Modelling Demand in Public Transport in SimMobility: Implementation Specifications

Author:	Tan Rui
Status:	Draft
Version:	0.8
Date:	3/23/2015

Table of Contents

0 Document Control

0.1 Summary

This document is the implementation specifications for modelling demand in DynaMIT 2.0 and SimMobility mid-term.

0.2 Document History

Version	Date	Author	Changes since last version
0.1	10 Mar 2014	Rui Tan	First Draft
0.2	17 Mar 2014	Rui Tan	Correct description error in Table 5-R_service_lines Added Filtering criteria in Section 4 Correct error-code in Section 4.1 Desired Path Format Added weight specification for each Label in terms of edge attributes in Section 4.2.1 Pseudo code for Labeling Approach
0.3	24 Mar 2014	Rui Tan	Updated Link Elimination approach in Section 4.3
0.4	02 Apr 2014	Rui Tan	Updated K shortest path approach in section 4.4 Updated Simulation Approach in Section 4.5
0.5	02 Jun 2014	Rui Tan	Added Section 5: Computing Path Attributes Added Section 6: Route Choice Model
0.6	17 Sep 2014	Rui Tan	Added Section 4 : Implementing stop pair choice set Updated Section 6 to include computing access walk and egress walk. Updated Section 7 to include stop-pair utility and stop pair selection probability. Updated Introduction and Implementation overview
0.7	27 Jan 2015	Rui Tan	Updated section 1 and Section 2 Updated Section 3: Network Data Model – new data model is described which contains access and egress links from origin nodes and destination nodes to public transport stops/stations Updated Section 4: Implementing choice set generation with respect to new network data model Updated Section 5: Computing path Attributes based on new network data model and new route choice model Updated Section 6: Route choice Model – multi-mode path selection. Removed Stop Pair selection.
0.8	23 Mar 2015	Rui Tan	Updated Section 3 for input data. Corrected table captions and added unites for edge attributes Updated Section 4.2 Labeling Approach with better explanation and cost function Updated 4.3 Link Elimination approach (The same with private side) Added 4.6 Path Feasibility Check

0.3 References

[1]“Tech_Specs_Public_Transport_Demand_DynaMIT_SimMobility”, version 0.4

[2]Yen, J. Y. (1971). Finding the k shortest loopless paths in a network. *management Science, 17*(11), 712-716.

[3]Tan, Rui; Robinson, S; Lee, D.H.; Ben-Akiva, M (2014) Evaluation of Choice Set Generation Algorithms for Modeling Route Choice with Smart Card Data in Large-Scale Public Transport Network.

0.4 Distribution

Document distributions will be recorded here.

0.5 Quality Assurance

Step	Description	Undertaken By	Date	Remarks
1	Quality Review			
2	Project Manager			
3	Executive Review			

1 Introduction

The purpose of the Implementation specification is to explain the key data model and implementation details for the demand models in modelling transport demand to a sufficient level that:

- A software programmer is able to code in Simmobility framework with reference to this document.
- The modeller is able to verify that what is being coded is what is required.

This document is arranged as follows. Section 2 provides an overview of this document to. Section3 specifies the network data model. Section 4 specifies the OD path set generation methods in detail. Section 5 will describe how to compute path attributes for the OD path choice sets. Section 6 explains the implementation of route choice model. Section 7 will describe the departure time choice model, followed by section 8 describing the en-route choice model and section 9 will explain the day-to-day learning model.

2 Implementation Overview

Figure 1 illustrates the necessary blocks for implementation of public transport demand models.

