Project Seminar
Master of Science
at Technical University of Munich

From A to B - Identifying Interesting Multimodal Paths

Referent:	Operations and Supply Chain Management Prof. Dr. Maximilian Schiffer Technical University of Munich
Advisor:	Patrick Klein
Handed in by:	Erik Hammer Matriculation Number: 03722558 Hock Yin Lam Matriculation Number: 03721471
Handed in at:	31.03.2021

Table of Contents

List of Figures

List of Tables

List of Abbreviations

LRI	Likelihood Ratio Index
MAE	Mean Absolute Error
MLE	Maximum Likelihood Estimation
MLL	Maximum Log Likelihood
MNL	Multinomial Logit
MXL	Mixed Logit
NL	Nested Logit
NR	Newton-Raphson
PB	Private Bike
PC	Private Car
RU	Representative Utility
SF	Scale Factor
SP	Stated Preference
SS	Simulated Survey

List of Symbols

$U(a, b)$	Uniform distribution with start value a and end value b
$L(\beta)$	Likelihood function evaluated at β
$LL(\beta)$	Log likelihood function evaluated at β
$LL(\beta')$	Log likelihood function evaluated at a vector of scaled β $\left(\beta' = \frac{\beta*}{\sigma}\right)$
$LL(\hat{\beta})$	Log likelihood function evaluated at a vector of β estimates
σ	Scale factor
λ	Step size for Newton-Raphson algorithm

1 Introduction

The number of different available mode of transportations has increased with the emergence of “modern” transportation options like bike sharing, car sharing and electric scooters. With the transformation to a mobility as a service model the multimodal transport network is more interconnected than ever which leads to many more possible route alternatives and an increase in complexity. In many industries like online advertisements, data about the user is collected and utilized to provide personalized recommendations for the user. In contrast most of the current routing tools only show the fastest and/or the shortest route [5]. Also, most of the time “modern” modes of transportation as described previously are not included in route alternatives. By omitting these modern modalities and only showing the shortest and/or fastest route, current routing tools do not yield the best utility for the user since his individual preferences are not considered in the route calculation. For example, a user may prefer a route which minimizes the environmental impact, offers high comfortability, and includes public transport and bike as modes of transportation, rather than the fastest route using a car. This is why a new era of modern routing tools is emerging in which interesting, personalized paths are generated by measuring the utility of each path alternative [7]. The path cost is represented as a utility function in which multiple criteria are weighted by the corresponding user preferences. As the user preferences cannot be measured directly, an approach is needed to infer them. One way to do this is to give the user a set of exclusive path alternatives from which they can choose from and observe the choice they made. From the choice selection their preferences can then be estimated. This kind of problem, based on utility theory and choice selection, is called discrete choice problem; of which a vast amount of research has already been done.

However, research involving transportation networks including car and bike sharing as mode of transportations is underdeveloped as of now. Car and bike have mostly been incorporated in the network as personalized vehicles but not in the context of public transportation. Furthermore, the common approach to initialize the preferences of an individual user is using Stated Preference (SP) surveys. But so far to the best of our knowledge, no preference parameters have been initialized by asking the user to directly input his preferences on a dimensionally reduced set of parameters.

With this work, we close the gap in this field, by combining the "Simulated Survey" (SS) method which estimates the user parameters based on the input of a dimensionally reduced set of parameters by the user and the multimodal transportation network including bike and car sharing as options.

By directly asking for the user input no other information needs to be collected for the initialization as opposed to the previously mentioned methods. This approach can potentially solve the issue of requiring a certain number of observations before the user parameters can be initialized. For this purpose, we introduce the simple dimensionality reduction method SS. Additionally, we use the A* algorithm based on a multi criteria weighted utility function as path cost to generate path alternatives in an extended multimodal network by adding car and bike sharing as possibilities. Lastly, we compare the performance of our model with various metrics between a SP survey and a SS method. The results of which show promising implications for an SS approach.

2 Review of Literature and Research

Modeling the multimodal network as a discrete choice problem has been a popular research topic in the recent years. Modesti and Sciomachen [4] study a multimodal network with public, car and walk as available modalities and utilizes an adjusted version of the Dijsktra algorithm for path generation. They use a utility function which includes quantitative as well as qualitative features. But instead of learning the user preferences they use fixed theoretical values for all users.

Since not all users have the same preferences, other approaches have been developed to learn the individual user preferences from the user choices. Arentze [7] develops an approximation method for Bayesian inference for learning individual user preferences with the goal to reduce the computational time. He proposes an algorithm based on sequential belief updating of preference parameters and replaces random sampling with equal-interval sampling. The proposed procedure initializes each preference parameter of the MNL model by sampling from choice observations from multiple users obtained through SP Survey and updates the individual user preference incrementally as soon as a new user choice from this specific user is observed.

Even though the approximation method is more computationally efficient than the conventional Bayesian inference method, updating the preference parameters by sampling from a multidimensional distribution can still be computationally taxing. Instead of initializing the preference parameters by sampling from multiple users, Nuzzolo and Comi [1] propose an approximation method by sampling from choices of the same user obtained from an SP survey. A batch updating algorithm is used to update the preference parameters after a certain number of observed user choices is reached. They also compare the performance of learning user preferences between Multinomial Logit (MNL), Mixed Logit (MXL) and Nested Logit (NL) models. Additionally, the performance between individual and average group utility models are investigated. The results imply that the individual should be preferred over average group utility model since it yields better path advice performances. Regarding the discrete choice model, they conclude that the MNL model is the most suitable approach for processing large amounts of data, since it produces satisfactory results given its simplicity.

Campigotto et al. [5] introduces a method called FAVOUR that utilizes SP for single users as well as user group information for initializing the preference parameters.

The FAVOUR method consists of 3 stages. In the first stage the users are grouped into classes based on answers on socioeconomic questions. In the second stage, SP surveys are personalized based on the obtained information in stage 1. The prior probability is further refined by mass-preferences prior transfer learning utilizing information from travelers in the same class. In the last stage the personalized preference parameters are incrementally updated with new user choices. The key contribution of their work is using the mass-preferences prior, in order to reduce the number of required questions for initializing good preference parameters.

2.1 Summary

Table 1 provides an overview of the key characteristics in the field of discrete choice modelling relating to the previously introduced publications and this paper. To the best of our knowledge the approach of initializing the preference parameters by asking the user to directly input his preferences on a dimensionally reduced set of parameters has not been explored. The previously introduced publications all utilizes the SP survey for parameter initialization. In addition, only one of the publications included car and bike sharing as available mode of transportations.

This paper closes this gap by comparing the performance of using direct user inputs and SP surveys for parameter initialization.

Table 1: Overview of discrete choice models

	I	II	III	IV	V
Model	RU	MNL	MNL, MXL, NL	MXL	MNL
Parameter Initialization	Fixed Values	SP – Multiple Users	SP – Single User	SP + Mass Preference Prior	SP – Single User & SS
Learning Method	-	Incremental	Batch	Incremental	Batch
Inference Technique	-	Bayesian	MLE	Bayesian	MLE
Path Generation	Dijkstra	-	-	-	A* Star
Multi Modal Network	PC Public Walk	PC Public	Public	PC, PB Public Walk Sharing	Public Walk Sharing

Indices I – IV signify publications as follows: (I) Modesti, Sciomachen (1997); (II) Arentze (2013); (III) Nuzzolo, Comi (2015); (IV) Campigotto et al. (2016); (V) This paper

3 Methodology

In the proceeding sections the structure of our model is described as well as the assumptions taken. Starting with 3.1, discrete choice model theory, selection, and representative utility function are derived. Next, referring to figure.1, section 3.2 encompasses path generation and explains the transportation network used in combination with an A* search method. In 3.3 the benchmark choice set assumptions are described which lay the foundation for our dimensional reduction and simulated survey approaches. Section 3.4 defines β-parameter estimation with maximum log likelihood function and details Newton-Raphson's method as the numerical solver. Finally, in 3.5, limitations of the model are discussed.

