Prospect Theory Multi-Agent Based Simulations for Non-Rational Route Choice Decision Making Modelling

Dissertation zur Erlangung des
naturwissenschaftlichen Doktorgrades
der Julius–Maximilians–Universität Würzburg

vorgelegt von

Gustavo Kuhn Andriotti

aus
Passo Fundo - RS - Brasilien

Würzburg, 2009

the corresponding counter was incremented.[21] This way the extra table can be used to construct the prospect. Then instead of using $\alpha_n r_n$, $\alpha_n pt(P_n)$[22] was used, where P_n corresponds to the prospect built based on the current reward. This is shown in eq. 5.13, where P_n is the prospect generate from an auxiliary "table" $\mathbf{C}$. The table $\mathbf{C}$ simply stores each received reward in pairs $\langle c, f\rangle$ where c is the reward value and f is the frequency with which the reward has appeared. This means that each time a reward is received the corresponding counter f is incremented or a new pair $\langle c, f\rangle$ is included with $f = 1$ – assuming that no other pair has a $c = r_n$, being r_n the reward received at step n.

$$\mathbf{C} = \{\mathbf{c} \in \mathbf{C} \mid \mathbf{c} = \langle c, f\rangle\} \tag{5.8}$$

$$\mathbf{C} = \{\forall \mathbf{c}, \mathbf{d} \in \mathbf{C} \mid c_{\mathbf{c}} \equiv c_{\mathbf{d}} \Rightarrow \mathbf{c} = \langle c_{\mathbf{c}}, f_{\mathbf{c}} + f_{\mathbf{d}}\rangle \wedge \mathbf{C} = \mathbf{C} - \{\mathbf{d}\}\} \tag{5.9}$$

$$\mathbf{C}_n = \mathbf{C}_{n-1} + \{\langle r_n, 1\rangle\} \tag{5.10}$$

$$P_n = \left\{\forall \langle c, f\rangle \in \mathbf{C} \exists \langle x, p\rangle \mid x = c \wedge p = \frac{f}{\sum_{\langle c,f\rangle \in \mathbf{C}_n} f}\right\} \tag{5.11}$$

$$Q_n(a) = (1 - \alpha_n) Q_{n-1}(a) + \alpha_n pt(P_n) \tag{5.12}$$

$$V_n \equiv \underset{a \in \mathcal{A}}{\operatorname{argmax}} [Q_n(a)] \tag{5.13}$$

It is possible to observe that the $Q(\bullet)$ function converges to the $pt(\bullet)$ values, depicted in Fig. 5.4a and 5.4b. The reason why the PT based $\mathcal{Q}$-Learning is faster (in converging) than the regular version is because of the score/reward table. This table gives more information than just an aggregation over the past rewards, which is the case of the value iterated strategy of the original $\mathcal{Q}$-Learning. This means that as soon as the table has a significant sample[23] over the reward function the $Q(\bullet)$ function returns a value close enough to the expected $pt(\bullet)$. The objective of the figures was not to compare the performance of both version but to show that both converge to the theoretical expected value. The formal definition of convergence is in Eq. 5.14 (for the EUT/normal version) and in Eq. 5.15 (for the PT version).

$$\lim_{n \to \infty} Q_n(a) = eut(r(a)) \tag{5.14}$$

$$\lim_{n \to \infty} Q_n(a) = pt(r(a)) \tag{5.15}$$

The proof of Eq. 5.15 is rather simple: $\lim_{n \to \infty} P_n = R(\bullet)$, i.e. the experienced the agent gets to the MDP (the higher n becomes) the more its P_n evolves closer to the PDF in $R(\bullet)$ and since this is the only prerequisite to return a correct $pt(\bullet)$ equivalent value then it converges to it in $n \to \infty$.[24]

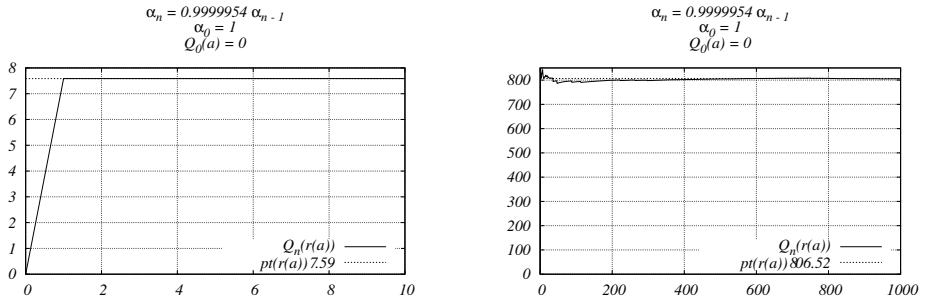
The information issue, i.e. the extra storage needed by the PT version is approached in the next section, where the issue not covered by the previous chapter is recapitulated: the editing phase.

[21] The method used should not be confused with the adopted for the agent $\mathcal{Q}$-Learning because it has several drawbacks, explained in the very next section. It is suitable, nevertheless, for this particular experiment. The reason for using a different algorithm is again to give a clear view of a single aspect (in this case the convergence of $Q(\bullet)$) instead of trying to explain every aspect together.

[22] The $pt(\bullet)$ function is the same as in Eq. 4.1.

[23] The frequencies stored in the extra table are close enough to the real PDF in $R(\bullet)$.

[24] The numerical and practical issues are ignored here, i.e. when n is actually ∞ all values in P_n are also numerically ∞ and therefore the computation of the probabilities is not possible. But as discussed before, $n = \infty$ is not an acceptable condition for any practical algorithm and therefore it can be said that when P_n reaches a close enough sample of $R(\bullet)$ its values can correctly construct $R(\bullet)$ and therefore the correct $pt(\bullet)$ value.



(a) Deterministic $r(a) = 10$ and $pt(r(a)) \sim 7.59$ (b) Stochastic $r(a)$ with $pt(r(a)) \sim 806.52$

Figure 5.4: Simple $\mathcal{Q}$-Learning $Q_n(a)$ evolution, using PT

5.5 Editing Phase

In Sec. 4.6.2 it is discussed why the standard clustering methods are not appropriate for coping with the editing phase. Therefore here a simple algorithm is proposed that combines part of the simplicity of the lossy counting and part of the strategy in [ALSS95]. This editing method is also inspired by [DS05], despite of the drawbacks discussed in Sec.4.6.

The algorithm is a stream clustering algorithm based on centroids (geometrical centres), i.e. the outcomes are seen as clusters and the derived prospect corresponds to the centroids of those clusters, having the sum of the outcomes' probabilities as the cluster probability and the mean value (of the outcomes) as the representative outcome. Because $R(\bullet)$ is unidimensional the clustering is made over a single dimension. This method also requires a separation threshold, called ϵ.

As a consequence of the clustering method, the reward function codomain is limited and partially known, because ϵ must be given. It means that the derived $\mathcal{Q}$-Learning algorithm needs to accumulate more knowledge about the world than the original version, because it must have an idea of the reward function codomain. This means that ϵ should not be calibrated but informed (by the modeller) to help the algorithm. This implies too, that the agent is not completely "blind" when looking at the reward function $R(\bullet)$ (it knows how to aggregate the rewards through ϵ).[25] Another consequence is that the $R(\bullet)$ function must map a closed interval in $\mathbb{R}$.[26]

This cluster structure is defined by Eq. 5.16, where $\mathbf{C}$ is the cluster structure, which is a set of pairs $\langle c, f\rangle$. These pairs represent the centroid value in c and the amount of points in that centroid in f. In this structure the centroids are unique, i.e. no two pairs share the same centroid value c.

$$\begin{aligned} \mathbf{C} &= \{\mathbf{c} \in \mathbf{C} \mid \mathbf{c} = \langle c, f\rangle \wedge c \in \mathbb{R} \wedge a \in \mathbb{N}^*\} \\ \mathbf{C} &= \{\forall\ \mathbf{c} \in \mathbf{C}, \nexists\ \mathbf{d} \in \mathbf{C} \mid c_{\mathbf{c}} = c_{\mathbf{d}}\} \end{aligned} \tag{5.16}$$

In Eq. 5.16 only the structure is formalised but not how to transform it into a prospect $\mathbf{x}$. This process is rather simple and shown in Eq. 5.17. There $\mathbf{x}_{\mathbf{C}}$ is the prospect generated from the cluster structure $\mathbf{C}$; $\langle c, f\rangle$ is a centroid/accumulator pair from $\mathbf{C}$; $\langle x, p\rangle$ is the resulting outcome/probability pair (for the corresponding $\langle c, f\rangle$); and x receives the c value and p receives the frequency f converted to the corresponding probability.

[25] The agent builds the prospects using the editing algorithms but the value ϵ is given by the modeller and not "discovered" by the agent.

[26] If the interval is not closed the amount of items in the extra table can grow unlimited. A simple example is to try to segment in a limited number of intervals (higher than 1) the entire $\mathbb{R}$: the number of intervals possible are 1 and ∞.

$$prospect(\mathbf{C}) = \mathbf{x_C} \tag{5.17}$$

$$\mathbf{x_C} = \left\{ \forall \langle c, f \rangle \in \mathbf{C}\ \exists \langle x, p \rangle \in \mathbf{x_C} \;\middle|\; x = c \wedge p = \frac{f}{\sum_{\langle c,f \rangle \in \mathbf{C}} f} \right\}$$

The last part of the algorithm builds the structure $\mathbf{C}$ from the rewards received by the agent, as it experiences it. A simple version of the algorithm is shown in Algo. 5.1. The function `Cluster(•)` receives as arguments: the current cluster structure, the clustering threshold, and the new reward to be included in the cluster. This algorithm avoids the linear growth in both memory and computational complexity. This means that once the reward function codomain is partitioned the amount of pairs in $\mathbf{C}$ stays constant. Another advantage of this algorithm is that it builds the structure as the rewards are received, which relaxes the codomain restrictions (the codomain must be limited but the limits must not be known beforehand).

Algorithm 5.1: `Cluster(•)`

Data: $\langle$centroid, accumulator$\rangle$ set $\mathbf{C}$
Data: the threshold ϵ
Data: the reward r to be included
Result: Updated set $\mathbf{C}$

```
1  c_min ← NIL ; /* c_min is the closest centroid to r */
2  Δd ← ∞ ; /* Δd is the distance between r and c_min */
   /* finds the closest centroid to the reward r */
3  forall pair c ∈ C do
4  |  if Distance(c, r) < Δd then
5  |  |  Δd ← Distance(c, r);
6  |  |  c_min ← c;
7  |  end
8  end
   /* checks if the centroid is suitable for aggregation */
9  if Δd ≤ ε then
10 |  c_{c_min} ← (r + a_{c_min} c_{c_min}) / (1 + a_{c_min}) ; /* new centroid value */
11 |  a_{c_min} ← a_{c_min} + 1 ; /* increments counter/accumulator */
12 else/* it is a new centroid */
13 |  C ← ⟨r, 1⟩ ; /* add the new centroid */
14 end
```

The editing method presented has also drawbacks: it has a fixed threshold that may not be the most appropriate clustering method.[27] Another issue is its strong dependency on the initial values. If the algorithm is used as in Algo. 5.1 then the first rewards must be a characteristic draw from $R(\bullet)$, i.e. a well distributed sequential draw from $R(\bullet)$. For example, let $\epsilon = 5$ and a reward sequence be $\langle 5, 15, 7, 13, 9, 11, 9, 11, 9, 11 \rangle$. In this case $\mathbf{C} = \{\langle 7.8, 5\rangle, \langle 12.2, 5\rangle\}$, which is clearly not an acceptable clustering. A better clustering would be $\mathbf{C} = \{\langle 6, 2\rangle, \langle 10, 6\rangle, \langle 14, 2\rangle\}$

To overcome this problem some modifications are possible. The most simple is to fix where the centroids can be; in the example, let only multiples of the ϵ to be the centroids (such as: -5, 0, 5, 10, and so on).[28] Another approach would be to have, besides the threshold, the sample size that is considered statistical significant. Then before this sample is available the algorithm keeps all outcomes. Thus the behaviour would be: if the sample is not large enough then create the

[27] An ϵ depending on the region of $R(\bullet)$ may be needed. A simple example would be the Normal curve, where near 0 it is interesting to have a smaller ϵ than near the extremes.

[28] This approach simplifies the computational complexity of the algorithm but does not reflect the abstract concept of a centroid, which is the "centre" of the cluster.

cluster structure every step. Then, when it has reached a sample large enough, it switches to the Algo. 5.1 using the last created cluster as the starting structure. This last suggestion implies that the reward function acts as a random variable (if the PDF is not known) and that the sample size must be estimated/known.

5.5.1 Bias

It is also noteworthy saying that this clustering method is biased. The bias exists because it gives the same weight to new and old experiences. This means that a past too "bad" or too "good" result will be remembered as "fresh" and any other experience, even though it may not reflect the current world state, i.e. the rewards received recently. As it will be seen in chapter 8, this has a marginal impact on the results but may not be ignored. The improvement of the clustering method is left as future work.

5.6 Modified $\mathcal{Q}$-Learning

The modifications necessary are a combination of what was presented as the single-state $\mathcal{Q}$-Learning in Sec 5.4.2 and the clustering method from the previous section. As said before in Sec. 4.6, the editing phase has not been formalised and the standard clustering methods are not suitable for the task (Sec. 4.6.2). Therefore a new clustering method is proposed (Sec.5.5) to cope with the editing phase. This algorithm must be included into the modified $\mathcal{Q}$-Learning. The complete modified $\mathcal{Q}$-Learning is presented in Eq. 5.21. It is assumed that $\mathbf{C}_n \leftarrow \texttt{Cluster}(\mathbf{C}_{n-1}, r_n)$ represents the centroid set at step n and $\texttt{Cluster}(\bullet)$ the function in Algo. 5.1, then the modified $\mathcal{Q}$-Learning algorithm is given by Eq. 5.18, 5.19, 5.20, and 5.21; where $v(\bullet)$ and $\pi(\bullet)$ are the same distortion functions from the PT specification (Eq. 4.2 and 4.3). It is worth noticing that the $\mathcal{Q}$-Learning has no complexity increase because the only modifications are the inclusion of the set $\mathbf{C}$ and the $pt(\bullet)$ function. The $pt(\bullet)$ function depends on the size of $\mathbf{C}$ and on the function $\texttt{Cluster}(\bullet)$, which for its turn only depends on the size of $\mathbf{C}$. Yet $\mathbf{C}$ has a limited size (significantly smaller than the amount of iterations) meaning that no increase of computational complexity occurs.

$$pt(\mathbf{C}) = \sum_{\mathbf{c}\in\mathbf{C}} v(c_{\mathbf{c}})\pi\left(\frac{a_{\mathbf{c}}}{\sum_{\mathbf{c}\in\mathbf{C}} a_{\mathbf{c}}}\right) \tag{5.18}$$

$$\mathbf{C}_n = \texttt{Cluster}(\mathbf{C}_{n-1}, r_n) \tag{5.19}$$

$$Q_n(s,a) = (1-\alpha_n)Q_{n-1}(s,a) + \alpha_n\left[pt(\mathbf{C}_n) + \gamma \sum_{s'\in\mathcal{S}} V_{n-1}(s')\right] \tag{5.20}$$

$$V_n(s) \equiv \underset{a\in\mathcal{A}}{\operatorname{argmax}}\left[Q_n(s,a)\right] \tag{5.21}$$

To make it clear how the algorithm works a simple example is given step-by-step. The clustering parameter is $\epsilon = 5$ and in Tab. 5.2 and 5.3 is the evolution of each of the structures according to the received reward, in the second column of Tab. 5.2. For the $\mathcal{Q}$-Learning parameters it is assumed: $\alpha_n = \alpha_0^n$, $\alpha_0 = 0.9999954$, $\gamma = 0.9$, and $Q_0(\bullet) = 0$ – which are the parameters suggested in [Lit94].

In the first table (Tab. 5.2) it is shown what happens with the centroid set $\mathbf{C}_n$ as it incorporates each new reward (column "r_n" in Tab. 5.2). Then in Tab. 5.3 one can see how the $\mathbf{C}_n$ set is converted into a prospect, where the accumulators are converted into probabilities, and what is the $pt(\bullet)$ value for this prospect (columns "$prospect(\mathbf{C}_n)$" and "$pt(\mathbf{C})$" in Tab. 5.3, respectively). The last column in Tab. 5.3 shows the Q values at each step.

Table 5.2: Evolution of $\mathbf{C}_n$ (part 1)

n	r_n	$\mathbf{C}_n$	Operation in $\mathbf{C}$
1	100	$\{\langle 100, 1\rangle\}$	includes a new pair $\langle 100, 1\rangle$
2	102	$\{\langle 101, 2\rangle\}$	aggregates with pair $\langle 100, 1\rangle$
3	101	$\{\langle 101, 3\rangle\}$	aggregates with pair $\langle 101, 2\rangle$
4	101	$\{\langle 101, 4\rangle\}$	aggregates with pair $\langle 101, 3\rangle$
5	110	$\{\langle 101, 4\rangle, \langle 110, 1\rangle\}$	includes new pair $\langle 110, 1\rangle$
6	105	$\{\langle 101.8, 5\rangle, \langle 110, 1\rangle\}$	aggregates with pair $\langle 101, 4\rangle$
7	113	$\{\langle 101.8, 5\rangle, \langle 111.5, 2\rangle\}$	aggregates with pair $\langle 110, 1\rangle$
8	120	$\{\langle 101.8, 5\rangle, \langle 111.5, 2\rangle, \langle 120, 1\rangle\}$	includes new pair $\langle 120, 1\rangle$

Table 5.3: Evolution of $\mathbf{C}_n$ (part 2)

n	$prospect(\mathbf{C}_n)$	$pt(\mathbf{C})$	$Q_n(s, a)$
1	$\{\langle 100, 1\rangle\}$	57.54	57.54
2	$\{\langle 101, 1\rangle\}$	58.05	109.84
3	$\{\langle 101, 1\rangle\}$	58.05	156.90
4	$\{\langle 101, 1\rangle\}$	58.05	199.26
5	$\{\langle 101, 0.8\rangle, \langle 110, 0.2\rangle\}$	51.58	230.91
6	$\{\langle 101.8, 0.8\rangle, \langle 110, 0.2\rangle\}$	52.19	260.01
7	$\{\langle 101.8, 0.71\rangle, \langle 111.5, 0.28\rangle\}$	51.43	285.44
8	$\{\langle 101.8, 0.625\rangle, \langle 111.5, 0.25\rangle, \langle 120, 0.125\rangle\}$	60.97	317.86

5.7 Summary

In this chapter the $\mathcal{Q}$-Learning algorithm is approached and its modifications to conform the PT. It is argued that learning is necessary because the Markovian state transition function is not known beforehand and therefore the agents must learn it (Sec. 5.2). Then it is said that here the MDP approach is used (Sec. 5.3) because the amount of agents is high and it is not suppose that persons keep track of each single other competitor when its amount is high, which is also the opinion from von Neumann and Morgenstern (see quotation at the end of Sec. 5.3).

Following this discussion the standard $\mathcal{Q}$-Learning is presented (Sec. 5.4) as well as the first step towards its modification to be PT based (Sec. 5.4.2). Then the final issue is approached: the editing phase (Sec. 5.5). It is also important to mention that the clustering method adopted is biased, as discussed (Sec. 5.5.1). With all necessary algorithm it is then possible to present the modified $\mathcal{Q}$-Learning (Sec. 5.6), which includes a simple example to illustrate how the algorithm works. Therefore the complete $\mathcal{Q}$-Learning proposal to tackle the PT based learning algorithm is presented in this chapter.

Chapter 6

Traffic And Route Choice

In this chapter the general aspects of traffic modelling and then the specific aspects of route choice modelling is presented. The modelling framework used here is the four-step model, which is addressed first in this chapter. When the "big picture" is shown then the specifics of the route choice problem are discussed and how it is tackled in this work. It also includes the mapping from traffic to Markov Decision Process (MDP), which is necessary since the MDP is the adopted modelling approach for the agent (Sec. 5.3).

6.1 Concepts Review

The only necessary concept necessary here is the MDP modelling adopted by the learning algorithm (Sec. 5.3). This means that it is necessary to map each element of the MDP to the corresponding traffic element. The basic property of an MDP is that it does no depend on historical evolution of the states, i.e. the next state depends only on the current state and the action of the agent. Specially important is to define which are the actions, the states, and the rewards.

6.2 Traffic Modelling

The goal of any given model is to better understand the modelled phenomenon. This means that a model is only useful if it casts some light over the phenomenon and helps to explain its observed behaviour and also to predict future behaviour under other conditions. In a nutshell: understanding and prediction are the key features of a good model. According to [HB07], for traffic this translates into understanding its dynamic and also its behaviour in a future condition.

The practical applications of a traffic model are diverse and depend on who is looking at the model. For traffic authorities the goal is to understand the network use (link[1] load) and how conditions will change, e.g., when a new highway is built or the impact of a new industry facility for the vicinities of it. For logistic companies it is to find optimal ways to navigate through the network with minimal cost (avoid overloaded links for example) and help to plan future paths according to the future conditions. For public transportation companies it is interesting for planning schedules, and, of course, profitability (how may persons are in each line at each time), to plan future extensions (such as new lines or schedules) to improve satisfaction and profit. For transportation researchers a model helps to understand how people make decisions to fulfil their transportation needs.

In the previous paragraph, except for the last example, all applications are basically interested into link load and its behaviour. This is also the focus in this text, since this seems to be the only available standard against which a model can be thoroughly evaluated. The reason relies on the

[1] Link is the term to specify a segment in a street that connects two crossings and link load is the measurement of the amount of vehicles in this segment at given time.

available data, which is mainly link load. Nevertheless some hope can be expected from projects such as the Berkeley's Mobile Millennium Project[2] that collects a comprehensive route decision dataset, although their findings are still to be published. For the time being, the only practical benchmark for traffic models is link load, i.e. a model is as good as it is capable of reproducing the network load.

6.3 The Four-Step Model

The most well-known traffic modelling technique is the four-step model [Bat07, McN07] (FSM). This model requires, as the name says, four steps to be fulfilled before modelling traffic. These steps are:

Trip Generation Based on various data (such as socio-demographic and land use surveys) determine which are the potential origins and destinations.

Trip Distribution When all origins and destinations are realised, it is necessary to establish how many individuals go from each origin to each destination. This means that the origin destination matrix (OD matrix) is specified.

Mode Split Once all transportation needs are determined it is necessary to decide, for each trip, which transportation mode will be used. For instance how many individuals (for each OD pair) are using a private vehicle, a public transportation (and its kind), or another transportation mode such as car-pool or bicycle.

Assignment When all previous steps are fulfilled, then the modeller must find the routes for each trip (with its corresponding mode) that better corresponds to the observed link loads (occupation).

Even though assignment is seen as a single step by the FSM, it can be sub-divided into other four steps [Bat07]:

Route Choice For each OD pair a route will be assigned.

Load Aggregation The link load will be aggregated based on the chosen routes, i.e. how many routes share each one of the links.

Capacity Evaluation Since each link has a specific capacity, the corresponding side-effects must be taken into account, such as travel-time penalties.

Cost Evaluation For each OD pair the travel cost, based on the routes, will be evaluated and given as feedback.

The whole process is usually iterative running until an equilibrium is reached, i.e. the individuals can no longer ameliorate their costs. The Wardrop [War52] equilibrium definition is normally used, which is also known as user equilibrium (UE). When the equilibrium is reached, the model can be evaluated, verifying how good it actually reflects the observations. Since link load/occupation is the most largely available data from a network this is used along with a fitness function (Sec. 2.5.1) to inform how the model fits the data. In the praxis this means that a model must ultimately reproduce the observed link loads through route choice modelling, although the ultimate goal is to use the model to forecast future conditions of the network or how will the load be distributed in some future time. The logic behind the model is this: if all steps have been correct and the model reproduces the observed network load (with an acceptable accuracy) then the decision model must be correct. This means that the route choice model also correctly captures how people decide to go from A to B (the transportation need). Thus, because the way how people make decision remains the same,[3] when the scenario changes the results (link load) given by the model will be correct.

[2] `http://traffic.berkeley.edu/`, launched on November 10th of 2008.
[3] This is based on the utility theory, which says that the utility function is unlikely to change (chapter 2).

6.3.1 Route Choice Problem

From the four steps in the FSM the last one (assignment) is the most critical because it is the only that just observing the behaviour does not bring much information to the modelling process. In comparison, to find out which are the origins and destinations (the trip generation step) is rather simple and can be estimated by locating the several zones in the modelled region (living, shopping, and working zones) and any future situation can also be estimated by the land use planning. This information is all what is needed to build the necessary model, because the observations are the parameters. For the next step (trip distribution) the same is true, even though harder to acquire, knowing the amount of individuals in each zone is sufficient. These parameters can be collected from demographic and socio-economic surveys. The mode split can also be estimated by the number of registered vehicles, public transportation use, and so on. It is fair to say that this is not as easy as the previous two but the observation gives sufficient details about its nature.

The last step is the most difficult one because only observing it (assuming that all routes are known) does not give much insight on how the routes were chosen. Since the goal is to capture how people choose their routes. It is assumed that it is unlikely to change over time. Thus, having all routes provides the gold standard against which the model must be evaluated, not the model itself. This means that a choice model is still necessary.

A route choice is a discrete choice model, where a set of finite and countable choices (choice set) is given and the individual chooses the element that best suits him/her. If this choice problem can be modelled using a utility function it means that a mathematical function can be built that mimics the individuals' choices. Since a utility function is used it is meaningful to discuss about its modelling and inevitably about its states of equilibrium.

6.3.2 Equilibrium And Utility Functions

The concept of user-equilibrium is only meaningful with an associated fitness function (in this case the utility function) that says when a change in the choice does not improve the expected reward (the equilibrium definition). It is then meaningful to discuss about rationality and non-rationality in this context. According to [CS09], the rational equilibrium is said to be the objective user optimal decision and for the non-rational equilibrium it is said to be the subjective optimal decision. But regardless of the theory the optimal choice is always the one that yields the maximum utility, according to the used utility function.

In traffic as already discussed, the standard approach is to assume rational travellers, on the other hand it seems that the Prospect Theory (PT) is gaining more and more attention from the traffic researchers. But in a recent article [CS09] a continuous derivation of the Cumulative Prospect Theory (CPT) is investigated for modelling a five link network. The main argument for the use of the CPT is that it is – arguably – a better approach to model how people evaluate choices under uncertainty, i.e. when the costs are not known beforehand. On the other hand, the very existence of an equilibrium in traffic is questioned in [Goo98], where it is argued that an equilibrium cannot be supposed valid for traffic when the data is not collected in such state and no evidence can be shown to support that the system moves towards the equilibrium. That said, it is supposed here that an equilibrium exists and, as in [CS09], this equilibrium is the one established by the perceived cost, i.e. the utility function does not reflect necessarily the real cost of a route, but what the traveller/agent perceives as its cost, its subjective cost.

