Simulation of Genetic Algorithms : Traffic Light Efficiency

Abstract

Traffic is a problem in many urban areas worldwide. Traffic flow is dictated by certain devices such as traffic lights. These traffic lights signal when each lane is able to pass through the intersection. Often, static schedules interfere with ideal traffic flow. The purpose of this project was to find a way to make intersections controlled with traffic lights more efficient. This goal was accomplished through the creation of a genetic algorithm, which enhances an input algorithm through genetic principles to produce the “fittest” algorithm. The program was comprised of two major elements: coding in Java and coding in Simulation of Urban Mobility (SUMO), which is an environment that simulates real traffic. The Java code called upon the SUMO simulation via a command prompt which ran the simulation, received the output, altered the algorithm, and looped. The SUMO component initialized a simulation in which a 1 x 1 street layout was created, each intersection with its own traffic light. Each loop enhanced the input algorithm by altering the scheduling string (dictates the light changes). After the looped simulations were executed, the data was then analyzed. This was accomplished by creating an algorithm based upon "regular" practice – timed traffic lights – and comparing the output which was comprised of the total time it took for all vehicles to exit the system and the average time it took each individual vehicle to exit the system. These different variables: the time it took the average vehicle to exit the system and total time for all vehicles to exit the system, where then graphed together to provide a visual aid. The genetic algorithm did improve traffic light and traffic flow efficiency in comparison to traditional scheduling methods.

Introduction

- Traffic light scheduling is inflexible and produces traffic problems as a result
- Therefore, a flexible algorithm must be utilized to adapt to a changing environment; an algorithm which is efficient in most any situation
- A traffic light scheduling algorithm that takes a certain set of given parameters (number of cars, speed, acceleration. Etc.) and creates an efficient schedule would yield the best results
- As such, a genetic algorithm was created in an attempt to solve all of these complexities and issues in Java code
- A simulation portion was needed to yield data for the genetic algorithm
- Simulation of Urban Mobility (SUMO) was used for the purpose of traffic simulation

Engineering Goals

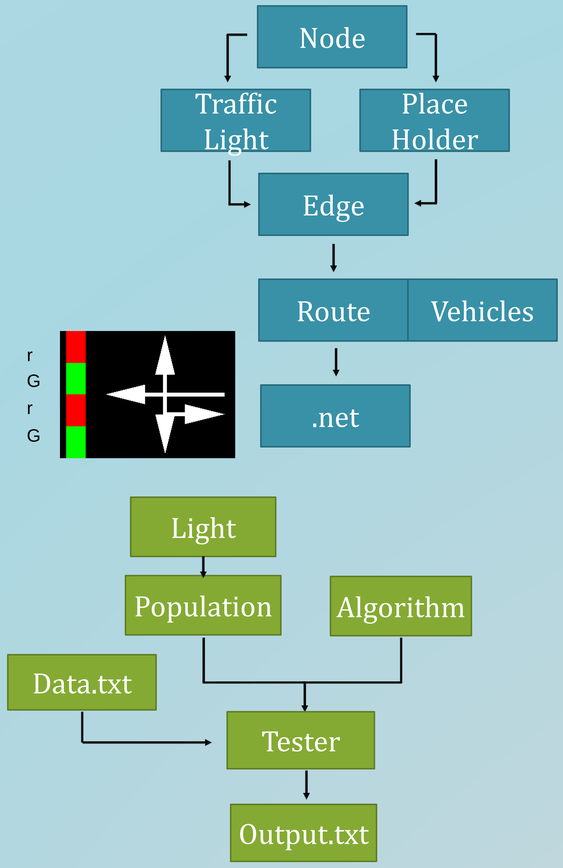
- Create a genetic algorithm in Java
- Create a SUMO simulation environment that contained a traffic light and vehicles
- Combine the two aforementioned components into a program that allows the genetic algorithm to function by running multiple times
- Utilize this genetic algorithm to create a more efficient algorithm based upon the original schedule

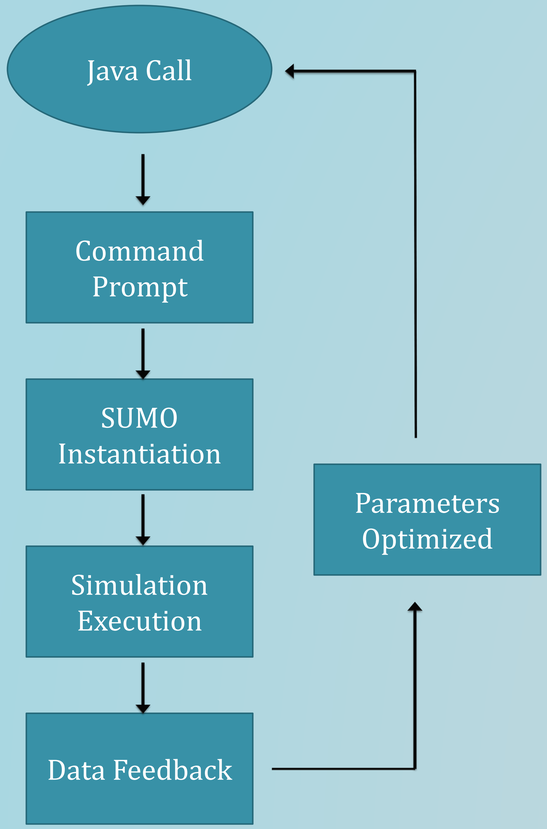
Program Overview

- Two main portions
  1. Java code – Genetic Algorithm
  2. SUMO code – Simulation Environment

