Project Documentation

Date: 2017/11/29

Topic: Genetic Algorithm Scheduling

Task we did today: Decided the problem/topic we are going to solve and initially translated the question into Genetic Algorithm model and terms.

Task tomorrow: Discuss about the details of how to generate (crossover) the next generation, and what constraints we can add to the initial problem.

**Model Mapping:**

Gene: each task and its corresponding start time

Genome: a sequence of tasks and the start times of those tasks relative to one another

First Generation: random start times within the precedence constraints

New Generation: the cross over result of the previous generation

Fitness Function: includes constraints

**Material From Wikipedia**

To apply a genetic algorithm to a scheduling problem we must first represent it as a genome. One way to represent a scheduling genome is to define a sequence of tasks and the start times of those tasks relative to one another. Each task and its corresponding start time represents a gene.

A specific sequence of tasks and start times (genes) represents one genome in our population. To make sure that our genome is a [feasible solution](https://en.wikipedia.org/wiki/Candidate_solution" \o "Candidate solution) we must take care that it obeys our precedence constraints. We generate an initial population using random start times within the precedence constraints. With genetic algorithms we then take this initial population and cross it, combining genomes along with a small amount of randomness (mutation). The offspring of this combination is selected based on a [fitness function](https://en.wikipedia.org/wiki/Fitness_function" \o "Fitness function) that includes one or many of our constraints, such as minimizing time and minimizing defects. We let this process continue either for a pre-allotted time or until we find a solution that fits our minimum criteria. Overall each successive generation will have a greater average fitness, i.e. taking less time with higher quality than the preceding generations. In scheduling problems, as with other genetic algorithm solutions, we must make sure that we do not select offspring that are infeasible, such as offspring that violate our precedence constraint. We of course may have to add further fitness values such as minimizing costs; however, each constraint that we add greatly increases the search space and lowers the number of solutions that are good matches.