# Technical Memo

|  |  |
|---|---|
| **To:** | SHRP2 - C10 Tri-Agency Project Implementation Files |
| **From:** | Billy Charlton & Lisa Zorn |
| **Date:** | July 15, 2015 |
| **Subject:** | Fast-Trips Development Plan - Working Draft |

## Introduction

The purpose of this document is to propose a list of development priorities for the Python version of Fast-Trips, now that we know more about it.

Decisions need to be made around the order of priorities with the overall objective of front-loading performance and usability features as much as possible while still meeting the schedule for the feature development needs of other tasks.

Considerations include when each feature is needed for other tasks and the overall usability of this version of Fast-Trips.  Related tasks include Task 4, which will use it for choice set generation in September, and Task 8, which will use it for integration, beginning in the fall.

## Background

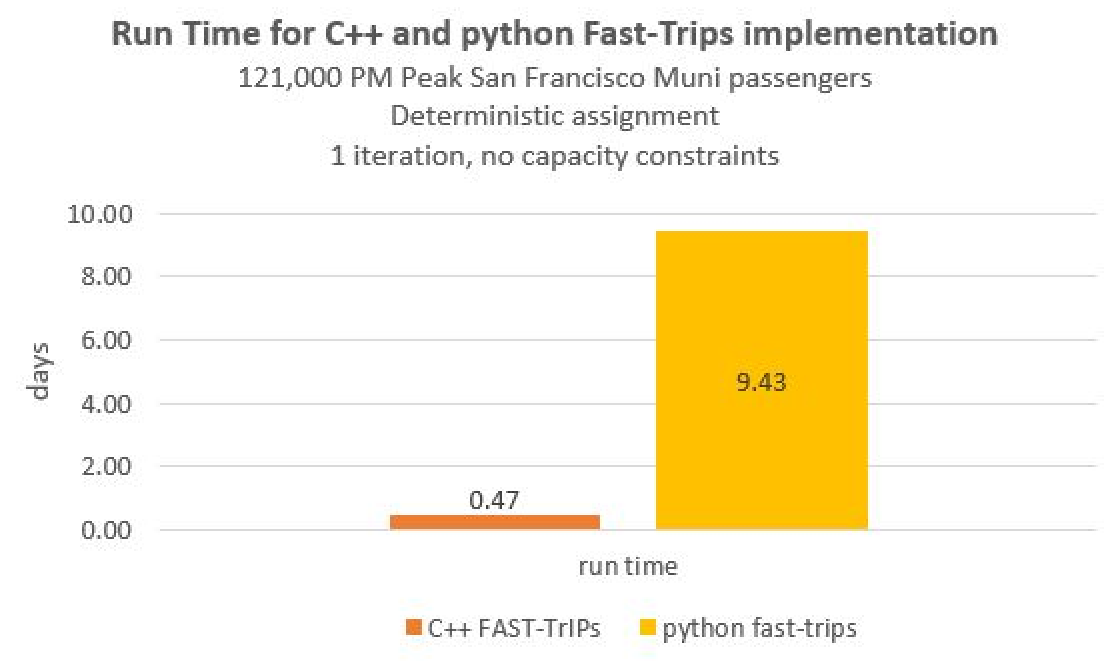
The original workplan highlighted the following items for Fast-Trips development in rough order:

1. Translation to Python
2. Read new input formats for networks and demand
3. Control capabilities to turn on/off various features
4. Crowding implementation
5. Multi-class
6. Skimming / choice set generation
7. Intermodal travel capabilities (similar to what was used at SACOG implementation)
8. Performance

The intermodal travel capabilities feature is desired to be able to represent the dynamic filling of park and ride lots and the skimming / choice set generation capabilities are needed in order to estimate the transit route choice model (Task 4) and to integrate it into SF-CHAMP and SoundCast (Task 8).

At the time of writing the workplan we noted that “we knew we didn’t know all that there is to know” and that we would revisit this priority list following the translation into Python when Lisa would have had the opportunity to read through all the code.

Lisa completed the straight translation of the Fast-TrIPs C++ code to Python in May. In the process, she fixed one bug present in the C++ code. She was able to verify the translation by running sets of 200 people through the network on both the Python and C++ versions of the code. In the process, she also noted that the crowding capabilities and multi-class assignment capabilities were already essentially “there.”



While those are two steps forward, Lisa also confirmed what we had suspected to be the case with a direct Python translation: it is slow. The Python version is orders of magnitude slower which would jeopardize its overall usefulness. This is not unexpected and there are numerous reasons for this including that Python works quickest when things are vectorized rather than when things run in for-loops (as they do in Fast-Trips). Fast-Trips was written for C++, not Python, so we are not surprised that we will need to change some of the internal structure to get it to work well in Python.

## Proposed Fast-Trips Development Plan

Because the performance was so slow, and because a speed-up may require changing some of the internal structure, we propose to elevate “performance” to the top of our list before tackling other features. The proposed new development plan is as follows in order of attack:

1. Critical Performance (sub six-hour run time for entire Bay Area, all time periods)
2. Read new input formats for networks and demand
3. Control capabilities to turn on/off various features and adjust parameters
4. Skimming / choice set generation
5. Intermodal travel capabilities (AKA park and ride) similar to what was used at SACOG implementation
6. Nice to have performance

## Details on Performance Attack

Performance is the foremost issue right now. Current run times are expected to be 1,666 hours per crowding iteration, so a 300x+ speedup is needed to meet the current performance target of six hours. The team decided to undertake the following items in order to better understand the performance issue:

- Investigate what is taking so long with SnakeViz
- Explore using Pandas for simulation
- Explore what else can be vectorized into NumPy / Pandas arrays
- Explore how far away Fast-Trips is from a traditional graph-search and consider employing something that does the graph search work in C such as:
    - Pandana
    - Graph-Tool
- Consider a Map-Reduce approach
- Consider compiling part of it in Cython or as a C extension if necessary

To date, several of these items have been undertaken.

- Using Pandas to Simulate resulted in a 2X speed improvement.
- **Using Cython for several methods** resulted in another 2X speed improvement; however, this has much more potential. Billy will be spearheading this. It is possible that in the end we will end up with a very fast c-based codebase with a Python wrapper.
- Investigation into Pandana revealed that it is unlikely that this tool would help.
- Investigations with SnakeViz revealed that approximately ⅓ of the time was spent label-setting and ⅔ of time was spent in path-finding. The entire process should be parallelizable because each passenger path can be found in parallel (this process includes both label-setting and path-finding).