Public\_transit code documentation

In order to run the public\_transit file, one must first build a workspace in Eclipse. The structure of the files within the public\_transit file is recognized only by Eclipse. This was done both for ease of use and due to the fact that the previous simulations had all been written using Eclipse.

The following is a detailed description of each file in the public\_transit folder.

**src**

The source (src) file contains 2 folders which in turn contain all of the source code for the simulations. The first simulation is the ridership of boarding/alighting on buses.

**ridership**

In the ridership folder we have several files but the main ones to focus on are BusTest.java, BusInfo.java and Ridership.java. Ridership.java was built by Tim Sider. The BusTest.java file was built between both myself and Mr. Sider. It currently takes in input of sorted.csv (which can be found in the input folder). This file can but should not be changed. The output file is ridership\_all.csv (which can be found in the output folder). This contains the simulation output of boarding/alighting for all buses in the Montreal. It also relates bus i.d. and bus lines for future reference. As many of these simulations as is necessary can be run. Each iteration should take about 3 hours for all of the data.

The BusInfo.java file was created in order to facilitate use of the data. Like the BusTest.java file it is written using common Java OOP structure but it does not include any unnecessary class variables. This meaning that all necessary variables can be accessed using a BusInfo object. After data organization, the main function of this file is to compute time, speed and distances. Speed is generated from the calculateSpeed() method which can be done on a per object basis. Time is generated by the generateTime() which intakes BusInfo[] but does not output anything. It just adds the time to the time variable for each object in the array. This procedural form of programming was done due to the unavoidable calculation between objects. Distances were calculated using ArcGIS software and all exceptions have been accounted for. generateDistance() intakes BusInfo[] and adds a distance column to the end. The final output file with distances is labeled as distance\_all\_FINAL.csv. Due to the large amount of exceptions calculated in the distance file, if more ridership simulations are made, it is highly recommended to simply copy-paste the distance column to the new ridership file. This will avoid any future errors involved in both GIS and the software.

Summary

BusInfo.java – Input: sorted.csv (do not change this)

Output: ridership\_all.csv (can be run in 3 hours for as many simulations as is required)

BusTest.java – Input: ridership\_all.csv (any ridership of similar format can be used)

Output: distance\_all\_FINAL.csv (includes distance column at the end which should be copy-pasted into any future ridership files being made).

Ridership.java – ridership model used by BusInfo.java

**gasemissions**

The gas emissions folder contains various source code files as well but only 2 of these are of importance. EFTable.java and Emissions.java.

EFTable.java is a class which creates an object that intakes the emission factor data from the given emission factor tables. All of these tables can be found in the EF\_New folder. This file contains emissions factors specified by grade, season, bus year, road type and bus type. This class allows each table to be taken into the RAM only once while running the file. This means that the program does not need to continuously read through the files in the EF\_New folder. Also, the setSpeed\_passengers() file restructures the data in order to interpolate between all values. This allows the array to be accessed by getSpeed\_passengers() using single values. The EF tables are further described below in the EF\_New section.

Emissions.java is the file that connects the bus lines directly to the emissions and further computes both active and idle emissions of each bus between each bus stop in the Montreal area. The generateEmission() functions intakes Emissions[] and then fills in the total emissions variable by summing up the idle and active emissions which can be found on lines 185 and 188 of Emissions.java. Furthermore, this function also linearly interpolates between bus years between the EF\_New files. The peopleTime() function calculates the time the bus spends idle using boarding and alighting found in the ridership simulation. It then takes the larger of boarding or alighting and creates a standard normal distribution to compute the final idle time of the bus at each stop. Also, the generateYear() method uses the randomStandardYear() and randomArticulatedYear() function to randomly generate a bus based on the given Montreal bus distributions. This inputs distance\_all\_FINAL.csv and outputs EF\_all.csv. The EF\_all.csv file contains all of the class variables of the Emissions.java file including the final output emissions of each bus in grams.

Emissions.java – Input: distance\_all\_FINAL.csv (which uses EFTable.java)

Output: EF\_all.csv (all bus stops with emissions and other variables)

**EF\_New**

This folders contains files developed by Ahsan Alam. They contain emission factors both in grams/mile for active emission and grams/hour for idle emissions. Each file is uniquely separated by road grade, road type, bus year, bus type and season. Within each file we find number of passengers on the x-axis and bus velocity (mile/hour) on the y-axis. Based on these 2 values, an emission factor can be picked out of the table. Due to the missing passengers and missing bus years, the EFTable.java class interpolates between the passenger values and creates and array with the interpolated values. The Emissions.java class then also interpolates between file years when calculating the final emissions.

**Inputs**

The Inputs folder contains the bus\_line\_distances folder which contains all of the distances between each bus stop for each bus line in Montreal. This data was obtained from ArcGIS software using maps developed by McGill transportation research teams. The stm\_data folder is all of the data that was directly obtained from the STM(Socité de transport de Montréal ) developer site. This, together with field data, is what all simulations and data are based off of. The distance\_all\_FINAL.csv file can also be found here which contains one simulation and all of the correct distances are also included in the last column.

**Outputs**

The GISPoints folder contains all of the longitudes and latitudes of each bus stop. These files were used in order to later calculate distances using ArcGIS. The Bus\_Line\_NOREPEAT file was used because it did not repeat the same bus lines at different times of day with the same distances. The ridership\_all.csv file can be found here which contains a ridership simulation without the distances. The final EF\_all.csv file is contained here.