Regarding the utility function, the only variable is the travel-time. But before presenting the travel-time calculation method (in the next section) it is necessary to discuss how relevant the travel-time function is for the decision process. Recalling the discussion about the utility function in chapter 2 it is said that the utility function forms an ordinal scale and that any monotone affine transformation is possible. That said, the only important feature of a travel-time function for the route choice model is that it must be proportional to the link load, i.e. the more vehicles in the link the higher is the travel-time in that particular link. The argument is the following: suppose that the real travel-time function[4] (if such modelling is possible) is an exponential curve,

[4] The very nature of the travel-time as a function can be questioned because. Since it is independent from the

such as the one supposed by the Bureau of Public Roads [Tra00] (BPR) and that the drivers take only the travel-time into consideration. For the utility function side, only the ordering is relevant, i.e. how the links/routes are scored (which is the best, the second best, and so on). With this in consideration a utility function that uses the exponential values of the travel-time function is equivalent to another function that uses the logarithm of the travel-time, because the logarithm does not change the ranking of the links/routes (even though the actual utility values are different).

Now suppose that the travel-time function is hypothetically linear. Then again both the examples of utility function explained before are still equivalent and valid, because the travel-time is still proportional to the amount of vehicles and the utility functions still rank more occupied links below less occupied. This means that taking the point-of-view of the agents and the decision-making, the specific travel-time function is almost irrelevant. It must only reflect the same behaviour observed in reality, which is higher occupation implies higher travel-time.

This remark has a caveat nevertheless. If a travel-time function is used that is only loosely related to the real travel-time function then this model cannot be used to model problems where the departure and arrival times are relevant. In this case none of the so called day-activity planning models can use this function. Another issue is the lack of a specific travel-time function for urban traffic. The BPR function is explicitly said to only model highway traffic [Tra00]. A more specific modelling is attempted in [DG08] using fundamental diagrams [RPM04][5] for urban links in San Francisco (USA) and Yokohama (Japan). The approach was to extract four linear regions from the estimated fundamental diagrams and then to calculate the travel-time according to the linear region corresponding to the occupation. For compatibility reasons the BPR method is used, which is presented in Sec. 6.4.1.

6.4 Modelling Traffic Assignment

In this work it is assumed that the first three steps are already fulfilled. This means that the model needs as input the OD matrix (steps 1,2, and 3) and the objective is to model the traffic assignment. To accomplish this task some prerequisites are necessary. First, the choice set must be provided, i.e. the route set for each OD pair.[6] The method used is the so called Link Elimination Shortest-Path [ACaERSMM93] (LESP) with one modification: only one link is eliminated per algorithmic run – this is formally presented in Sec. A.5. Second, a utility function structure is provided and here only the travel-time is taken into account. This means that it is supposed, in this first model, that only travel-time is relevant for the individual route evaluation. Since an iterative process is used to achieve the network's final state, an equilibrium definition is necessary. To cope with that both concepts are used: rational user-equilibrium (EU) and the subjective user-equilibrium, as [CS09] defined it. The first definition is obviously used for the rational models and the second for the non-rational PT based model proposed.

As mentioned in the previous paragraph the only variable in the utility function is the travel-time, but travel-time is a cost. Then the simplest way to extract the worthiness of travel-time is in Eq. 6.2, where: r is the route being evaluated, $l \in r$ is a link used in the route r, and t_l is the time consumed to transverse[7] the link l. In a simpler explanation: the utility of a route is equal the negative value of the route's travel-time that is the sum of the consumed time in each of its links. This leaves the t_l to be calculated that is done by the function $bpr(\bullet)$ (Eq. 6.3) explained

specific drivers in the link. Therefore the travel-time, per definition, is not a function (the counter-domain is not unique for the same domain value). This can be better understood if different quotes of aggressive (move faster) and non-aggressive (slower) drivers are mixed and, with a fixed amount of drivers each time, made to travel the same link. It is expected that this will produce different behaviours and travel-times as well.

[5] Fundamental diagrams is a graphic plot that has the vehicular density as the x-axis and flow in the y-axis. Density, for its turn is simply the amount of vehicles currently on the link divided by the link capacity (the maximum amount of vehicles that fit on the link). The flow, on the other hand, is the amount of vehicles per time unit that travels the link.

[6] Because in this first approach no modus is modelled, it is assumed that each OD pair has an associated modus.

[7] Transverse is the technical term to designate the time consumed to travel a link.

in the very next section.

$$u(r) = v_{travel-time}(r) \tag{6.1}$$
$$v_{travel-time}(r) = -\sum_{l \in r} t_l \tag{6.2}$$

The network is not, however, used as it comes to the simulation. First the input graph is transformed into a super-network (Sec. A.2) that stratifies the input graph according to its multiple modi. This means that each modus has its own network, with the relevant links for this particular modus, and then the different networks are interconnected, so that modus change is allowed. This structure provides a better way to identify and control the modus change when generating a path in the network.[8] Therefore a modified shortest-path algorithm is used (Sec. A.4) for generating routes.

6.4.1 Travel-Time Calculation

For the travel-time calculation the formula provided by the Bureau of Public Roads [Tra00] (BPR) is used. The reason for its adoption is compatibility, i.e. because most of the commercial traffic simulation programs, such as VISUM,[9] also use it. This function is in Eq. 6.3 below, where, according to [Avi06], $\theta = \rho = 2.0$, l_{load} is the occupation of link l, C_l is the capacity of l, l_{length} is the length of l, v_{max} is the maximum speed allowed, and l_{lanes} the amount of lanes of l.

$$bpr(l) = t_{freeflow}(l) \times \left(1 + \theta * \left(\frac{l_{load}}{C_l}\right)^{\rho}\right) \tag{6.3}$$
$$t_{freeflow}(l) = l_{length}/v_{max} \tag{6.4}$$
$$C_l = 0.3\,(l_{lenght} l_{lanes}) \tag{6.5}$$

It is, however, worth noticing that this is not appropriated for urban traffic scenarios because it was explicitly developed for highway traffic. The travel-time curve for a single lane link with $100m$ is depicted in Fig. 6.1 and the fundamental diagram in Fig. 6.2.

6.4.2 Translating Route Choice To Link Load

As explained in Sec. 6.3 the method for translating route choice to link load is to collect all decision (routes) and to count how many drivers took the same link and then calculate the corresponding travel-time informing that back to the drivers. The formalization of this concept is made below:

1. Ask all agents about their route choices: $\mathbf{a} = \langle a_0, a_1, \ldots, a_n \rangle$, where $\mathbf{a}$ is the joint action vector and a_i is the route choice of agent i.
2. Reset all link loads: $\forall e \in E_{\mathcal{S}} \Rightarrow e_{load} = 0$, where $\mathcal{S}$ is the graph representing the network, $E_{\mathcal{S}}$ the link set of the graph, and e_{load} is the link occupation (amount of agents occupying this particular link).
3. Set the new link loads: $\forall e \in E_{\mathcal{S}} \Rightarrow e_{load} = \sum_{a \in \mathbf{a}} \delta_e$, where a is a route (in this case a list of links) and $\delta_e = \begin{cases} 1 & iff\ e \in a \\ 0 & \text{other else} \end{cases}$.
4. Inform back to the agents the travel experience: $\mathbf{R} = \langle \mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_n \rangle$, where $\mathbf{r}_i = \langle r_0, r_1, \ldots r_m \rangle$ and $r_j = bpr(e_j \in a_i)$ (Eq. 6.3). This means that the agent receives back a tuple $(\mathbf{r}_i)$ where each element corresponds to the travel-time, which is calculated by function $bpr(\bullet)$, for all links (e_j) of the chosen action (a_i).

[8] A Context-Free Grammar is used for this purposes (Sec. A.3).
[9] `http://www.ptvag.com/`

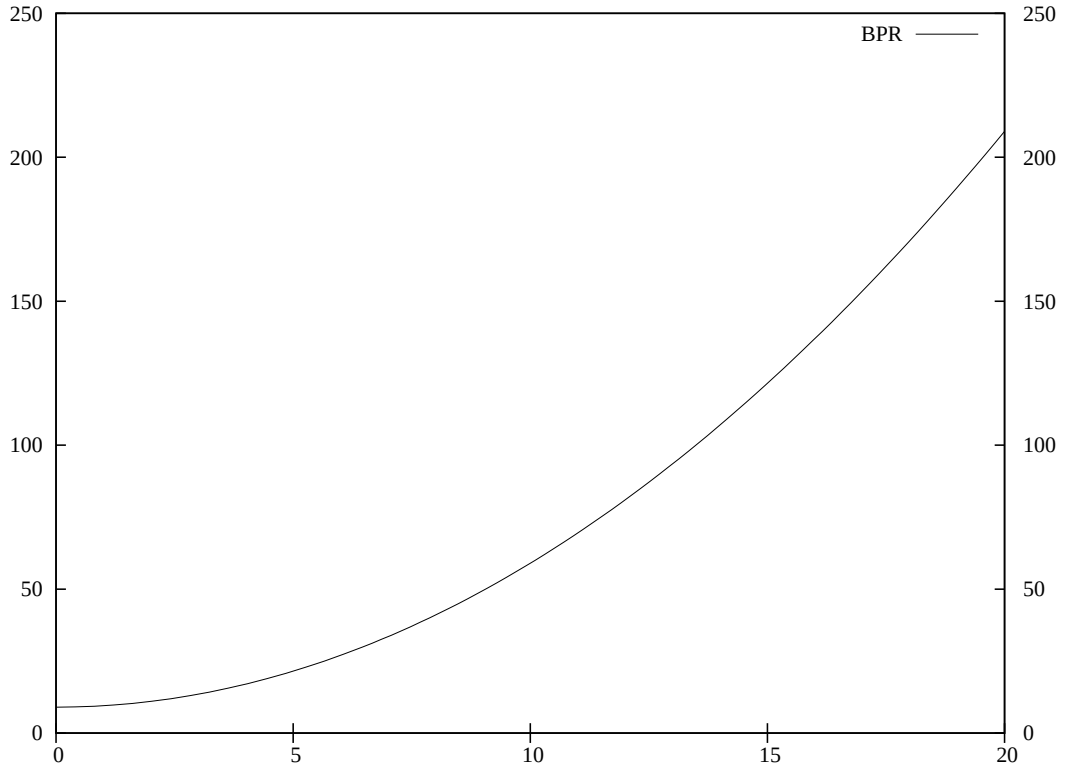


Figure 6.1: Travel-time function

6.5 Traffic As An MDP

As said in Sec. 6.1 and explained in Sec. 5.3 the world is, for the agents, an MDP instance. This means that the traffic assignment elements must be mapped into the MDP elements. First, each step is defined as transportation event, i.e. a step encloses an entire trip/travel (from an origin to a destination). Second, the traffic network must provide the states and rewards to the agents.

To cope with the last requirement, the traffic network is transformed into a super-network [CFCLB03, FCHLvN03, FCvNB04], which is formalised in Sec. A.2. A super-network can be said to be a stratified labelled digraph, i.e. each modus is represented by one stratus where all links have the particular modus of the stratus. The links are also labelled according to which modus they have. This structure is useful because it not only provides means to assign special costs (such as the cost of buying a bus ticket) but also to provide a multi-modal navigation through the network. This means that using the super-network representation this work can be easily extended to provide not only the traffic assignment modelling but also the modus split modelling (left as future work).

The states, as already mentioned, are the OD pairs, i.e. a state is defined by an origin and a destination node/vertex. Therefore the action set is the route set for the given state (OD pair).

The last element of the MDP specification is the reward function. This is done by informing the travel-time experience of each link in the agent's chosen route. Formally, the traffic instance of the MDP tuple $\langle S, A, T, R \rangle$ is in Eq. 6.6. In this definition some other structures are defined, which are: $\mathcal{S}$ for the input super-network; $V_{\mathcal{S}}$ for the vertex set of $\mathcal{S}$; `AllPaths`$(\bullet)$ an algorithm that generates all paths for a given OD pair (an example is in [MMS90]); r is a route; $l \in r$ is a link in route r; $bpr(\bullet)$ a function that returns the travel-time for the given edge/link.

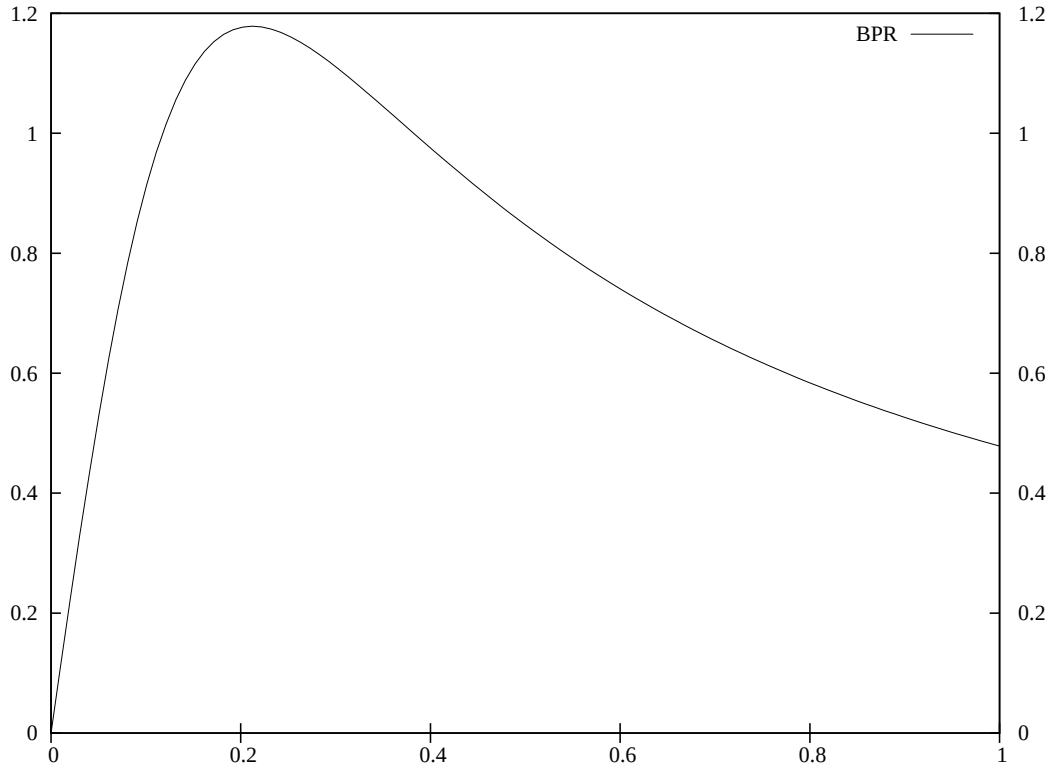


Figure 6.2: Fundamental diagram

$$
\begin{aligned}
MDP &= \langle S, A, T, R \rangle \\
S &= \{\forall \langle o, d \rangle \in V_{\mathcal{S}} \times V_{\mathcal{S}} | o, d \in V_{\mathcal{S}} \wedge o \neq d\} \\
A &= \cup_{\langle o,d \rangle \in S} A_{\langle o,d \rangle} \\
A_{\langle o,d \rangle} &\subseteq \texttt{AllPaths}(o, d) \\
T(\langle o, d \rangle, r, \langle o^{'}, d^{'} \rangle) &= \begin{cases} 1.0 & iff \langle o^{'}, d^{'} \rangle \equiv \langle o, d \rangle \forall r \in A \\ 0.0 & \text{other else} \end{cases} \\
R(s, r) &= -\sum_{l \in r} bpr(l)
\end{aligned}
\tag{6.6}
$$

In Eq. 6.6 the action set $A_{\langle o,d \rangle}$ is only a subset of $\texttt{AllPaths}(o, d)$ because it depends on the path generation algorithm and, as argued before, the path generator used is the LESP (Sec. 6.4).

6.6 Summary

In this chapter the traffic modelling is approached. The modelling paradigm used is the four-step model (FSM), which is the standard approach (Sec. 6.3). The focus is on the route choice problem (Sec. 6.3.1), which is the last step in the FSM. It is also stressed that modelling traffic assignment usually means to reproduce the link loads observed in the data because that is normally what is available. In Sec. 6.4 it is discussed how route choices are translated into link occupation and then into travel-time experiences. In the last section, the mapping from traffic to an MPD instance is made. This step is necessary to make it possible to use the $\mathcal{Q}$-Learning formalism (chapter 5) for modelling the route choice problem, which is the scenario used to evaluate this work.

Chapter 7

Agent Architecture

In this chapter the elements are put together into an agent architecture. These elements are the learning engine (Sec. 5.6), the split reasoning (Sec. 4.2.1), sensory (in this case travel-time feedback, Sec. 6.4.1), and memory management (explained in details in this chapter). In the proposed architecture the concepts of split reasoning (discussed in chapter 4) and non-rationality, as well as rationality, are incorporated. A quick note about the typefaces here employed: regular typeface is used as in "System 1" it refers to the idea evoked by the concept and the typewriter typeface appears as in "`System 1`" it refers to the architectural mechanism that, in this case, cope with the "System 1" concept.

7.1 Concepts Review

Here the only concept that is needed from the previous chapters is the bi-parted reasoning system (Sec. 4.2.1). The bi-parted system means that the decision-making is performed at two different levels. The first, called System 1, is the intuitive level and corresponds to the "quick-and-dirty" decision-making. This part accounts for the decision with which the agent is accustomed with, i.e. the decisions made by System 1 refer to problems that are more-or-less known by the agent and whose solutions are rapidly available. The second level is the System 2 that accounts for the decisions that need a more elaborated decision process. At this level the options are collected and analysed before any choice is made. This is a more cumbersome process and therefore the agent, when making the decision using the System 2, tries to truly optimise its choice.

7.2 Why An Agent Architecture

From what has presented it may seem unnecessary to have an agent architecture since the elements to be put together present low or no complexity. The main reason is to organise the concepts and make it simpler to understand how data, reasoning, and sensory fit together. Another one, not less important, is to make it simpler to extend this architecture into one that includes new features (some examples are given in chapter 10). The last major contribution of an architecture is to lay the foundations for implementation by explicitly pointing out the basic building blocks.

It also makes it simpler to show how the specific features used here are, such as the split reasoning (chapter 4) and the flexibility to switch between the Expected Utility Theory (EUT) and the Prospect Theory (PT). Some secondary features are: the switch between exploration (experimenting with the world) and exploitation (using the own world model to make decisions); as well as to have an intuitive decision-making level (required by the split reasoning model).

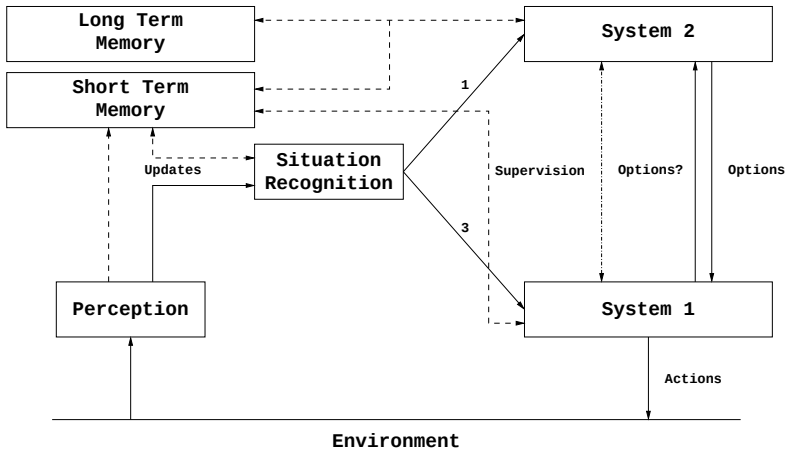


Figure 7.1: Proposed agent architecture

7.3 Proposed Agent Architecture

The proposed architecture is depicted in Fig. 7.1. In this architecture two reasoning levels are present and they are named `System 1` and `System 2` after [TK83]. As discussed in Sec. 4.2.1; the bi-parted decision-making engine is supported by several researchers and medical evidences, therefore here only two decision-making modules are used. The first is the intuitive level (System 1) that account for known situations and for those whose options are already available. The decision-making at this level is sometimes described as a decision that "pops-up" to mind and in general it is said to be "quick-and-dirty". This means that the decision is not necessarily the best (whenever the criteria are) but the best among the ones that are promptly available to the agent.

In System 2 on the other hand the highest reasoning is situated, where more options are analysed (or even generated) and their estimated outcomes taken into account. This means that the System 2 is responsible for, at a first glance, supporting the System 1 in the decisions that it (the System 1) cannot make. In this case System 2 acts as a solution provider when the System 1 falls short. Another function of the System 2 is to supervise the activities of System 1, i.e. if the outcome (or expected outcome) of System 1 is below the agent's acceptance level (Sec. 4.2.1) it takes over the decision processes and corrects the System 1's decision.

Another point is that here a split memory mechanism is proposed where each module has its own memory. The `Short Term Memory` is the only memory available to `System 1` while `System 2` has access to both, the short and long term memories. The reactive behaviour is in `System 1` and, as explained in Sec. 4.2.1, the `System 2` is only triggered when `System 1` performed poorly or does not know how to decide, which is not the case in reactive responses. In this model the supervision (dash-dotted line identified with `Supervision` in Fig. 7.1) is explicit depicted as well as the option supplier (the extra command line `Options?` from `System 1` to `System 2` in Fig. 7.1).

7.3.1 Memories

As discussed in Sec. 4.2.1, the bi-parted reasoning system asks for a decision from the intuition level (System 1) and this first decision can be corrected by the reasoning level (System 2) and that if the System 1 cannot solve the problem it is escalated to the System 2. It is also known [AF00] that as easy an information is available as probable it is that it will be used – this is called accessibility bias. In practical terms it means that if a decision must be made and the short-term memory already contains any options it is probable that the choice this will be made among them. Then, if no option is promptly available, the System 2 will be triggered to supply an option set from which a decision will be made. Therefore to support this decision-making mechanism a suitable memory organisation must be provided. In this case the memory is split into two: short and long

term memories.

Short Term Memory

To support the intuitive decision-making module (`System 1`), the `Short Term Memory` is provided and it has some defined features. First, it is restricted in size, i.e. how many options it can hold. Therefore, according to [Mil56], the `Short Term Memory` is limited to 7 items per context, and the context here is represented by a state, i.e. each different state is interpreted as a different context. Specializing this concept for traffic, it means that each origin destination pair (OD pair) corresponds to a state and, the routes being the available actions, each state holds a maximum of 7 routes. So the state "home-to-office" holds a maximum of 7 routes and "office-to-groceries-store" holds a different set of routes, which again can hold a maximum of 7 routes.

Along with the short term memory items a score (a single numerical value) is also kept with each one of them. This score is the "intuitive" utility of each item. This means that when the `System 1` makes a decision it ranks each of the items in the `Short Term Memory` and chooses the very best according to the scores/utilities associated with each item. For the current application the scores kept with each item are the latest evaluation made by the $\mathcal{Q}$-Learning, i.e. the $V_n(s)$ values (Eq. 5.3 and 5.21). This way the supervision of `System 2` is "pro-active" and keeps the `System 1` always up-to-date with the actual utilities.

It must be noticed that this is not entirely correct under the psychological point-of-view, because a separation between them exists and the scores in the short term memory are not promptly updated. This means that the `System 2` must have an internal threshold that establishes when the utility, or expected reward, does not meet the "acceptable" standards. However, no model for this behaviour is available and establishing an "ad-hoc" interpretation of this will introduce another variable in the model and one that will probably be wrong. Thus this lack of decoupling between `System 1` and `System 2`.

Aside from this, the short term memory suppose to be up-to-date, i.e. it must contains relatively "fresh" items. It means that items that are old are not supposed to be in it. To cope with that, in `Short Term Memory` the items are also tagged with their age, i.e. how long the item in the `Short Term Memory` was not being used. In practice, this means that each time the agent is in a given state and it makes a decision the non-chosen options have their age increased and the chosen has its age set to zero. As soon as an item reaches the maximum age threshold it is excluded from the `Short Term Memory`. This threshold, in the application, is set to be a week, supposing that each state is reached once a day. This means that, if a route hasn't been used a week long it no longer belongs to the short term memory, it is not "fresh" any more.

Another feature of the `Short Term Memory` is that if the `System 2` decides that new items are necessary and the memory is full, then items must be excluded. This means that the items in the `Short Term Memory` can be recycled and items with poor performance (with the lowest scores) are removed to make room for new items coming from `System 2`.

Long Term Memory

For the long term memory no limit is built in and it can hold as many items as available. This memory is the information storage, i.e. all data available and necessary for the high reasoning decision-making (System 2) is registered there. This is more a general storage space without a pre-established form. Translating to the used `Long Term Memory` for the traffic scenarios, it contains the $Q(\bullet)$ function/table (Eq. 5.2 and 5.20) from the $\mathcal{Q}$-Learning algorithm as well as the vectors $\mathbf{C}$ (Eq. 5.19 and Algo. 5.1). It also contains a table that associates links with travel-times, i.e. for each used link its travel-time is recorded and stored for use of the route generation module (explained later). As it can be noticed, the `Long Term Memory` is heterogeneous and can be seen as collection of different types of information sets.

7.3.2 Action Choice Generation

The choice set, i.e. the set of the possible actions for any given state can be beforehand given or on-demand generated. Either way it must be provided to the System 2, which is not responsible for its generation. The strategy adopted depends on the which algorithm is used to be the `System 2`. Because here the $\mathcal{Q}$-Learning is used and the test scenario is traffic assignment some advantage can be taken from this set-up. In essence the choice generation is, for traffic assignment, a route generator and for this task the algorithm used is a variation of the Link-Elimination Shortest Path (LESP), explained in Sec. 6.4. Then a simplification is made: not all possible routes are generated for each possible OD pair, but enough routes to fill the `Short Term Memory`.

Then, as the routes are being forgotten or recycled (as explained before), new routes are generated on-demand, at the request of `System 2`. This event, of demanding new routes, are only triggered when the exploring behaviour is activated (introduction of chapter 5). Where new options are generated until the `Short Term Memory` capacity is exhausted and than an action is chosen randomly from the options available in the `Short Term Memory`. This is explained later when presenting the `System 1`.

7.3.3 Environment

It was already explained, in Sec. 5.3, that the environment is assumed an MDP instance. This means that the `Perception` receives, as feedback, the current state and reward received for the previous chosen action. It also means that the `System 1` must inform an action for the current state for each simulation step. In traffic, the states are the transportation needs, i.e. the OD pairs, for which the agent must choose a route. As a feedback it also receives the reward for the last route (a function of the travel-time as explained in Sec. 6.4). A slight modification is made here, where the agent receives the travel-time reward corresponding to each of the links that compound the chosen route and the final reward is the sum of links' travel-times converted to utilities (Eq. 6.2).

7.3.4 System 1

Some of the aspects of the `System 1` were already mentioned but its internal mechanisms were not fully explained, which is the subject of this section. The `System 1` is meant to behave as a "quick-and-dirty" decision-maker (an effortless decision) and it is helped by the `Short Term Memory`. Thus, the basic algorithm implemented for `System 1` is in Fig. 7.2.

The algorithm is rather simple; it just selects the best option currently available in the `Short Term Memory`. In the case none is available it requests a new option from `System 2`. When the decision is made it is then informed to the `Environment`. The decision's score is, in the next step, then up-dated by the `System 2` that becomes the feedback from `Perception`.

7.3.5 Situation Recognition

Before explaining the `System 2` it is necessary to discuss the `Situation Recognition`. The main task is to inform the decision-making modules, `System 1` and `System 2`, about the current state and that a new decision is necessary. Then it looks if the `System 1` can make the decision, i.e. if the corresponding `Short Term Memory` has any item in it. In case it does, the `System 1` is responsible for making the decision. But if the corresponding `Short Term Memory` is empty then the `System 2` is informed to make the decision.

Another responsibility of the `Situation Recognition` is to forward actions' feedbacks, i.e. rewards/costs from the last action, to the `System 2`. It is also in this module that the "time" track is kept, i.e. it is responsible for keeping track of where in time the agent is. In the MDP world it means to know what is the current step.