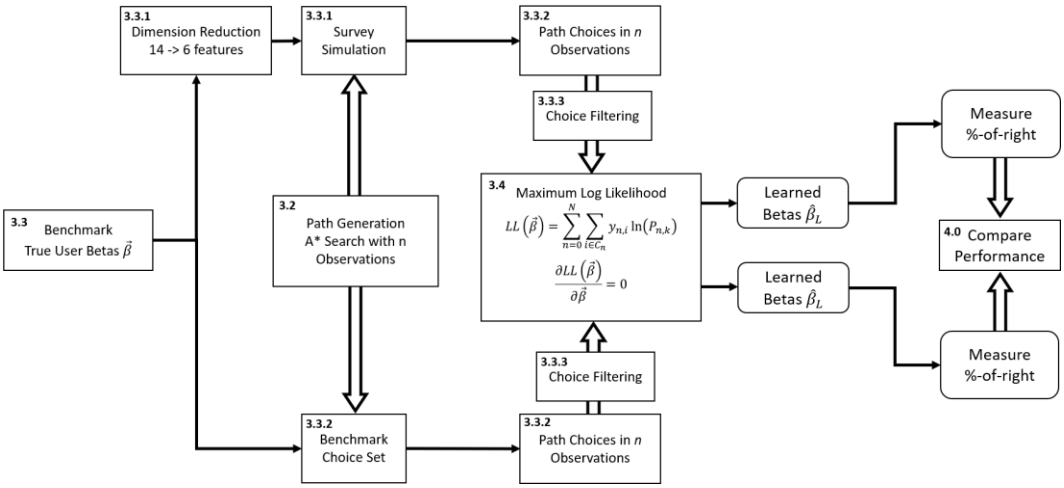

Figure 1: Methodology Overview

3.1 Discrete Choice Model

Discrete choice modelling describes a problem in which a decision maker must make a choice among a set of presented choice alternatives. In our setting the decision maker is a traveller who wants to find a route from a source to a target location. The choice set, from which the traveller can choose from, consists of different path alternatives from the same source-target-location pair. In general, discrete choice models are based on the assumption of utility maximization, which means that the decision maker selects the choice that maximizes his utility. A model derived under this assumption belongs to the category of random utility models. Note that the explanations for the discrete choice model in this subchapter are referenced from Kenneth E. Train [2].

3.1.1 Random Utility Model

A random utility model in which a decision maker can choose among J alternatives in observation n can be represented with the following formula:

$$U_{nj} = V_{nj} + \varepsilon_{nj} \tag{1}$$

As the decision maker wants to maximize their utility, they will only select an alternative k over alternative j, if and only if $U_{nk} > U_{nj}\ \forall k \neq j$. U_{nj} represents the true utility of the decision maker which is only known to themself. The researcher can only observe a part of the features, the explanatory variables $x_{nj}\ \forall j$, and a part of the features of the decision maker β and combine them into a function $V_{nj}(x_{nj}, \beta)$. This function describes the observable effect on the utility function and is also called representative utility. The unobservable part of the true utility function is represented by the random error term ε_{nj}. With this model a choice probability P_{nk} that the decision maker selects alternative k in observation n can be derived as:

$$\begin{aligned} P_{nk} &= \text{Prob}\big(\varepsilon_{nj} - \varepsilon_{nk} < V_{nk} - V_{nj} \forall j \neq k\big) \\ &= \int_{\varepsilon} I\big(\varepsilon_{nj} - \varepsilon_{nk} < V_{nk} - V_{nj} \forall j \neq k\big) f(\varepsilon_n) d\varepsilon_n \end{aligned} \tag{2}$$

Different choice models are derived based on different assumptions about the probability density function of the random error term $f(\varepsilon_n)$. In the next subsection we will explain how the MNL model is derived.

3.1.2 Multinomial Logit

The choice probability for the MNL model is derived by specifying that each random error term ε_n is independently, identically distributed extreme value. This distribution is known as Gumbel and type I extreme value and is assumed to have a variance of $\pi^2/6$. The probability density (3) and cumulative distribution (4) function of this distribution are given as:

$$f\big(\varepsilon_{nj}\big) = e^{-\varepsilon_{nj}} e^{-e^{-\varepsilon_{nj}}} \tag{3}$$

$$F\big(\varepsilon_{nj}\big) = e^{-e^{-\varepsilon_{nj}}}. \tag{4}$$

Based on these functions the choice probability of the MNL model can then be derived as:

$$P_{nk} = \frac{e^{V_{nk}}}{\sum_j e^{V_{nj}}} \tag{5}$$

in which V_{nk} is the representative utility of the chosen alternative k.

The independence specification of the random error terms is seen as the biggest drawback of MNL. In our problem setting, it implies that the error terms are uncorrelated over path alternatives for the same user. The MNL can be seen as an ideal specification of the representative utility in which the error terms are essentially white noise. Despite this critical specification, MNL is the most used discrete choice model due to its convenient close form formulation of the choice probability [2] and is also the preferred model in this paper.

3.1.3 Representative Utility Function

The representative utility function in this paper is specified as linear in parameters β. These parameters reflect the user preference for each explanatory variable x_{nj}. For observation n and path alternative j representative utility is denoted as:

$$V_{nj} = \beta_{num} * num_{nj} + \sum_{m \in M} [\beta_{co2} * CO2_{njm} + \beta_{dm} * d_{njm} + \beta_{tm} * t_{njm} + \beta_{cm} * c_{njm}] \quad (6)$$

It consists of the 5 below listed features:

Table 2: Explanatory Variables

num	Number of transfers between modalities
$co2$	CO2 emission
d_m	Discomfort on modality m
t_m	Travel time on modality m
c_m	Monetary cost on mode m

Since the features discomfort, travel time and monetary costs are differentiated per modality, the total number of explanatory variables, and therefore β parameters are 14. If $\vec{\beta}$ and the explanatory variables $\vec{x}$ are written in a vectorized form, the shorthand notation for the representative utility is:

$$V_{nj} = \vec{\beta}\vec{x}_{nj} \quad (7)$$

The following paragraph shortly explains how we determine the values of the explanatory variables. The number of transfers increases by 1 each time the mode of transportation is changed on the path e.g., if the traveler switches from bike to car. The number of transfers is calculated dynamically since it cannot be calculated prior to running the A* algorithm. The value of the number of transfers depends on which

modality has been picked at the previous node and which modality is picked for the current edge.

The calculation for CO2 emission for public transport and car are calculated based on average fuel consumption per kilometer that is referenced from the website De Lijn [11]. For public transport, the fuel consumption is calculated by averaging the fuel consumption of bus, tram, U-Bahn, and S-Bahn since they are not differentiated in the network. On the other hand, the CO2 emission for biking and walking are considered by understanding that physical exhaustion leads to an increased CO2 emission by the body. We assumed that walking is half as exhausting as biking. The CO2 emission from biking is taken from the website dasfahrradblog [12].

Table 3: CO2 emission (CO2_EM) values in g/km by modality

Public	Bus: 75.0	39.125
	Tram: 23.0	
	U-Bahn: 30.5	
	S-Bahn: 28.0	
Car		127.0
Bike		21.0
Walk		10.5

The formula for the CO2 calculation is:

$$co2 = length * CO2\_EM \tag{8}$$

The travel time feature represents an idealistic travel time. It does not take traffic for cars and bikes, and for public transport waiting time into consideration. The values are directly taken from the edge network.

The calculation of the discomfort feature assumes that the discomfort of traveling with a certain modality increases in the order: Car, public, bike, walk. The selected values here are purely theoretical and other possible values could be used.

Table 4: Discomfort factor (DF) by modality

Public	1.2
Car	1.0
Bike	2.0
Walk	3.0

The formula for the discomfort calculation is:

$$d = length * DF \tag{9}$$

The monetary cost is made up of 2 parts. The first part represents the opportunity cost of missing earning money due to traveling and is identical for all modalities. The opportunity cost (OC) is set as the minimum wage (9,50 €/hour) in Germany. The second part is differentiated between modalities. For bike and car sharing the prices are taken from pricing models of common car and bike sharing platforms [10][13]. For walking no additional costs are added. Finally, for public transport we made a simplification by taking the cost of a day ticket as reference and convert it to cost per second.

