**TMG Bill Toolbox – Extract Emme Paths**

**Namespace:** bill\_toolbox.extractpaths

**Authors:** Bilal Yusuf, Yuan Hong (Bill) Sun

**Date Completed:** July 25th, 2017

**Programming Language:** Python 2.7

1. **Background**
   1. **General Concept**

In path choice modelling, one needs to compare the observed paths taken in real life from a survey to the paths generated by the computer simulation. This is to analyze the similarity of the modelled paths and the observed paths, to ensure the correctness of the modelling.

For a path choice analysis of the Greater Toronto Area (GTA), Emme Modeller [1] was used in conjunction with XTMF [2] to allow for repeated transit assignments using different sets of parameters, and the path choices generated by each assignment is compared to the observed paths gathered by Transportation Tomorrow Survey (TTS) [3] in 2012. XTMF uses a particle swarm optimization program to generate different values for the parameters that optimize the similarity between the two data sets.

* 1. **Path Proportion**

Every path is defined by its origin-destination (OD) pair, as well as all the segments of public transit routes taken (all the other path details are omitted for simplicity). A proportion value, ranging from 0.0 to 1.0, represents the probability that a path is taken in Emme for a certain origin-destination pair; however, paths with proportion values that are too low (<0.1%) can be neglected since they are very unlikely to be taken, and would not significantly affect the final results. From testing, it was found that not limiting the path proportions would allow Emme to generate path files that are too large to be analyzed efficiently, as many paths with extremely small proportions are included.

1. **Overview**
   1. **Program Specifications**

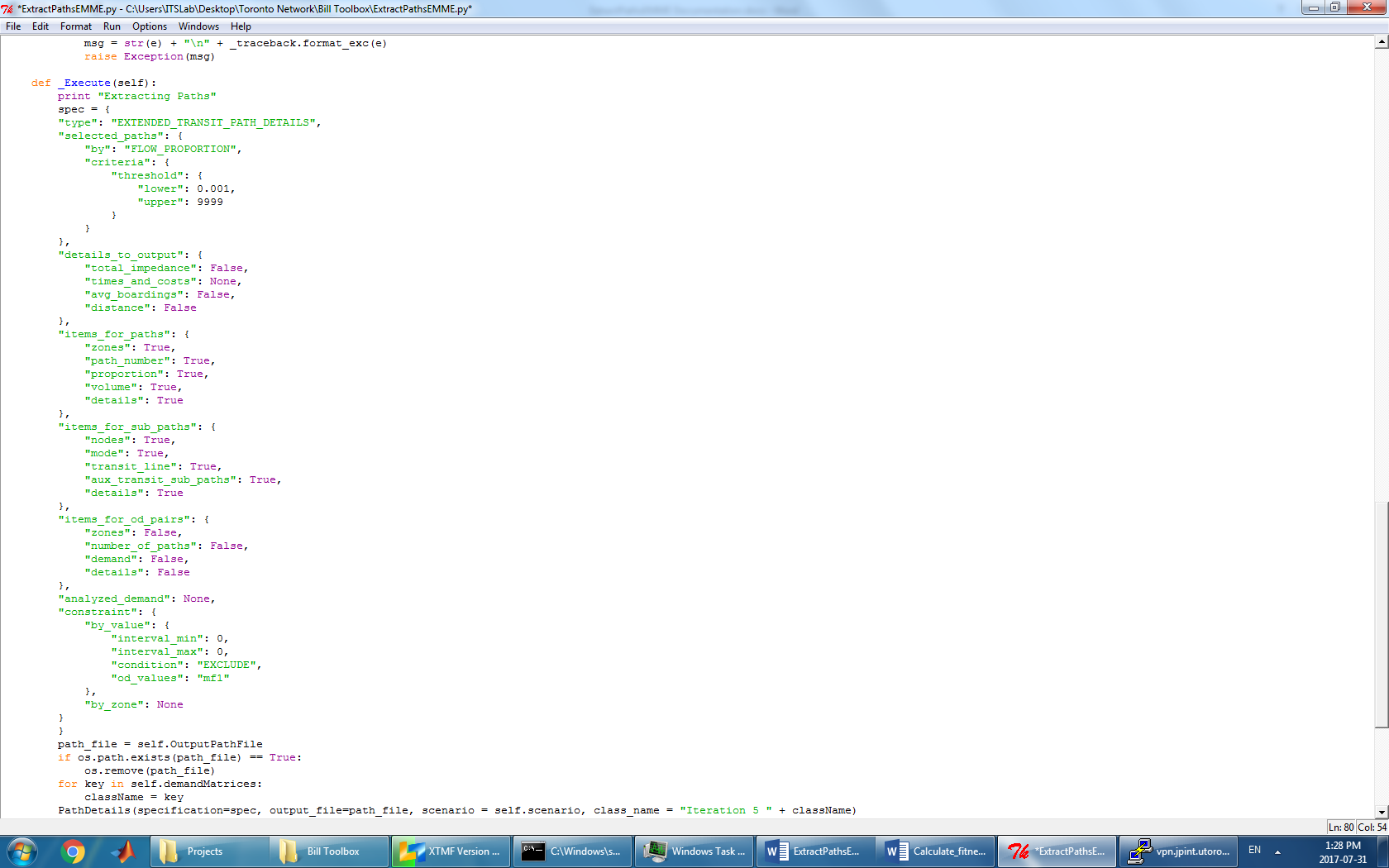
The Calculate Fitness program is programmed using Python 2.7, the default programming language used by Emme 4. It is developed to be used alongside XTMF, by calling it from an “EmmeTool” module. This program cannot be ran separately on Python, or ran directly using Emme.