This time tracking function also incorporates the ageing mechanism of the `Short Term Memory`. Therefore, the `Situation Recognition` recognises that time is passing and that the corresponding entries in the `Short Term Memory` must have their age increased. Thus, if any entry reaches the age

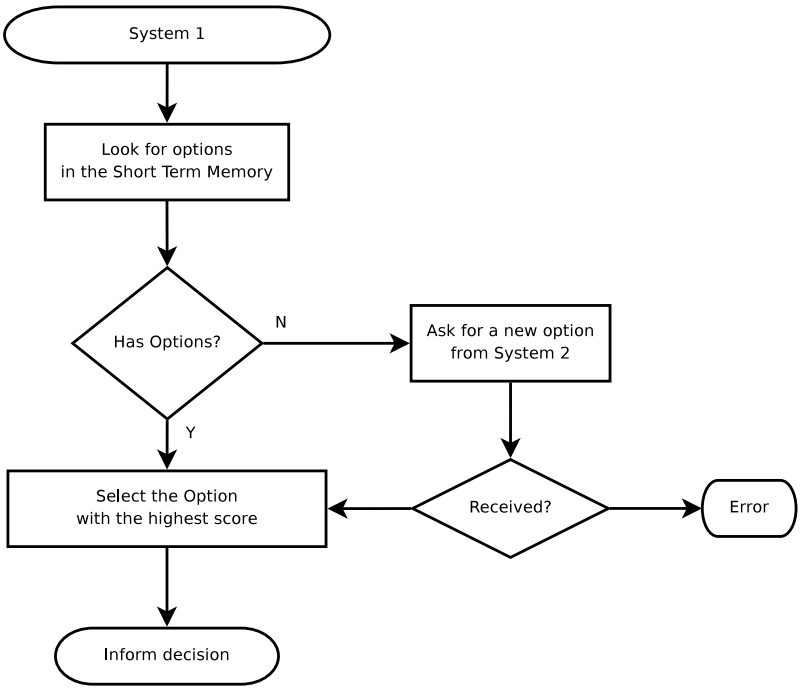


Figure 7.2: System 1

threshold the `System 2` is informed about this and acts accordingly. The basic ageing algorithm is presented in Fig. 7.3.

7.3.6 System 2

The System 2 (Sec. 4.2.1) is the highest reasoning processes. This means that at this level the cumbersome reasoning is made, where, when necessary, information is analysed and a more elaborated decision is made. At the `System 2` the decisions that cannot be handled by the `System 1` are made. It also corrects/supervises the activities of the `System 1`.

By cumbersome reasoning it is meant to analyse a multi-dimensional information space and to extract from it the relevant choices that lead to a decision. In the case of $\mathcal{Q}$-Learning, it means to analyse the multi-dimensional $Q(\bullet)$ table/function, evaluating the best choice and, in the case of the PT modified $\mathcal{Q}$-Learning, also to analyse the various vectors $\mathbf{C}$. For the traffic specific case, it also means to provide the route generator (Sec. 6.4) with updated link weights, which are the travel-times collected as feedback by the `Perception` and stored in the `Long Term Memory`.

The supervision is made in two opportunities. The first is the already mentioned "pro-active" supervision that updates the `Short Term Memory` scores when it receives the feedback from the last action. The second supervision or correction is when the `System 2` decides to explore instead of exploit. Then `System 2` takes over the decision process and randomly chooses an option to be the next action. It can also ask the action generator for more options, which in case of traffic means new routes, when they are necessary.

7.4 Summary

Here the agent architecture is presented. Its main objective is to lay the foundations for the practical application of the different concepts presented here. How to organise the memory, the decision flows, and how the different modules interact with each other (Sec. 7.2). This architecture is designed to accommodated the envisioned agent aspects (Sec. 7.3). This means that an MDP

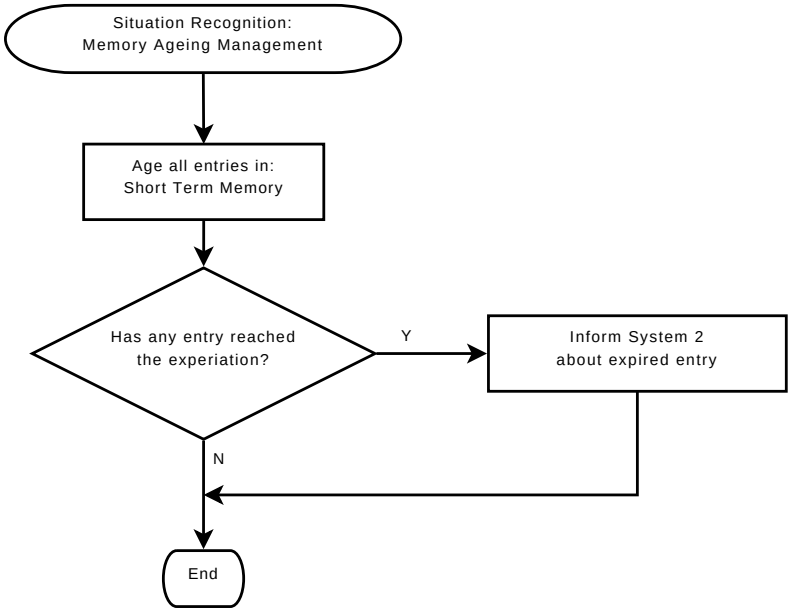


Figure 7.3: Memory ageing process

world is assumed and special attention is given to the interaction between the two reasoning systems: System 1 and 2 (Sec. 7.3.4, 7.3.5, and 7.3.6).

Chapter 8

Evaluation

In this chapter the results and evaluation of the agent architecture and learning algorithm are made. The objective is to evaluate in which conditions the Prospect Theory based $\mathcal{Q}$-Learning diverge from the rational behaviour. Then to verify if it is indeed better at reproducing real data and has a last point to see if the whole framework can scale to a real world scenario.

8.1 Concepts Review

The first point is the $\mathcal{Q}$-Learning algorithm (chapter 5) an its modification to be based on the Prospect Theory (Sec. 5.6). An important issue is the clustering algorithm (Sec. 5.5) used to cope with the editing phase necessary but not formalised for the Prospect Theory (Sec. 4.6). One may not forget that the clustering algorithm is biased and this bias was discussed in Sec. 5.5 and again when presenting the results (Sec. 8.8.1). Another concept is the bi-parted reasoning system (Sec. 4.2.1) which is also approached by the architecture (Sec. 7.3.6). The bi-parted system refers to the split of the decision-making process into two levels. The first, called System 1, is the "quick-and-dirty" decision-making that is triggered when the problem is well known by the agent and whose options are promptly available to the agent (in the short-term memory, Sec. 7.3.1 and 7.3.4). The second is called System 2 and here accounts for the learning algorithm, i.e. the $\mathcal{Q}$-Learning (chapter 5).

8.2 Evaluation Methodology

The objective of the experiments done and described here are to first show how and in which conditions the Prospect Theory (PT) based $\mathcal{Q}$-Learning (chapter 5) diverge from the rational behaviour. This means that the main goal is to provide the conditions under which the rationality is violated and why. For the experiments the traffic scenarios were adopted. Here three algorithms were used for the `System 2` reasoning engine: the standard $\mathcal{Q}$-Learning (STD), the so-called clustered $\mathcal{Q}$-Learning ($CluEUT$), and the clustered PT based $\mathcal{Q}$-Learning ($CluPT$). The standard $\mathcal{Q}$-Learning (Sec. 5.4) is the gold standard for the rational behaviour, i.e. this is the algorithm that behaves according to the von Neumann definition of rationality [vNM07] and this was proven correct in [WD92]. This means that if the presented architecture works then the standard $\mathcal{Q}$-Learning must converge to the expected rational behaviour. Second, because the clustering method, presented in Sec. 5.5, was not proven to be correct the claims must be supported by evidences. Therefore, the standard $\mathcal{Q}$-Learning was modified to use the clustering method (Sec. 5.5) proposed for the editing phase (Sec. 4.6) and it must, as the standard $\mathcal{Q}$-Learning, converge to the rational behaviour. Then, assuming that the previous two assumptions are correct, the PT modified $\mathcal{Q}$-Learning (Sec. 5.6) can be evaluated.

What is meant by the previous paragraph is the following. Given a simple enough scenario, where the expected rational behaviour is known, the validity of the agent architecture can be

demonstrated. This is done by using a reasoning that has a well known behaviour and then comparing this expected behaviour with the results produced using the architecture. Then the distortions, if any, imposed by the architecture can be evaluated by analysing the deviations observed in the simulation results and the expected results. For this step the standard $\mathcal{Q}$-Learning was chosen as the gold standard.

The second point is to evaluate the distortion introduced by the clustering method, i.e. to verify how biased it is. This bias can be measured by comparing the results of two equivalent reasoning engine: one with and the other without the clustering. These engines are the standard $\mathcal{Q}$-Learning and the modified clustered $\mathcal{Q}$-Learning. If the results of these two engines are not equivalent then the clustering method can be said biased and that it introduces distortions in the reasoning of the agent.

The last point is, if the architecture does not introduce any "artefact" then no deviation is expected between the simulated and predicted results. If the clustering method also shows no or an acceptable bias, then the PT based engine can be evaluated. Unfortunately the validity of the PT can only be established with real data, which in this case is sparse. To tackle this the data from [SSC+05, Chm05] is used to evaluate the PT based $\mathcal{Q}$-Learning, but this experiment does not give any clue about the working of the PT in traffic and which are the critical conditions for it. Therefore, some extended experiments were done to show how the PT based $\mathcal{Q}$-Learning operates and in which conditions it deviates from rationality.

Two scenarios are proposed, the commuter scenario with two routes, which is inspired by the El Farol Bar [Art94] Problem, and the city of Burgdorf (Switzerland). The commuter scenario is chosen because some real data is available in [SSC+05, Chm05] and also because it is simple enough to be understood (Sec.8.8 and 8.9). Using this scenario all analysis are made, investigating several different set-ups to verify the influence of the PT in traffic. Besides this scenario, the city of Burgdorf is also simulated (Sec. 8.10) because some data is available and its size can show that the simulation framework do scale. However, the main objective of simulating Burgdorf is to show that the agent architecture can scale to real-world scenario, not to fit the data.[1]

8.3 Different Algorithm For System 2

The different variations of the $\mathcal{Q}$-Learning used in the experiments are: *STD* referring to the standard $\mathcal{Q}$-Learning, *CluEUT* to the cluster modified $\mathcal{Q}$-Learning, and *CluPT* to the clustered PT based $\mathcal{Q}$-Learning. A compact view of the different modifications in the $\mathcal{Q}$-Learning algorithm is presented in Eq. 8.5. There $V_n(s)$ is the decision for state s at step n, $a \in \mathcal{A}$ is a route, α_n is the learning factor for step n, r_n is the travel-time for the last route, $d(r_n)$ is the reasoning selection function, and $\texttt{Cluster}(\bullet)$ corresponds to Algo. 5.1.

$$eut(\mathbf{C}) = \sum_{\mathbf{c}\in\mathbf{C}} c_{\mathbf{c}} \frac{a_{\mathbf{c}}}{\sum_{\mathbf{c}\in\mathbf{C}} a_{\mathbf{c}}} \tag{8.1}$$

$$pt(\mathbf{C}) = \sum_{\mathbf{c}\in\mathbf{C}} v(c_{\mathbf{c}})\pi\left(\frac{a_{\mathbf{c}}}{\sum_{\mathbf{c}\in\mathbf{C}} a_{\mathbf{c}}}\right) \tag{8.2}$$

$$d(r_n) = \begin{cases} r_n & \text{standard } \mathcal{Q}\text{-Learning} & STD \\ eut(\texttt{Cluster}(\mathbf{C}_{n-1}, r_n)) & \text{EUT based } \mathcal{Q}\text{-Learning} & CluEUT \\ pt(\texttt{Cluster}(\mathbf{C}_{n-1}, r_n)) & \text{PT based } \mathcal{Q}\text{-Learning} & CluPT \end{cases} \tag{8.3}$$

$$Q_n(s,a) = (1-\alpha_n)Q_{n-1}(s,a) + \alpha_n\left[-d(r_n) + \gamma \sum_{s'\in\mathcal{S}} V_{n-1}(s')\right] \tag{8.4}$$

$$V_n(s) \equiv \underset{a\in\mathcal{A}}{\operatorname{argmax}}\left[Q_n(s,a)\right] \tag{8.5}$$

[1] When this scenario is presented, it will be explained why it cannot be used for the PT evaluation (Sec. 8.10).

In the reasoning selection function $d(r_n)$ (Eq. 8.3) some clarification is necessary. First it is has a "minus" in Eq. 8.4 because the reward is actually a cost (travel-time) and therefore the effective reward is the negative of the cost (Sec. 6.4). Second, this function triggers the reward manipulation according to the reasoning engine being in use (STD, $CluEUT$, or $CluPT$). The first instance, the STD, the working is the one corresponding to standard $\mathcal{Q}$-Learning where the reward is used directly. The second case, the $CluEUT$, is the clustered version of the STD. This modification works by, as the $CluPT$, accumulating the rewards received into the cluster structure $\mathbf{C}$ and uses the same function to perform the clustering: the function `Cluster`($\bullet$) (Sec. 5.5). But the difference between $CluEUT$ and $CluPT$ is that the former calculates the rational utility derived from the clustering structure (the function $eut(\bullet)$ from Eq. 3.3 in Sec. 3.3) and the $CluPT$ calculates the non-rational utility from the PT (Eq. 4.1).

The function of $CluEUT$ is to verify if the clustering method is biased and by how much. This is so because $CluEUT$ is supposed to yields the same results as STD, i.e. to behave rational.

8.4 Common Parameters Across The Experiments

The parameters used for all experiments are presented in Tab. 8.1, where the reasoning engines correspond to the ones presented in Eq. 8.3.

Table 8.1: Simulation parameters

Parameter	Value
Short-term memory capacity	7
Short-term memory maximum age	7
Long-term memory list capacity	∞
Reasoning engines	Eq. 8.3: $\{STD, CluEUT, CluPT\}$
$\mathcal{Q}$-Learning α	Eq. 5.5: $\alpha_{init} = 1$ and $\alpha_{final} = 0.1$
$\mathcal{Q}$-Learning exploration	Eq. 5.5: $\alpha_{init} = 0.2$ and $\alpha_{final} = 0.1$
$\mathcal{Q}$-Learning γ	0.9
$\mathcal{Q}$-Learning $V_0(\bullet)$	0.0
$\mathcal{Q}$-Learning $Q_0(\bullet)$	0.0
Cost function	Eq. 6.3: $bpr(\bullet)$
Reward function	Eq. 6.2: $-bpr(\bullet)$

In all experiments, before any data were collected, one last simulation step was run, where no learning or exploring occur. This means that the agents, on this one last iteration, are only exploiting their knowledge. This strategy is used to eliminate any stochastic "artefact" that does not come from the agent's accumulated knowledge, i.e. that does not come from the individual $V_n(s)$ (Eq. 8.5).

8.5 Calibration

An equilibrium can be defined as the existence of an attractor [Mil85] and the probability of reaching this attractor is 1. This assumes the existence of a stable state space towards which the function being calibrated inexorably moves. The problem using MASim is that this equilibrium is usually not a specific point in the search space, it is likelier to have a multi-point or a steady state equilibrium. This is an issue for most of the standard calibration methods available because they expect a single point for each parameter configuration set. To make it more clear, standard calibration methods expect the fitness function to be an actual mathematical function, i.e. only one point in the codomain for each parameter set. This is not true for MASim, where a parameter set yields potentially multiple points in the codomain, one for each simulation.

The problem with the equilibrium has no easy solution for MASim. One alternative, which may not be acceptable, is to increase the parameter's step to the point where all steady states

are disjointed. This makes the multi-point equilibrium behaves as a single point because of the "disjointedness" of it. In this case any gradient oriented procedure will not get "confused" about which configuration set is better than the other.

The previous method has two main disadvantages. The first is that it may miss the optimal solution because it allows only disjointed configuration sets. Second, it assumes that the step size for "disjointedness" is known beforehand, which can be hard to estimate.

Another way to bypass the hysteresis issue (multiple equilibrium points) is to aggregate several simulation runs for the same configuration set and let this aggregated point be the result. This, however, makes the calibration process even more cumbersome, bringing two other problems: how to aggregate and how much simulations are enough. Unfortunately no general valid answer can be used. In some cases, for instance, the mean value suits the problem and in others the median is more appropriated. For the amount of simulations the answer is easy but not simple: to make as much repetitions as necessary so that the aggregation points are disjoint.

The calibration of MASim is the subject in [FKP04, FKP06] where a top-down approach with feedback is adopted. The idea is to break the fitness function into multiple sub-functions (each tackling a facet or region of the search space) and then calibrating the agents in groups according to these sub-goals. As the sub-goals are optimised/calibrated so is the macro level (the final calibration). This method greatly reduces the search space but not all scenarios can be optimised using this procedure, as in traffic assignment.

One way to break the calibration task into sub-goals is to approach the calibration problem by calibrating each OD pair as a sub-task. The problem with it is that the resulting calibration has a multitude of parameters, potentially one set for each OD pair, which greatly increases the scenario analysis complexity. This approach is however an improvement over using the traditional methods in the way described before, which requires several simulations or making the parameters discrete enough.

8.6 Difficulties In Comparing With Microeconomics

Besides the theoretical fundamentals for the validity of this discrete choice modelling framework one could speculate about its experimental validity. An obvious method to support the validity with evidences would be to use this framework (with the rational reasoning engine) and compare its results for the same problem modelled by a microeconomic model (Sec. 3.4). The appropriated model to use is a Multi Nominal Logit (MNL) model (Sec. 3.4), which does not model the correlation among the options. Then, assuming similar results, it can be said that the frameworks are equivalent.

The two main issues of this methodology are: first, the option generation and, second, the calibration methods. The latter is related to the calibration problems when using Multi-Agent Simulations (MASim) (Sec. 8.5) but it can be bypassed with the appropriated adjustments. Yet both models (microeconomics and MASim) must be calibrated under the same techniques and configurations, i.e. same fitness function and parameters' variation steps. This is more or less a matter of attention to the details. The first issue is rather more complicated.

In one sentence, the option generation for both models must be equivalent. This means that they must use the same algorithm under the same metrics. This imposes a practical difficulty that is on transferring the used infrastructure (the super-network, Sec. 6.5 and A.2) and the manipulation algorithms (the route generator, Sec. 6.5 and A.5) to another implemented framework (such as the BIOGEME[2] [Bie03, Bie08] or R[3] [CNZ99]). Therefore this comparison is let out of the scope of this work, since microeconomics are models for rational choice as well as the standard $\mathcal{Q}$-Learning [Wat89, WD92], which is equivalent to the MNL.

[2] `http://biogeme.epfl.ch/`
[3] `http://www.r-project.org/`

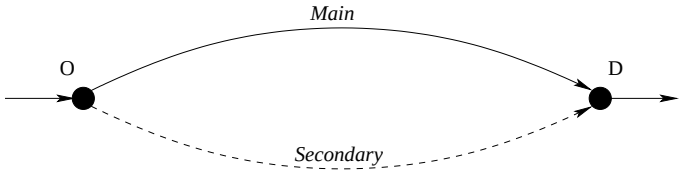


Figure 8.1: El Farol traffic scenario

8.7 Results Analysis

The concern is on the investigation about the applicability of a non-rational model for traffic assignment. This means that the user equilibrium state from both rational and non-rational models are compared.

The objective is to compare the user equilibrium, or Wardrop's Equilibrium [War52], from both models (rational and non-rational). If both have the same equilibrium point this means that both are most likely to be equivalent and therefore the use of non-rationality through the PT does not give an alternative to the EUT and so no need for this framework (since it is more complex than an MNL model). But, on the other hand, if the equilibrium points differ it means that the PT is indeed an option worth of investigation. To answer these questions several experiments (under different configurations) were made to identify the conditions necessary for the PT to deviate from EUT, i.e. when the PT is relevant (Sec. 8.8). To verify the algorithms' practical applicability, the Selten scenario (Sec. 8.9) using the data from [SSC+05, Chm05] was simulated. Finally, the scalability of the framework is analysed in Sec. 8.10.

Notice that for the El Farol Bar inspired scenario (Sec.8.8) and for the Burgdorf scenario (Sec. 8.10) the travel-time is calculated by $bpr(\bullet)$ function (Sec. 6.4.1). But in the Selten scenario (Sec. 8.9) a different travel-time function is used, which is presented and explained in the corresponding section.

8.8 El Farol Bar Inspired Scenario

The El Farol Bar [Art94] problem is a minority game, i.e. several agents must choose among two options, but the more agents choose the same choice the worse they will perform doing so. In the original problem, two choices are given to the agents: go to the bar (the El Farol bar) or stay at home. The bar had a limited capacity, so if the amount of agents in the bar is higher than the ideal capacity, then the agents that decided to stay at home win. If, on the other hand, the capacity was not exceeded then the agents at the bar win.

To transfer for the traffic is rather simple and already made in [BBA+00], there two routes were given and one route had a higher capacity than the other. Then, if the agent is in a route whose capacity has been exceeded it experiences congestion and is penalised with a higher travel-time than the agents on the other route, whose capacity has not been exceeded. This scenario is a binary choice where agents must achieve equilibrium and the rational choice says that the agents tend to reach the perfect split, where both routes have the same travel-time. The hypothesis in this experiment is that the rational behaviour (EUT) reaches this equilibrium but the non-rational does not.

A schematic view of the scenario is in Fig. 8.1, where the main route is identified by *Main* and the secondary by *Secondary*. The origin is identified by O and the destination by D.

8.8.1 Clustering Bias

It will be evident in the results that the clustering made by function `Cluster`$(\bullet)$ (Algo.5.1) is biased. The bias is towards the *Main* route, meaning that it tends to penalise the *Secondary* route. The reason for this is in its "memory" of bad results. The *Secondary* route is two times

more sensible than the *Main* route because the first has half the capacity of the second. Then if x agents exceed the perfect split in *Secondary* they are more severely penalise than if they where in the *Main* route. This discrepancy associated with the clustering "memory" tend to "poison" the lottery/prospect associated with the *Secondary* route.

However, the bias does not vary too much and is fairly homogeneous in the experiments. Therefore, in the results an extra-column was include (μ^b_{CluPT}) that represent the deviation of the $CluPT$, excluding the bias observed in $CluEUT$. This is calculated by Eq. 8.6, where from the values in $CluPT$ the bias observed in $CluEUT$ is removed (μ^b_{CluEUT}). The bias in the $CluEUT$ (μ^b_{CluEUT}) is the difference between the expected value (μ_{EUT}) and the actual obtained value (μ_{CluEUT}). To overcome the bias it is necessary, as explicitly shown in the next section, several simulation steps, 1000 or more.

$$\mu^b_{CluPT} = \mu_{CluPT} - \mu^b_{CluEUT} \tag{8.6}$$

$$\mu^b_{CluEUT} = \mu_{CluEUT} - \mu_{EUT} \tag{8.7}$$

The same effect does not happen in the STD because the learning factor α_n guarantees that "bad" scores are attenuated as the agent get experienced with the routes and these "bad" scores are rather seldom. This, however, is not the case with the clustering function. There the scores receive no special treatment if they are old or new, they are all "remembered" the same. No simple solution was found for this issue and therefore left as future work.

To make the bias evident, in all results the error is given in field "*err*" by each reasoning engine. This error is calculated by: $err = \mu_\bullet - \mu_{EUT}$.

8.8.2 Scenario Experiments

This simple scenario was used to investigate the influence of the different parameters. The parameters were: simulation horizon, amount of agents, and target density at equilibrium. For all experiments the theoretical expected value is given, identified by μ_{EUT}.

The horizon experiments were designed to verify which is the minimal simulation horizon necessary to reach behavioural stability, i.e. the minimum horizon where this variable (the horizon) does not bias the agent's behaviour. It was also verified if the agents still remain in the same behaviour when this horizon is stretched beyond the minimum.

For this scenario, when the perfect split (2/3 for the *Main* and the remaining 1/3 for the *Secondary* route) is reached all agents experience the same travel-time regardless the chosen route. It also follows that both routes have the same density.[4]

The next experiment varied the agent amount and was designed to evaluate if the amount of agents has any influence on the agents' behaviour, i.e. if the deviations from rationality are attenuated or accentuated by increasing the amount of agents.

The so called target density at equilibrium experiments were made modifying the routes length according to the amount of agents to yield the "target density", when the perfect split is reached. These experiments are helpful to evaluate the agent's behaviour under free-flow, regime, and congestion conditions, i.e. how they respond (according to the reasoning engine used) to the different stages that the traffic conditions can assume. It can also show when the PT based $\mathcal{Q}$-Learning deviates from the rational behaviour.

All simulations were repeated 100 times and the results aggregated into the mean occupation ($\mu_\bullet$) and the standard deviation ($\sigma_\bullet$). When a parameter is varied all others are kept fixed. For each experiment set the complete configuration is presented. The parameter that are not explicitly given are the same as in Tab. 8.1 from Sec. 8.4, such as the travel-time cost function, which is the $bpr(\bullet)$ (Eq. 6.3 in Sec. 6.4.1). In all experiments both routes have the same length but the *Main* route has two lanes and the *Secondary* only one. Another point is why, excepting for Sec. 8.8.2, the

[4] Density means that each route can accommodate a limited amount of vehicles and therefore the density says how much of this amount is occupied. If a route have space for, say, a maximum of 100 vehicles and 49 are using this route in a given moment, then the corresponding density is $49/100 = 0.49$.

target density is 0.3. This value was chosen because it is located right after the congestion region of the travel-time function $bpr(\bullet)$ (Sec. 6.4.1). This can be appreciated by looking at Fig. 6.2 where 0.3 is right after the curve's "knee" and this value seems to highlight the PT behaviour.

For the clustering method the value was adjusted according to Eq. 8.8, where sp is the shortest path and $length(sp)$ returns the length, in meters, of the sp. The maximum speed is $v_{max} = 120km/h \sim 33.3m/s$. Because the route length depends on the amount of agents, given that target density is fixed, so does the clustering parameter. For 100 agents it corresponds to $33.3s$ for a target density of 0.3.

$$\epsilon = 2\frac{length(sp)}{v_{max}} \tag{8.8}$$

Horizon

The horizon experiments are designed to verify if the route choices are influenced by the amount of experience accumulated by the agent, i.e. if the behaviour is experience dependent. It also presents an opportunity to verify if the clustering method for building prospect can distort the decisions, comparing the standard $\mathcal{Q}$-Learning with its cluster based equivalent. The fixed parameters are presented in Tab. 8.2 and the experiment variables in Tab. 8.3. The results for this experiment are shown in Tab. 8.4, where $\mu_\bullet$ refers to the mean occupation and $\sigma_\bullet$ to the standard deviation.