- SUMO
  - Created using .xml files
  - Nodes – represent a point in the simulation; create edges
  - Edges – represent the edge of the road; create routes
  - Routes – represent the road; also contain the creation of vehicles within the simulation
  - TLS – contains the schedule for the traffic light, each set of four dictates one of the four lights
    Ex. “GGggrrGrGGggrrrr”
  - NETCONVERT – creates a .net file which compiles the prior .xml files
  - SUMO – allows for the TLS file to be implemented within the simulation

- Java
  - Light – the individual that adapts and is created randomly
  - Population – the combination of all Lights from a certain generation
  - Algorithm – uses the math and allows the population to adapt
  - Tester – summated all aforementioned classes and allowed the output to be written as a file and read incoming data from past iterations





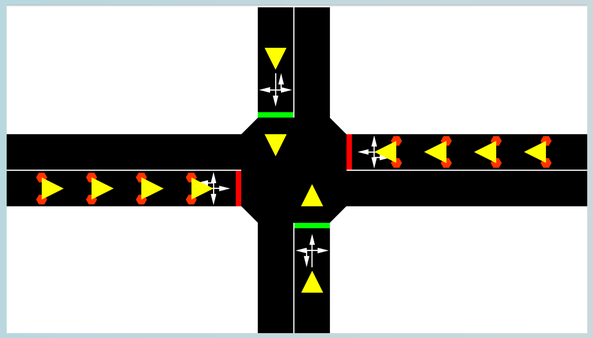
SUMO Code

```
<nodes>
        <node id="N" x="0.0" y="100.0" />
```

```
<edges>
        <edge from="N" id="Ntocenter" to="center" />
        <edge from="center" id="centertoS" to="S" />
```

```
<routes>
        <route id="up" edges="Stocenter centertoN"/
<vehicle id="1" type="Car" route="left" depart="0"/>
```

```
<additional>
<tlLogic id="center" programID="finalcode" offset="0" type="static">
<phase duration="31" state="GGggrrGrGGggrrrr"/>
<phase duration="30" state="ryyGGrGrryyGGGG"/>
</tlLogic>
</additional>
```



Screenshot of SUMO GUI

Genetic Algorithm

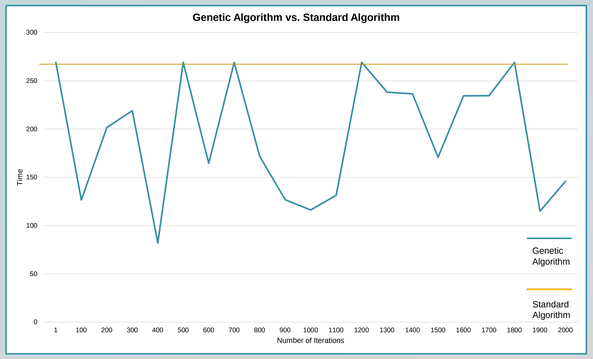
- An algorithm which uses the tenets of heredity to create the “best” algorithm given a certain set of parameters
- Two main components
  1. “Fitness” Calculator – assesses the relative efficiency of the individual
  2. Pool of Individuals – The set of individuals which “breed”
- Creates a pool with individuals that have random values
  - The randomness allows for mutations
- “Breeds” these different individuals to create a generation
- Calculates the fitness level of each individual
- Selects the most fit to selectively breed
- Process repeated thousands of times to “evolve”

Program Limitations and Assumptions

- SUMO modeled traffic accurately in relation to the real-world
- Allowing for the data to apply to real-world applications
- Genetic Algorithm was not given full array of parameters to change, limiting the results
- SUMO was found to be sensitive in that a single change in the traffic light string could cause a dramatic shift in traffic time

Results

- The genetic algorithm was found to increase efficiency by 92.1% after 2000 iterations.
- This shows the potential for the genetic algorithm to improve the efficiency even though the function was oscillating, suggesting that the genetic algorithm parameters were too constrictive



Future Work / Work Cited

- Expand setting to include multiple traffic lights to allow the development of an algorithm which makes the lights function together
- Implement a real-time data collecting device to allow the algorithm to function real-time and adapt to changing conditions
- Collect real life data to eliminate confounding factors

- Al-Khateeb, Khalid. "Dynamic Traffic Light Sequence Algorithm Using RFID." *Dynamic Traffic Light Sequence Algorithm Using RFID*. N.p., n.d. Web. 27 Feb. 2014."
- Java Platform SE 6." Oracle Documentation. Web. 12 Dec. 2013.
- Hardesty, Larry. "Eliminating Unexplained Traffic Jams." *MIT's News Office*. MIT, 27 Oct. 2013. Web. 27 Feb. 2014.