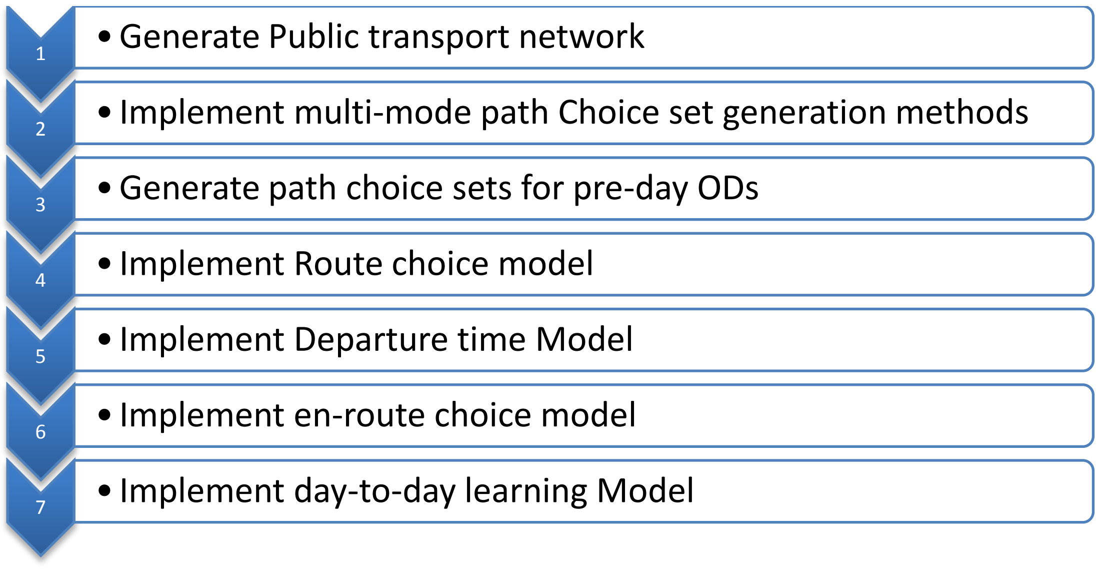
1
• Generate Public transport network
2
• Implement multi-mode path Choice set generation methods
3
• Generate path choice sets for pre-day ODs
4
• Implement Route choice model
5
• Implement Departure time Model
6
• Implement en-route choice model
7
• Implement day-to-day learning Model

Figure 1: Implementation overview of public transport demand models

3 Network Data Model

3.1 Input Data

The Network data used is based on Google Transit Network Data. It consists information on stops and service line schedules including trains (MRT, LRT) and buses.

All_nodes.csv file contains all Simmobility nodes, bus stops, MRT/LRT stations. They are having the following format as listed inTable 1.

Variable	Description
Stop_id	stop id for stops as specified by LTA. Examples: “NS27”, “CC15/NS17” for train stations; “14429”, for bus stops, “N_11456” for simmobility nodes. Note: For SimMobility Nodes, it is having the format of “N_” + the nodeId assigned to it in pre-day. Such modification is needed to avoid duplicated stop_id with bus stop_id
Stop_code	stop code for stops as specified by LTA. Examples: “NS27”, “CC15/NS17”
Stop_name	Name of stops as specified by LTA. Examples: “Marina Bay”
Stop_desc	Description of stops, usually the street where the stop is located. Example: “Marina Street”
Stop_lat	Latitude of stops. Example: 1.2761
Stop_lon	Longitude of stops. Example: 103.85467
stopType	#---stopType: #-----------0 Simmobility nodes (origin/dest) #-----------1 Bus Stops #-----------2 MRT/LRT Stations

Table 1: Data format for nodes including MRT/LRT stations and bus stops

Train service schedule information and bus services schedule information is extracted from the following data:

Variable	Description
trip_id	Service trip id. Example “100_saturday_1-S”
Arrival_time	Scheduled arrival time at the specified stop in stop_id. Example: “05:30:00”
Departure_time	Scheduled departure time at the specified stop in stop_id. Example: “05:30:00”
Stop_id	Stop id for the current stop along service line as specified in trip_id. Example “66009”
Stop_sequence	The sequence number for current stop as specified in stop_id along the service line as specified in trip_id

Table 2: Data format for train service lines and bus service lines

All_links.csv file contains all edges that links up Simmobility nodes, bus stops, and MRT/LRT stations. They are having the following format as listed in .