Table 8.2: Fixed parameters for horizon experiments

Parameter	Value
Agent amount	100 agents
Target density	0.30
Route length	$555.6m$
Route capacity *Main*/*Secondary*	222/111 vehicles
Scenario capacity	333 agents
Equilibrium travel-time	$19.67s$
Equilibrium mean speed	$28.25m/s(101.7km/h)$
Clustering ϵ	$33.33s$

Table 8.3: Simulation parameters for horizon experiments

Parameter	Values
Simulation horizon	$\{10, 50, 100, 500, 1000, 1500, 2000\}$
Reasoning engine	$\{STD, CluEUT, CluPT\}$

From the results in Tab. 8.4 it can be seen that a horizon higher than 50 is not necessary for the STD, which is the reasoning gold standard. But that is not true for STD and $CluEUT$, meaning that the clustering method is biased. This bias is however similar across the experiments, as argued before. Another interesting feature is that the standard deviation does not necessarily decreases with the increase of the horizon, showing that multiple simulations and aggregation is necessary to extract the real behaviour.[5] The results also show that $CluPT$ ($\mu_{CluPT}(\sigma_{CluPT})$) consistently deviates – for horizons higher than 50 – from its rational counterparts (STD and $CluEUT$). The standard deviation is fairly similar between $CluEUT$ and $CluPT$ (for horizons higher than 50).

The results for a horizon lower than 50 is somehow divergent, to say the least. An explanation could not be found to justify them, except that with less than 50 iterations is too low for the learning algorithm to adapt itself to the scenario, which is a known issue for learning algorithm, they need the necessary experience to start profiting from the environment.

[5] Note that the $\sigma_\bullet$ does not come from the exploration rate (Sec. 5.4.1).

Table 8.4: Occupation results of the *Main* route for horizon experiments with 100 agents

Horizon	$\mu_{STD}(\sigma_{STD}) : err$	$\mu_{CluEUT}(\sigma_{CluEUT}) : err$	$\mu_{CluPT}(\sigma_{CluPT}) : err$	μ^b_{CluPT}
10	$51.04(6.92) : -15.63$	$51.35(6.98) : -15.32$	$50.98(7.15) : -15.67$	66.30
50	$65.71(16.16) : -0.96$	$73.30(8.20) : +6.63$	$78.0(8.65) : +11.33$	71.37
100	$66.82(8.44) : +0.15$	$73.93(7.91) : +7.27$	$83.24(9.22) : +16.57$	75.98
200	$66.80(7.07) : +0.13$	$76.06(8.08) : +9.39$	$83.85(9.01) : +17.18$	74.46
300	$66.39(7.01) : -0.28$	$75.76(8.04) : +9.09$	$85.34(9.11) : +18.67$	76.25
400	$66.13(6.87) : -0.54$	$72.70(7.69) : +6.03$	$84.77(9.10) : +18.10$	78.74
500	$65.94(7.01) : -0.73$	$71.41(7.52) : +4.74$	$83.22(9.04) : +16.55$	78.48
1000	$66.12(6.99) : -0.55$	$67.69(7.23) : +1.02$	$75.44(8.19) : +8.77$	74.41
1500	$65.65(7.10) : -1.01$	$67.06(7.13) : +0.39$	$74.36(8.08) : +7.69$	73.97
2000	$66.24(7.12) : -0.41$	$66.86(6.97) : +0.19$	$74.51(8.30) : +7.84$	74.32
μ_{EUT}	$66.\overline{6}$			

However, the critical amount of simulation steps is 1000 because under this horizon the bias of the clustering method can be considered marginal. This means that at leas 1000 steps for the this two route scenario is necessary for investigation the agent behaviour, because under this limit the clustering bias is too much an influence in the results.

Agent Amount

The objective in varying the amount of agents is to verify if it plays a role and if the behaviour is consistent across different agent populations. The fixed parameters, valid for all simulations, are in Tab. 8.5 with its derived properties in Tab. 8.7. The variable parameters are shown in Tab. 8.6. Because the amount of agents implies in different route properties they are presented in Tab. 8.7. In this table is important to observe that the clustering parameter ϵ varies and this is due to Eq. 8.8, since the target density is kept constant at 0.3 but the agent amount varies (as more agents as longer the length and as higher the ϵ as well).

Table 8.5: Fixed parameters for agent population experiments

Parameter	Value
Simulation horizon	1000 steps
Target density	0.3

Table 8.6: Simulation parameters for agent population experiments

Parameter	Values
Agent amount	$\{10, 50, 100, 500\}$
Reasoning engine	$\{STD, CluEUT, CluPT\}$

The results for the different agent populations are depicted in Tab. 8.8 and are consistent with the expected results: rational behaviour near the μ_{EUT} and the non-rational PT based agents having a consistent deviation from it (μ^b_{CluPT}). The reasons for the higher occupation in the *Main* and not in the *Secondary* route are explained by the next experiment.

Target Density

The fixed parameters are depicted in Tab. 8.9 and the variable parameters in Tab. 8.10. For convenience, the derived properties for the different densities are shown in Tab. 8.11. The density experiments (Tab. 8.12) show that it does not worth investigating scenarios where the density is

Table 8.7: Derived parameters from agent amount experiments with target density 0.3

Agents	Clustering ϵ	Length	Equilibrium		Capacity in *veh*		
			Travel-time	Mean speed	*Main*	*Sec.*	Total
10	$3.33s$	$55.6m$	$1.97s$	$28.25m/s(102km/h)$	22	11	33
50	$16.67s$	$277.8m$	$9.83s$	$28.25m/s(102km/h)$	111	55	166
100	$33.33s$	$555.6m$	$19.67s$	$28.25m/s(102km/h)$	222	111	333
500	$166.67s$	$2777.8m$	$98.33s$	$28.25m/s(102km/h)$	1111	555	1666

Table 8.8: Occupation results of the *Main* route for agent amount experiments and target density of 0.3

Agents	$\mu_{STD}(\sigma_{STD}) : err$	$\mu_{CluEUT}(\sigma_{CluEUT}) : err$	$\mu_{CluPT}(\sigma_{CluPT}) : err$	μ^b_{CluPT}	μ_{EUT}
10	$6.65(1.03) : -0.01$	$6.92(1.01) : +0.25$	$8.25(1.17) : +1.59$	8.00	$6.\overline{6}$
50	$33.33(3.68) : +0.00$	$34.06(3.75) : +0.73$	$38.35(4.44) : +5.02$	37.62	$33.\overline{3}$
100	$66.12(6.99) : -0.53$	$67.25(7.13) : +0.59$	$76.93(8.32) : +10.26$	76.35	$66.\overline{6}$
500	$330.96(34.28) : -2.37$	$336.79(34.01) : +3.45$	$379.16(38.88) : +45.83$	375.70	$333.\overline{3}$

not above the limit of the free-flow region ($d \geq 0.2$). The reason, first, is that in the free-flow region not enough variability is produced and the experienced travel-times are too similar.

Table 8.9: Fixed parameters for target density experiments

Parameter	Value
Agent amount	100 agents
Simulation horizon	1000 steps

It is again important to explicitly say that the clustering parameter ϵ changes from target density to target density (Tab. 8.11). This is due to the Eq. 8.8 that depends on the route length that depends on the target density and agent amount. Since here the agent amount is kept constant at 100 but the density varies it also does the length and consequently the clustering parameter.

Another limit is to avoid high densities such as 0.9 that leaves almost no room for variations, i.e. the routes are so saturated that the experienced travel-time tends to repeat itself and that is why when the density increases the deviation (in μ^b_{CluPT}) decreases (Tab. 8.12). Because the routes are already saturated (in the congestion region) little to no difference is made in the experienced travel-time. So the best density value to appreciate the differences between the behaviours is in the region between 0.2 and 0.6 (Tab. 8.12).

A last point that needs an explanation is why the overloaded route (occupation above the equilibrium) is the *Main* and not the *Secondary* route. The explanation is in the $\pi(\bullet)$ function curvature (Fig. 4.1b), which is an inverted "S". The point is that the *Secondary* route is much more sensible to any variation in its occupation than the *Main*. This means that it is prone to have extreme travel-times (congestion region) with low frequencies, which translates into low probabilities. Because of that, the agent (through the PT evaluation method) is penalising the *Secondary* while tolerating frequent "bad" outcomes in the *Main* route.

8.9 The Selten Scenario

The Selten Scenario is a reproduction of the scenario proposed in [SSC+05, Chm05].[6] There, a traffic scenario with two routes was designed, similar to the one presented in previous section. For these scenario 18 individuals were asked to decide between two routes and they were monetarily

[6] This scenario is also used in [KB04] but there the impact of giving the agents information about the traffic is evaluated and therefore not quite the application here.

Table 8.10: Simulation parameters for target density experiments

Parameter	Values
Target density	$\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$
Reasoning engine	$\{STD, CluEUT, CluPT\}$

Table 8.11: Derived parameters from target density experiments and 100 agents

Dens.	Clustering ϵ	Length	Equilibrium		Capacity in *veh*		
			Travel-time	Mean speed	*Main*	*Sec.*	Total
0.1	$100.10s$	$1666.7m$	$50.050s$	$33.3m/s(120km/h)$	667	333	1000
0.2	$50.05s$	$833.3m$	$25.03s$	$33.3m/s(120km/h)$	333	167	500
0.3	$33.37s$	$555.6m$	$16.68s$	$33.3m/s(120km/h)$	222	111	333
0.4	$25.03s$	$416.7m$	$16.68s$	$33.3m/s(120km/h)$	167	83	250
0.5	$20.02s$	$333.3m$	$12.51s$	$33.3m/s(120km/h)$	133	67	200
0.6	$16.68s$	$277.8m$	$10.01s$	$33.3m/s(120km/h)$	111	56	167
0.7	$14.30s$	$238.1m$	$8.34s$	$27.0m/s(97km/h)$	95	48	143
0.8	$12.51s$	$208.3m$	$8.83s$	$16.0m/s(57km/h)$	83	42	125
0.9	$11.12s$	$185.2m$	$26.28s$	$7.0m/s(25km/h)$	74	37	111

rewarded for choosing the fastest route (shortest travel-time). The 18 individuals were competing with each other and 100 rounds of decisions were performed.

The scenario is the same as in Fig. 8.1 but the Eq. 8.9 and 8.10 were used for calculating the travel-time (in minutes) for the *Main* and *Secondary* route respectively. There t_m is the travel-time in minutes for the *Main* route, n_m is the amount of individuals that chose the *Main* route, t_s is the travel-time in minutes for the *Secondary* route, and n_s is the amount of individuals in the *Secondary* route.

$$t_m = 6 + 2n_m \tag{8.9}$$

$$t_s = 12 + 3n_s \tag{8.10}$$

The travel-time functions are depicted in Fig. 8.2 and Fig. 8.3, where Fig. 8.2 shows them in a regular plot and in Fig. 8.3 modifying t_m to $t_m = 6 + 2(18 - n_s)$. The rational equilibrium for this scenario is when 6 individuals choose the *Secondary* and 12 the *Main* route, which yields a 30 minutes travel-time for both routes. This is the same ratio used in the previous scenario (2/3 and 1/3).

For this scenario four experiments were made but from them only two are of interest here. These two are the following: first the individuals were only informed to make their decisions and to try to choose the fastest route. In the second experiment they were told that the *Main* route has higher capacity than the *Secondary*. The data collected from both experiments are reproduced in Tab. 8.13, which was extracted from [SSC+05, Chm05], in Tab. 18 from page 68 from the rows identified by "Variation I" and "Variation II" respectively.[7] In Tab. 8.13 μ_s and σ_s correspond to the mean amount of individuals in the *Secondary* route and the standard deviation. The column "Deviation" corresponds to the deviation from the expected rational behaviour. There the line "Not-informed" corresponds to the first experiment, where the individuals were not informed about the differences in the routes' capacities. The "Informed" line corresponds the second experiment, where the individuals were aware of the routes' capacities. The last line, "μ_{EUT}" corresponds to the expected rational equilibrium.

It is interesting to observe that the data is more than one standard deviation "under" the expected rational behaviour. Replicating the same experiment using the three engines being

[7] The thesis of Chmura is available at `http://www.ub.uni-duisburg.de/ETD-db/theses/available/duett-05152005-222337/`.

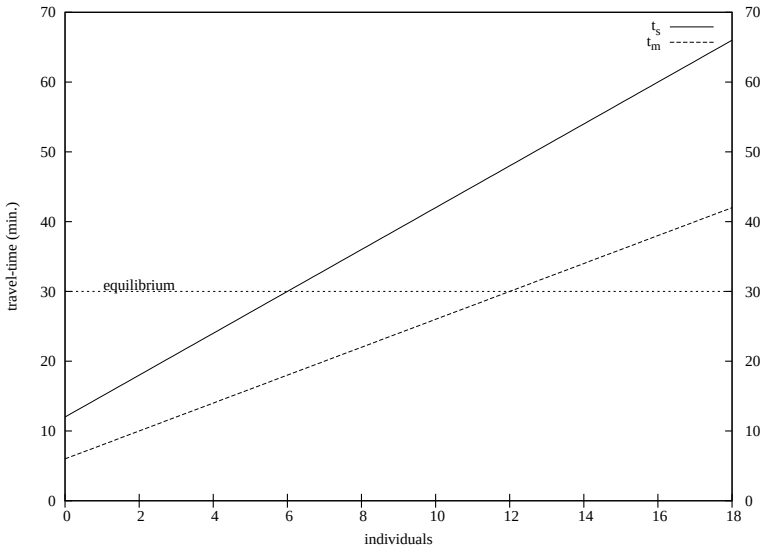


Figure 8.2: Travel-time functions

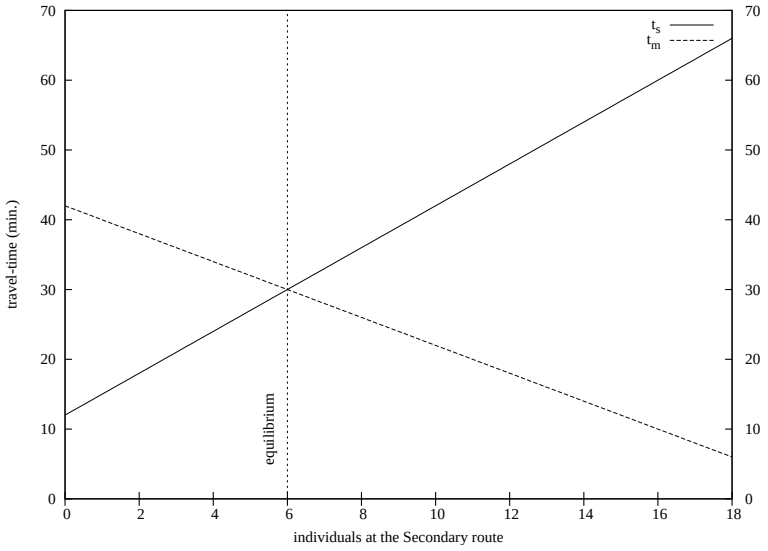


Figure 8.3: Travel-time functions by individuals in the *Secondary* route

Table 8.12: Occupation results of the *Main* route for density experiments and 100 agents

Density	$\mu_{STD}(\sigma_{STD}) : err$	$\mu_{CluEUT}(\sigma_{CluEUT}) : err$	$\mu_{CluPT}(\sigma_{CluPT}) : err$	μ^{b}_{CluPT}
0.1	66.45(7.09) : −0.21	67.77(7.13) : +1.10	69.19(7.24) : +2.53	68.09
0.2	66.15(7.04) : −0.50	67.35(7.13) : +0.68	75.57(8.23) : +8.90	74.89
0.3	66.12(6.99) : −0.53	67.25(7.13) : +0.59	76.93(8.32) : +10.26	76.35
0.4	66.43(7.05) : −0.23	67.33(7.11) : +0.66	74.05(8.28) : +7.39	73.39
0.5	66.57(7.12) : −0.09	67.41(7.15) : +0.74	72.99(7.93) : +6.32	72.25
0.6	66.25(7.11) : −0.40	67.89(7.08) : +1.22	73.48(7.83) : +6.81	72.25
0.7	66.18(7.12) : −0.47	67.55(7.10) : +0.88	73.88(8.13) : +7.21	73.00
0.8	65.74(7.13) : −0.92	67.66(7.12) : +0.99	74.24(7.94) : +7.58	73.25
0.9	65.85(6.96) : −0.81	67.77(7.08) : +1.10	73.20(7.97) : +6.54	72.10
μ_{EUT}	$66.\overline{6}$			

Table 8.13: Table 18 from [Chm05] for the secondary route and 18 persons

Type	μ_s	σ_s	Deviation
Not-informed	4, 50	1.38	−1.50
Informed	4, 44	1.01	−1.56
μ_{EUT}	6.00		

tested here the results, in Tab. 8.14, are quite compelling. The simulate values were obtained by fixing the horizon at 1000, repeating the experiment 30 times, and setting the clustering threshold to 5 minutes. In Tab. 8.14 the column "Error" was added to show how much the mean (μ_s) of the simulated values deviate from the data.

Table 8.14: Simulation results for the secondary route and 18 agents

Engine	μ_s	σ_s	Error	
			Non-Informed (4.50)	Informed (4.44)
STD	5.61	1.47	1.11	1.17
CluEUT	5.16	1.30	0.66	0.72
CluPT	4.10	1.20	−0.40	−0.34
μ_{EUT}	6.00			

The simulated results show that the *CluPT* is closer to the real data but only by a small margin when looking only at the error magnitude in the column "Error" (Tab. 8.14). However, when looking at where the *CluPT* got wrong then it is the clear "winner". The data shows that the individuals tend to sub-utilise the *Secondary* route, i.e. avoid it even when it means to get a worse travel-time. This is exactly what the *CluPT* agents do, they avoid the *Secondary* route and it can be said that the *CluPT* agents "exaggerate" the behaviour observed in the data. That is not how *STD* and *CluEUT* behave. They tend to approach the rational user equilibrium (6.00). In this case, the *CluPT* is indeed the best approach for modelling this scenario.

Even though the results supports the claims in this thesis the experiments performed in [SSC+05, Chm05] have some issues. The first issue regards the travel-time functions, they are very unlikely to have any practical validity. Second, the participants are rewarded directly with money which is not the case in traffic, where the monetary utility is a few steps away from the transportation experience. This is shown to be a very important issue regarding the human attitude towards accomplishing tasks [MA06, Ari08]. The third criticism is that the 100 decisions were made on the same day, where in reality it is done once or twice a day. However, the data and the simulations are not being disputed here, just its validity as a reasonable traffic scenario. The point being made is that this scenario shows that the PT based non-rational behaviour suits better to reproduce the behaviour observed in the data but cannot be said to correspond to a real situation in traffic.

A note on the apparent lack of bias in the clustered reasoning engines, *CluEUT* and *CluPT*. It has not vanished but has been attenuated by the linear nature of the travel-time functions (Eq. 8.9 and 8.10).

8.10 The Burgdorf Scenario

The objective in simulating the city of Burgdorf (Switzerland) was to verify if the framework scale to a real-world scenario. The data was kindly made available by Mr. Guido Rindsfüser from Emch+Berger Holding AG.[8] The city network is in Fig. 8.4 and some figures about it in Tab. 8.15. The figures presented in Tab. 8.15 need some explanation. The values annotated with "(Network)" are the figures of the network, resulting from converting the input graph into the super-network structure (Sec. 6.5 and A.2). Where "(Target)" is given, it corresponds only to the private vehicles modus, from which the data is available, and "(Data)" is the effective amount of data available. This means that from the total 522 edges/links (corresponding to the private modus) only 122 are available to evaluation, which corresponds to roughly 23% of the target links.

Being more detailed. The vertices, edges, and maximum occupancy values with the "(Network)" label refer to values extracted from the input data after transforming it into a super-network (Sec. 6.5 and A.2). This means that if a link allows private vehicles, public transportation, and pedestrian traffic it is represented three times (one for each modus). The vertex and edge amounts are extracted from the input graph, provided with the data, and the maximum occupancy is a derived measure. This was calculated assuming that a vehicle has a mean size of $5m$ and then summing up all link lengths (multiplied by the corresponding lane amount) and dividing by the assumed car length.

The values labelled with "(Target)" refer to the elements, in the input data, related to the regular vehicular traffic. This means that some links do not allow private vehicles, such as pedestrian only zones or special public transportation lines. All links that does not allow private vehicles to travel through them are excluded from the figures corresponding to "(Target)".

The last label is "(Data)" and it corresponds to the elements in the "(Target)" that have an associated occupancy data. This means that from the 522 links in the "(Target)" only 122 were annotated with occupancy data, for the other 400 links no data was collected. The "Occupancy Sum(Data)" was calculated by summing up all occupancy data available and "Mean Occupancy(Data)" was calculated by dividing up the "Occupancy Sum(Data)" by the 122 links.

Table 8.15: Burgdorf scenario figures

Property	Amount
OD pairs	1339
Agents	7357
Vertices(Network)	756
Vertices(Target)	211
Edges(Network)	2570
Edges(Target)	522
Edges(Data)	122
Maximum Occupancy(Network)	5687.93
Maximum Occupancy(Target)	1949.47
Occupancy Sum(Data)	462.03
Mean Occupancy(Data)	3.78

Moreover, the load data shows that only 3 of the 122 links[9] (line "Edges(Data)" in Tab. 8.15) have a density higher than 0.3 and the conditions to observe any deviation are to have high

[8] `http://www.emchberger.ch/`

[9] To calculate this the density of all links that have occupancy data was calculated: $density = (lanes \times length)/size_{car}$), where $size_{car} = 5m$.

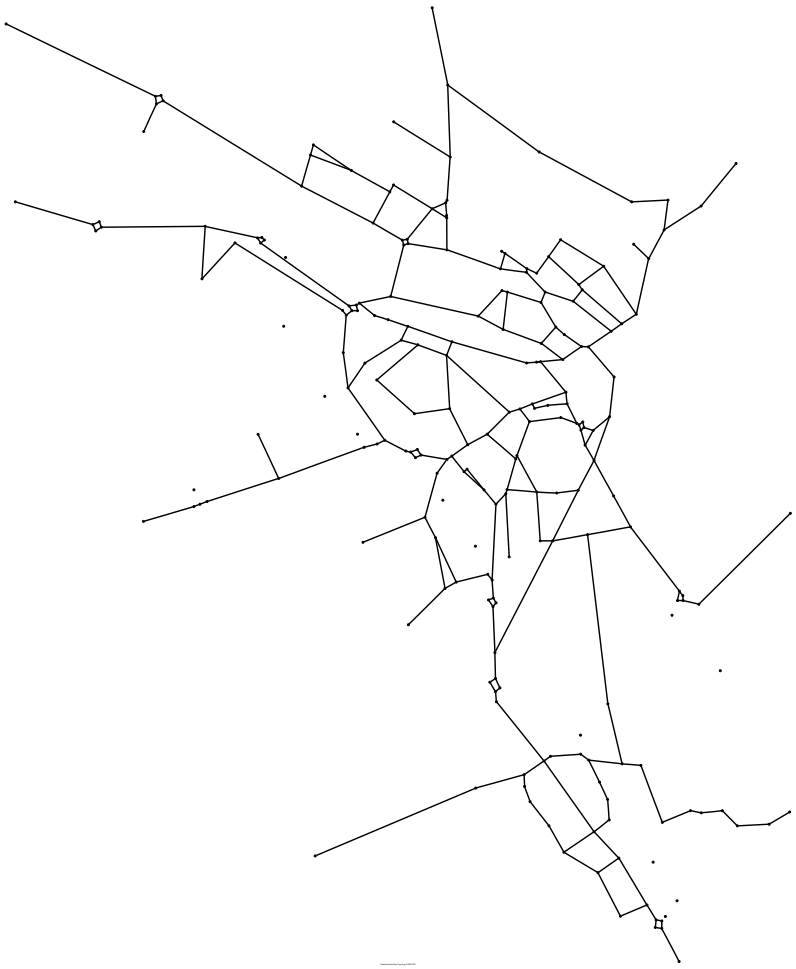


Figure 8.4: Burgdorf scenario

variability in the travel-time, as demonstrated by the experiments in Sec. 8.8. This means that situations near complete congestion (very high densities) and free-flow (very low densities) do not present any of the conditions where the PT based reasoning deviates from the rational behaviour. Even if the travel-time function is changed, so that the maximum flow occurs at the density of 0.2 for example, does only make a difference at those 3 links, which represent about 2.5% of data (from the 121 links) and about 0.6% of the total amount (from the 522 links). This means that the difference in the final evaluation will be barely noticed, if it can be at all. It is important to stress that this not a manipulation made here but an analysis made over the given data.

Another issue is the OD matrix. It isn't quite clear what the numbers, associated with each OD pair, represent because summing up all available space in the network it yields about 5688 vehicles at the same time (line "Maximum Occupancy(Network)" in Tab. 8.15).[10] But that is the capacity of the network (assuming that it is acceptable that cars touch each others, bump to bump). If the actual counts are summed up, then it is reduced to 462 vehicles or to $462/122 \times 522 \sim 1978$, extrapolating for the other links. This means that the 7357 agents being simulated cannot possibly represent a particular moment in time, such as a rush-hour. A strategy would be to multiply, at each OD pair, the amount of vehicles/agents by a factor, such as $1978/7357 \sim 0.27$, but this would make several OD pairs disappear (some OD pairs have only 1 vehicle/agent assigned). To avoid any manipulation, that could compromise the validity of the experiments, the data was used as it was provided and for the travel-time calculations the BPR formula (Sec. 6.4.1) $bpr(\bullet)$ was used (Eq. 6.3). As expected, all links are over-saturated (densities higher than 1.0) and therefore showing no noticeable difference among the reasoning engines. The reason is simple, at over-saturation the variability is low, i.e. every agent tend to experience the same travel-time over and over again. This means that little to none stochastic variation is present in the prospects/clusters, leading to a decision that is equivalent to the rational decision. Nevertheless the results in Tab. 8.16 are presented to show that such a scenario can be simulated and that the whole framework does scale to real-world scenarios. There each simulation has an horizon of 1000 steps[11] and each simulation was repeated 20 times. The column "MSE" represents the final Mean Squared Error (MSE) values, "$\sqrt{MSE}$" the absolute error (not squared), "Expected ($\sqrt{MSE}$)" is what the data yields as expected for the previous column, "Error" is difference between the expected and the actual values (the simulated minus the expected value), and "Error factor" represents how many times the expected value have been missed (the division between the simulated and expected values).

Table 8.16: Burgdorf scenario MSE results

Engine	MSE	$\sqrt{MSE}$	$\overline{occup}$	Error factor	Time (h)	Mem. (MB)
STD	683.70	26.15	3.78	6.92	15.40	309.96
$CluEUT$	662.58	25.74	3.78	6.81	19.90	608.17
$CluPT$	694.34	26.35	3.78	6.97	17.07	498.51

Because of the several issues mentioned before the results can only be used to say that a complete city can be simulated using the framework presented here. Any other claim will be bogus. It is tempting to say that the $CluPT$ is worse than the others, but this is deceiving because the "MSE" column shows the mean squared error. When looking at the absolute mean error, in column $\sqrt{MSE}$ from Tab. 8.16, this difference drops but says that all engines are failing by a "catastrophic" margin (column "Error factor" in Tab. 8.16). The value in $CluPT$ (26.35) says that it missed, on average, by 22.56 vehicles in each of the 122 links, whose data is available. Then taking the average occupancy in the 122 links it gives about 3.78 vehicles per link (line "Mean Occupancy(Data)" in Tab. 8.15 and "$\overline{occup}$" in Tab. 8.16). This means that missing these 3.78 vehicles by 25.74, for the $CluEUT$, which is 6.81 times more vehicles (column "Error factor"

[10] The calculation done was: $total = 1/car_{lenght} \sum_{l \in L} l_{length} l_{lanes}$, where $car_{length} = 5m$, $l \in L$ is a link in the link set (each of the 522 links), l_{length} is the link length in meters, and l_{lanes} is the amount of lanes for this link.