Table 5: Price factor (PF) in €/s by modality

Public	0.0001
Car	0.002
Bike	0.0013
Walk	0

Finally, the price calculation is

$$c = travel\ time * (OC + PF) \tag{10}$$

3.2 Path Generation

Similar to the work reported by Nuzzolo and Comi [1], path generation in our model relies on a random representative utility cost function to find unique source/target (S/T) path alternatives. These alternatives may be found by various shortest path algorithms such as Dijkstra as demonstrated by Modesti and Sciomachen [4], which work well on smaller networks but become too slow in large graphs such as in our case. To improve speed, we transform Dijkstra into an A* search in which branches are pruned via a heuristic function instead of endlessly exploring the graph. No matter what path search method is used for solving (S/T) pairs, the key property in discrete choice context is the random utility aspect. By building a dataset of highly random path alternatives, we improve the chances that some path alternatives will appeal strongly to a user's preference. Since a user's preference is unknown, having a dataset of diverse mutually exclusive alternatives improves the likelihood that the model can estimate a user's β-parameters to a high degree. This idea is explained further in section 3.4.

3.2.1 Random Beta Generation

Recall that in (7) the $\vec{\beta}$ is dot product multiplied with explanatory variables $\vec{x}$ to determine path utility. During path generation, in a particular observation *n*, the (S/T) pair is fixed for all J alternatives, this limits the scope of possible $\vec{x}$ that are inherent to the graph networks i.e., $\vec{x}$ is not fully random. Therefore, this leaves only $\vec{\beta}$ to be modified per alternative. Thus, to ensure path diversity for each (S/T) pair, random beta values with high variability are generated.

The formula used for random $\vec{\beta}$ is as follows:

$$K = \{0,1,2,\ldots,13\} \tag{11}$$

$$B = \{x_i \,|x_i \,\epsilon\, U(1,10e3)\};\ \ \forall i \epsilon K \tag{12}$$

In which B is a set of 14 random uniformly distributed numbers normalized from 0 to 1:

$$\vec{\beta} = \frac{B_i}{\sum_{k\epsilon K} B_k};\ \forall i \epsilon K \tag{13}$$

Although $\vec{\beta}$ is reasonable to use in this current state, path diversity can be improved by amplifying differences in β ratios via:

$$B = \{\beta_i^p \,|p \,\epsilon\, U(1,8)\};\ \ \forall i \epsilon K \tag{14}$$

Figure 2 shows an example of four $\vec{\beta}$ after normalizing B with (13).

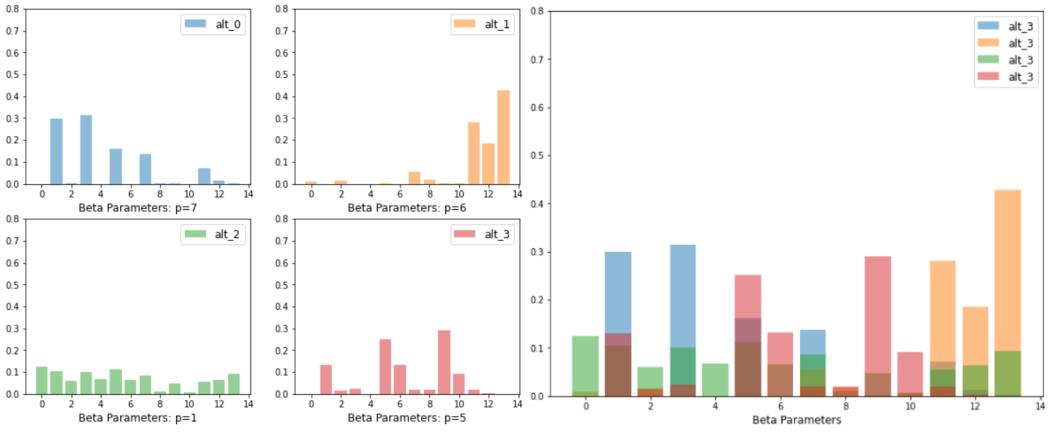

Figure 2: Random Beta Generation

3.2.2 Transportation Network

In order to create a multimodal transportation network, multiple graphs need to be merged together. In this paper the focus is on the city of Munich with four main graph networks: walk, bike, drive, and public. Note that we only consider car and bike sharing as available modalities and not private cars and bikes. The first 3 networks were gathered easy enough, utilizing the graph_from_place() function built into osmnx's library. The remaining public network though was not readily available from online sources in multidigraph format and required generation routines described in the next paragraph. Figure 3 captures a plot of each graph.

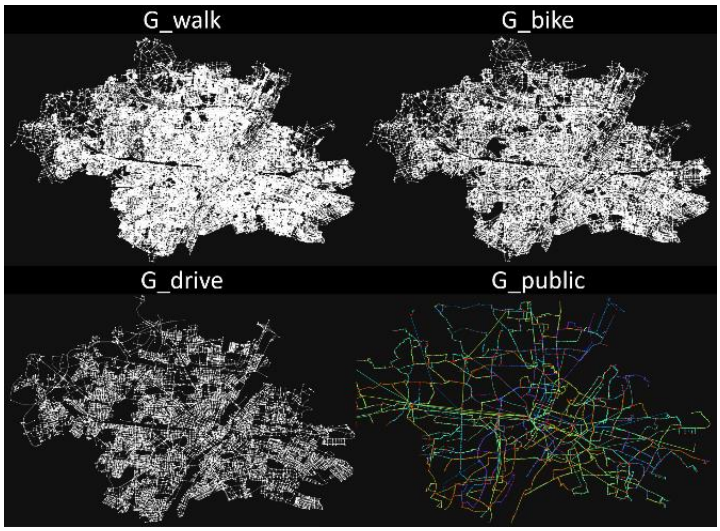

Figure 3: Graph Networks

The public transportation data was sourced from wiki.tum.de [14] in csv format. The files contain info such as stop-id (with lat and lon coordinates), line reference, sequence order, etc.

To guarantee that the public network connects to the other modalities, we iterate through each stop-id and find the nearest walk node from G_walk using get_nearest_node(). Once a node is found and less than 200m from the original stop-id coordinate, we assume this walk node becomes the stop-id[1] else, the stop-id is considered outside the city boundaries and ignored. From this point routes for Tram, U-Bahn, S-Bahn are created by simply straight-line connecting stop-id's in sequence order for each line reference-id. These modality routes are added to G_public. For creating bus routes, implementation takes a more realistic approach. Instead of straight-line connecting two stop-ids together, shortest_path() is used on the G_drive graph to find the likely route a bus would take. This process only works if first the stop-ids are properly connected to G_drive. The routine that establishes the

[1] Note: the majority of stop-ids are less than 50m from a walk node.

connections (stop-ids to G_drive) is illustrated in figure 4 and explained further in the next paragraph.

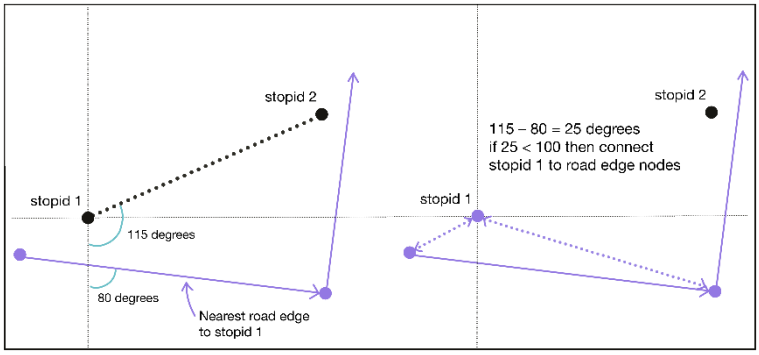

Figure 4: Orientation-Check Routine

Before connecting a stop-id to the nearest road edge, an orientation check is performed to determine if the road edge is headed in the general direction towards the next stop id in the bus route sequence. If the orientation difference is < 100 degrees then the connections are made, if not, the road edge is temporarily deleted from G_drive and the next nearest edge is found. This method is continued for each bus route until all route stop-ids are connected to G_drive. Finally, the full bus routes can be generated via shortest_path() running in stop-id sequence order and are added to G_public.