The final edges are having the following format:

Variable	Description
start_stops	Stop id for the starting vertex
end_stops	Stop_id for the ending vertex
r_type	type of current edge. Example: “Bus”, “LRT”, “Walk”, "Shuttle bus", "Cycle", and etc
road_Index	Index for road type: 0 for Bus, 1 for MRT/LRT, 2 for Walk
road_edge_id	Strings of passing road segments by current edge as specified in edge_id. Example:”4/15/35/43”, each number corresponds to a road segment between two adjacent stops along service lines. Road_edge_id containing single “edge_id” are referring to the road segment between two adjacent stops along service lines as specified in R_service_lines. Road_edge_id containing more than 1 “edge_id” are virtual edges representing traversing the edges sequentially without a transfer in between.
r_service_lines	if the edge is a route segement, it contains services line/direction that travelling on this route segment. If this edge is a walking leg, it is string “Walk”
edge_id	Id for current edge
link_travel_time	Estimated travel time on current edge. If it is a transit leg such as RTS or Bus, it is the estimated in-vehicle travel time on current route segment. If it is a walking leg, it is the estimated walking time on current walking leg. The unit is in seconds.
transit_time	Estimated travel time on transit legs for bus and MRT/LRT. It is extracted from google schedule files. The unit is in seconds.
walk_time	Estimated walk time for walk legs. It is estimated based on direct distance and 4km/h walking speed. The unit is in seconds.
wait_time	Estimated waiting time to embark on current edge. Data obtained from scheduled service line information. The unit is in seconds.
transfer_penalty	Transfer penalty-used to impose penalty on transfers for shortest path searching which assumes path cost is the addition of edge attributes. The unit is in seconds.
day_transit_time	Estimated transit time for buses and MRT/LRT in day time. Night bus services will have very large value in this column. The unit is in seconds.
dist	Estimated distance from travel time table. The unit is in km.

Table 3: Edge Data format for SimMobility network

Note that the edge here is corresponding to both the route segments and walking legs as defined in Reference [1], section 5.2.

4 IMPLEMENTING PATH CHOICE SET GENERATION METHODS

Four choice set generation methods will be implemented, and there are: Labelling approach, Link Elimination Approach, K shortest Paths Approach and Simulation Approach. The paths will be generated for each SimMobility OD pairs.

4.1 Desired Path format

As the shortest paths generated at this step will be subsequently sent to supply side for simulation, the desired path format is a list of transfer points along the path, starting from the origin point and ends at the destination point.

The desired paths choice set $P_{S_s S_t}$ for OD pair $\{S_s, S_t\}$ contains set of paths specified in the following format:

$$p_i = \{S_s, S_1^i, S_2^i, S_3^i, \ldots\ldots S_{Q_k}^i, S_t\}$$

Where p_i is the *i*th path in $P_{S_s S_t}$, S_s is the starting stop, $S_1^i, S_2^i, S_3^i, \ldots\ldots S_{Q_k}^i$ are the transfer nodes in sequence where Q_k denotes the number of the last transfer node before reaching ending stop S_t.

A necessary conversion from the generated path format by embedded shortest path search algorithm to the desired format is needed.

4.2 Labelling Approach

Labeling approach searches shortest paths based on different cost functions as defined as "Labels". In current implementation, the following labels can be included.