The program takes exactly two arguments when called from XTMF:

1. Emme scenario number as an integer
2. Location of Emme-generated path file as a string
   1. **Structure Overview**

The program is structured under the main class called ExtractPathsEMME (\_m.Tool ()). The \_\_call\_\_ () function defines all the variables, and the \_Execute () function specifies the format for all the extracted paths and extracts the paths by calling the Path Details tool in Emme. The *spec* dictionary under \_Execute () details the path format and specifications.

The program selects a minimum set of information for the paths to improve runtime efficiency. The proportion for the paths are limited to greater or equal to 0.1% to filter out paths that are unlikely to be taken. This makes the Emme-generated path files a lot smaller in size, thus improving the runtime.

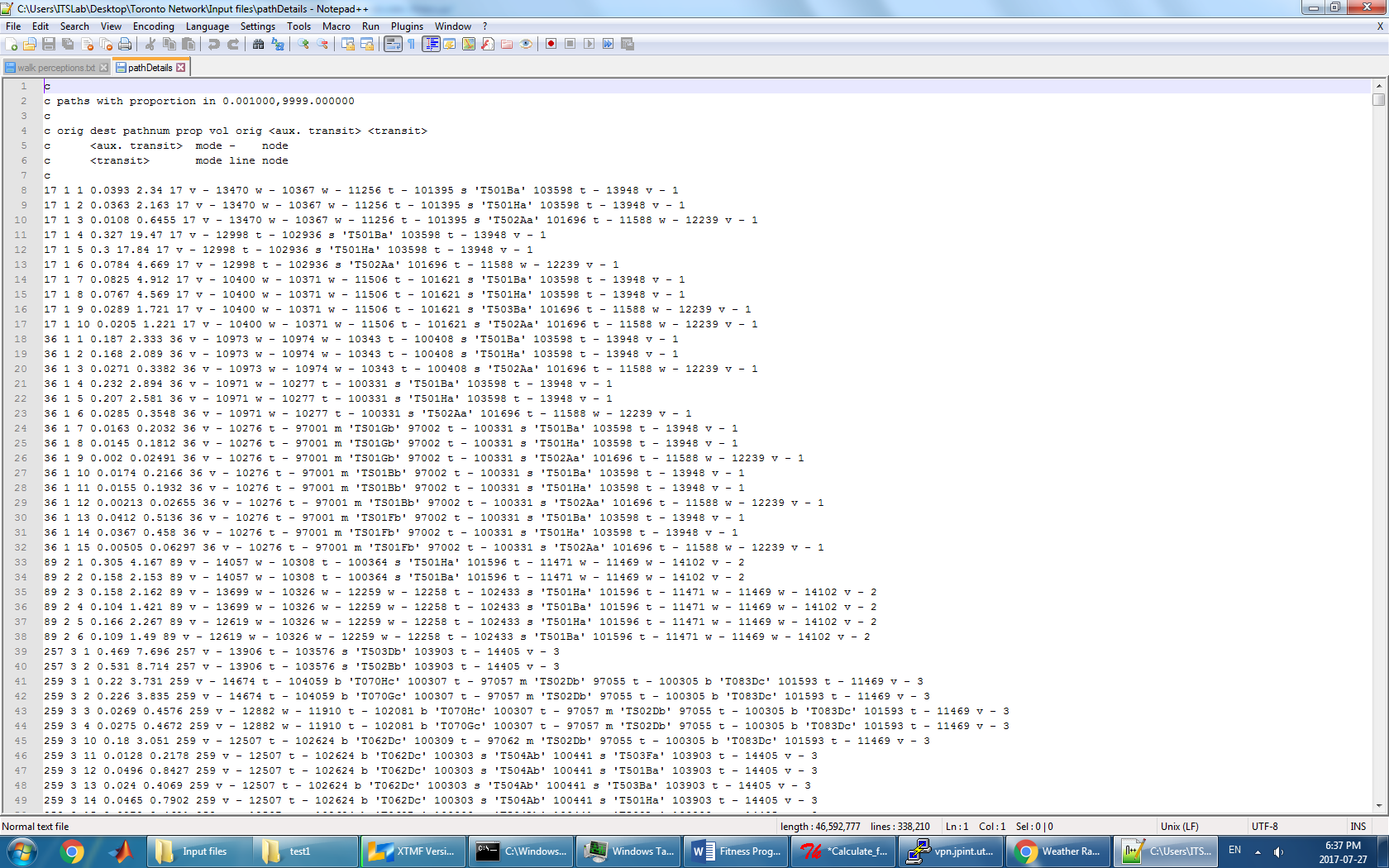


1. **Program Details**
   1. ***spec* Dictionary**

The *spec* dictionary specifies the path format and properties. Based on the requirements of the Path Details tool in Emme, it had to be in the format shown on the right. The attributes that are chosen to be displayed are selected “True”. All other attributes are selected “False” or “None”. Note that at the top, under “selected\_paths” and “criteria”, the paths are selected by proportion (threshold) with a lower bound of 0.001 (0.1%) to filter out paths that are very unlikely to be taken.