3.2.2.1 Pre-processing

In order to generate a combined node and edge network from the 4 modality graph networks (G_public, G_drive, G_walk and G_bike), some pre-processing for the edge networks has to be done. An edge with the same source and target node can be contained in all 4 modality graph networks. But since the value for certain features like travel time is different for each modality, these values are stored in a new edge feature in form of a dictionary differentiated by modality. For example, an edge with the same source and target node contained in all graph networks has different values for each feature. In the combined edge network these values are stored in a dictionary as a new feature like illustrated in the following table:

Table 6: Example edge feature values for different graph networks

	Co2	Travel time	Discomfort	Cost
Public	4.3	2	116	0.018
Car	12.3	7	97	0.03
Bike	2.0	7	193	0.002
Walk	0.5	35	290	0.08

Using this methodology, the different modality edge networks are combined into one edge network containing all modalities. The combined node network is simply created by combining all nodes with a unique id from the different modality node networks.

Finally, after the combined edge network is obtained, the values of the featuresare normalized using a minimum maximum scaler

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{15}$$

before they are used in the path cost calculation since the scale of the features are all different.

3.2.3 A* Search

Unlike Dijkstra the A* algorithm is an informed search technique and can be much more cost-efficient regarding time and space complexity. The Dijkstra algorithm only looks at the current path cost to node $g(n)$ for the next node selection without any additional information. In contrast the A* algorithm uses an evaluation function $f(n)$. It also contains the current path cost $g(n)$ but additionally includes a heuristic function $h(n)$.

$$f(n) = g(n) + h(n) \tag{16}$$

The heuristic function estimates the cost to reach the target node. For A* search to be optimal the heuristic function has to be admissible [6]. Admissible means that the heuristic function always under-approximates the actual cost:

$$h(n) \leq g(n, target) \tag{17}$$

In this equation $g(n, target)$ represents the actual cost to the target node. From another perspective it means, that $f(n)$ never overestimates the cost to the target. Thus, the algorithm only searches for paths that has a lower cost to the goal than those previously found. Essentially not promising nodes are pruned, since only nodes are expanded for which the cost is lower than the cost of the current optimal path. With a well-defined heuristic function, pruning increases the time and space efficiency by a lot. The definition of the heuristic function is problem specific. Therefore, we need to find a heuristic function that fits our problem.

The path cost $g(n)$ in our problem setting is the representative utility function $V(n)$ (6). All features except number of transfers in $V(n)$ are dependent on the length of the path. Thus, we decided to use the great circle distance to the target node to

calculate the heuristic function. The great circle distance is the shortest distance between two points on a sphere. In the implementation the great circle distance is precalculated for every node to the selected target nodes. Then during the run of the A* algorithm the heuristic function is calculated by using those precalculated length len_{heu} values.

$$h(n) = \beta_{co2} * len_{heu} * CO2\_EM_m + \beta_{dm} * len_{heu} * DF + \beta_{tm} * \frac{len_{heu}}{speed_m} + \beta_{cm} * \frac{len_{heu}}{speed_m} * (OC + PF) \quad (18)$$

The formula for $h(n)$ is the same as $V(n)$, with omission of the number of transfers as the only difference. Number of transfers is not included for simplicity reason, since it is difficult to find an admissible approximation for it. By using the great circle distance to calculate each feature for the heuristic function, its value is guaranteed to be lower than the actual cost to the target node. After each feature is calculated the values are normalized by applying the same scaler used in the pre-processing. Finally, the evaluation function in our A* algorithm is defined as:

$$f(n) = V(n) + h(n) \quad (19)$$

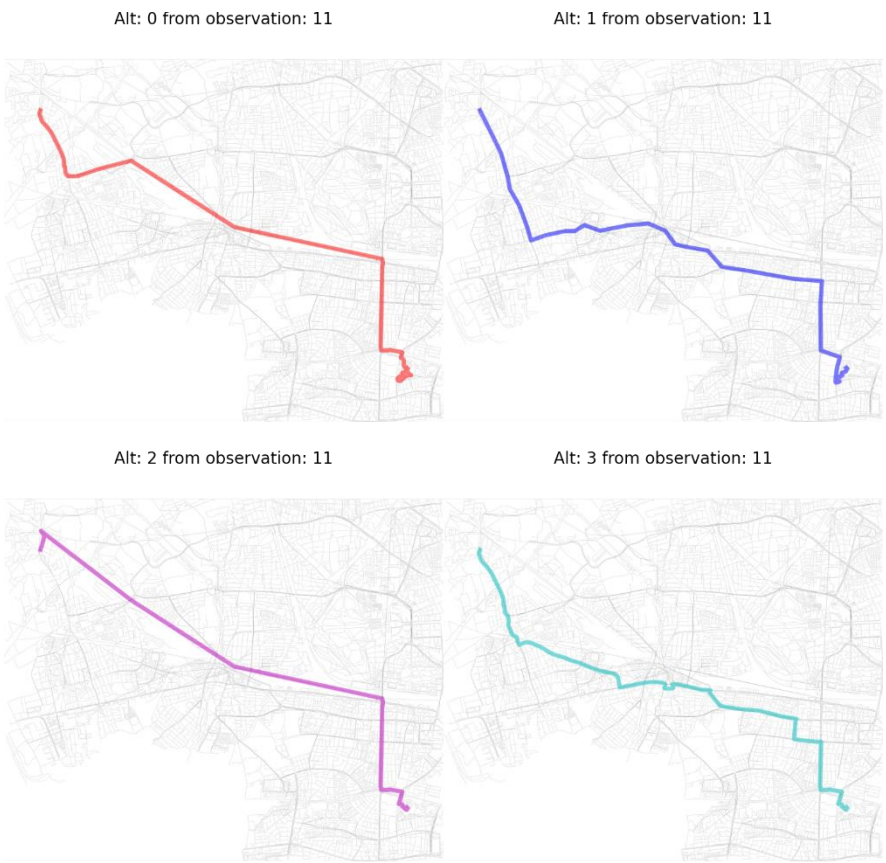

Figure 5: Example of 4 alternatives from A* Search

3.3 Benchmarking & Simulated Survey

Because the results of our model are not being compared to previous discrete choice research in this papers domain, benchmarking the model requires an internal assumption that we know a user's true β-parameters. Through this assumption we devise a method to create and score the performance of a SS, based on a reduced set of preference information. How the SS is created is discussed in the next paragraph while the performance metrics are reviewed later in the results section. Also, in this section we develop choice filters to compare performance of β-parameter estimation and lastly, we briefly explain the scale parameter used in this model.

3.3.1 Simulated Survey

During this project's research many ideas were explored in creation of a survey simulation. Ideas relating to finding relative percentile differences in the path explanatory variables and calculating the accuracy of sum product outcomes of these differences to a user's reduced set of preference sliders. The sliders represented averseness towards car, public, bike, walk, number of transfers, and sustainability. Ultimately though a simpler approach was chosen due to better performing results and easier comprehension. Therefore, these percentile difference methods will not be discussed further in this paper due to their complexity and worse performance.