Labels	Label Description
1	Minimal total in-vehicle travel time
2	Minimal number of transfers
3	Minimal walking distance
4	Maximized travel on MRT
5	Maximized travel on Bus
6	Minimal waiting time at transfers
7	Minimal total travel time1 (in-vehicle +waiting +walking)
8	Minimal total travel time 2(in-vehilce+10waiting+5walking)
9	Minimal total travel time 3(in-vehilce+10waiting+10walking)
10	Minimal total travel time 4(in-vehilce+10waiting+20walking)

Table 4: Labels for implementation

4.2.1 Pseudo code

Below is an example of pseudo code for labeling approach:

```
Initialization:
    Searched Paths Q=empty.
    Search Index i=1
Procedure:
    Step 1: Generate cost function of label i,
```

Different cost functions for different labels could be realized by using the same cost function but varing the edge attributes. Cost = day_transit_time + walk_time + wait_time + transfer_penalty. Edge attributes are updated as follows to find different paths: (Note that, when searching a path for a different label, always reset the temp_links to origin links: All_links.)

Label 1. Minimal total in-vehicle travel time:

temp_links[, walk_time:=0L]

temp_links[,wait_time:=0L]

Label 2. Minimal number of transfers

temp_links[, day_transit_time:=0L]

temp_links[, walk_time:=0L]

temp_links[, transfer_penalty:=1L]

temp_links[,wait_time:=0L]

Label 3. Minimal walking distance

temp_links[r_type=="Walk", walk_time:=Big_Value]

Label 4. Maximized travel on MRT

temp_links[r_type!="RTS",transfer_penalty:=Big_Value]

Label 5. Maximized travel on Bus
temp_links[r_type!="Bus",transfer_penalty:=Big_Value]

Label 6. Minimal waiting time at transfers

temp_links[, day_transit_time:=0L]

temp_links[, walk_time:=0L]

Label 7. Minimal total travel time1 (in-vehicle +waiting +walking)

Label 8. Minimal total travel time2 (in-vehicle +10waiting +5walking)

temp_links[, wait_time:=10L*wait_time]

temp_links[, walk_time:=5L*walk_time]

Label 9. Minimal total travel time3 (in-vehicle +5waiting +5walking)

temp_links[, wait_time:=10L*wait_time]

temp_links[, walk_time:=10L*walk_time]

Label 10. Minimal total travel time4 (in-vehicle +10waiting +20walking)

```
                temp_links[, wait_time:=10L*wait_time]
                temp_links[, walk_time:=20L*walk_time]

    Step 2: Search shortest path p_i based on current cost function
    Step 3: If p_i does not exist in Q, Q=Union{Q, p_i)
    Step 4: Increment I and check whether stop criteria has been met
Stop Criteria:
    I>10
```

4.3 Link Elimination Approach

Link Elimination algorithm searches alternative paths by eliminating edges along the found paths. During the search for the least cost path it is assumed that people will always chose the current least cost path given the current eliminated network conditions.

edges along the first shortest path i, has been eliminated once.

4.3.1 Pseudo code

Below is an example of pseudo code for Link Elimination approach:

```
Initialization:
    Cost function = day_transit_time+transfer_penalty+wait_time + walk_time
    Searched Paths Q=empty.
    Search Index i=1
Procedure:
    Step 1: search shortest path on network,denote it as path p_i, update Q=union(Q, p_i)
    Step 2: for each edge e along shorest path path p_i, eliminate this edge
    Step 3: search shortest path on eliminated network, denote the found path as p_j, update Q=union(Q, p_j)
    Step 4, resume network. and check whether stop criteria has been met. If no, go back to Step 2
Stop Criteria:
    All edges along the initial shortest path p_i has been eliminated once.
```

4.4 K shortest Path Approach

The K-shortest path algorithm of reference 2- Yen (1971) is adopted in this paper, with the following modifications to make it suitable for multimodal PT network: 1) instead of comparing node sequences, it has been modified to compare link sequences for determining links to eliminate, as parallel links are existing in our PT network; K could be set to 30 as the maximum size of observed choice sets is only 15 from the smart card data.