[11] This value is not a definitive value, it must be investigated but because the objective was to verify only the scalability of the framework it was not further investigated.

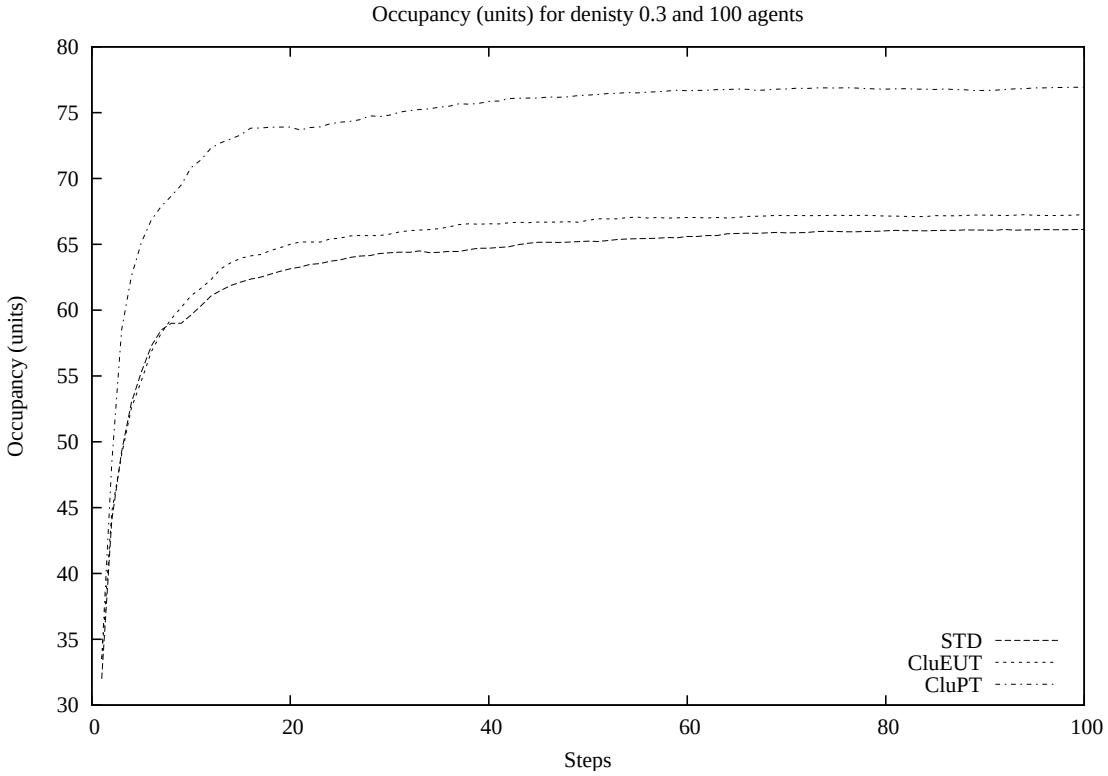


Figure 8.5: Average occupancy evolution

in Tab. 8.16), is almost as bad as missing it by 26.35 (for *CluPT*), which is 6.97 times higher.

A note on performance (the last two columns in Tab. 8.16). This the experiments were run on an Intel® Pentium® 4 3.20GHz with 2GB Ram. The discrepancy in *CluEUT* is because the simulation was run on another machine with the same processor but with a concurrent simulation (the processor was not dedicated for the simulations). The second difference is that the machine where *CluEUT* was run has an inferior I/O performance. The I/O performance is important because data was constantly be collected and log files from the simulations were produced (to later analysis for possible errors).

8.11 Conclusion

From the results presented with different parameters it can be seen how the different parameters are relevant when considering the use of the PT for agent modelling. The first conclusion is that the amount of agents does not have a relevant influence in the agent behaviour. Another point is the influence of the density, or load level. The higher the load the more relevant the consideration of the PT is, excluding the extreme situations where it reaches over-saturation. Another relevant aspect is the shape of function $\pi(\bullet)$ that determines, in this particular case, which route will be stressed.[12]

Some practical aspects are also relevant and they are the simulation horizon and the repetition amount. The horizon must be at least as high as the minimum required, in this case under 1000 steps, to guarantee that the implementation artefacts do not greatly influence the results, for the two route scenario. It remains to be validated for more complex scenarios. As the results show the most severe artefact is the bias in the clustering method. For the repetition amount, which were

[12] For a visual impression of function $\pi(\bullet)$ with different γ values see Fig. B.1.

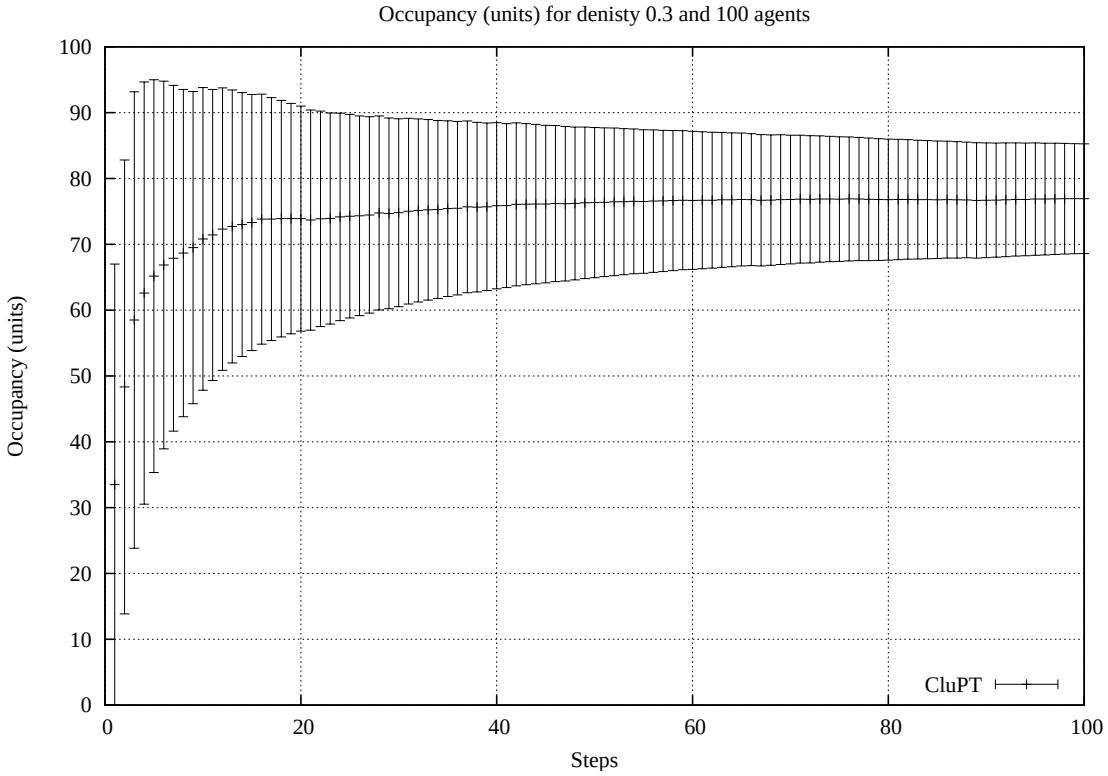


Figure 8.6: Mean occupancy and standard deviation for *CluPT* engine

fixed in 100, it is less critical. To visually observe the influence of the repetition amount please address to the Fig. 8.5 and 8.6. Where the average occupancy for the *Main* route using different reasoning engines is plotted (Fig. 8.5). Then the same plot but for only the *CluPT* including its standard deviation (Fig. 8.6). For these plots the horizon is 1000, target density 0.3, and agent amount 100.

Besides analysing the conditions under which the PT based agents do deviates from the rational behaviour, it is also correct to say that they behave more closely to the real data than their rational counterparts. As seen in Sec. 8.9, the *CluPT* agents are closer to the real data than the rational based agents. The non-rational agents not only have a lower error but also show the same behaviour, avoidance of the *Second* route, which supports the claims that non-rationality is a better approach for the human decision-making modelling. A last point is that the framework does scale up to allow simulations of real-world scenarios, as shown by the last section.

Summarising what the results show the following claims are made. First the conditions necessary for the experiments were established with the different experiments in Sec. 8.8. Then a similar scenario was used to show that the non-rational behaviour (the *CluPT* engine) is indeed better at reproducing real data (Sec. 8.9). The last claim is that the framework does scale for real world scenarios and that is demonstrated in Sec. 8.10.

Chapter 9

State-Of-The-Art And Related Work

As delineated by the previous chapters this thesis has a multidisciplinary scope and it can be broken into three sub-themes. The first is the utility based discrete choice modelling (covered by chapters 2, 3, and 4), the second is agent learning and architecture (in chapters 5 and 7), and the third is the traffic assignment problem (chapter 6). In this chapter these sub-fields are again approached, presenting the advancements in each area and how they relate to this thesis.

9.1 Discrete Choice Modelling

In the previous chapters only utility based choice modelling theories were presented but other theories exist that are not based on utility functions. Consequently, first the utility based theories are reviewed and then the non-utility based alternative is presented.

9.1.1 Utility Based Modelling

The models based on the utility theory [Fis70] (UT) can be subdivided between rational and non-rational. By rationality it is meant the models that are based on the work of von Neumann and Morgenstern, as presented in chapter 3. The non-rational models, on the other hand, are the models that do not follow this definition, for instance the Prospect Theory [KT79] (PT) presented in chapter 4.

Rationality Based Utility Models

The first modelling technique under this definition is the Expected Utility Theory [vN28, vNM07] (EUT), which is the base for all others. As discussed chapter 3, this theory has some drawbacks such as requiring complete knowledge about the individual's internal decision process. To cope with such restrictions, improvements known as Random Utility Models [GP06] (RUM) were proposed. The main development of RUM is to allow partial knowledge about the decision process of the individuals being modelled, because under the RUM it is admitted that some factors cannot be modelled. But these factors are still relevant because they can disrupt the ranking established by the utility calculated based on the known factors, as mentioned in Sec. 2.5.2 and 3.4.

The models mentioned and briefly presented in Sec. 3.4 are improvements over the EUT. They are also more advanced than the model presented in this thesis because they, except for Logit [Ber44, Luc59] and Multi Nominal Logit [McF74] (MNL), assume that the options need some treatment to make them iid (independent from each other). This correlation capturing is not present in this work. Here, as in the MNL, no structure is given to extract the correlation between the options. This is a point that is left as future work because it requires a complete new

study about how agents may perceive correlation and how they could learn and extract them from the options.

The complexity and refinement in the most advanced models such as the Path-Size Logit [BAB99] or Probit [Bli34a, Bli34b] are beyond the scope of this thesis, but "better" models.[1] In a recent improvement [FB09] it is proposed that instead of using additive errors the use of multiplicative errors is evaluated. This means that instead of a utility as in $u = v + \varepsilon$ (Eq.2.5) it multiplies the error: $u = v\varepsilon$. They also report that the multiplicative model represents the data better than the additive model (in this case an MNL model). This does not mean that their improvements may not be used in the non-rational model presented here, but that they must be carefully studied and brought to light in relation to learning algorithms.

Non-Rational Utility Models

The most prominent non-rational utility based model is the Prospect Theory [KT79] (PT) and its improvements, as mentioned in chapter 4 (specially Sec. 4.7). The difference between PT and EUT is in how the utility is calculated and which assumptions are made about how u is aggregated: compare $eut(\bullet)$ (Eq. 3.3) and $pt(\bullet)$ (Eq. 4.1). Because the PT does not observe all the axioms from EUT the improvements mentioned in previous section cannot be directly incorporated into the PT, even though both theories are similar. In this regard the PT still lacks development to capture the correlation from the options to allow a broader use.[2] Among the improvements of the PT is the Cumulative Prospect Theory [TK92, WT93] (CPT), presented in Sec. 4.7.1. The CPT can be easily incorporated into the presented agent learning algorithm (chapter 5) and architecture (chapter 7) but no experiment was made using it because it is not clear if it is applicable for all modelling problems, as already discussed. The use of CPT into the $\mathcal{Q}$-Learning is rather simple: instead of using $pt(\bullet)$ in modified $\mathcal{Q}$-Learning (Sec. 5.6) the $cpt(\bullet)$ function (Eq.4.4) is used.

The PT, however, is not the only utility based non-rational theory. The Theory of Small Feedbacks [BE03] is another alternative. The problem with it is that it has some problems with large horizons decisions.[3] Because of this issue no further attempts to use this theory were made.

9.1.2 Non-Utility Based Modelling

For the non-utility based modelling an example is the Fast and Frugal Way [GG96] with its extensions [GG02, HG05]. This model is based on a hierarchical decision process, where the decision problem is as follows. The decision problems are always binary, i.e. compare two options and choose the best. For these choice problems a set of binary criteria is given, where each of them can assume the values y (for *yes*) or n (for *no*), meaning that the criterion is fulfilled (case y is assigned) or not (case n is assigned). It is assumed that the decision problem can be modelled by establishing a sequence of such binary criteria, with their corresponding y and n values. This means that in a hypothetical choice problem $\mathbf{X} = \{A, B\}$ five criteria are given, say $\langle V, W, X, Y, Z\rangle$. Each of these attributes/criteria is then applied to each option (A, and B). From this evaluation a string of y and n (for *yes* and *no*) is returned by the model such as: $s_A = \langle y, n, n, y, y\rangle$ and $s_B = \langle y, n, y, n, n\rangle$. This means that: the criterion V is met by both options (the first y on each string); then the criterion W is equally not fulfilled (both with n); and then the first difference appears, where B has the criterion X fulfilled but A has not (the third character in the sequences). In this case, according to the theory, the option B is the choice made.

The algorithm works by choosing the "winner" by comparing pairwise the two options' strings of *yes* and *no*. If both options have the same answer for the same criterion the algorithm compares the next criterion and does that until the options have different answers, winning the option with

[1] It is said "better" because these models refine the structure of the option set more carefully than it is done in this thesis.

[2] The PT requires the same assumption as in EUT, i.e. all options are independent from each other.

[3] The model has been implemented and tested but it "degenerates" when the iteration horizon is extrapolated beyond of what is used in the original article [BE03]. The authors of [BE03] were notified [Bar07] about the problem, including the evidences, but no answer has been received.

an y. The problem with this theory is that it is only possible to have binary choices (as the Logit model) and only binary criteria are allowed (no numerical comparison is considered).

9.2 Traffic Assignment

As explained in chapter 6, the traffic assignment step is the most complex in the four-step modelling (FSM) concept. Considering the context of this work it is necessary to mention the traditional approach and its state-of-the art as well as the work in the artificial intelligence (AI) field. Under what is here called the traditional approach it is meant the microeconomics approach, i.e. the use of microeconomic methods to model the fourth step in the FSM. However, these techniques are macroscopic. In the AI field the focus was on the multi-agent related work, to which this thesis is more closely related.

9.2.1 Microeconomics

The developments in the microeconomics discrete choice modelling are close related to the use of microeconomics for traffic modelling. When the MNL [McF74] model was developed, the objective was to model modus choice (the third step in the FSM) for shopping behaviour (mainly to model the split between transit and private vehicle choice). Later the Nested-Logit [BA73] model was developed. This model was a significant improvement for both econometrics and traffic modelling and is the central point of the book [BAL85], considered one of the milestones in traffic assignment modelling using microeconomics. Other subsequent models as Cross-Nested-Logit [Vov97] and Path-Size Logit [BAB99, BF05] were also developed for modelling traffic assignment.

The common points in them are that they are all macroscopic and stochastic models, i.e. the route distribution is given as a result for a given $\boldsymbol{\beta}$ vector (Sec. 2.5). Recalling the chapter 6, such models return how many individuals, in each origin destination pair (OD pair), took each of the available routes (connecting O to D). The differences are, as explained in Sec. 9.1.1, in how the correlations among the routes are explicitly represented. The advantage in using microeconomic models is that the influence of each parameter (looking at the $\boldsymbol{\beta}^*$, Sec. 2.5) is explicit and can give answers to questions like: "how relevant is the price of a bus ticket when compared with the travel-time?"[4] Those models can then be used in testing hypothetical scenarios, where the bus ticket price is increased or decreased and the impact of this change on link load can be evaluated. The main disadvantage of these models is that all of them assume the rational behaviour (EUT) as a model for human behaviour.

Some models for non-rational traffic assignment models were also proposed but for simple and synthetic scenarios, with the exception of [SK04]. There the PT was used to model the departure time choice for the commuter scenario for the Otsu city in Japan. This is not exactly traffic assignment and can be said to be more adequately classified as part of a day-activity planning strategy. In [Avi06] on the other hand, the traffic assignment problem for a synthetic binary route choice was modelled using CPT and compared with EUT. The objective was to show that the equilibria are not compatible, i.e. CPT does not have the same equilibrium point as EUT does and therefore it is worth investigation (the use of CPT for traffic assignment).

The most closely related work to this thesis is the work done by Connors and Sumalee [CS09] where CPT is used to model traffic assignment. Their work assumes a variation on the interpretation of the Wardrop [War52] equilibrium, as explained in chapter 6. The main differences between their approach and the one here is that there the travel-time calculation is similar to the one used here, as already mentioned, and a stochastic component is attached to it. This stochastic element is a normally distributed "uncertainty". Another great difference is that it is a entirely macroscopic approach and its utility function has some drawbacks: it can return values that oscillates

[4] Assuming that the price value used in the utility function is normalized with the travel-time value (both are on the same range), then comparing the β_{price} with $\beta_{travel-time}$ gives the relevance directly.

about the origin, i.e. it may mix positive and negative utilities, whose behaviour is not very clear.[5]

The decision to adopt CPT in [CS09] seems to be entirely based on the use of a continuous distribution function to describe the prospects and it appears easier to derive the equations for the CPT than for the PT. Since in [CS09] no real-world scenario is modelled the assumption of CPT model does seem to be based on evidences. This is the same situation in [Avi06], which also uses CPT, and does not justify its adoption by using evidences as well. Therefore it is fair to say that the choice between CPT and PT remains to be investigated when modelling route choice.

9.2.2 Artificial Intelligence

The use of heuristics and specially learning for traffic assignment is an active research field. In [AP05] individuals were invited to participate in a binary route choice problem. For the hypothetical scenario the individuals suppose to choose the route that perform better. The travel-time functions were specified by a Probability Distribution Function (PDF) for each route, so that the travel-time experienced at each decision "round" varied. After collecting the decision data, several models where built to check which of them better reproduces the data – the best on fitting the data of the real decision-makers. The compared family of models were EUT, RUM, CPT, Bayesian Learning, and Reinforcement Learning, being the latest the best approach. However, the experiments can be said to be biased because in some of the tested scenarios the initial assumptions were changed along the experiment (the travel-time PDFs changed during the experiment), which could be the reason why learning performed better than the other approaches.

In [BBA$^+$00] a synthetic commuter scenario[6] with two routes was designed to verify the performance of different decision-making strategies, being these strategies based on the past two experiences. The objective is always to choose the route that has the lowest amount of agents (a minority game). The commuter scenario is also the one used in [KB02, BK03] where the optimal distribution is to have 2/3 of the agents on the main route and the remaining 1/3 on the secondary route. In the experiments performed two types of reinforcement based heuristics were used: the first based on own experience and the other taking into account a forecast given by an external source. The focus, on the first experiment, was to evaluate the best learning frequency, i.e. how often should the agent re-evaluate the route scores (using the reinforcement heuristic). On the second experiment the agent's task was extended to also evaluate the quality of the forecast, i.e. how reliable it was. The findings show that, first, a rather frequent re-evaluation is better than a seldom and, second, that the forecast reliability is inversely correlated with the amount of agents receiving this information. Then, again exploring the commuter scenario, in [KBW03] a different set-up was used, where both routes had the same capacity, i.e. the agents had to choose between physically equivalent routes. The difference between [KBW03] and [BK03] is that the agents willing to use a forecast information can shop among the different types of information. For each agent four types of information where made available, from a very simple binary information about the presence or not of congestion on the routes to the very elaborated traffic density for each route. The agents must then pay for the information with different prices according to the complexity of the information. The objective was to evaluate which would be the most efficient type of information for the agents, i.e. the type of information with the best cost/benefit rate. The surprising finding was that mean speed turned out to be a misleading information, where the agents using it performed worse than the ones using no information.

In [SSC$^+$05, Chm05] a scenario similar to [BK03, KB04] were used to evaluate how good a reinforcement learning reproduces the data produced by the following set-up. Eighteen individuals participated in the experiment to choose the fastest route, given that one route is "naturally" faster than the other. The experiment had an horizon of 200 choices and different scenarios were proposed. First no information about the routes' nature was given, i.e. the individuals were not

[5] The problem is not that no mathematical treatment exist but that it depends on empirical evidences to support its application, which is the argument in [LF91, Sch03].

[6] Commuter scenario means that for a given familiar OD pair (usually from home to work and vice-versa) the individuals are experienced with the available options and must decide which route to take. The choice is normally to minimise the travel-time.

explicitly informed that one route was faster than the other. Then in a second experiment this information was given. The third was similar to the first experiment but a travel-time disturbance is introduced, simulating the situation where a construction site appears interchangeably in one of the routes. The last is similar to the third but with the additional information about which route is usually the faster. It is important to say that the individuals where monetarily rewarded for their correct decisions. The real data for the experiment without disturbance is the most interesting and closely related to this work. There the distribution is not the one of the rational equilibrium. The real data yields mean occupation, in the secondary route, of 4.50[7] (no information about the different route times) and 4.44 (full disclosure about the nature of each route) and at the rational equilibrium 6.00 is expected. Even though the results in [SSC+05, Chm05] are similar to the results found here in chapter 8 (for the two-route scenario) a critic must be made. It cannot be assumed that persons behave in traffic as in the computer simulations. The first reason is the time between the experiences: in traffic it is usually once a day and in the computer are 200 simulations a day. Second, in the computer simulations the participants are monetarily rewarded for their decisions, which is not the case in traffic.[8]

In another work [BK05] the Braess [Bra68, BNW05] paradox was investigated and how effective a control system would be in avoiding its emergence. To test the hypothesis, a Braess scenario was simulated where two types of drivers where present: informed and non-informed drivers. By informed it is meant that the informed drivers received traffic information from the control system about the network state. The results were that when drivers receive the information about the network state, the network state gets closer to the global optimum, but even closer if the received information is manipulated (inducing the drivers to take different routes). This work only marginally related to one presented here because no Braess condition is investigated here, but remains as future work.

The influence of a control system is also the subject in [Wah02], where a two-route scenario, along with other scenarios, is used for investigating the use of Advanced Traveller Information Systems (ATIS). There the impact of the use of ATIS is tested using Multi-Agent Simulations (MASim) but the findings are strictly hypothetical. The objective was to verify how can information about the traffic state influence the drivers behaviour. One remark, which is also present in [KBW03], is that when the amount of informed drivers (that use ATIS) grows the system performance degrades. The main reason is because when ATIS gives a forecast about future conditions it does not take into account the reaction of the drivers to this very given forecast. In this case what happens is that the more users receive a forecast, and decide accordingly, the lower the reliability is for the forecast. The relation of the work in [Wah02] and the one here resides in the use of the two-route scenario and the use of MASim.

Yet all these approaches are still based on the fundamentals of game-theory [vN28, vNM07] (the rational behaviour). One exception is the Small-Feedbacks [BE03] model which is not based on rationality, but showed some problems (as discussed early, in Sec. 9.1.1).

[7] This data corresponds to the mean occupation for the secondary route in [Chm05], Tab. 18 at page 68 in the rows "Variation I" and "Variation II". The thesis of Chmura is available at `http://www.ub.uni-duisburg.de/ETD-db/theses/available/duett-05152005-222337/`.

[8] In traffic the monetary reward is at least one step away (considering gasoline consumption) or several (considering vehicle depreciation, maintenance, insurance, etc). It is known that human behaviour changes when money is not directly handled (see [MA06, Ari08] for further information) and as further way the money reward/cost from the action is as less aware people are of it.

Chapter 10

Conclusion And Future Work

This thesis proposes a novel approach for modelling human decision-makers using Multi-Agent Simulations (MASim) with a non-rational behaviour. This non-rational behaviour is here based in the Prospect Theory [KT79] (PT), chapter 4, which was compared to the rational behaviour in the Expected Utility Theory [vNM07] (EUT), chapter 3. This model was used to design a modified $\mathcal{Q}$-Learning algorithm (chapter 5). The PT based $\mathcal{Q}$-Learning was then integrated into an agent architecture (chapter 7).

The designed architecture with the different reasoning engines ($\mathcal{Q}$-Learning variations) where then used to simulate traffic assignment (chapter 6). The results (chapter 8) show that, as expected, the non-rational behaviour manifests itself only under some conditions. Theoretically these conditions are the existence of variability in the outcomes received by the agent (Sec. 4.5.2), which builds an outcome probability distribution called lottery/prospect. The variability makes it possible to have outcomes located at one or both "bumps" of the function $\pi(\bullet)$ (Eq. 4.3 and Fig. 4.1b). Some experiments, organised in sets, were performed to evaluate the proposed agent reasoning and architecture. The first set (Sec. 8.8) was designed to show which are the theoretical conditions for the occurrence of divergences between rational and non-rational behaviour in a practical scenario, in this case traffic. It also aimed at showing the possible problems with the proposed agent architecture and which are simulation parameters necessary to minimise the influence of the bias imposed by the simulation framework. From this first set it was shown that the proposed agent architecture does have some limitations (requiring a minimal amount of agents and a minimum value for simulation horizon). But it also shows that once these limitations are circumvent, by a high enough agent amount and high enough simulation horizon, the simulation framework works as expected. In the specific case of traffic the main conclusion is that the PT based reasoning deviates from the rational behaviour only when the agents experience congestions. This means that in scenarios where no traffic congestion is present no difference is expected between a rational (EUT based) and a non-rational (PT based) decision-making behaviour. On the other hand, it is also clear that only when congestions appear it is worth investigating the traffic conditions because if no congestion occurs no intervention is necessary.

The first experiments show that the congestion tends to be concentrated where most of the agents are, i.e. if a route has a higher capacity than another it also tends to be used more intensively. Consequently, "good" routes tend to be considered too "good" and to be overused, even though rationally seen it is a "bad" judgement. But this is only speculative and hypothetical. Therefore a second set of experiments was performed (Sec. 8.9) with a similar scenario but that provided real data [SSC+05, Chm05]. The data shows exactly the same tendency, i.e. to overestimate how good a "good" route is. This behaviour is the same as the data, which is the same for the non-rational agents. Not only the PT based $\mathcal{Q}$-Learning shows the same tendency as it fits a slightly better the data than its rational counterparts (Tab. 8.14).

Finally, the third experiment set (Sec. 8.10) shows that the proposed agent reasoning, inserted in the agent architecture, can be used for simulating real-world scenarios. There, the city of Burgdorf in Switzerland was simulated, whose data was kindly provided by Guido Rindsfüser from

Emch+Berger.[1] Unfortunately, the data provided does not seem to be suitable for the simulated scenario, as extensive discussed in Sec. 8.10. Nonetheless, it serves the purpose of showing the scalability of the proposed architecture.

10.1 Future Work

This work is just the beginning. Several aspects of the non-rational decision-making are left for investigation and some of them, the ones considered more important, are discussed here.