To obtain a choice set from our simplified SS, recall the RU function from (6). This utility contains 14 features and 14 β-parameters. In the same way random β's were created for path generation in section 3.2.1, likewise, the same equations are used for generating a true user β set: $\vec{\beta}_{user}$. Taking the $\vec{\beta}_{user}$ benchmark, we now dimensionally reduce from 14 to 6 preferences in which these 6 preferences represent the averseness towards a particular modality, number of transfers, and sustainability. The dimensional reduction is done by taking a 3-way average of each modalities β-parameters $\beta_{dm}, \beta_{tm}, \beta_{cm}$ and then simply copying β_{num} and β_{co2} from $\vec{\beta}_{user}$. Now, instead of using these 6 parameters outright to generate a choice set, for convenience we recreate a set of 14 β-parameters by duplicating the 3-way averages 3 times for each modality. Shown formally below and visually in figure 6, $\vec{\beta}_{survey}$ can be compared to $\vec{\beta}_{user}$.

$$\vec{\beta}_{survey} = \{ \tfrac{1}{3}\textstyle\sum_{b=0}^{3}\beta_b, \cdots, \ \cdots, \quad \text{(Car)} \tag{20}$$

$\frac{1}{3}\sum_{b=3}^{6}\beta_b, \cdots, \ \cdots,$ (Public)

$\frac{1}{3}\sum_{b=6}^{9}\beta_b, \cdots, \ \cdots,$ (Bike)

$\frac{1}{3}\sum_{b=9}^{12}\beta_b, \cdots, \ \cdots,$ (Walk)

$\beta_{12}, \ \beta_{13}\}$ (# Transfers, Sustainability)

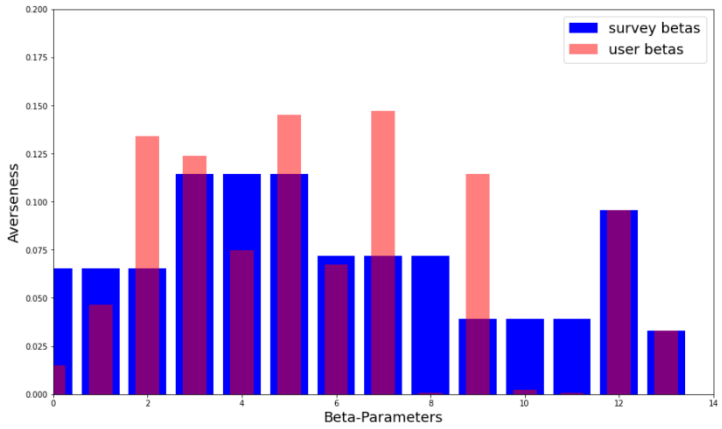

Figure 6: $\vec{\boldsymbol{\beta}}_{\boldsymbol{survey}}$ compared to $\vec{\boldsymbol{\beta}}_{\boldsymbol{user}}$.

3.3.2 Choice Set

With $\vec{\beta}_{user}$ chosen, the benchmark or SP choice set can be calculated using the following transformation:

$$N = \{0,1,2, \dots, \dots, \dots, 199\} \tag{21}$$

$$C_n = X_n \cdot \vec{\beta}^{\mathsf{T}}{}_{user} \quad \forall n \epsilon N \tag{22}$$

Correspondingly to how the benchmark choice set is calculated in (22), the SS betas $\vec{\beta}_{survey}$ use the same formula to generate a choice set. Furthermore, the choice set binary variables for both $\vec{\beta}_{user}$ and $\vec{\beta}_{survey}$ defined as $y_{n,i}$ can now be obtained by storing which alternative i has the max utility.

$$y_{n,i} = \{x_i |\, x_i \epsilon \{0,1\}\}; \quad x_i = \begin{cases} 1 \; if \; C_{ni} = max\{C_n\} \\ 0 \; if \; C_{ni} \neq max\{C_n\} \end{cases} \quad \forall i \epsilon C_n; \; \forall n \epsilon N \tag{23}$$

3.3.3 Choice Set Filtering

Through experimentation it was found that it is not always ideal to estimate β's using the entire choice set. Therefore, before β estimation is run, a choice filtering method is used to eliminate observations/surveys with low path diversity as this seems to improve β estimation. Low path diversity reveals itself as a low probability of choosing one alternative over another referring to equation (5). In our

modelling, two choice filters have been developed. The first choice filter works by specifying the number of observations to keep (the observation target). Simply put, the filter first calculates the choice probabilities for each observation and then removes the lowest probabilities until the observation target number is reached. To check that our assumptions hold for filter 1, we implement a second filter in which 100 observations are randomly chosen and not based on a probability threshold. The outcomes of the two filter methods are presented later in the results section.

3.3.4 Scale Factor

In MNL models, the scale factor can be a constant that is used to normalize the error variance to some number, which in turn normalizes the scale of utility [2].

$$U_{nj} = \frac{\vec{x}_n \cdot \beta}{\sigma} + \frac{\varepsilon_{nj}}{\sigma} \tag{24}$$

Normalization is usually required when externally benchmarking models or datasets from another research [9]. In this paper, scaling utility is not normally required since the results are not being compared to other research, but through experimentation we observed inconsistent performance when scaling was not considered. Therefore, we include a scale factor in our results.

Under iid assumption the scale parameter is typically found by setting the unobservable error terms variance $Var(\varepsilon_{nj})$ to $\pi^2/6$. In our model, since we do not know the exact variance of ε_{nj}, and because the scale of utility is irrelevant, Train [2] states that it is left up to the researcher to arbitrarily choose a scale parameter for convenience. By experimenting with our dataset and reviewing the performance of β estimation convergence, we developed a straightforward method to estimate $Var(\varepsilon_{nj})$ and thereby determine a scale parameter.

Since we know that in MNL the choice probability function (5) is derived from the Gumbel cumulative density function (4), we can set the error term variance to the Gumbel variance (25)

$$\begin{aligned} Var(\varepsilon_{nj}) &= \sigma^2(\pi^2/6) \\ \sigma &= \sqrt{\frac{Var(\varepsilon_{nj})}{(\pi^2/6)}} \end{aligned} \tag{25}$$

This gives us a scale factor σ equation in which if we assume an estimate for $Var(\varepsilon_{nj})$ we can derive σ. Our $Var(\varepsilon_{nj})$ is estimated based on an average representative utility variance for a distribution of random users:

$$\begin{gathered} \mathrm{D} = \{0, 1, 2, \dots, \dots, \dots, 1000\} \\ B = \{x_i | x_i \, \epsilon \, equation\,(14)\}; \quad \forall \mathrm{i} \epsilon D \\ Var(\varepsilon_{nj}) \sim Avg\{Var\big(V_n(\vec{x}_n, \vec{\beta}\big)\}; \;\; \forall n \epsilon N; \; \forall \vec{\beta} \epsilon B \end{gathered} \tag{26}$$

In the above equation a new $\vec{\beta}_{randomUser}$ is generated after N observations for a total of 1000 runs to obtain a distribution of variance terms. Taking the mean of the variance distribution we use this number as our estimated $Var(\varepsilon_{nj})$ and calculate a scale factor (25) for the model (24).

3.4 Maximum Log Likelihood

In the literature reviewed for this project, it was discovered that MNL models are typically estimated through the use of maximum likelihood estimators (MLE). The general form is shown here:

$$L(\beta) = \prod_{n=0}^{N} \prod_{i} (P_{ni})^{y_{ni}} \tag{27}$$

For mathematical convenience, taking the log of (27) is re-expressed as the log likelihood function, in which the choice probability function P_{ni} is from (5):

$$LL(\beta) = \sum_{n=0}^{N} \sum_{i \in C_n} y_{n,i} \, ln(P_{ni}) \tag{28}$$

Both textbooks from Ben-Akiva, Lerman [3] and Train [2] point to maximum log likelihood (MLL) as an effective means for estimating β-parameters. In Trains 2003 text, he cites Mcfadden's [8] derivation that LL(β) is globally concave for linear-in parameters utility. Furthermore, Train explains that global concavity can be guaranteed if the second partial derivative or hessian matrix of MLL is negative definite for all β-parameters.

$$\nabla^2 \mathcal{L}(\beta_k) = \frac{\partial^2 LL(\beta')}{\partial \beta_k \partial \beta_k} \; < \; 0 \quad \forall k \epsilon K \tag{29}$$

This concavity check (21) is important because in MLL the goal is to find what values of β set the first partial derivatives equal to zero. In other words, maximizing the log likelihood function. If the likelihood function is globally concave, then there

can only be one maximum. Maximizing LL(β) is illustrated in figure 7 and defined as follows:

$$\nabla \mathcal{L}(\beta_k) = \frac{\partial LL(\beta')}{\partial \beta_k} = 0 \quad \forall k \epsilon K \tag{30}$$