4.4.1 Pseudo code

```
Initialization:

    Cost function = day_transit_time+transfer_penalty+wait_time + walk_time

    Search shortest path p_1,

    Searched Path Q={p_1}

    temp paths list B={empty}

    Number of Paths limit K,

    Search Index k=1

Procedure:

    Step 1:Increment k-Search for k-th shortest path:


        Identify edge sequences to compare from p_(k-1). Denote edges in p_(k-1) as
    edge sequence{e_1, e_2, e_3,….e_q(k-1)} where e_q(k-1) is the SECOND last edge
    in p_(k-1) in sequence. q(k-1) is the [number of edges in p_(k-1) -1]

            • Initial search
              a) Eliminate first edge as in {p_1, p_2,…p_(k-1)} from the network
              b) Search for shortest path, denote as p_temp, if p_temp does not exist
                 in Q, or B, B=union{B, p_temp}. Otherwise, do nothing.
            • If q(k-1)>=1, start subsequent search:

        For i in 1: q(k-1)

        {

            a) check whether the sub-path {e_1, e_2,…,e_i} coincides with sub-path
               consisting of the first i edges in sequence for j =1 ,2,3, k-1.If coincides,
               eliminate edge e_(i+1) along that path.
            b) Denote subpath {e_1, e_2, …e_(i-1)} as S_i, Search for shortest path
               R_i with origin at Ending Vertex of (edge_i) and destine at the same
               destination.
            c) concatenate S_i and R_i to obtain A_i_k,
               if A_i_k does not exist in B nor exist in Q,
                       add A_i_k to B
                       B=union{B, A_i_k}
            }
        }
    Step 2: Compute path cost for each path in B, record the least cost path A_j as p_k.
        Remove A_j from B
Stop Criteria:
    B is empty, or k=K
```

4.5 Simulation Approach

In-vehicle travel time and walking time are both randomized following an independent and identically distributed normal distribution with mean equals to its original value and standard deviation is set to 5 times the original value. To avoid drawing a negative value, the absolute value is taken. 50 draws of randomized travel times were performed for each sample OD. The selection of sampling distribution and number of draws takes into consideration of the maximum size of observed choice set, coverage, and computational time.

4.5.1 Pseudo code

```
Initialization:
    Draws = N, assign distribution = f(u, m)
    function = Link_Travel_time+transfer_penalty+Waiting_time
    Searched Path Q={}
    Draw Index n=1
Procedure:
    Step 1: Resume Network
    Step 2: Incremental n and
    Step 3: Randomize edge cost (day_transit_time, walk_time, wait_time) by drawing
        the new edge cost from distribution f depends on current edge cost.
        For example, new_link_travel_time:=abs(norm(link_travel_time,
        5*link_travel_time))   absolute value of a normal distribution with mean =
        link_travel_time and standard deviation = 5* link_travel_time
    Step 3: Search shortest path p_n based on randomized network
    Step 3: If p_n does not exist in Q, Q=Union{Q, p_n)
Stop Criteria:
    n>=N
```

4.6 Path Feasibility Check

All paths need to be checked before putting into a choice set for route choice modeling. A feasible path should satisfy:

1) the total number of transfer <=6

2) No two consecutive walking edges along the path

3) Does not walk back to any SimMobility nodes from bus stop/stations in the middle of the path. (note that the access/egress walk transfer penalty is set to be much heavier than transfer walk to avoid such infeasibility.)

5 COMPUTING PATH ATTRIBUTES

Path attributes is computed based on the path composition and link attributes. Given a path consisting of a sequence of edges: {r1, r2, r3, r4….}, the following path attributes are to be computed:

- Total In_vehicle Travel Time
- Total Waiting Time
- Total Walking Time
- Total Number of Transfers
- Total Distance
- Total Cost
- Path-Size

5.1 Pseudo code for Computing Total In-Vehicle Time, Total Walking Time. Total Waiting Time, Total Number of Transfers, total distance

Initialization:

```
    Total in-vehicle time = 0.0

    Total Walking Time = 0.0

    Total Waiting Time = 0.0

    NumofTransfer=-1,

    TotalDistance = 0.0

    Get Path edge sequence: P={r_1, r_2, r_3,…, r_K} where K is the maximum number
of edges along path

    Compute index i = 1
```

Procedure:

```
    Total Waiting Time = Total Waiting Time + Wait_time(r_i)

    Total in-vehicle time = total in-vehicle time + day_transit_time(r_i)

    Total Walk Time = Total Walk Time + walk_time(r_i)
    NumofTransfer = NumofTransfer + 1
    TotalDistance = TotalDistance + dist(r_i)
    Increment i
Stop Criteria:
    i>K
```

Assign total in-vehicle time, total walking time, total waiting time, number of transfer and total distance to the path.