The first and most important is to improve the clustering method so that less simulation runs are necessary to overcome its bias. A second point is to include an explicit structure for expressing the correlation among the options in the choice set This is important because usually options are correlated (specially in traffic) and this correlation must be modelled by the agent because the PT as the EUT require the options to be iid (Sec. 3.4). To tackle this point the Support Theory [TK94, RT97, Nar04] seems to be the most advanced model. But it may be too complex for a realistic use in MASim and therefore an approach similar of to the one used in microeconomics (Sec. 3.4) seems reasonable.

Another deficiency in the PT is that it is calibrated for monetary outcomes and its use with other types of outcomes remains unknown. In traffic, it means to collect real route decision data, without the monetary reward used in [SSC+05, Chm05], and then to calibrate the PT parameters, the α, β, λ, and γ values (Eq. 4.1). Again in traffic, the correct use of the *status quo* must be investigated. In other words, how people set their internal *status quo* value: is it the absolute 0, the last experienced travel-time, or the estimated travel-time. Moreover, is it route dependant or is it a common value for all routes going from A to B. Those questions can only be answered investigating the attitudes of individuals fulfilling their transportation needs.

Advancing one more step into the traffic modelling, how individuals make decision about the transportation modus: why to use the car or bus or bicycle or even deciding to walk. The structure supporting this was already done here using the super-network structure (Sec. A.2), which includes the necessary algorithms for navigating in such structure (Sec. A.4). But the decision model is not yet available.

In another direction, how can this agent architecture be adapted for other decision problems, such as stock markets and consumption behaviour. If it can, is this a better model than the rational model, does the data express deviations from the rational choice that corresponds to the PT behaviour. Moreover, are other psychological characteristics necessary for a better modelling. As explained in [Kah02], some of them are: framing, anchoring, accessibility, and attribute substitution. Framing corresponds to the "wording" of the problem presentation, anchoring is the presence of an external reference point for the choice. By accessibility it is meant how easy the solution can be recovered from memory and attribute substitution is a consequence of the accessibility, where instead of using the necessary attributes to evaluate an option some correlated attribute is used that is more easily accessible.

As one can see, much research is still ahead and this work is just scratching the surface. Hopefully it will bring awareness to the use of non-rational models instead of the rational ones, for modelling human decision-makers, as well as to show that the non-rational models are indeed better in modelling persons than the rational models.

[1] `http://www.emchberger.ch/`

Bibliography

[ACaERSMM93] José Augusto Azevedo, Maria Emília O. Santos Costa, Joaquim Jo ao E. R. Silvestre Madeira, and Ernesto Q. Vieira Martins. An algorithm for the ranking of shortest paths. *European Journal of Operational Research*, 69(1):97–106, August 1993.

[AF00] I. Ajzen and M. Fishbein. Attitudes and the attitude-behavior relation: Reasoned and automatic processes. *European review of social psychology*, 11:1–33, 2000.

[AH79] M. Allais and G. M. Hagen. *Expected Utility Hypotheses and the Allais Paradox: Contemporary Discussions of the Decisions Under Uncertainty with Allais' Rejoinder*, volume 21 of *Theory and Decision Library*. Kluwer Academic, September 1979.

[AH02] J. Aycock and R.N. Horspool. Pratical earley parsing. *The Computer Journal*, 45(6):620–630, 2002.

[All53] M. Allais. Le comportement de l'homme rationnel devant le risque: Critique des postulats et axiomes de l'École américaine. *Econometrica*, 21(4):503–546, October 1953. Translated and reprinted in [AH79].

[ALSS95] Rakesh Agrawal, King-Ip Lin, Harpreet S. Sawhney, and Kyuseok Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *VLDB '95: Proceedings of the 21th International Conference on Very Large Data Bases*, pages 490–501, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[ALSU06] Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison Wesley, 2nd edition, August 2006.

[AOR05] Dan Ariely, Axel Ockenfels, and Alvin Roth. An experimental analysis of ending rules in internet auctions. *RAND Journal of Economics*, 36(4):890–907, Winter 2005.

[AP05] Erel Avineri and Joseph N. Prashker. Sensitivity to travel time variability: Travelers' learning perspective. *Transportation Research Part C: Emerging Technologies*, 13(2):157–183, April 2005.

[Ari08] Dan Ariely. *Predictably Irrational: The Hidden Forces That Shape Our Decisions*. Harper Collins Publishers, 77–85 Fulham Palace Road, Hammersmith, London W6 8JB, March 2008.

[Art94] W. Brian Arthur. Inductive reasoning and bounded rationality. *The American Economic Review*, 84(2):406–411, May 1994. Papers and Proceedings of the Hundred and Sixth Annual Meeting of the American Economic Association.

[Avi06] Erel Avineri. The effect of reference point on stochastic network equilibrium. *Transportation Science*, 40(4):409–420, 2006.

[BA73] M. Ben-Akiva. *The structure of travel demand models*. PhD thesis, MIT, 1973.

[BAB99] M. Ben-Akiva and M. Bierlaire. Discrete choice methods and their applications to short-term travel decisions. In Randolph W. Hall, editor, *Handbook of Transportation Science*, Kluwer's International Series Operations Research Management Science, book 2, pages 5–34. Kluwer, 1999.

[BAL85] Moshe Ben-Akiva and S. Lerman. *Discrete Choice Analysis Theory and Application to Travel Demand*. MIT Press, Cambridge, MA., 1985.

[BAN06] M. Balmer, K. W. Axhausen, and K. Nagel. A demand generation framework for large scale micro simulations. In *85th Annual Meeting of the Transportation Research Board, TRB*, Washington, D.C., USA, January 2006.

[Bar07] Greg Barron. personal communication, September 12th 2007.

[Bat07] John Bates. History of demand modelling. In David A Hensher and Kenneth J Button, editors, *Handbook of Transport Modelling*, volume 1, chapter 2, pages 11–34. Emerald Group Publishing Limited, 2nd edition, October 2007.

[Bay63] Thomas Bayes. An essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, f. r. s. communicated by mr. price, in a letter to john canton, a. m. f. r. s. *Philosophical Transactions of the Royal Society of London*, 53:370–418, 1763.

[BBA+00] Ana L.C. Bazzan, Rafael H. Bordini, Gustavo K. Andriotti, Rosa Vicari, and Joachim Wahle. Wayward agents in a commuting scenario (personalities in the minority game). In Edmund Durfee, editor, *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS'2000),10–12 July, Boston*, pages 55–62, Los Alamitos, CA, 2000. IEEE Computer Society.

[BBAR06] Shlomo Bekhor, Moshe E. Ben-Akiva, and M. Scott Ramming. Evaluation of choice set generation algorithms for route choice models. *Annals of Operations Research*, 144(1):235–247, May 2006.

[BE03] Greg Barron and Ido Erev. Small feedback-based decisions and their limited correspondence to description-based decisions. *Journal of Behavioral Decision Making*, 16(3):215–233, July 2003.

[Bel57a] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, dover paperback (2003) edition, 1957.

[Bel57b] Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6, 1957.

[Ber38] Daniel Bernoulli. Specimen theoriae novae de mensura sortis. *Commentarii Academiae Scientiarum Imperiales Petropolitanae*, 5:175–192, 1738. Translated and reprinted in [Ber54].

[Ber44] Joseph Berkson. Application to the logistic function to bio-assay. *Journal of the American Statistical Association*, 39(227):357–365, September 1944.

[Ber54] Daniel Bernoulli. Exposition of a new theory on the measurement of risk. *Econometrica*, 22(1):23–36, January 1954.

[Ber93] James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer, 2nd edition, March 1993.

[BF05] Michel Bierlaire and Emma Frejinger. Route choice models with subpath components. In *Proceedings of Swiss Transport Research Conference (STRC) 2005*, Monte Verità, Ascona, Switzerland, March 2005.

[BHL05] Piet H. L. Bovy and Sascha Hoogendoorn-Lanser. Modelling route choice behaviour in multi-modal transport networks. *Transportation*, 32(4):341–368, July 2005.

[Bie03] Michel Bierlaire. BIOGEME: a free package for the estimation of discrete choice models. In *Proceedings of the 3rd Swiss Transport Research Conference*, Monte Verita, Ascona, Switzerland, 2003.

[Bie05] Herman J. Bierens. *Introduction to the Mathematical and Statistical Foundations of Econometrics*. Cambridge University Press, digital edition, January 2005.

[Bie08] Michel Bierlaire. *An introduction to BIOGEME Version 1.7*. EPFL, digital edition, August 2008. Version 1.7, last visited at 27.11.2008.

[BJM98] Christopher L. Barrett, Riko Jacob, and Madhav V. Marathe. Formal language constrained path problems. In S. Arnborg and L. Ivansson, editors, *Algorithm Theory SWAT'98*, volume LNCS 1432, pages 234–245. Springer Berlin / Heidelberg, 1998.

[BK03] A. L. C. Bazzan and F. Klügl. Route decision behaviour in a commuting scenario: Simple heuristics adaptation and effect of traffic forecast. In *Proceedings of the Euroworkshop on Behavioural Responses to ITS*, Eindhoven, 2003.

[BK05] Ana L.C. Bazzan and Franziska Klügl. Case studies on the braess paradox: Simulating route recommendation and learning in abstract and microscopic models. *Transportation Research Part C: Emerging Technologies*, 13(4):299–319, August 2005.

[Bli34a] C. I. Bliss. The method of probits. *Science*, 79(2037):28–39, January 1934. Corrected in [**?**].

[Bli34b] C. I. Bliss. The method of probits – a correction. *Science*, 79(2037):409–410, January 1934.

[BMR+08] M. Balmer, K. Meister, M. Rieser, K. Nagel, and K.W. Axhausen. Agent-based simulation of travel demand: Structure and computational performance of matsim-t. In *2nd TRB Conference on Innovations in Travel Modeling*, Portlan, USA, June 2008.

[BNW05] Dietrich Braess, Anna Nagurney, and Tina Wakolbinger. On a paradox of traffic planning. *Transportation Science*, 39(4):446–450, November 2005. joint translation of "Über ein Paradoxon der Verkehrsplanung" [Bra68] with Anna Nagurney, Tina Wakolbinger.

[Bra68] Dietrich Braess. Über ein paradoxon der verkehrsplanung. *Unternehmensforschung*, 12:258–268, March 1968.

[BRV+05] M. Balmer, M. Rieser, A. Vogel, K.W. Axhausen, and K. Nagel. Generating day plans based on origin-destination matrices. a comparison between visum and matsim based on kanton zurich data. In *Proceedings of Swiss Transport Research Conference (STRC) 2005*, Monte Verità, Ascona, Switzerland, March 2005.

[BS90] Piet H.L. Bovy and Eliahu Stern. *Route Choice: Wayfinding in Transport Networks*. Kluwer Academic Publishers, August 1990.

[CFCLB03] Kristof Carlier, Stella Fiorenzo-Catalano, Charles Lindveld, and Piet Bovy. A supernetwork approach towards multimodal travel modeling. In *In Proceedings of the 82nd TRB annual meeting*, pages 1–16, Washington D.C., 2003. National Academy Press.

[CH94] Colin F. Camerer and Teck-Hua Ho. Violations of the betweenness axiom and nonlinearity in probability. *Journal of Risk and Uncertainty*, 8(2):167–196, March 1994.

[Chm05] Thorsten Chmura. *Analyse, Modellierung und Simulationen von Routenwahlverhalten*. PhD thesis, Universität Duisburg-Essen, Physik, Astronomie - Universitaet Duisburg-Essen, April 2005.

[Cho54] Gustave Choquet. Theory of capacities. *Annales de l'institut Fourier*, 5:131–295, 1954.

[Cho56] Noam Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2:113–124, 1956.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, second edition, September 2001.

[CNZ99] Francisco Cribari-Neto and Spyros G. Zarkos. R: Yet another econometric programming environment. *Journal of Applied Econometrics*, 14(3):319–329, May/June 1999.

[CS70] John Cocke and Jacob T. Schwartz. Programming languages and their compilers: Preliminary notes. Technical report, Courant Institute of Mathematical Sciences, New York University, 1970.

[CS09] Richard D. Connors and Agachai Sumalee. A network equilibrium model with travellers' perception of stochastic travel times. *Transportation Research Part B: Methodological*, In Press, Corrected Proof, 2009.

[DG08] Carlos F. Daganzo and Nikolas Geroliminis. An analytical approximation for the macroscopic fundamental diagram of urban traffic. *Transportation Research Part B: Methodological*, 42(9):771–781, 2008.

[Dia71] Robert B. Dial. A probabilistic multipath traffic assignment model which obviates path enumeration. *Transportation Research*, 5(2):83–111, June 1971.

[Dij55] E. W. Dijkstra. A note on two problems in connexion with graphs. In *Numerische Mathematik*, volume I, pages 269–271, 1955.

[DMKSD06] Benedetto De Martino, Dharshan Kumaran, Ben Seymour, and Raymond J. Dolan. Frames, biases, and rational decision-making in the human brain. *Science*, 313(5787):684–687, 2006.

[DS05] G.B. Davies and S.E. Satchell. Continuous cumulative prospect theory and individual asset allocation. Cambridge Working Papers in Economics CWPE 0467, Faculty of Economics (formerly DAE), University of Cambridge, April 2005. available at http://www.dectech.org/People_Greg.html.

[Ear68] Jay Earley. *An Efficient Context-Free Parsing Algorithm*. PhD thesis, Carnegie-Mellon Univeristy, Pittsburgh PA, Dept. Of Computer Science, August 1968. Accession Number: AD0677685.

[Ear70] J. Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970.

[Edw92] A. W. F. Edwards. *Likelihood*. The Johns Hopkins University Press, Baltimore, expanded edition, October 1992.

[EH99] N. Christofides E. Hadjiconstantinou. An efficient implementation of an algorithm for finding k shortest simple paths. *Networks*, 34(2):88–101, 1999.

[Ell61] Daniel Ellsberg. Risk, ambiguity, and the savage axioms. *The Quarterly Journal of Economics*, 75(4):643–669, November 1961.

[Epp98] David Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28(2):652–673, 1998.

[Eps94] Seymour Epstein. Integration of the cognitive and the psychodynamic unconscious. *American Psychologist*, 49(8):709–724, August 1994.

[ER07] Ido Erev and Alvin E. Roth. Multi-agent learning and the descriptive value of simple models. *Artificial Intelligence*, 171(7):423–428, May 2007.

[FB09] M. Fosgerau and M. Bierlaire. Discrete choice models with multiplicative error terms. *Transportation Research Part B: Methodological*, 43(5):494–505, 2009.

[FCHLvN03] Stella Fiorenzo-Catalano, Sascha Hoogendoorn-Lanser, and Rob van Nes. Choice set composition modelling in multi-modal travelling. In *10th International Conference on Travel Behaviour Research*, Lucerne, August 2003.

[FCvNB04] Stella Fiorenzo-Catalano, Rob van Nes, and Piet H.L Bovy. Choice set generation for multi-modal travel analysis. *European Journal of Transport and Infrastructure Research*, 4(2):195–209, 2004.

[Fis22] Ronald Aylmer Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London*, 222(A):309–368, 1922.

[Fis70] Peter C. Fishburn. *Utility Theory For Decision Making*. Operations Research Society of America. Publications in operations research. Wiley, 1970.

[Fis81] Peter C. Fishburn. Subjective expected utility: A review of normative theories. *Theory and Decision*, 13(2):139–199, June 1981.

[FKP04] Manuel Fehler, Franziska Klügl, and Frank Puppe. Techniques for analysis and calibration of multi-agent simulations. In *Engineering Societies in the Agents World V*, number 3451 in Lecture Notes in Computer Science, pages 305–321. Springer Berlin / Heidelberg, 2004.

[FKP06] Manuel Fehler, Franziska Klügl, and Frank Puppe. Approaches for resolving the dilemma between model structure refinement and parameter calibration in agent-based simulations. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 120–122, New York, NY, USA, May 2006. ACM.

[FT87] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3):596–615, July 1987.

[FW97] H.P. Fennema and P.P. Wakker. Original and cumulative prospect theory: A discussion of empirical differences. *Journal of Behavioral Decision Making*, 10(1):53–64, 1997.

[GG96] Gerd Gigerenzer and Daniel G. Goldstein. Reasoning the fast and frugal way: Models of bounded rationality. *Psychological Review*, 103(4):650–669, 1996.

[GG02] Daniel G. Goldstein and Gerd Gigerenzer. Models of ecological rationality: The recognition heuristic. *Psychological Review*, 109(1):75–90, January 2002.

[Goo98] P. Goodwin. The end of equilibrium. In T. Gärling, T. Laitila, and K. Westin, editors, *Theoretical Foundations of Travel Choice Modeling*, pages 185–195. Pergamon, 1st edition, June 1998. 978-0080430621.

[GP06] Faruk Gul and Wolfgang Pesendorfer. Random expected utility. *Econometrica*, 74(1):121–146, 2006.

[GSP+89] Gerd Gigerenzer, Zeno Swijtink, Theodore Porter, Lorraine Daston, John Beatty, and Lorenz Kruger. *The Empire of Chance: How Probability Changed Science and Everyday Life*. Cambridge University Press, The Pitt Building, Trumpington Stree, Cambridge CB2 1RP, UK, 1989. Reprint in 1990.

[HB07] David A Hensher and Kenneth J Button, editors. *Handbook of Transport Modelling*, volume 1. Emerald Group Publishing Limited, 2nd edition, October 2007.

[HBZ04] E. A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, pages 709–715, San Jose, California, July 2004. Association for the Advancement of Artificial Intelligence, AAAI Press.

[HG05] John M.C. Hutchinson and Gerd Gigerenzer. Simple heuristics and rules of thumb: Where psychologists and behavioural biologists might meet. *Behavioural Processes*, 69(2):97–124, May 2005. Proceedings of the meeting of the Society for the Quantitative Analyses of Behavior (SQAB 2004).

[HK05] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, CA, USA, second edition, November 2005.

[HMS03] John E. Hershberger, Matthew Maxel, and Subhash Suri. Finding the k shortest simple paths: a new algorithm and its implementation. In *Proceedings 5th Workshop Algorithm Engineering & Experiments (ALENEX)*, pages 26–36. SIAM, January 2003.

[HMU01] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, second edition, 2001.

[Hoc05] H.H. Hochmair. Investigating the effectiveness of the least-angle strategy for wayfinding in unknown street networks. *Environment and Planning B, Planning and Design*, 32(5):673–691, 2005.

[Hor84] Joel L. Horowitz. The stability of stochastic equilibrium in a two-link transportation network. *Transportation Research Part B: Methodological*, 18(1):13–28, February 1984.

[HSB07] John Hershberger, Subhash Suri, and Amit Bhosle. On the difficulty of some shortest path problems. *ACM Trans. Algorithms*, 3(1):5, 2007.

[JM03] Víctor M. Jiménez and Andrés Marzal. A lazy version of eppstein's k shortest paths algorithm. In *Proceedings Experimental and Efficient Algorithms: Second International Workshop, WEA 2003*, pages 179–190, Ascona, Switzerland, May 2003.

[Kah02] Daniel Kahneman. Maps of bounded rationality: A perspective on intuitive judgment and choice. In Tore Frängsmyr, editor, *The Nobel Prizes 2002*, pages 449–489, Aula Magna, Stockholm University, December 2002. Nobel Foundation. presented by Professor Torsten Persson, Chairman of the Prize Committee.

[Kas65] T. Kasami. An efficient recognition and syntax-analysis algorithm for context-free languages. Technical report, Air Force Cambridge Research Lab, Bedford, MA, 1965. Scientific report AFCRL-65-758.

[KB02] F. Klügl and A. L. C. Bazzan. Simulation of adaptive agents: Learning heuristics for route choice in a commuter scenario. In *Proceedings of the first international joint conference on Autonomous Agents and Multi-Agent Systems, AAMAS*, volume 1, pages 217–218, Bologna, Italy, July 2002. New York: ACM Press. Extended Abstract.

[KB04] Franziska Klügl and Ana L. C. Bazzan. Route decision behaviour in a commuting scenario: Simple heuristics adaptation and effect of traffic forecast. *Journal of Artificial Societies and Social Simulation*, 7(1), January 2004.

[KBW03] F. Klügl, A. L. C. Bazzan, and J. Wahle. Selection of information types based on personal utility - a testbed for traffic information markets. In *Proceedings of the second international joint conference on Autonomous Agents and Multi-Agent Systems, AAMAS*, Melbourne, Australia, July 2003. New York: ACM Press.

[KF02] D. Kahneman and S. Frederick. Representativeness revisited: Attribute substitution in intuitive judgment. In T. Gilovich, D. Griffin, and D. Kahneman, editors, *Heuristics and Biases: The Psychology of Intuitive Judgment*, chapter 2, pages 49–81. Cambridge University Press, New York, 2002.

[KF05] D. Kahneman and S. Frederick. A model of heuristic judgment. In K.J. Holyoak and R.G. Morrison, editors, *The Cambridge Handbook of Thinking and Reasoning*, chapter 12, pages 267–293. Cambridge University Press, 2005.

[KF06] D. Kahneman and S. Frederick. Frames and brains: elicitation and control of response tendencies. *Trends in Cognitive Sciences*, 11(2):45–46, February 2006.

[Kni21] Frank H. Knight. *Risk, Uncertainty, and Profit.* Number 31 in Hart, Schaffner, and Marx Prize Essays. Houghton Mifflin Company, Boston and New York, 1921.

[Knu98] Donald E. Knuth. *Art of Computer Programming*, volume 3. Addison-Wesley Professional, 3rd edition, April 1998. Sorting and Searching.

[KR88] Brian W. Kernighan and Dennis M. Ritchie. *C Programming Language.* Prentice Hall Software. Prentice Hall PTR, 2nd edition, April 1988.

[KR00] John H. Kagel and Alvin E. Roth. The dynamics of reorganization in matching markets: A laboratory experiment motivated by a natural experiment. *The Quarterly Journal of Economics*, 115(1):201–235, February 2000.

[KT79] Daniel Kahneman and Amos Tversky. Prospect theory: An analysis of decision under risk. *Econometrica*, 47(2):263–292, March 1979.

[LF91] R. Duncan Luce and Peter C. Fishburn. Rank- and sign-dependent linear utility models for finite first-order gambles. *Journal of Risk and Uncertainty*, 4(1):29–59, January 1991.

[Lit94] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 157–163, New Brunswick, NJ, 1994. Morgan Kaufmann.

[Lov91] William S. Lovejoy. Computationally feasible bounds for partially observed markov decision processes. *Operations Research*, 39(1):162–175, January/February 1991.

[Luc59] R. Duncan Luce. *Individual Choice Behavior: A Theoretical Analysis*. Dover Publications, New York, 1959.

[MA06] Nina Mazar and Dan Ariely. Dishonesty in everyday life and its policy implications. *Journal of Public Policy and Marketing*, 25(1):117–126, Spring 2006.

[Man77] Charles F. Manski. The structure of random utility models. *Theory and Decision*, 8(3):229–254, July 1977.

[Mar60] J. Marschak. Binary choice constraints on random utility indications. In K. Arrow, editor, *Stanford Symposium on Mathematical Methods in the Social Sciences*, pages 312–329, Stanford, CA, 1960. Stanford University Press.

[Mar71] A. A. Markov. Extension of the limit theorems of probability theory to a sum of variables connected in a chain. In R. Howard, editor, *Dynamic Probabilistic Systems*, chapter Appendix B. John Wiley and Sons, August 1971. reprinted.

[Mar04] Robert Martin. The st. petersburg paradox. Online encyclopedia, Stanford University, http://plato.stanford.edu/entries/paradox-stpetersburg/, July 2004. Stanford Encyclopedia of Philosophy.

[McF74] D. McFadden. Conditional logit analysis of qualitative choice behavior. *Frontiers of Econometrics*, 1974.

[McF99] Daniel McFadden. Rationality for economists? *Journal of Risk and Uncertainty*, 19(1–3):73–105, December 1999.

[McN07] Michale G. McNally. The four step model. In David A Hensher and Kenneth J Button, editors, *Handbook of Transport Modelling*, volume 1, chapter 3, pages 35–41. Emerald Group Publishing Limited, 2nd edition, October 2007.

[Mil56] George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97, 1956.

[Mil85] John Milnor. On the concept of attractor. *Communications in Mathematical Physics*, 99(2):177–195, July 1985.

[MM02] Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 346–357. VLDB Endowment, 2002.

[MMS90] M. Migliore, V. Martorana, and F. Sciortino. An algorithm to find all paths between two nodes in a graph. *J. Comput. Phys.*, 87(1):231–236, 1990.

[MT00] D. McFadden and K. Train. Mixed mnl models of discrete response. *Journal of Applied Econometrics*, 15:447–470, 2000.

[Nar04] Louis Narens. A new foundation for support theory. Technical Report MBS 04-04, Mathematical Behavioral Sciences at UC Irvine, University of California, Irvine, January 2004.

[NCK+06] Ehren L. Newman, Jeremy B. Caplan, Matthew P. Kirschen, Igor O. Korolev, Robert Sekuler, and Michael J. Kahana. Learning your way around town: How virtual taxicab drivers learn to use both layout and landmark information. *Cognition*, 2006. In Press, Corrected Proof, Available online 1 August 2006.

[Ned99] Zhivko Prodanov Nedev. Finding an even simple path in a directed planar graph. *SIAM Journal on Computing*, 29(2):685–695, 1999.

[Osb03] Martin J. Osborne. *An Introduction to Game Theory*. Oxford University Press, New York, New York, USA, August 2003.

[Ram02] Michael Scott Ramming. *Network Knowledge and Route Choice*. PhD thesis, Massachusetts Institute of Technology, February 2002.

[Reb89] Arthur S. Reber. Implicit learning and tacit knowledge. *Journal of Experimental Psychology: General*, 118(3):219–235, September 1989.

[Ric59] F. J. Richards. A flexible growth function for empirical use. *Journal of Experimental Botany*, 10(2):290–301, 1959.

[RPM04] Roger P. Roess, Elena S. Prassas, and William R. McShane. *Traffic Engineering*. Pearson Education, Inc., Upper Saddle River, New Jersey 07458, third (international) edition, 2004.

[RT97] Yuval Rottenstreich and Amos Tversky. Unpacking, repacking, and anchoring: Advances in support theory. *Psychological Review*, 104(2):406–415, April 1997.

[Rub07] Ariel Rubinstein. Instinctive and cognitive reasoning: A study of response times. *Economic Journal*, 117(523):1243–1259, October 2007.

[RW02] M. Raubal and S. Winter. Enriching wayfinding instructions with local landmarks. In Max J. Egenhofer and David M. Mark, editors, *Geographic Information Science*, volume 2478 of *Lecture Notes in Computer Science*, pages 243–259. Springer, Berlin, 2002.

[RW08] Marc Oliver Rieger and Mei Wang. Prospect theory for continuous distributions. *Journal of Risk and Uncertainty*, 2008. to appear.

[Sch03] Ulrich Schmidt. Reference dependence in cumulative prospect theory. *Journal of Mathematical Psychology*, 47(2):122–131, 2003.

[Sha53] L. S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences of the United States of America*, 39(10):1095–1100, October 1953.

[Sim55] Herbert A. Simon. A behavioral model of rational choice. *The Quarterly Journal of Economics*, 69(1):99–118, February 1955.

[Sim56] Herbert A. Simon. Rational choice and the structure of the environment. *Psychological Review*, 63:129–138, 1956.

[Sim79] Herbert A. Simon. Rational decision making in business organizations. *The American Economic Review*, 69(4):493–513, September 1979.