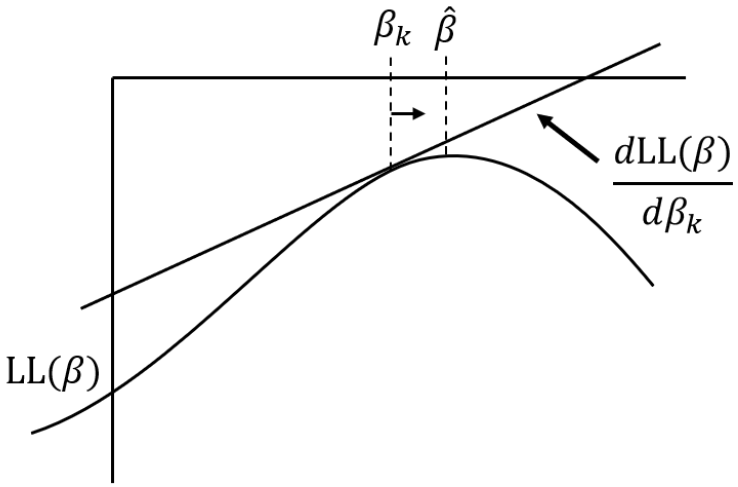

Figure 7: Maximizing Log Likelihood

Referring back to (28), since the β's are unknown and need to be estimated, the only inputs to solve MLL are the explanatory variables $\vec{x}$ from (7) and the binary choice variable $y_{n,i}$. This choice variable $y_{n,i}$ is also known as the dummy variable and as discussed previously in section 3.3.2 (23), contains the index of the chosen alternative by the user in the n^{th} observation. 1 for chosen alternative and 0 if not chosen. With these inputs we seek to estimate a total of 14 betas, therefore 14 equations to solve jointly. Discussed in the next section, a numerical estimation procedure is used to solve these partial derivative equations as they are typically too complex to solve algebraically.

3.4.1 Newton-Raphson

Many statistical packages exist for solving MLL but in this project we decided to build the solver from scratch for two main reasons. First, having access to the internals of the derivative calculations we can easily debug and adjust parameters. Second, working though the math develops an intuition about the working principals of MLL and builds a good foundation for future discrete choice projects. In the following paragraphs we summarize the main points of Newton-Raphson's (NR) method starting with the iteration routine and ending with convergence criteria. The reader is encouraged to explore chapter 8 of Train (2003) for further details about this numerical estimation method.

The core of Newton-Raphson is the iteration formula which specifies the new estimates for $\hat{\beta}$ based on the current value subtracted by the ratio of the first and second partial derivatives of LL($\hat{\beta}$). The λ is the step size and is discussed in later paragraphs.

$$\beta_{k+1} = \beta_k - \left(\lambda \cdot \frac{\nabla \mathcal{L}(\beta_k)}{\nabla^2 \mathcal{L}(\beta_k)}\right) \quad \forall k \epsilon K \tag{31}$$

At the start of the routine all the β_k's are initialized to zero as the first 'guess' and from this the partial derivatives are calculated. The first partial derivative is relatively straight forward. Shown below the equation is expanded out for a single β_k.

$$LL(\hat{\beta}) = \sum_{n=0}^{N} \sum_{i \in C_n} y_{ni} \left(\hat{\beta} \cdot x_{nj} - ln\left(a_n(\hat{\beta})\right)\right) \; : \; a_n(\hat{\beta}) = \sum_{j \in C_n} e^{\hat{\beta} \cdot x_{nj}}$$

$$\frac{LL(\hat{\beta})}{\partial \beta_k} \rightarrow \frac{\partial(\hat{\beta} \cdot x_{nj})}{\partial \beta_k} = x_{nik}$$

$$\frac{\partial LL(\hat{\beta})}{\partial \beta_k} \rightarrow \frac{\partial ln\left(a_n(\hat{\beta})\right)}{\partial \beta_k} = \frac{1}{a_n(\hat{\beta})} \cdot \frac{\partial a_n(\hat{\beta})}{\partial \beta_k} \tag{32}$$

$$let \;\; a_n(\hat{\beta}) = g_n \;\; and \;\; \frac{\partial a_n(\hat{\beta})}{\partial \beta_k} = f_n$$

$$\nabla \mathcal{L}(\beta_k) = \sum_{n=0}^{N} \sum_{i \in C_n} y_{ni} \left(x_{nik} - \frac{f_n}{g_n}\right)$$

Next, the second partial derivatives are calculated using the quotient rule. Following from the end of (32) and again expanded out for a single β_k.

$$\frac{\partial^2 LL(\hat{\beta})}{\partial \beta_k \partial \beta_k} = \sum_{n=0}^{N} \sum_{i \in C_n} y_{ni} \left(\frac{\partial x_{nik}}{\partial \beta_k} - \frac{\frac{\partial f_n(\hat{\beta})}{\partial \beta_k} \cdot g_n \; - \; f_n \cdot \frac{\partial g_n(\hat{\beta})}{\partial \beta_k}}{g_n^2} \right) \tag{33}$$

$$\left(\frac{\partial f_n(\hat{\beta})}{\partial \beta_k} \cdot g_n\right) = \sum_{j \in C_n} \sum_{jj \in C_n} \sum_{kk \in \mathrm{K}} x_{n,j,k}^2 \cdot e^{x_{n,j,kk} + x_{n,jj,kk}}$$

$$\left(f_n \cdot \frac{\partial g_n(\hat{\beta})}{\partial \beta_k}\right) = \sum_{j \in C_n} \sum_{jj \in C_n} \sum_{\mathrm{kk} \in \hat{\beta}} \left(x_{n,j,k} \cdot x_{n,jj,k}\right) \cdot e^{x_{n,j,kk} + x_{n,jj,kk}}$$

$$\nabla^2 \mathcal{L}(\beta_k) = \sum_{n=0}^{N} \sum_{i \in C_n} y_{ni} \left(0 - \frac{f_n' \cdot g_n - f_n \cdot g_n'}{g_n^2}\right)$$

Concluding the derivative procedures, the next β_{k+1} is calculated using (31). These new $\hat{\beta}$ estimates need to be checked against all convergence criteria, but before doing so we first must confirm that the new estimates have not overshot the maximum of the log likelihood function. This is done by solving (28) and checking that LL($\hat{\beta}_{+1}$) is greater than the previous LL($\hat{\beta}$). If for example the maximum is overshot, then instead of recalculating the partial derivatives, we can simply reduce the step size λ and recalculate $\beta_{k+1}$from (31). This sub procedure is run until LL($\hat{\beta}_{+1}$) is greater than the previous. Figure 8 provides more insight into this routine. Furthermore, the step size provides some intuition into how $\nabla \mathcal{L}(\hat{\beta})$ and $\nabla^2 \mathcal{L}(\hat{\beta})$ contribute to the beta estimations. One could say $\nabla \mathcal{L}(\beta_k)$ tells which direction and $\nabla^2 \mathcal{L}(\beta_k)$ by how much to adjust each β_k.

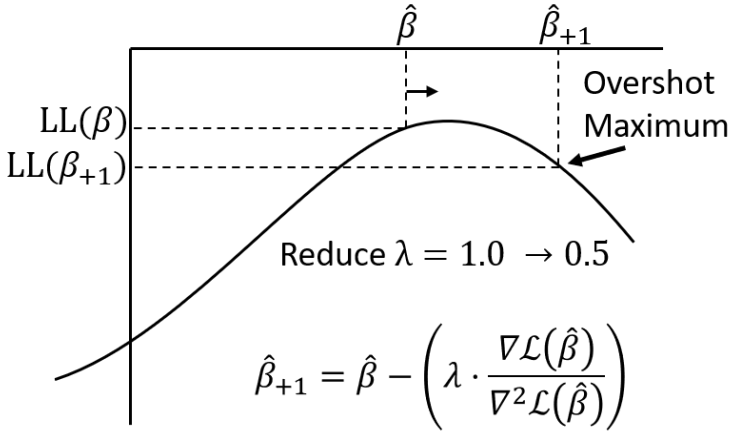

Figure 8: Step Size-Overshot Routine

Lastly, after the overshot sub-routine, the convergence criteria are checked. Although not absolutely necessary, our variant of NR implements four checks listed below:

$$\sqrt{\frac{1}{K}\sum_{k=0}^{K}\left(\beta_{k_{new}} - \beta_{k_{prev}}\right)^2} \; < \; 10e-4 \tag{34}$$

$$\left|\frac{\beta_{k_{new}} - \beta_{k_{prev}}}{\beta_{k_{prev}}}\right| \; < \; 10e-2 \quad \forall k \epsilon K \tag{35}$$

$$\left|\nabla\mathcal{L}(\beta_k) \cdot \sum_{kk=0}^{K} \frac{-\nabla\mathcal{L}(\beta_{kk})}{\nabla^2\mathcal{L}(\beta_k)}\right| \; < \; 10e-5 \quad \forall k \epsilon K \tag{36}$$

$$|\nabla\mathcal{L}(\beta_k)| \; < \; 10e-5 \quad \forall k \epsilon K \tag{37}$$

Typically, criteria (35) ends NR but this can easily be adjusted via the listed thresholds. For further reading on these convergence criteria see page 82 of Ben-Akiva, Lerman [3] for (34), (35) and page 202 of Train [2] for (36).

