slower it runs.

I guess I need to define some terminology for this next part. I’ll call checkpoint 1 or cp1 the points where it completes a cost and prints in the console how many circuits it found and how long it took and how many circuits are in that level. You have cp1 in the version I sent you. I’ll call cp2 the one I did last night. At cp2 it prints the progress toward completing the level.

Cp1 appears at the completion of a cost. Cp2 appears either 300.0 seconds after the most recent cp1 or cp2. So if the completion of a cost is taking more than 300 seconds, it will print progress reports, but if it is less than 5 minutes it will not print progress reports.

So now that that is explained, I noticed that you can approximate the time it takes to build a single circuit by looking at cp2 and comparing it to the cp before it since it checks if 300 seconds has passed at the start of building a new circuit (this may slightly slow things down since it has to check so often). Combining circuits with a cost of 6.0 with things that cost less than it takes a little less than 10 minutes to complete. I noticed that cp2 appears at exactly (with rounding) 300.00 seconds after the previous cp1 from completion of 5.0 costs. So this means that it is essentially taking no time to build each circuit. However a couple of checkpoints into 7.0 cost, when a lot of data is starting to be stored, cp2 is appearing about 301.5ish seconds after its most recent checkpoint. Of course this varies based on where the program is at the time of doing the time check. If it happens to finish the last circuit at 299.9999 seconds then it will need to wait a full loop before printing progress. If it finishes the previous one at 300.00001 then it will immediately print the progress report.

Anyway, this means that it is slowing down considerably since it is spending about 1.5 seconds per circuit as opposed to like <0.001 seconds when sortedByCost is a lot smaller in lower costs. I had an idea that might fix this problem, but it might be difficult to code and may take a while, so I wanted to ask you before I started it to see if it was worth it. The idea is that at cp1 and cp2 it would also “dump” all of its contents in a text file so it is stored in the hard drive rather than the working memory. There are two gains and possibly two risks from this that I know of:

 For one, the program is sped up because it is not holding as many things as before. It will put the components of sortedByCost that it isn't using in a file every checkpoint. So it is really only holding the circuits in alphaCost, and betaCost.

Which brings us to the first risk. Each time you move to the next cost, you need to load the circuits from the file. Since the file would probably just contain the name of each circuit rather than each property, I would need to make it find all the properties for each circuit just from the name, but this would only be done once each time we are changing cost that we are looking at, and saving it to a file should be rather quick.

The second thing is that it allows for the option of continuing the program from the last checkpoint if I adapt the function to allow for it. Since essentially all values will be stored in text, if your computer overheats, or crashes, or you just want to stop the process temporarily because it is slowing down your computer, you can do that and resume it at the last checkpoint rather than starting over. Which seems like a major gain.

The second risk is that the file size of the circuits in sortedByCost might get big and if you tried to open the file and see its actual contents (which I don't see why it would be necessary) that would probably freeze up your computer because the file would be so big.

I don’t know how difficult it is to program that ability in. I don’t know if it will be easier than it seems since it is already planned out or longer because I am underestimating it. I don’t think it would take more than a week worst-case scenario, but I don’t know if you want me to put the time into it or if you want me to just move on to more important tasks like working with Jing and Alex.

Arinze

I will wait for a response to decide whether or not I should put more time into the program. Last night, I also changed the naming convention for the files so the date is first. This makes it so that all files with the same date are grouped together when sorted by name rather than grouping by max cost reached as would be done before with the old names. This was not mentioned in the email because the email was long enough as it is.

While I wait for a response, I will work on the SRA tool kit I downloaded.

I followed instructions from https://github.com/ncbi/sra-tools/wiki/Building-and-Installing-from-Source.

What I entered into the cmd:

$ cd C:\Users\Arinze\Desktop\UROP2015Copy\sratoolkit.2.5.2-win64\bin

$ cd ncbi

$ git clone https://github.com/ncbi/ngs.git

$ git clone <https://github.com/ncbi/ncbi-vdb.git>

These ones were giving me trouble so I had to figure out how to read git to the system path.

$ cd ngs  
$ ./configure

./configure is also giving me a trouble.

I glanced back at the circuit builder and I noticed that the calculation for percentage complete was off because it was increasing then decreasing. It was because of the order by which it made new circuits. It would go through alphaCircuits once and then go through each of the beta costs for each element of alphaCircuits. I changed it so it goes through each element of alphaCircuits once for each betaCost. This is allows for the percentage complete calculation to be more accurate. I commented out the old version so it is still there in case I want to convert it back. I also changed the checkpoint2 time to 150.0 seconds so I can see updates more frequently. Using the new method of ordering circuits, the program was more quickly able to determine that storing 9.0 cost circuits (using nor at cost 1.0) would take too much memory. It crashed. This at least tells me that trying to run it with the other method would crash too and I shouldn’t waste my time trying it unless there is a way to store the values outside of the program.

SRA tools isn’t working because in the beginning of the summer I screwed up the cmd Paths while trying to make antlr work, so I will look into sequential logic.

There are two kinds of sequential circuits: synchronous and asynchronous. Synchronous depends on a clock. Whenever the clock changes, the circuit updates it values at each wire. The problem is that the clock cannot change faster than the propagation delay. The propagation delay is however long it takes for each element in the circuit to receive the signal. Something that is 5 gates in will receive the signal that the clock changed faster than something 10 gates in. This makes me wonder what happens when something depends on its output and the input. Would the wire’s value be ‘flickering’ while waiting for the net clock? Or do we pause the state until the next input change. In a real cell, we would not be able to pause the outputs and wait until the next input. I read a bit more and apparently the portions of the circuit that complete their operations first are idle until the next clock pulse which must wait for the last portion of the circuit to finish.

Asynchronous circuits are circuits that don’t use a clock. The value of a wire changes as soon as an input is received for a gate. Due to propagation delay, this can be problematic if a gate receives one of its inputs before the other and performs a calculation based on that input.

I’m not sure which one Bryan would want me to focus on. I will have to wait and ask him. In the mean time, I can work on a way of saving the circuits in sortedByCost in a file so they are out of the working memory. I will probably just start by planning how it will be organized and not actually write any code yet.

Alex and I talked about what we are working on and discussed what problems we are running into and possible solutions.

I also spoke to Jing and she explained what specifically would need to be done for her project. And we went over in more detail what each file type contains and what each number in the equations mean. I will start the actual programming tomorrow.

With Alex, I exchanged codes with him and his code runs on my computer. I still need to take a more detailed look at his code and make sure I understand what is happening.