[Sim82] Herbert Simon. *Models of Bounded Rationality*. MIT Press, 1982.

[Sim86] Herber A. Simon. Rationality in psycghology and economics. *The Journal of Business*, 59(4):209–224, October 1986.

[Sim92] Herbert A. Simon. Rational decision-making in business organizations. In Assar Lindbeck, editor, *Nobel Lectures, Economics 1969-1980*, pages 343–371, Singapore, December 1992. Nobel Foundation, World Scientific Publishing Co. Nobel Prize in Economics 1978.

[Sip05] Michael Sipser. *Introduction to the Theory of Computation*. Course Technology, second edition, February 2005.

[SK04] Metin Senbil and Ryuichi Kitamura. Reference points in commuter departure time choice: A prospect theoretic test of alternative decision frames. *Journal of Intelligent Transportation Systems*, 8(1):19–31, January 2004.

[SLB08] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, New York, New York, USA, December 2008.

[Slo96] Steven A. Sloman. The empirical case for two systems of reasoning. *Psychological Bulletin*, 119(1):3–22, January 1996.

[Slo02] Steven A. Sloman. Two systems of reasoning. In T. Gilovich, D. Griffin, and D. Kahneman, editors, *Heuristics and Biases: the Psychology of Intuitive Judgment*, pages 379–396. Cambridge University Press, Cambridge, 2002.

[SPG03] Y. Shoham, R. Powers, and T. Grenager. Multi-agent reinforcement learning: a critical survey. Technical report, Stanford University, 2003.

[SPG04] Yoav Shoham, Robert Powers, and Trond Grenager. On the agenda(s) of research on multi-agent learning. In *AAAI 2004 Fall Symposium on Artificial Multi-Agent Learning*. AAAI Press, 2004.

[SPG07] Yoav Shoham, Robert Powers, and Trond Grenager. If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171(7):365–377, 2007. special issue on Foundations of Multi-Agent Learning.

[SSC+05] R. Selten, M. Schreckenberg, T. Chmura, T. Pitz, S. Kube, S.F. Hafstein, R. Chrobok, A. Pottmeier, and J. Wahle. Experimental investigation of day-to-day route-choice behavior and network simulations of autobahn traffic in north rhine-westphalia. In M. Schreckenberg and R. Selten, editors, *Human Behaviour and Traffic Networks*, pages 1–21. Springer, Heidelberg, June 2005. 978-3540212201.

[Ste46] S. S. Stevens. On the theory of scales of measurement. *Science*, 103(2684):677–680, June 1946.

[Sto07] Peter Stone. Multiagent learning is not the answer. it is the question. *Artificial Intelligence*, 171(7):402–405, May 2007. Foundations of Multi-Agent Learning.

[Str00] Bjarne Stroustrup. *The C++ Programming Language*. Addison Wesley, 201 W. 103rd Street, Indianapolis, IN 46290, special edition, 2000.

[Sut84] Richard S. Sutton. *Temporal credit assignment in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, July 1984.

[Sut88] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, August 1988.

[SV00] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, June 2000.

[TK83] Amos Tversky and Daniel Kahneman. Extensional versus intuitive reasoning: The conjunction fallacy in probability judgment. *Psychological Review*, 90(4):293–315, October 1983.

[TK92] Amos Tversky and Daniel Kahneman. Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk And Uncertainty*, 5(4):297–323, October 1992.

[TK94] Amos Tversky and Derek J. Koehler. Support theory: A nonextensional representation of subjective probability. *Psychological Review*, 101(4):547–567, October 1994.

[Tra00] Transportation Research Board. *Highway Capacity Manual: 2000*. Transportation Research Board, Washington, D.C., lslf edition, December 2000.

[vdZC05] N.J. van der Zijpp and S. Fiorenzo Catalano. Path enumeration by finding the constrained k-shortest paths. *Transportation Research Part B: Methodological*, 39(6):545–563, July 2005.

[vN28] John von Neumann. Zur theorie der gesellschaftspiele. *Mathematische Annalen*, 100(1):295–320, December 1928.

[vNM07] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton University, 60th anniversary edition edition, March 2007. 1st Edition: 1944.

[Vov97] Peter Vovsha. The cross-nested logit model: Application to mode choice in the tel-aviv metropolitan area. *Transportation Research Record*, 1607:6–15, 1997.

[Wah02] Joachim Wahle. *Information in Intelligent Transportation Systems - Information in Intelligenten Transportsystemen*. PhD thesis, Gerhard-Mercator-Universität, Physik, Astronomie - Gerhard-Mercator-Universität, January 2002.

[War52] John Glen Wardrop. Some theoretical aspects of road traffic research. In *Proceedings of Institution of Civil Engineers, Part II*, volume 1, pages 325–378, London, 1952.

[Wat89] Christopher J. C. H. Watkins. *Learning with Delayed Rewards*. PhD thesis, Cambridge University, 1989. Psychology Department.

[WD92] Christopher J. C. H. Watkins and Peter Dayan. Technical note: Q-learning. *Machine Learning*, 8(3):279–292, 5 1992.

[WG96] George Wu and Richard Gonzalez. Curvature of the probability weighting function. *Management Science*, 42(12):1676–1690, 1996.

[Wim76] W. C. Wimsatt. Reductionism, levels of organization and the mind-body problem. In G. G. Globus, G. Maxwell, and I. Savodnik, editors, *Consciousness and the Brain: A Scientific and Philosophic Inquiry*, pages 199–267. Plenum Press, New York, 1976.

[Woo03] Jeffrey Wooldridge. *Introductory Econometrics - A Modern Approach*. South-Western, 2nd edition, 2003.

[WT93] Peter Wakker and Amos Tversky. An axiomatization of cumulative prospect theory. *Journal of Risk and Uncertainty*, 7(2):147–175, October 1993.

[You67] Daniel H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208, 1967.

Appendix A

Technologies And Algorithms

This chapter is an appendix because it is only marginally related to the subject of this thesis, even though it is important for its infrastructure. Three technologies are here presented, namely: super-networks; Context-Free Grammars (CFG), for navigating in a super-network; and path generation algorithms for the super-network. A super-network can be seen as an edge-labelled weighted directed graph (digraph) that is generated from another (compacter) edge-labelled weighted digraph. The other technologies are modifications of existing algorithms for using this special graph (the super-network).

The motivation for adopting the super-networks in the traffic network representation is because it presents an efficient alternative for multi-modal route generation. This concept was informally presented in [CFCLB03, FCHLvN03, FCvNB04] and based on this structure it is here formalised and an efficient way to "navigate" in it is also developed (using a modified Dijkstra [Dij55] algorithm). The advantage in using the methods described here is twofold: performance and flexibility. In [vdZC05] the authors use a restricted form of super-networks and from it they generate several routes, eliminating the invalid ones afterwards – the method shown here does not generate any invalid route. In another approach to the multi-modal route choice [BHL05], the "multi" is restricted to a maximum of three modi combination (*Access + Main + Egress*) – the new approach shown here has not such restriction. Expanding the amount of modi, in [BJM98] it is suggested the use of a grammar; but a prohibitive approach is used there by adopting a bottom-up parser, which needs the whole word to give any evaluation about its validity (almost the same procedure as in [vdZC05]). All these issues are not present in the solution presented here.

A.1 Notation, Typography, And Conventions

Before presenting the algorithms it is necessary to establish a common notation that crosses several fields. These definitions for the set theory are:

- Upper case Roman letters, as in A, B, and C, represent sets.
- Lower case Roman letters, such as a, b, and c, represent elements in sets.
- Greek letters have a context dependant meaning, i.e. their meaning will be explained as they are presented (can be sets, usually in upper case Greek letters as Σ, or elements, usually in lower case as α).
- Upper case Roman letters in different typefaces are used to represent data-structures or set of sets, as in $\mathcal{N}$ (a set of networks); but the type faces as in $\mathfrak{N}$ are used exclusively for sets of sets. In this particular case $\mathcal{N}$ is one of the possible sets in $\mathfrak{N}$ – $\mathfrak{N}$ is the universal set of $\mathcal{N}$.
- $|A|$ corresponds to the amount of elements in the set A.

- A^B represents a map from B into A, i.e. being A and B sets then a structure exists that maps elements of B into elements of A.
- $\texttt{Get} : \mathfrak{X} \times B \mapsto A$ is the function that recovers the A element mapped by the given B element, or NIL if no A element is associated with the B element. The association is stored into a map $X \in \mathfrak{X}$, where $\mathfrak{X}$ is the set of all possible maps.
- $\texttt{Put} : \mathfrak{X} \times B \times A \mapsto \{\}$ is the function to maps an element of B into an element of A, stored in $X \in \mathfrak{X}$.
- A^n is an n-ary Cartesian product, i.e. it represents an ordered tuple of n elements from the set/type A, with $n \in \mathbb{N}^*$.
- $\texttt{Get} : A^n \times [1, n] \mapsto A$ is the function that returns the ith element (second argument) of a tuple (first argument) – it is assumed that the first element's index is 1 and the last n.
- $A \backslash B$ represents the set A without the elements present in B: $A \backslash B = A - B$.

For the graph theory, the definitions are:

- $G(V, E)$ represents a graph with vertex set V and edge set E. It is also possible to refer to these elements by subscriptions: V_G is the vertex set of graph G and E_G its edge set. If "$G(V, E)$" is in the right-side of an attribution, as in $G^{'} \leftarrow G(V, E)$, it means that a new graph is created with vertex set V and edge set E.
- $\mathfrak{G}$ is the set of all graphs. This notation is used to specify domain and codomain of functions. The same logic is valid for $\mathfrak{V}$ and $\mathfrak{E}$ to denote the set of all sets of vertices and all sets of edges.
- $\texttt{Source} : E \mapsto V$ returns for a digraph the corresponding source vertex for the given edge.
- $\texttt{Target} : E \mapsto V$ as in `Source()` but returns the target vertex.
- $\texttt{In} : V \mapsto \mathfrak{E}$ is a function that returns for a digraph all edges that have the given vertex as target: $\texttt{In}(v) = E^{'}$ and $E^{'} = \{\forall e \in E^{'} \mid \texttt{Target}(e) = v\}$.
- $\texttt{Out} : V \mapsto \mathfrak{E}$ as in `In()` but returns all edges that have the given vertex as source: $\texttt{Out}(v) = E^{'}$ and $E^{'} = \{\forall e \in E^{'} \mid \texttt{Source}(e) = v\}$.

A.2 Super-Network

A super-network, after [CFCLB03, FCHLvN03, FCvNB04], is a way to structure a weighted digraph specially designed for representing a traffic network. The principle is to stratify a graph according to its modi, i.e. each stratus corresponds to a network that contains only the edges associated with the modus it represents. Then several interconnected networks form a super-network. In Fig. A.1 this process (in a schematic view) can be seen, where G is the original graph – representing a traffic network – and $\mathcal{S}$ is the resulting super-network, with its several networks (N_1, N_2, and N_3).

For simplicity the word network is used to refer to a sub-graph/network of a super-network and this last (super-network) refers to the resulting structure of a graph after this "segmentation" process. Unfortunately neither a formal definition of the super-network structure nor the transformation algorithms were given. The guidelines are to have each modus represented in a network in the super-network. Therefore here the definition of super-networks is made (in a precise mathematical definition) as well as the super-network generation algorithm (derived from the definition).

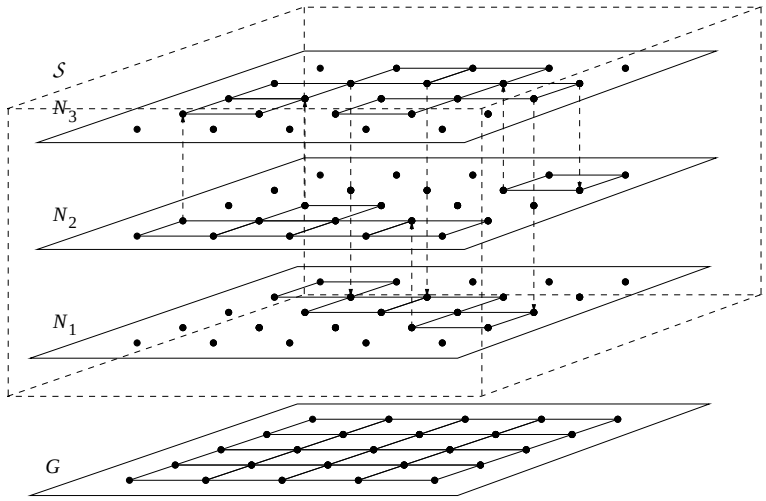


Figure A.1: From a digraph G to a super-network $\mathcal{S}$.

A.2.1 Super-network: Formalisation

To formalise the concept of super-networks some definitions are necessary. First it is assumed that a super-network $\mathcal{S}$ is derived or generated from an initial labelled digraph G. Moreover, each edge $e \in E_G$ has a set of labels $L_e \neq \{\}$ and each label supposedly corresponds to a modus – therefore from now on the label concept instead of the modus concept is used in the context of networks and super-networks. The set of all labels in a graph is: $L_G = \cup_{e \in E_G} L_e$.

A super-network for its turn is defined by $\mathcal{S}(G, \mathcal{N}, S_0, C, \mathcal{L})$, where G is the original digraph, $\mathcal{N}$ the network set, $S_0 \in \mathcal{N}$ is a special network (the start network), C the connecting edge set (explain latter), and $\mathcal{L}$ the label set. It is also expected that a function $\texttt{Labels} : E_{\mathcal{S}} \mapsto \mathfrak{L}$ is given that returns the labels associated with a given edge, where $\mathfrak{L}$ is the set of label sets and E_G is the edge set from G. Then $E_{\mathcal{S}} = E_G \cup E_{\mathcal{N}} \cup C$, where $E_{\mathcal{N}} = \cup_{N \in \mathcal{N}} E_N$ is the set of all edges of all networks in the network set $\mathcal{N}$ (E_N is the edge set of the network N). The same semantic has also $\texttt{Labels} : \mathfrak{E} \mapsto \mathfrak{L}$ but for a set of edges, returning $L = \cup_{e \in E} \texttt{Labels}(e)$.

A network $N(V, E) \in \mathcal{N}$ is also a digraph and resents a modus/label from the traffic network, given by G. Therefore a function is necessary: $\texttt{Equiv} : V_{\mathcal{N}} \cup E_{\mathcal{N}} \mapsto V_G \cup E_G$. This function returns the reference element of G that is being represented by the given network element. It implies that all elements in a network have a corresponding element in G, i.e.: $\nexists x \in V_{\mathcal{N}} \cup E_{\mathcal{N}} \mid \texttt{Equiv}(x) = NIL$. With this function it can also be defined $\texttt{IsEquiv} : V_{\mathcal{N}} \cup E_{\mathcal{N}} \times V_G \cup E_G \mapsto \{true, false\}$ that asserts if two elements are semantically equivalent, i.e. if the first argument semantically corresponds to the second element (in G). This function is formally defined as: $\texttt{IsEquiv}(x, y) \equiv (\texttt{Equiv}(x) = y) \wedge (\texttt{Equiv}(x) \neq NIL)$.

Let a super-network $\mathcal{S}$ has an initial graph G, a set of networks $\mathcal{N}$, a set of labels $\mathcal{L}$, and a set of connecting edges C, then all statements below hold.

- Every network maps all vertices from G exactly once.
- If an edge from G is represented/mapped in a network it is the only representation in that particular network, maintaining its semantic.
- Each network has only one label and no two networks have the same label, i.e. exactly one network per label.

The above mentioned statements are formally expressed in Eq. A.1.

$$
\begin{aligned}
&\forall N \in \mathcal{N} \Rightarrow |V_N| = |V_G| \\
&\forall N \in \mathcal{N}, \forall v_N \in V_N, \exists v_G \in V_G, \nexists v_G^{'} \in V_G \mid \\
&\qquad \texttt{IsEquiv}(v_N, v_G) = true \wedge \texttt{IsEquiv}(v_N, v_G^{'}) = true \wedge v_G \equiv v_G^{'} \\
&\forall N \in \mathcal{N} \Rightarrow \forall v_N \in V_N, \nexists v_N^{'} \in V_N \mid \texttt{Equiv}(v_N) = \texttt{Equiv}(v_N^{'}) \wedge v_N \neq v_N^{'} \\
&\forall N \in \mathcal{N}, \forall e_N \in E_N \Rightarrow \texttt{Equiv}(\texttt{Source}(e_N)) = \texttt{Source}(\texttt{Equiv}(v_N) \\
&\qquad \wedge \texttt{Equiv}(\texttt{Target}(e_N)) = \texttt{Target}(\texttt{Equiv}(v_N) \\
&\forall N \in \mathcal{N} \Rightarrow \forall e_N, e_N^{'} \in E_N, \exists e_G \in E_G \mid e_N \neq e_N^{'} \wedge \texttt{IsEquiv}(e_N, e_G) \\
&\qquad \oplus \texttt{IsEquiv}(e_N^{'}, e_G) \\
&\forall N \in \mathcal{N} \Rightarrow \forall e_N, e_N^{'} \in E_N \mid \texttt{Labels}(e_N) = \texttt{Labels}(e_N^{'}) \wedge \ |\texttt{Labels}(e_N)| \ = 1 \\
&\forall N, M \in \mathcal{N}, N \not\equiv M \Rightarrow \forall e_N \in E_N \ \nexists e_M \in E_M \mid \texttt{Labels}(e_N) = \texttt{Labels}(e_M)
\end{aligned}
\tag{A.1}
$$

Another special element of the super-network structure is the start network S_0. This network must also respect all rules in Eq. A.1 but with one extra restriction: $E_{S_0} = \{\}$, i.e. it has no edges and it is the only network with an empty edge set: $\nexists N \in \mathcal{N} \mid E_N = \{\} \wedge N \neq S_0$.

To completely specify a super-network the sets C and $\mathcal{L}$ must be defined. The set C is a special edge set that have no correspondence with any edge in G, they are the connecting edges. These edges have labels too and their set is L_C and $L_C \subset \mathcal{L} \wedge L_C \cap L_G = \{\}$, meaning that L_C is disjoint from the regular labels (L_G). Thus the set C is defined below and formally in Eq. A.2.

- Each network is connected to all other networks only through equivalent vertices and just one connecting edge $c \in C$ connects any two vertices
- The edges in C have no equivalent representation in G, another formalisation would be: $\forall c \in C \mid \texttt{Equiv}(c) = NIL$.
- The label $\alpha \in L_C$ is reserved for connecting the start network S_0 to all others.
- Similar to α, ω is also reserved for connecting the other networks back to the start network.
- The edges in C do not belong to any network: $C \cap E_{\mathcal{N}} = \{\}$.

$$
\begin{aligned}
&\forall N, M \in \mathcal{N} \ \exists c \in C \mid \texttt{Source}(c) \in V_N \wedge \texttt{Target}(c) \in V_M \wedge N \neq M \\
&\qquad \wedge \texttt{Equiv}(\texttt{Source}(c)) = \texttt{Equiv}(\texttt{Target}(c)) \\
&\forall c \in C \ \ \nexists c^{'} \in C \mid \texttt{Source}(c) = \texttt{Source}(c^{'}) \wedge \texttt{Target}(c) = \texttt{Target}(c^{'}) \\
&\forall c \in C \ \ \nexists e_G \in G \mid \texttt{IsEquiv}(c, e_G) = true \\
&\forall v \in V_{S_0} \ \exists c, c^{'} \in C \mid \texttt{Source}(c) = v \wedge \texttt{Labels}(c) = \{\alpha\} \\
&\qquad \wedge \texttt{Target}(c^{'}) = v \wedge \texttt{Labels}(c^{'}) = \{\omega\} \\
&\nexists c \in C \mid (\texttt{Labels}(c) = \{\alpha\} \wedge \texttt{Source}(c) \notin V_{S_0}) \\
&\qquad \vee (\texttt{Labels}(c) = \{\omega\} \wedge \texttt{Target}(c) \notin V_{S_0})
\end{aligned}
\tag{A.2}
$$

It also implies that $|L_C| \geq 3$, since all edges have labels and $\alpha \in L_C$ as well as $\omega \in L_C$.

The super-network definition can be optimised by adding the rule in Eq. A.3 for excluding unnecessary vertices in $\mathcal{N}$. It is assumed that the super-network was already generated and the function $\texttt{Remove} : V_{\mathcal{N}} \mapsto \mathfrak{S}$ is provided with the following behaviour (where $\mathfrak{S}$ the set of all super-networks): the function removes the given vertex as well as any edge connected to it, returning a new super-network without these elements.

$$
\forall N \in \mathcal{N} \backslash \{S_0\} \ \forall v \in V_N \mid \texttt{In}(v) \subset C \wedge \texttt{Out}(v) \subset C \Rightarrow \texttt{Remove}(v) \tag{A.3}
$$

What Eq. A.3 does is to select all vertices that have only connecting edges and eliminate them. They would be like the vertices in the super-network $\mathcal{S}$ from Fig. A.1 that have no full lines reaching them.

A.2.2 Super-network: Generation Algorithm

In this section the implemented algorithm for generating a super-network that attains to the rules in Sec. A.2.1 (including the optimisation step of Eq. A.3), is presented. For the algorithms extra functions are necessary:

- `Copy` : $V_G \mapsto V_N$ It creates a semantically equivalent copy from the given original vertex (from graph G) to be used in a network $N \in \mathcal{N}$.
- `Copy` : $E_G \times V_N \times V_N \times \mathcal{L} \mapsto E_N$ As the previous function it also makes a semantically equivalent copy but from an edges in G. The difference is that the edge is copied but its source set to be the second argument (a vertex resulting from the previous function), the target the third argument, and the label set by the fourth argument.
- `Connect` : $V_N \times V_N \times L_C \mapsto C$ This function connects two vertices assigning a label and returning a connecting edge – it uses the first argument as source, the second as the target, and the third as the label.
- `ConnLabel` : $L_G \times L_G \mapsto L_C$ For the connecting edges it is necessary a label $l \in L_C$ and this label is only predefined for the edges connecting the S_0 to and from other networks – α and ω, respectively. To recover the other labels this function is necessary. Because each network has only one label (except for the start network S_0) it is enough to specify as arguments the source label (first argument) and the target label (second).

With these above defined functions, a super-network is generated by the function `Supernet`(G) from Algo. A.1, where G is the input digraph, from which a super-network is wanted. This algorithm generates a super-network that obeys all rules in Sec. A.2.1, including Eq. A.3.

A.2.3 Super-network: Example

To give an idea of how a super-network looks like a series of images are here presented, actually one of the many possible visual representations. To avoid visual "pollution" in the illustrative figures neither arrows (to indicate edges' directions) nor multiple edges for the same vertex pair are depicted. This means that in Fig. A.2 each edge should be "seen" as actually two, one for each possible direction.

The process of creating a super-network starts with an input labelled digraph (as in Fig. A.2). In this figure each edge carries its labels, a subset from $L_G = \{b, c, w\}$. From this graph (in Fig. A.2) the creation process "extracts" the different networks, one for each label. The network set $\mathcal{N}$ (before including the connecting edges) is in Fig. A.3, where the start network S_0 called *start* is already present. For convenience each network is marked with its corresponding label.

The next step is to connect all networks, the creation of the edge set C. In Fig. A.4 an aesthetic view of the final super-network $\mathcal{S}$ can be seen, which was generated from the graph in Fig. A.2. There, the dashed lines represent the connecting edges in a "reduced" form, to keep it visually clear (the edges in C). The dashed lines (in Fig. A.4) are a reminder of what they really are, depicted in Fig. A.5. There, all connections between any two networks, including the start network S_0, are depicted (observe that each edge represents a pair of directed edges, one for each direction).

To make it even clearer, in Fig. A.6 a possible path in this super-network is shown. The path "jumps" over networks using the connecting edges (in this representations all other edges where omitted to avoid confusion). It can be noticed that the path starts and ends in the start network.

A.2.4 Navigation

For the navigation in the super-network some prerequisites are necessary:

- All paths must start and end in the start network S_0.

