Time Plan

July 3, 2018

Any of the first five points can be done simultaneously, mostly independent of each other.

1 Literature study. Initial knowledge base buildup

2 weeks.

Bidirectional single-rail train tracks are in many senses analogous to underground mining traffic and will be researched. Hopefully their concepts can be implemented. A lot of parameters are different though. Trains talk about stations, dwell times at stations, and usually have perfectly known speeds and positions. We also have the case of absolute priority and connectivity issues to take into account. So there's no guarantee that railway algorithms are a suitable match for underground mines.

To brace for the railway algorithms unsuitability, a look into other solutions is necessary. Algorithms with Incomplete-Information is a large and active field of study, and our situation definitely falls into that category. Some research into this area is also needed.

And for good measure, machine learning can always prove useful!

2 Log parser. Take HybridPositioning (i.e. real life data) logs and turn into a feasible format for simulation

1-2 weeks

The real life logs are too big. To speed up simulation they need to be pruned and only store data relevant to the simulation. Right now logs are appended with state data every second. These can be merged into linear or curved equations over time. A lot of data is HypbridP. specific and can be omitted entirely.

3 Visual Layer. Simple way of visualizing events and logs. Doesn't have to be fancy.

1-2 weeks

The alternatives are to either create 2D or 3D models, and to either have animation or simply stills. Preferably one would use Animated 3D scenes. Doing this however would need a 3D engine rather than simply plots. A drawback with a 3D engine is that publication-grade vector graphics might be hard to produce. There are potential solutions to this, but their feasibility will have to be explored in the future.

The visual layer will foremost be attempted to be implemented in Julia, simply because it is an exiting and promising language. GLVisualise is a 3D engine in Julia, based on OpenGL, that can produce a 3D animated scene for us. If Julia, as a new language, proves to be too problematic, or GLVisualise proves to be inadequate for our needs, C++ and pure OpenGL is a reliable fallback that I have much experience with.

4 Truncate logs to "routes". Thus a single route can be initialized, and will simulate a car traveling through the mine.

1-2 weeks

When logs are pruned and there's a proper way of displaying them, chop them up into sections. The logs are usually of a few hours worth of travel in the mine. A much more interesting and useful format would be a short route form point A to point B. Routes over specific sections can then be initialized and tested with. A problem might be that specific routes that are wanted do not exist in the data. Possible solutions might be to implement a most basic simulation of mine traversal, that might not be true to life, to actually go down into the mine to collect more data, or to simply work around and find similar section elsewhere that does have data for it.

5 Make simulator. Able to initialize several routes, simulate collisions, make cars halt at meeting-points, simulate network coverage, poor positioning techniques. Able to switch strategies easily.

4-5 weeks

Once again an implementation in Julia will try to be developed. With its emphasis on mathematical computation hopefully Julia will ease in processing the large node networks that mines make up. A backup would be simple C++, as in the above case.

A problem will be simulating connectivity accurately. Connectivity can be implied by the logs, as they sometimes show a loss of connection. However the question is if one should try to emulate the behaviour shown in the logs, which might prove tricky and might not give that much more accuracy, or simply ballpark it, which would be easy. In addition to this a collision has to be realistically estimated and generated on the fly, without relying on specific logs. The entire solution is centered around avoiding collisions and their costs, after all! One could use machine learning to try and have to computer estimate these for you, or develop some sort of function. Regardless, the accuracy of the simulation will be paramount in order to properly evaluate any strategy!

This point reached by early October

6 Implement strategies based on knowledge gained from literature.

3 weeks

Will slot into a vehicle in a simulation, and preferably also run a global "master" instance that all the vehicles communicate with. This depends on the implementation. Regardless it must work with or without connection to the "master".

Can either be written in C++ or Julia. C++ gives the advantage of being seamlessly portable, and able to be built into any application on any platform. Thus when the Master Thesis is complete, any best solution can simply be extracted and iterated upon. A lot of in-house code already exists in C++ as well, and can be used in the project.

Writing in Julia would enable reuse of a lot of simulation code, since it will need a local representation of the mine node network. Single language projects might also ease development, although Julia should be able to call C++ functions seamlessly.

7 Compare strategies and varying conditions in simulator. Compile data.

1 week

Run a suit of tests on the strategy. The challenge is to provide scenarios that are applicable to the real world, and as such prove the real life value of the given strategy. This is a somewhat abstract concept, and "determining what constitutes a real world scenario" is outside the scope of this paper, and will simply have to be estimated. An informative test to run however is that of time lost to traffic management, as opposed to if the vehicles could run interrupted even from each other. A sort of actual-optimal time difference, and comparing to other algorithms.

8 Alternatively iterate strategies and comparisons.

2-3 times.

Depending on the strategy employed, an iteration might also consist of tweaking of variables or implementation. Although this might not take quite as much time.

9 Present findings.

1 day

$\sim$ January - February