Note 1:

Besides number of transfers and total distance, the rest attributes should be updated during each simulation runs in the future.

Note 2:

For more accurate distance computation, summation of segments in road network should be applied.

5.2 Pseudo code for Computing TotalCost

TotalCost should be computed based on the distance traveled on Bus and MRT/LRTs. For Example:

Denote effective distance as the distance traveled on Bus service lines along the path:

```
    paths[effective_dist<=3.2, path_cost:=0.77]
    paths[effective_dist>3.2 & effective_dist<=4.2, path_cost:=0.87]
    paths[effective_dist>4.2 & effective_dist<=5.2, path_cost:=0.98]
    paths[effective_dist>5.2 & effective_dist<=6.2, path_cost:=1.08]
    paths[effective_dist>6.2 & effective_dist<=7.2, path_cost:=1.16]
    paths[effective_dist>7.2 & effective_dist<=8.2, path_cost:=1.23]
    paths[effective_dist>8.2 & effective_dist<=9.2, path_cost:=1.29]
    paths[effective_dist>9.2 & effective_dist<=10.2, path_cost:=1.33]
    paths[effective_dist>10.2 & effective_dist<=11.2, path_cost:=1.37]
    paths[effective_dist>11.2 & effective_dist<=12.2, path_cost:=1.41]
    paths[effective_dist>12.2 & effective_dist<=13.2, path_cost:=1.45]
    paths[effective_dist>13.2 & effective_dist<=14.2, path_cost:=1.49]
    paths[effective_dist>14.2 & effective_dist<=15.2, path_cost:=1.53]
```

```
paths[effective_dist>15.2 & effective_dist<=16.2, path_cost:=1.57]
paths[effective_dist>16.2 & effective_dist<=17.2, path_cost:=1.61]
paths[effective_dist>17.2 & effective_dist<=18.2, path_cost:=1.65]
paths[effective_dist>18.2 & effective_dist<=19.2, path_cost:=1.69]
paths[effective_dist>19.2 & effective_dist<=20.2, path_cost:=1.72]
paths[effective_dist>20.2 & effective_dist<=21.2, path_cost:=1.75]
paths[effective_dist>21.2 & effective_dist<=22.2, path_cost:=1.78]
paths[effective_dist>22.2 & effective_dist<=23.2, path_cost:=1.81]
paths[effective_dist>23.2 & effective_dist<=24.2, path_cost:=1.83]
paths[effective_dist>24.2 & effective_dist<=25.2, path_cost:=1.85]
paths[effective_dist>25.2 & effective_dist<=26.2, path_cost:=1.87]
paths[effective_dist>26.2 & effective_dist<=27.2, path_cost:=1.88]
paths[effective_dist>27.2 & effective_dist<=28.2, path_cost:=1.89]
paths[effective_dist>28.2 & effective_dist<=29.2, path_cost:=1.90]
paths[effective_dist>29.2 & effective_dist<=30.2, path_cost:=1.91]
paths[effective_dist>30.0 & effective_dist<=31.2, path_cost:=1.92]
paths[effective_dist>31.2 & effective_dist<=32.2, path_cost:=1.93]
paths[effective_dist>32.2 & effective_dist<=33.2, path_cost:=1.94]
paths[effective_dist>33.2 & effective_dist<=34.2, path_cost:=1.95]
paths[effective_dist>34.2 & effective_dist<=35.2, path_cost:=1.96]
paths[effective_dist>35.2 & effective_dist<=36.2, path_cost:=1.97]
paths[effective_dist>36.2 & effective_dist<=37.2, path_cost:=1.98]
paths[effective_dist>37.2 & effective_dist<=38.2, path_cost:=1.99]
paths[effective_dist>38.2 & effective_dist<=39.2, path_cost:=2.00]
paths[effective_dist>39.2 & effective_dist<=40.2, path_cost:=2.01]
paths[effective_dist>40.2, path_cost:=2.02]
```