3.5 Limitations

The following list contains general simplifications that the model of this paper underlies:

- Public transport is not distinguished between bus, tram, train, and subway.
- Waiting time and interarrival time for public transport are not considered.
- Traffic and intersection stop times are ignored.
- Locations of bikes and cars for sharing are not considered, a user can access a bike/car if an edge is a bike/car road.

To make the approach introduced in this paper more realistic, the above listed simplifications should be worked on. Furthermore, regarding β-parameter estimation, the Newton-Raphson implementation is sensitive to $\vec{x}$ variable scale and can lead to erroneous results if the scales between features are unnormalized or too large. Further work could be done to detect and warn when $\vec{x}$ variable scale is incorrect as well as to benchmark this implementation against other already existing software packages.

4 Results

A number of experiments were run to observe how the performance of our simulated survey compares to assumed true user β's from a SP survey. For each experiment, a total of 200 observations were used with 4 path alternatives in each observation. In all results we compare performance against three choice filter configurations. The first and second being the probability threshold and random filters discussed in section 3.3.3 in which 100 observations are removed and the third being no filter i.e., all 200 observations used. In section 4.1 we showcase the %-right relative to the benchmark SP choice set and in 4.2 the NR convergence statistics are discussed. Lastly in 4.3, remaining insights are given with respect to overall model performance and potential future experiments.

4.1 Choice Filter Performance

Summarized in table 7, the %-of-right is calculated for each model and each filter variant. Performance of our SS method shows good results and nearly on par with the learned true user betas β's when comparing the %-of-right incl. first + second best. This performance leads us to believe that SS is taking advantage of inherent correlations between features since only 6/14 were used to derive the choice set.

Table 7: Performance of MNL models for different choice filters

Choice Filters	$\widehat{\boldsymbol{\beta}}_{user}$ (Learned)		$\widehat{\boldsymbol{\beta}}_{survey}$ (Learned)	
	%-of-right	%-of-right incl. first + second best	%-of-right	%-of-right incl. first + second best
Probability	87	96	83	95
Random	82	91	80	91
None	82.5	91.5	78	90.5

Further, the results from the random and none choice filters show reduced performance compared to the probability threshold filter. This outcome partially validates our hypothesis that removing observations with low path diversity improves $\boldsymbol{\beta}$-parameter estimation.

4.2 Newton-Raphson Convergence

For the convergence statistics in tables 8 & 9 we see similar trends to the %-of-right results, further increasing our confidence in our correlation and path diversity hypothesis. Looking at likelihood ratio index (LRI) which is similar to a goodness-of-fit measure, we see almost a 20% improvement when the probability threshold is

used compared to the other filters. We also observe the minimal differences in LL($\hat{\beta}$) values between the benchmark $\hat{\beta}$ estimates and SS $\hat{\beta}$ estimates, showcasing the good performance of our SS method relative to the benchmark.

Table 8: Convergence statistics for learned benchmark β's

Choice Filters	$\widehat{\boldsymbol{\beta}}_{user}$ (Learned)					
	# Iterations	MAE	LRI	SF	LL($\hat{\beta}$)	Avg $\nabla\mathcal{L}(\hat{\beta})$
Probability	91	0.0342	0.8068	12.1763	-26.7883	0.5961
Random	43	0.0330	0.6190	13.8947	-52.8141	1.9750
None	50	0.0269	0.6201	13.9504	-105.3404	2.2337

Table 9: Convergence statistics for learned survey β's

Choice Filters	$\widehat{\boldsymbol{\beta}}_{survey}$ (Learned)					
	# Iterations	MAE	LRI	SF	LL($\hat{\beta}$)	Avg $\nabla\mathcal{L}(\hat{\beta})$
Probability	74	0.0441	0.7805	12.2046	-30.4330	0.9322
Random	36	0.0336	0.5992	13.8947	-55.5568	2.4347
None	39	0.0312	0.5875	13.9504	-114.3813	4.2008

Also, to note, the scale factor (SF) varies slightly between filters due to changes in the explanatory matrix X (changing observations sets). These minor changes in SF do not affect the relative performance between models to a significant degree.

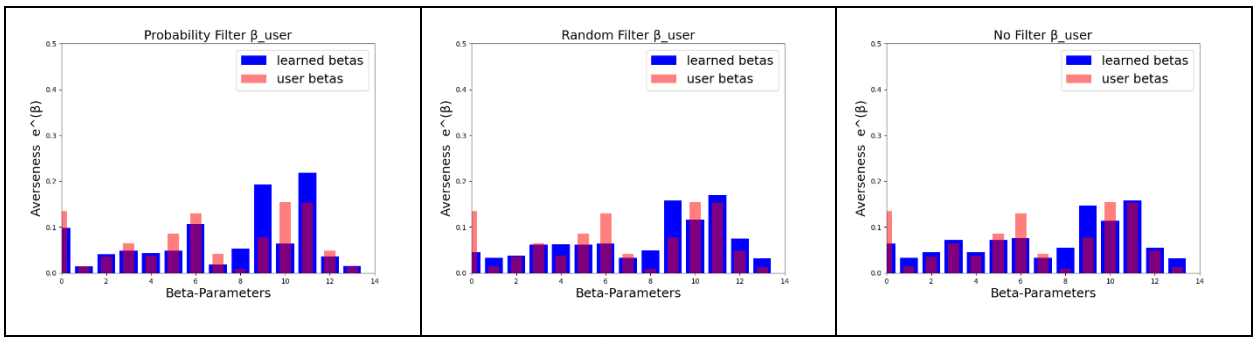

Figure 9: Learned $\hat{\beta}_{user}$ for each filter type

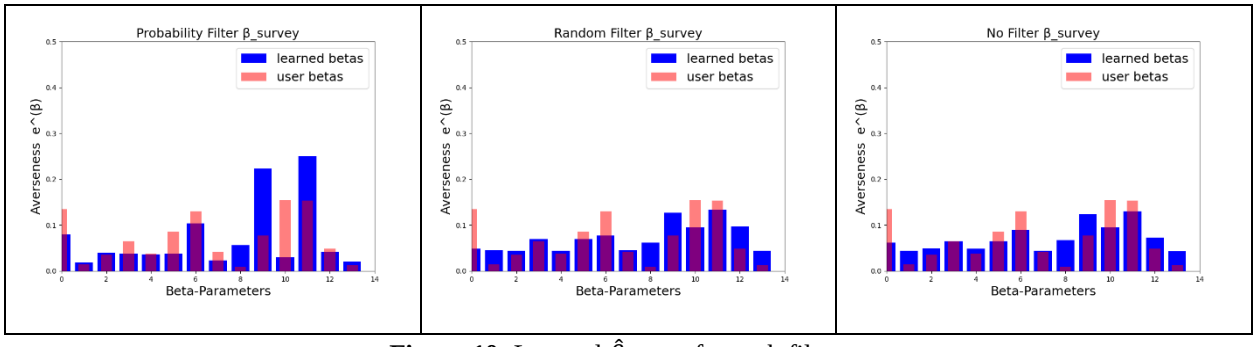

Figure 10: Learned $\hat{\beta}_{survey}$ for each filter type

In the above figures 9 & 10, the learned β's are plotted against the true user β's for visually comparing the effects of different choice filters.