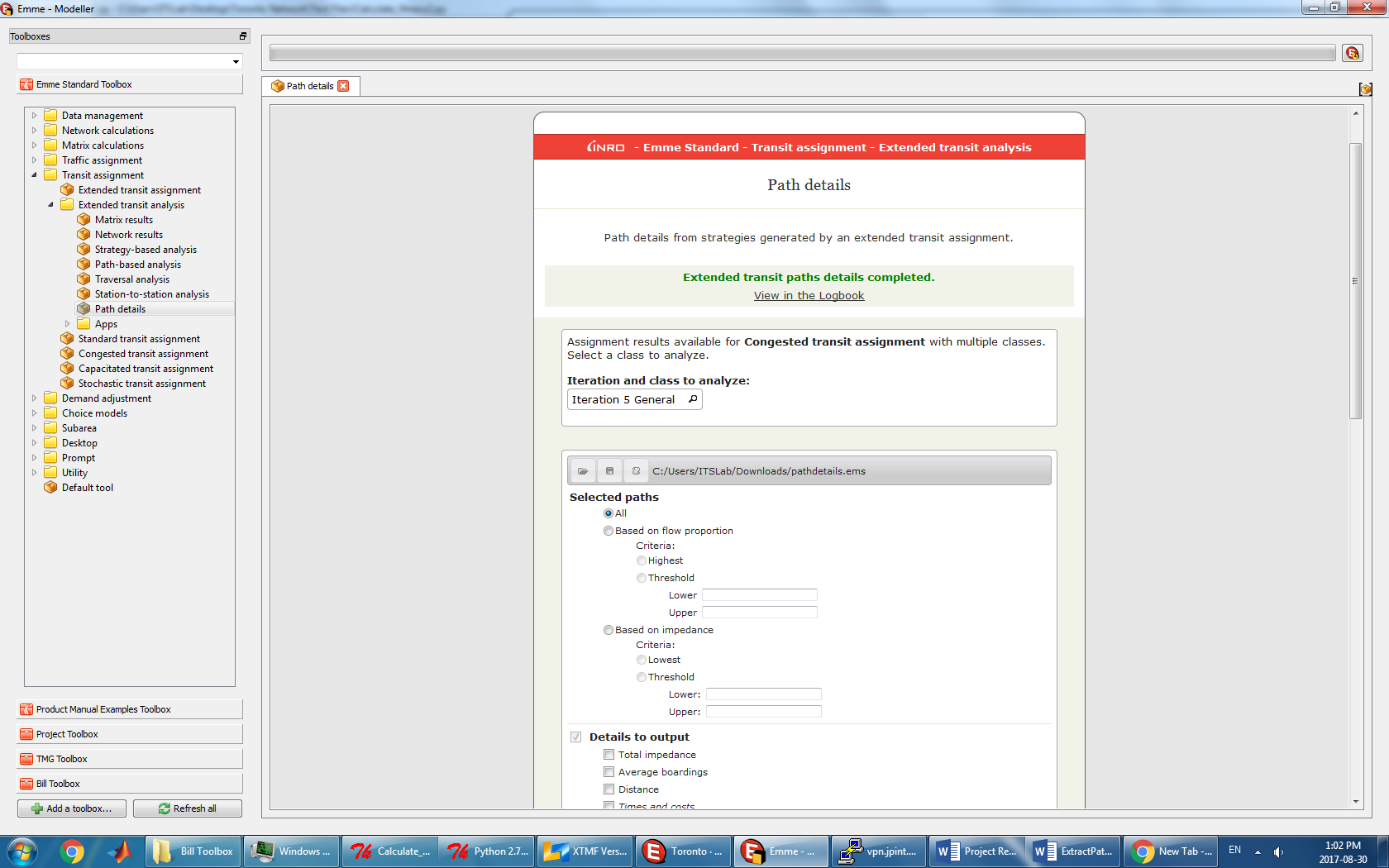
* 1. **Output Data**

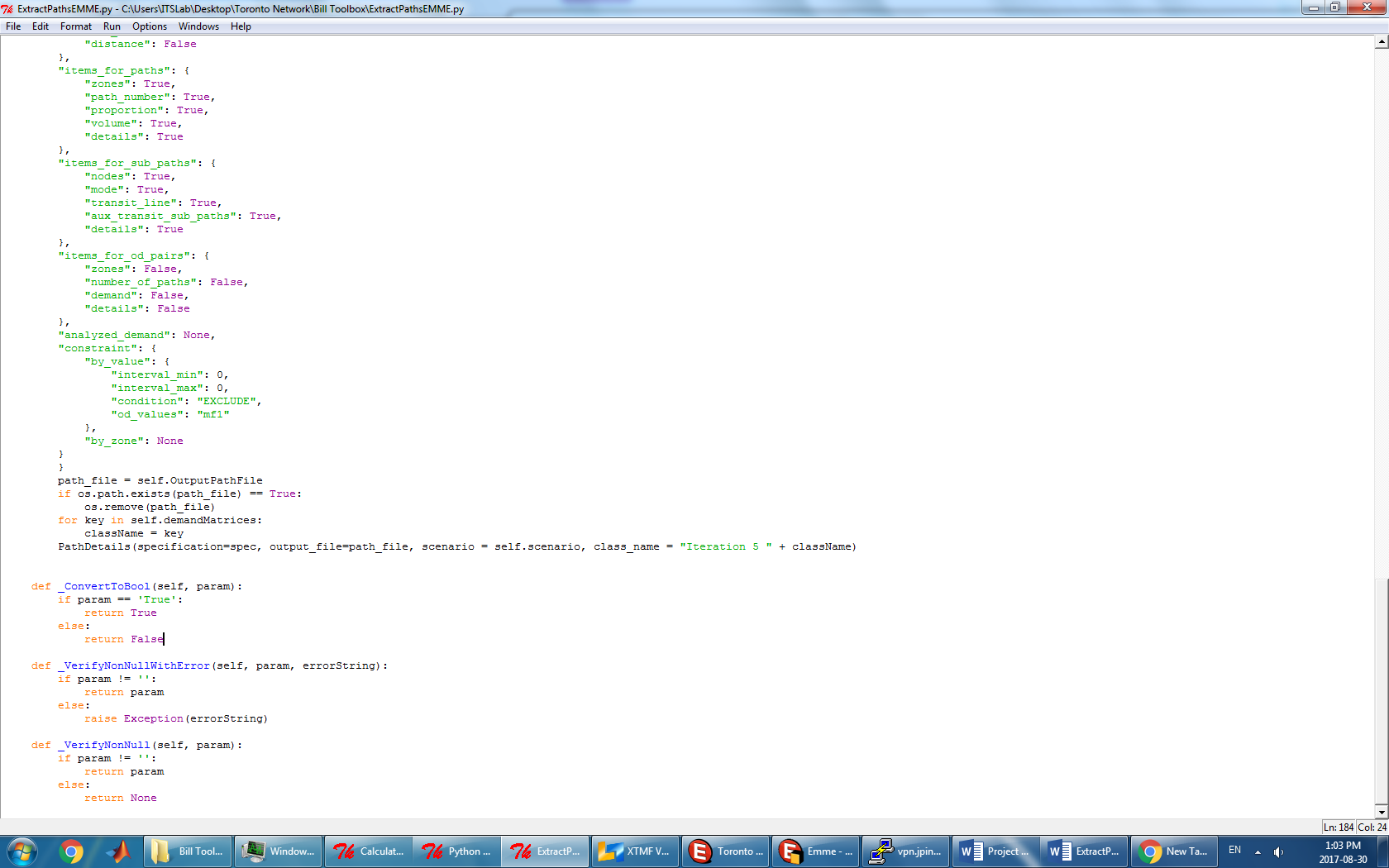
Below is a sample of the output Emme path details file. The first two columns are the origins and destinations. The third and fourth columns are proportions and volumes, respectively. Each path segment includes mode, transit line ID (or a dash if the mode of the segment is not transit), and the destination node.



* 1. **Iteration Number**

The congested transit assignment is ran with 6 iterations, numbered 0 to 5. As the iterations progress, the path data usually becomes better (more similar to the observed path data) as Emme better organizes them. Therefore, to get the best possible path data, the iteration number is set to 5 for all the model runs.





**Appendix**

1. Emme

<https://www.inrosoftware.com/en/products/emme/>

1. XTMF

<http://www.ecf.utoronto.ca/~miller/TMG-XTMF-Documentation.pdf>

1. TTS

<http://www.transportationtomorrow.on.ca/>