5.3 Pseudo code for Computing Path-Size

Computing the path-size involving checking the overlapped edges of current path with respect to other paths in the same choice set. Given a choice set C_n for OD pair n, it contains N_n paths with path i having the format of {r_1, r_2,…, r_Ki} where Ki is the maximum number of edges/route segments along path_i. The mathematical formulation of the path-size function is:

$$PS_{in} = \sum_{r \in \Gamma_i} \left(\frac{t_r}{T_i}\right) \frac{1}{\sum_{j \in C_n} \delta_{rj}} \qquad \textit{Eq. (5.1)}$$

Where PS_{in} is the path size of path i for OD pair n; r is the indexed route segment, Γ_i is the set of all route segments along path i; t_r is the travel time on route segment r; T_i is the total travel time on path i; C_n denotes the set of path alternatives for OD pair n; δ_{rj} equals to 1 if route segment $r$is on path j and 0 otherwise; $T^*_{C_n}$ denotes the total travel time of shortest path in C_n.

The pseudo code for computing path-size for each path i in choice set C_n is then:

```
Initialization:

        Get Path edge sequence for each path in C_n: p_i={r_1, r_2, r_3,…, r_Ki} where Ki is the maximum number of edges along path p_i. i in the path index in choice set C_n ranging from 1 to N_n

        i=1

Procedure:

        Path-size=0

        Path_travel_time=summation of all Link_travel_time(r) along path_i

        Sub-path-size = 0 # used to store the path-size component  for each edge

        Sub-N = 0, # used to store the number of overlapped edge in the choice set

        For each edge/route segment r along path p_i

        {

                Sub-path-size= Link_travel_time(r)/ Path_travel_time

                For each path j in choice set C_n

                {

                        If r exist in p_j, Sub-N = Sub-N+1

                }

                Sub-path-size=sub-path-size/sub-N

                Path-size=path-size+sub-path-size

                Assign Path-Size to path_i

        }
        Increment i
Stop Criteria:
        i>N_n
```

Note: This pseudo code is the most straightforward realization without any optimization. It is highly suggested to come out with other efficient code based on the used data structure.

6 Route Choice Model

The pre-trip route choice model for public transportation in Simmobility is a path-size logit model that selects a multi-mode path from a choice set. Each path alternative in the choice set contains a path from origin node to destination node.

The utility for OD path i in the choice set C_i is given as follows:

$$U_i = \boldsymbol{\beta}\boldsymbol{X}^{\boldsymbol{T}} + \varepsilon_i = V_i + \varepsilon_i \qquad Eq.\ (6.2)$$

where $\boldsymbol{X}$ represents total denotes vector of path attributes including total in-vehicle travel time, total walking time, total waiting time, total distance, denottotal number of transfer, path cost, and path size. $\boldsymbol{\beta}$ is the vector of coefficients given by modeler, and ε_i represents the unobserved error component in path utility. V_i is the deterministic part of the utility which will be used to compute path selection.

In an agent-based simulation environmental, there are two ways to assign the passengers to each path in the choice set. The first approach is to follow path selection probability and assign the same portion of passengers to each path. Another way is to simulate each agent's path selection.

6.1 OD Path selection probability:

The path selection can be computed by:

$$P(i|C_n) = \frac{\exp(V_i)}{\sum_{\in C_n} \exp(V_j)} \qquad Eq.\ (6.3)$$