4.3 Insights

Lastly, we compare path generation in figure 11 between $\hat{\beta}_{user}$ learned and $\hat{\beta}_{\text{survey}}$ learned and list the time travelled on each modality.

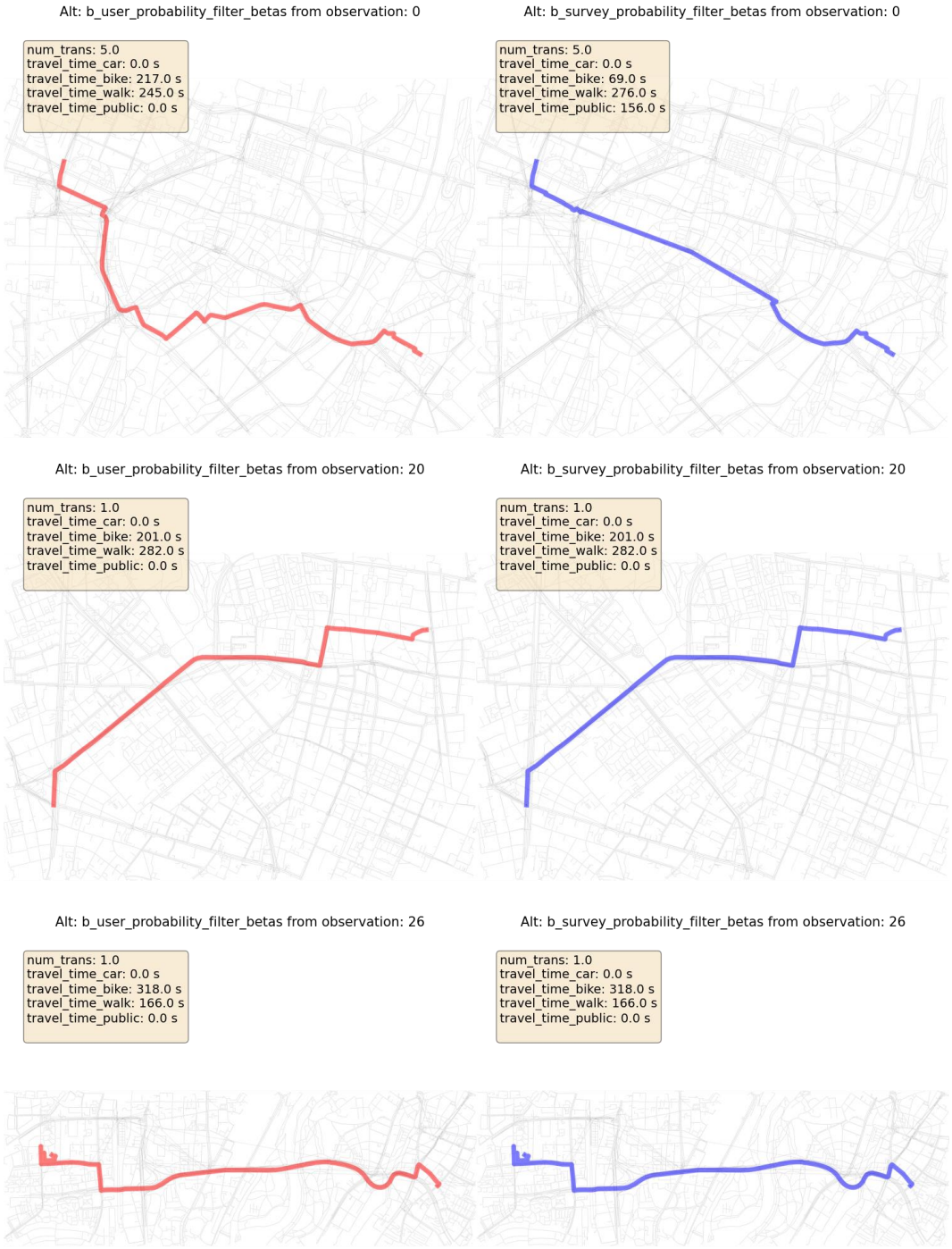

Figure 11: Path Generation using learned $\hat{\boldsymbol{\beta}}_{user}$ and $\hat{\boldsymbol{\beta}}_{\text{survey}}$

From these plots we can see that often the same routes are chosen between the two models.

Overall, the results of our modelling are promising for future work. With a simulated survey we capture the average preferences of a user and learn their approximate beta $\boldsymbol{\beta}$-parameters for a higher dimensional set of features. Seemingly, due to the natural correlations in some of the observable features, a user specifying average preference towards a particular modality captures a good portion of the information regarding that modalities time, cost, and discomfort. It would be interesting to see how our SS method performs once the simplifications listed in section 3.5 are resolved and if these correlations still remain or not.

5 Conclusion

In this paper we have introduced a model which uses direct user inputs on a dimensionally reduced set of parameters to initialize simulated survey. We have implemented a customized A* algorithm for which the path cost is based on a multi criteria weighted utility function to generate path alternatives as the choice sets. Furthermore, we utilize a self-implemented Newton-Raphson method for estimating the preference parameters. Our results have shown that our model performs reasonably well against the conventional method of initializing the preference parameters via SP survey. We believe that our proposal is promising, since the time required for the user to initialize the preference parameters can be reduced drastically. This could be an important step for users to be more willing to use a personalized routing tool since it lowers the entry barrier.

Going forward, it would be interesting to explore more sophisticated methods for the dimensionality reduction of preference parameters. Moreover, since our results are based on simulated choices, it would be beneficial to conduct an experiment with real users. Finally, for the model introduced in this paper to be applicable in the real-world, it is absolutely necessary to dismiss underlying simplifications such as overall accessibility of bikes and cars for sharing.

Reference List

[1] Agostino Nuzzolo, Antonio Comi. “Individual utility-based path suggestions in transit trip planners”. In: IET Intell. Transp. Syst., 2016, Vol. 10, Iss. 4, pp. 219–226 & The Institution of Engineering and Technology (2016)

[2] Kenneth E. Train. “Discrete Choice Methods with Simulation”. Textbook (2003)

[3] Moshe Ben-Akiva, Steven R. Lerman. “Discrete Choice Analysis”. The MIT Press Cambridge, Massachusetts London, England (1997)

[4] Paola Modesti, Anna Sciomachen. “A utility measure for finding multiobjective shortest paths in urban multimodal transportation networks”. In: European Journal of Operational Research 111 (1998).

[5] Paolo Campigotto et al. “Personalized and situation-aware multimodal route recommendations: the FAVOUR algorithm”. In: arXiv:1602.09076 [cs.AI] (2016).

[6] Stuart J. Russell, Peter Norvig. “Artificial Intelligence: A Modern Approach”. Textbook (2020).

[7] Theo A. Arentze. “Adaptive Personalized Travel Information Systems: A Bayesian Method to Learn Users’ Personal Preferences in Multimodal Transport Networks”. In: IEEE Transactions on Intelligent Transportation Systems, Vol. 14, No. 4, December (2013)

[8] Daniel Mcfadden. “The Measurement of Urban Travel Demand”. Journal of Public Economics 3 (1974) 303-328.

[9] Joffre Swait, Jordan Louviere. “The Role of the Scale Parameter in the Estimation and Comparison of Multinomial Logit Models”. Journal of Marketing Research Vol. 30, No. 3 (Aug. 1993)

[10] https://www.abendzeitung-muenchen.de/muenchen/bikesharing-muenchen-fuenf-anbieter-im-vergleich-art-468414#:~:text=Ein%20Tag%20kostet%2014%20Euro,oft%20f%C3%BCr%2012%20Stunden%20ausleihen.

[11] https://www.delijn.be/de/overdelijn/organisatie/zorgzaam-ondernemen/milieu/co2-uitstoot-voertuigen.html?vertaling=true

[12] https://dasfahrradblog.blogspot.com/2015/04/die-co2-bilanz-des-fahrrads.html#:~:text=Das%20Rad%20allein%20produziert%20demnach,22%20g%20CO2%20pro%20Kilometer.

[13] https://www.sixt.de/share/tarife/#/

[14] https://wiki.tum.de/display/msmmodels/Public+transportation+travel+times