Algorithm A.1: `Supernet(•)`

Data: G the input graph
Result: A new super-network $\mathcal{S}$

$\mathcal{N} \leftarrow \{\}$; `/* creates an empty network set */`
`/* creates the start network` S_0 `*/`
$S_0 \leftarrow \{\}$;
forall $v \in V_G$ **do**
  $S_0 \leftarrow S_0 \cup \{$ `Copy(`v`)` $\}$;
end
`/* creates the regular networks */`
$H \leftarrow \{\}$; `/* creates an empty map (`$\mathcal{N} \mapsto \mathcal{L}$`), with manipulation functions Get and Put */`
forall $lbl \in L_G$ **do**
  $N \leftarrow$ `Network(`G, lbl`)`;
  $\mathcal{N} \leftarrow \mathcal{N} \cup \{N\}$; `/* see Algo. A.2 */`
  `Put(`H, N, lbl`)` ; `/* associates the key` N `to the value` lbl `in map` H `*/`
end
$C \leftarrow \{\}$; `/* creates an empty edge set */`
`/* connects all regular networks */`
$\mathcal{L} \leftarrow L_G$;
forall $N \in \mathcal{N}$ **do**
  $l_N \leftarrow$ `Get(`H, N`)` ; `/* gets the label associated with the source network` N `*/`
  forall $M \in \mathcal{N} \backslash \{N\}$ **do**
    $l_M \leftarrow$ `Get(`H, M`)` ; `/* gets the label associated with the target network` M `*/`
    $l_C \leftarrow$ `ConnLabel(`l_N, l_M`)`;
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{l_C\}$;
    $C \leftarrow C \cup$ `ConnectNet(`N, M, l_C`)` ; `/* see Algo. A.3 */`
  end
end
`/* connects all regular networks to the start network */`
$\mathcal{L} \leftarrow \mathcal{L} \cup \{\alpha, \omega\}$;
forall $N \in \mathcal{N}$ **do**
  $C \leftarrow C \cup$ `ConnectNet(`S_0, N, α`)` $\cup$ `ConnectNet(`N, S_0, ω`)`;
end
$\mathcal{N} \leftarrow \mathcal{N} \cup \{S_0\}$;
$\mathcal{S} \leftarrow \mathcal{S}(G, \mathcal{N}, S_0, C, \mathcal{L})$; `/* creates a new super-network */`
return $\mathcal{S}$;

Algorithm A.2: `Network(•)`

Data: G the input graph
Data: lbl the corresponding label
Result: A new network N

$V_N \leftarrow$ `Vertices(`G, lbl`)` ; `/* see Algo. A.4 */`
$E_N \leftarrow$ `Edges(`G, lbl`)` ; `/* see Algo. A.5 */`
$N \leftarrow N(V_N, E_N)$; `/* creates a new network */`
return N;

Algorithm A.3: `ConnectNet(•)`

Data: N the source network
Data: M the target network
Data: *lbl* the label to assign to the edges
Result: Connecting edge set C

```
1 C ← {} ; /* creates an empty edge set */
  /* for all possible source vertices ...  */
2 forall s ∈ V_N do
  |  /* ...searches the appropriated target ...  */
3 |  forall t ∈ V_M do
  |  |  /* ...finding the appropriated target ...  */
4 |  |  if Equiv(s) = Equiv(t) then
  |  |  |  /* ...connects both */
5 |  |  |  C ← C ∪ { Connect(s, t, lbl) };
6 |  |  end
7 |  end
8 end
9 return C;
```

Algorithm A.4: `Vertices(•)`

Data: G the input graph
Data: *lbl* the corresponding label
Result: A vertex set V

```
1 V ← {} ; /* creates an empty vertex set */
2 forall v ∈ V_G do
  |  /* gets all labels reaching the node */
3 |  L ← Labels(In(v)) ∪ Labels(Out(v));
  |  /* just nodes that have edges with the given label lbl are worth of
  |     creation */
4 |  if lbl ∈ L then
5 |  |  V ← V ∪ { Copy(v) };
6 |  end
7 end
8 return V;
```

Algorithm A.5: `Edges(•)`

Data: G the input graph
Data: *lbl* the corresponding label
Result: A edge set E

```
1  E ← {} ; /* creates an empty edge set */
   /* for all possible source vertices s ...  */
2  forall s ∈ V_N do
       /* ...and all possible target vertices t ...  */
3      forall t ∈ V_N do
4          F ← Out(Equiv(s)) ∩ In(Equiv(t));
           /* ...and all edges between the given pair (s,t) ...  */
5          forall e ∈ F do
6              if lbl ∈ Labels(e) then
                   /* ...copies the edges */
7                  E ← E ∪ { Copy(e, s, t, lbl) };
8              end
9          end
10     end
11 end
12 return E;
```

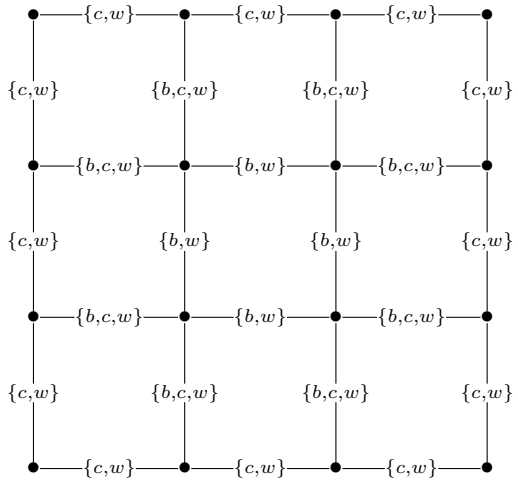


Figure A.2: Labelled graph

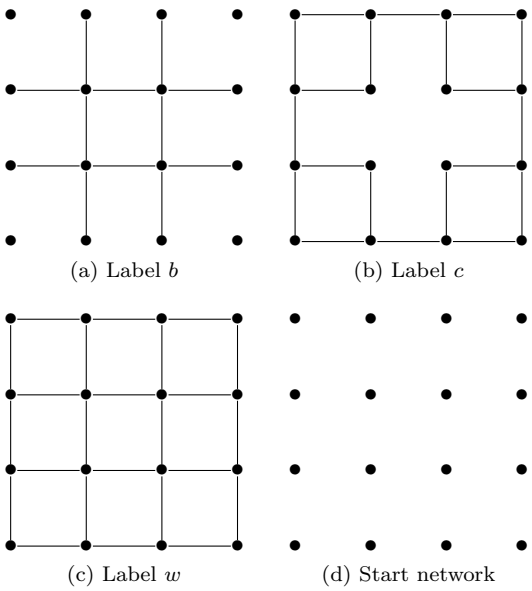


Figure A.3: Network view

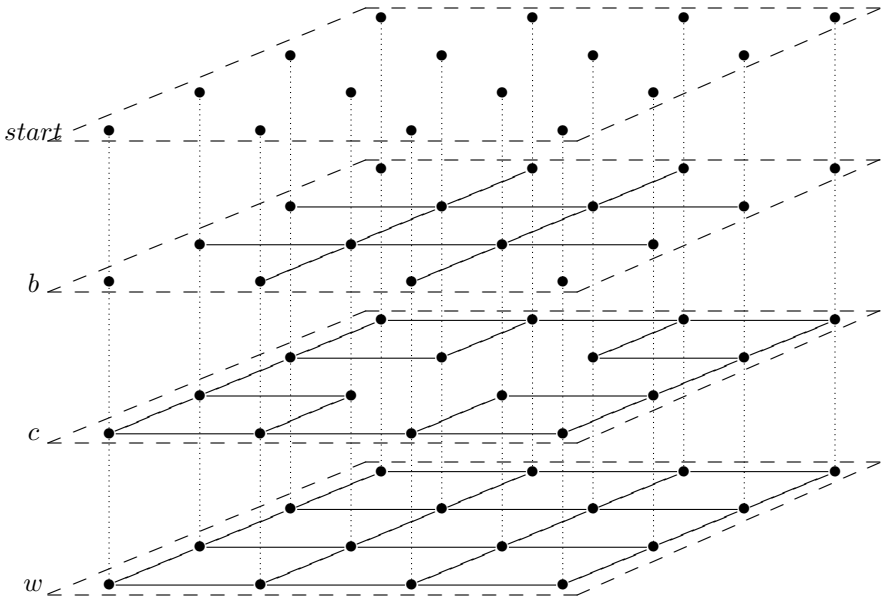


Figure A.4: Super-network in "aesthetic" view

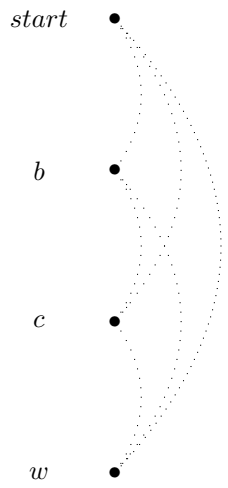


Figure A.5: One column "real" view from connecting edges

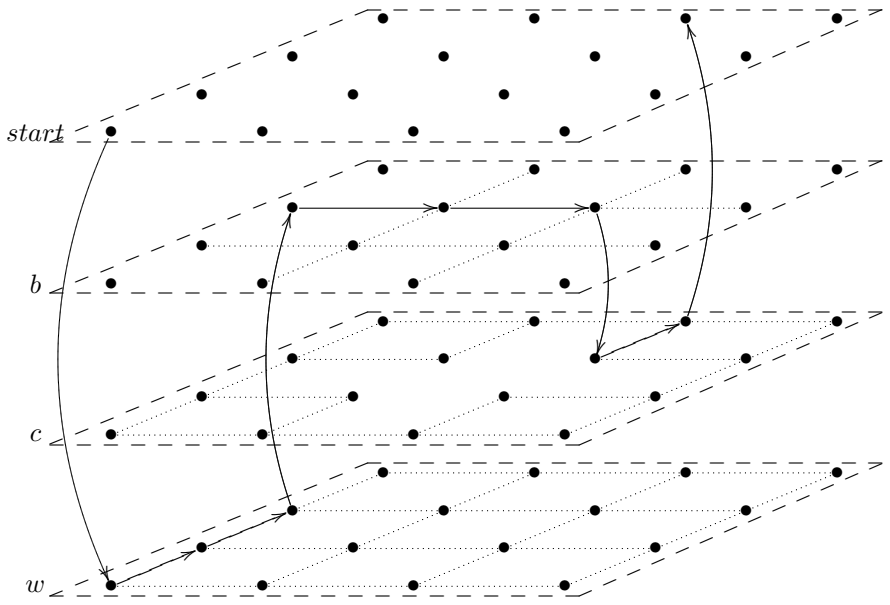


Figure A.6: A possible path in the super-network

- A grammar must be provided for path validation, which is based on the alphabet $\mathcal{L}$.
- The final label sequence must be a valid word according to the given grammar. This means that the sequence of labels (built by concatenating the labels from each edge in the path) must be a valid word according to the provided grammar.

A.3 Context-Free Grammars

In this section the use of Context-Free Grammars [HMU01] (CFG)[1] for the super-network navigation and path validation will be discussed. The use of grammars for navigating in graphs was already proposed by Nedev [Ned99], where CFGs were used for solving the *k disjoint paths problem for directed planar graphs* – here, nevertheless, another approach is proposed. The parsing process of a "label" word (hereafter called just word) demands three elements: the word itself, a grammar, and a parser. The parser takes the rules encoded in the grammar and evaluates if the word belongs to (can be generated by) the grammar. In case the grammar can be used to generate the word then the word is said to be accepted/valid. For the approach presented here only CFGs are allowed and the Earley parser [Ear68, Ear70] is adopted – actually an optimised version [AH02].

A CFG is defined as $R(V, \Sigma, P, \delta)$[2] where V is the finite non-terminal (or variable) set, Σ the finite terminal set (or alphabet), P the finite production rule set, and $\delta \in V$ the start symbol. Another property is that $V \cap \Sigma = \{\}$, i.e. no symbol is shared between Σ and V. Specific to the CFGs, the rules in P are in the form: $[V \rightarrow (V \cup \Sigma)^*]$. This means that in the left-side of the production rule must have exactly one non-terminal symbol and in the right-side none or any sequence of terminals and/or non-terminals.

The advantage of a CFG is in the trade-off between expressiveness and parser efficiency. A CFG is a powerful formalism and allows among other things to build syntax parsers for computer programming languages, such as C [KR88], C++ [Str00], or Java.[3] It has nevertheless restrictions as well. A CFG cannot restrict the total amount of a specific terminal – in traffic such rule would be something like: "no route is allowed to have more than 10 walk segments". Even though this particular case could be implemented in the Earley parser, without complexity increase. In general, any rule that requires a back-tracking, i.e. to go back and re-analyse the terminal sequence, cannot be implemented using a CFG. This implies that a CFG is not a general purpose constraint expressing formalism. It means that if a CFG is not enough for expressing the wanted constrains some post-generation filtering will be necessary.

A.3.1 Context-Free Grammars: Earley Parser

The parser chosen for the task is the Earley parser [Ear68, Ear70, AH02] that is a top-down parser, i.e. it evaluates prefixes and builds its evaluation (acceptance or rejection) based on the read terminal and the previous read word (previous terminal sequence). A counter-example would be the Cocke-Younger-Kasami [CS70, Kas65, You67] (CYK) algorithm which is bottom-up (needs the whole word before evaluating it) and also needs the grammar to be in the Chomsky normal form [Sip05, HMU01] (CNF) (CFGs can be automatic transformed to conform the CNF). The advantages of the Earley are overwhelming to even consider any other parser. First, it can parse any CFG in any form (just the original parser, but even for the optimal the modifications are minimal in comparison to CNF requirements). Second, its complexity for any CFG is $O(n^3)$, for non-ambiguous CFGs is $O(n^2)$, and $O(n)$ for most LR(K) grammars – the CYK algorithm has complexity $\Omega(n^3)$ for all sub-cases of CFGs. A derived advantage is: if the grammar is in fact

[1] In Chomsky [Cho56] classification system, CFGs are type 2 languages. More about grammars can be found in [Sip05, HMU01].

[2] In the praxis the letter G is used to symbolise a grammar but to avoid conflict with the input graph, also referred as G, the letter R, from rules, is used.

[3] `http://java.sun.com/`

a sub-case of a CFG the parser will perform accordingly – for more information about LR(K) grammars and other sub-cases, please refer to [ALSU06].

The Earley algorithm is a look-ahead algorithm too, it first tries to predict which terminal comes next and when it reads it, the algorithm eliminates the invalid hypothesis. For the optimised version [AH02] it is assumed that the grammar was modified to include the following production: $[\delta^{'} \rightarrow \delta]$ and $\delta^{'}$ included in V as well as made the start symbol. In the parser, the basic structure is the so called Earley set of dot-productions (or Earley's dot notation); it can also be called Earley state. A dot-production has the aspect in Eq. A.4, where $A \in V$ is a non-terminal; $\rho, \tau \in (V \cup \Sigma)^*$ are sequences of terminals and/or non-terminals; j the set where the dot-production was created; and $\bullet$ the current parsing position for the rule (in Eq. A.4 it means that ρ was already parsed and τ is still to be parsed). The notation used is: upper-case letters to represent non-terminals in V; lower-case letters for terminals in Σ; and Greek letters for sequences of terminals, non-terminals, or a mixture of both.

$$[A \rightarrow \rho \bullet \tau, j] \tag{A.4}$$

Each Earley set has a sequence and they are identified by S_j, meaning that it refers to the jth set in the sequence. A word is accepted if after reading the last terminal the corresponding set has a dot-production as $[\delta^{'} \rightarrow \delta\bullet, 0]$ (the start production has completed its parsing process). The initial set S_0 is always created before scanning any terminal and it starts by including the dot-production $[\delta^{'} \rightarrow \bullet\delta, 0]$. After that the three parsing procedures below are executed (in order). Assuming that the input word is x and x_i corresponds to the terminal at position i and x_1 is the first terminal:

Scanner If $[A \rightarrow \rho \bullet a\tau, j]$ is in S_i and $x_{i+1} = a$, then add $[A \rightarrow \rho a \bullet \tau, j]$ to S_{i+1}.

Predictor If $[A \rightarrow \rho \bullet B\tau, j]$ is in S_i, then add $[B \rightarrow \bullet\rho, i]$ to S_{i+1}.

Completer If $[A \rightarrow \rho\bullet, j]$ is in S_i, then add $[B \rightarrow \tau A\bullet, k]$ to S_i for all dot-productions $[B \rightarrow \mu \bullet A\varphi, k]$ in S_j.

For a visual impression of this parsing process, assume the example grammar in Eq. A.5 that is modified into the grammar in Eq. A.6.

$$\begin{aligned}
R &= (V, \Sigma, P, \delta) && \text{The original grammar} \\
V &= \{E\} && \text{The non-terminal set} \\
\Sigma &= \{+, n\} && \text{The terminal set} \\
P &= \{E \rightarrow E + E | n\} && \text{Production rules} \\
\delta &= E && \text{Start symbol } E
\end{aligned} \tag{A.5}$$

$$\begin{aligned}
R^{'} &= (V^{'}, \Sigma, P^{'}, \delta^{'}) && \text{The modified grammar} \\
V^{'} &= \{E, S^{'}\} && \text{The non-terminal set} \\
\Sigma &= \{+, n\} && \text{The terminal set} \\
P^{'} &= \{E \rightarrow E + E | n, S^{'} \rightarrow E\} && \text{Production rules} \\
\delta^{'} &= S^{'} && \text{Start symbol } S^{'}
\end{aligned} \tag{A.6}$$

The word to parse is $x = n + n$ and the parsing process is shown in Tab. A.1. There, the first column refers to the dot-production's number (used just for clarification); then comes the dot-product followed by a code, where the letters refer to: S = Scanner, P = Predictor, and C = Completer procedure. The letters (indicating the procedure) are also followed by a reference, such as "P $S_0(0)$" which means that the resulting dot-production was created by the Predictor procedure run over the first dot-production of set S_0. As already explained, the first set S_0 is initialised with the inclusion of the first dot-production $[S^{'} \rightarrow \bullet E, 0]$, indicated in Tab. A.1a by

"Added". In this case when the parser reads the first terminal (Tab. A.1b) it can be seen that just n is a valid word and therefore it is accepted (indicated by the presence of the $[S' \rightarrow E\bullet, 0]$ dot-product in the second item from set S_1). Because the input wasn't consumed the algorithm keeps parsing until it reaches the end of word (set S_3 in Tab. A.1d) and as a result it is a valid word (this is again confirmed by the presence of the dot-production $[S' \rightarrow E\bullet, 0]$, in S_3).

Table A.1: Parsing of $x = n + n$

(a) $S_0 : \bullet n + n$

(0)	S'	$\rightarrow$	$\bullet E$	$,0$	Added
(1)	E	$\rightarrow$	$\bullet E + E$	$,0$	P $S_0(0)$
(2)	E	$\rightarrow$	$\bullet n$	$,0$	P $S_0(0)$

(b) $S_1 : n \bullet + n$

(0)	E	$\rightarrow$	$n\bullet$	$,0$	S $S_0(2)$
(1)	S'	$\rightarrow$	$E\bullet$	$,0$	C $S_0(0)$
(2)	E	$\rightarrow$	$E \bullet + E$	$,0$	C $S_0(1)$

(c) $S_2 : n + \bullet n$

(0)	E	$\rightarrow$	$E + \bullet E$	$,0$	S $S_1(2)$
(1)	E	$\rightarrow$	$\bullet E + E$	$,2$	P $S_2(0)$
(2)	E	$\rightarrow$	$\bullet n$	$,2$	P $S_2(0)$

(d) $S_3 : n + n\bullet$

(0)	E	$\rightarrow$	$n\bullet$	$,2$	S $S_2(2)$
(1)	E	$\rightarrow$	$E + E\bullet$	$,0$	C $S_2(0)$
(2)	E	$\rightarrow$	$E \bullet + E$	$,2$	C $S_2(1)$
(3)	S'	$\rightarrow$	$E\bullet$	$,0$	C $S_0(0)$

A.3.2 Context-Free Grammars: Procedures For Automatic Grammar Generators

It is strongly suggested the use of grammar automatic generators. In this section the procedures for generating grammars automatically are presented. For the procedures a function that was informally presented in Sec. A.2.2 is necessary: `ConnLabel`($\bullet$). Additionally to this, the functions below are also necessary. There the symbol $\mathfrak{R}$ represents the set of all grammars; $\mathfrak{P}$ the set of all production sets; $\mathfrak{X}$ the set of all terminal and non-terminal sets; and X a terminal or a non-terminal.

- `AddProd` : $\mathfrak{R} \times \mathfrak{P} \mapsto \mathfrak{R}$ This function adds a production rule (second argument) to the given grammar (first argument), from which a new derived grammar is returned. This modified grammar has the production added to P_R with all necessary non-terminal symbols that are not already present in V_R (either in the left- or right-side of the production).
- `NewVar` : $\mathfrak{X} \mapsto X$ With this function it is possible to generate a new symbol, i.e. a symbol that is not present in the given symbol set (argument). The objective is to use the returned symbol as a non-terminal. A typical function call would be: `NewVar`($V_R \cup \Sigma_R$).
- `NewProd` : $V \times X^n \mapsto P$ It creates a new production having a non-terminal (first argument) as the production's left-side and a sequence of terminals and non-terminals (second argument) as the right-side.
- `Arrangements` : $\mathfrak{X} \times \mathbb{N}^* \mapsto (X^n)^n$ This function returns all possible Arrangements for the input symbol set (first argument) with the given size (second argument). An example would be:

$$\begin{aligned}
\texttt{Arrangements}(\{A,B,C\},1) &= \{(A),(B),(C)\} \\
\texttt{Arrangements}(\{A,B,C\},2) &= \{(A,B),(B,A),(A,C),(C,A),(B,C),(C,B)\} \\
\texttt{Arrangements}(\{A,B,C\},3) &= \{(A,B,C),(A,C,B), \\
&\quad\ (B,A,C),(B,C,A), \\
&\quad\ (C,A,B),(C,B,A)\}
\end{aligned}$$

To generate a simple grammar see Algo. A.6 but because the resulting grammar is not optimised and has some ambiguities it is recommended the use of an automatic grammar simplifier [HMU01].

Algorithm A.6: `Grammar(•)`

Data: Σ the regular terminal set
Data: $\Sigma^{'}$ all connecting terminals
Data: α the special terminal from start network to all others
Data: ω the special terminal from all others networks to the start
Data: $\mathcal{C} : (\Sigma^n)^n$ terminal combinations
Result: The grammar R

$N \leftarrow \Sigma \cup \Sigma^{'} \cup \{\alpha, \omega\}$; `/* all symbols (all terminals) */`
$S \leftarrow$ `NewVar(`N`)` ; `/* the start symbol */`
$N \leftarrow N \cup \{S\}$;
$R \leftarrow R(\{\}, \Sigma \cup \Sigma^{'} \cup \{\alpha, \omega\}, \{\}, S)$; `/* creates a new grammar, with empty production set */`
`/* one production rule for each combination */`
forall $c \in \mathcal{C}$ **do**
  $v \leftarrow$ `NewVar(`N`)`;
  $N \leftarrow N \cup \{v\}$;
  $P \leftarrow$ `LabelComb(`R, c, v`)` ; `/* see Algo. A.7 */`
  `/* all productions */`
  forall $p \in P$ **do**
    $R \leftarrow$ `AddProd(`R, P`)`;
  end
  `/* adds combination production to start symbol:` $[S \rightarrow v]$ `*/`
  $R \leftarrow$ `AddProd(`R, `NewProd(`S, (v)`))`;
end
$N \leftarrow \Sigma_R \cup V_R$;
`/* creates a new start symbol to include the connection with the start network */`
$S^{'} \leftarrow$ `NewVar(`N`)`;
$N \leftarrow N \cup \{S^{'}\}$;
$R \leftarrow$ `AddProd(`R, `NewProd(`$S^{'}$, (α, S, ω)`))` ; `/* encapsulation:` $[S^{'} \rightarrow \alpha S \omega]$ `*/`
`/* creates a new start symbol to conform the prerequisites of the parser */`
$S^{''} \leftarrow$ `NewVar(`N`)` ; `/*` $[S^{''} \rightarrow S^{'}]$ `*/`
$R \leftarrow$ `AddProd(`R, `NewProd(`$S^{''}$, $(S^{'})$`))`;
$R \leftarrow R(V_R, \Sigma_R, P_R, S^{''})$;
return R;

Algorithm A.7: `LabelComb(•)`

Data: R the grammar
Data: Σ the terminal set to combine
Data: S the left-side symbol
Result: A production set P

```
   /* creates the recursive production rules for each terminal */
 1 N ← V_R ∪ Σ ; /* holds the already used symbols */
 2 L ← RecurLbls(Σ, N, S) ; /* see Algo. A.9 */
 3 P ← P_L;
 4 N ← N ∪ N_L;
 5 M ← M_L;
 6 E ← E_L;
   /* non-terminals for all combinations */
 7 A ← VarsComb( Σ, N, S, E) ; /* see Algo. A.8 */
 8 P ← P ∪ P_A;
 9 C ← C_L;
10 N ← N ∪ N_A;
11 F ← F_L;
12 E ← E ∪ E_A;
   /* adds the production rules to the combination non-terminals */
13 forall c ∈ C do
14 |  r ← () ; /* creates an empty production right-side symbol sequence */
15 |  for i → 1 to |c| − 1 do
16 |  |  r ← r+ Get(M, Get(c, i)) ; /* adds to the right-side the corresponding
   |  |  simple recursive production */
17 |  |  r ← r+ ConnLabel(Get(c, i), Get(c, i + 1)) ; /* adds the necessary
   |  |  transition terminal from current terminal to the next in the
   |  |  combination */
18 |  end
   |  /* the last symbol in the combination is generated by any non-terminal
   |     whose production starts with that terminal */
19 |  forall e ∈ Get(E, Get(c, |c|)) do
   |  |  /* creates the production corresponding to the non-terminal associated
   |  |     with the combination */
20 |  |  P → P ∪ { NewProd(Get(F, c), (r, e)) };
21 |  end
22 end
23 return P;
```

Algorithm A.8: `VarsComb(•)`

Data: Σ the terminal set
Data: N already used symbols
Data: S the left-side symbol
Data: map E from terminals to non-terminal set, whose productions start with the generation of terminal set as key
Result: A production set P
Result: C combination set, with all possible terminal combinations
Result: N updated
Result: map F that maps terminal combinations to corresponding non-terminal
Result: map E updated

```
/* generates the non-terminals for all possible combinations */
C ← {} ; /* creates an empty set ((Σ^n)^n) */
F ← {} ; /* creates an empty map (N^(Σ^N)) */
for i ← 2 to |Σ| do
    forall c ∈ Arrangements(Σ, i) do
        t ← Get(c, 0);
        /* creates a new non-terminal to hold the combination generator symbol
           */
        v ← NewVar(N);
        N ← N ∪ {v};
        C ← C ∪ {c};
        /* adds v as a non-terminal that generates a production that starts
           generating the terminal t */
        Put(E, t, Get(E, t) ∪ {v});
        /* associates the non-terminal v with the combination c */
        Put(F, v, c);
        /* adds a production to generates the specific combination */
        P ← P ∪ { NewProd(S, (v)) };
    end
end
return (P, C, N, F, E);
```

Algorithm A.9: `RecurLbls(•)`

Data: Σ the terminal set
Data: N already used symbols
Data: S the left-side symbol
Result: A production set P
Result: N updated
Result: map M from terminal to non-terminal, whose production is the simple recursion production
Result: map E from terminals to non-terminal set, whose productions start with the generation of terminal set as key

$P \leftarrow \{\}$; `/* creates an empty production set */`
$M \leftarrow \{\}$; `/* creates an empty map (` N^{Σ} `) */`
$E \leftarrow \{\}$; `/* creates an empty map (` $(N^n)^{\Sigma}$ `) */`
`/* creates the recursive production rules for each terminal */`
forall $t \in \Sigma$ **do**
  $v \leftarrow$ `NewVar(`N`)` ; `/* creates a new non-terminal to hold the recursion generator symbol */`
  $N \leftarrow N \cup \{v\}$;
  `Put(`M`,` t`,` v`)` ; `/* associates the terminal` t `with the non-terminal` v `*/`
  `/* add` v `as a non-terminal that generates a production that starts generating the terminal` t `*/`
  `Put(`E`,` t`, Get(`E`,` t`)` $\cup \{v\}$`)`;
  $P \leftarrow P \cup$ `RecurProd(`v`,` t`)` ; `/* see Algo. A.10 */`
  `/* adds a simple production to permit single symbol strings */`
  $P \leftarrow P \cup \{$ `NewProd(`S`,` (v)`)` $\}$;
end
return (P, N, M, E);

Algorithm A.10: `RecurProd(•)`

Data: N left-side non-terminal
Data: t the terminal
Result: A production set P

$P \leftarrow \{\}$; `/* creates an empty production set */`
$P \leftarrow P \cup \{$ `NewProd(`N`,` (t)`)` $\}$; `/* simple production:` $N \rightarrow t$ `*/`
$P \leftarrow P \cup \{$ `NewProd(`N`,` (N,t)`)` $\}$; `/* recursive production:` $N \rightarrow Nt$ `*/`
return P ; `/*` $N \rightarrow Nt|t$ `*/`

A.3.3 Context-Free Grammars: Earley Parser Validity States

Using the Earley parser presented, two functions are built to test the validity of any state, i.e. in any state the validity of the already read word can be checked. This property is explored by the shortest-path algorithm presented next. The two functions are: $\texttt{IsValid} : \mathfrak{S} \mapsto \{true, false\}$, where $\mathfrak{S}$ is the set of all Earley sets. The algorithm for this function is simple and to explain it, it is assumed that π is the validity criterion (in the example of Eq. A.6 it would be $\pi = [S^{'} \rightarrow E\bullet, 0]$); then $\texttt{IsValid}(S_j) = true \equiv \pi \in S_j$, other else it returns false.

For the second function the optimised version [AH02] of Earley parser is needed. In this version another restriction is necessary: elimination of ϵ-productions (a step in the grammar optimisation routine) and elimination of ϵ completely. This means that grammars that accept empty strings are not allowed (this is easily overcome if the ϵ-production elimination checks if the empty word is accepted and then "flags" the grammar to accept empty strings, this is the approach adopted). The parser is below presented:[4]

Scanner If $[A \rightarrow \rho \bullet a\tau, j]$ is in S_i and $x_{i+1} = a$, then add $[A \rightarrow \rho a \bullet \tau, j]$ to S_{i+1}.

Completer If $[A \rightarrow \rho\bullet, j]$ is in S_i, then add $[B \rightarrow \tau A\bullet, k]$ to S_i for all dot-productions $[B \rightarrow \mu \bullet A\varphi, m]$ in S_j.

Predictor If $[A \rightarrow \rho \bullet B\tau, j]$ is in S_i, then add $[B \rightarrow \bullet\rho, i]$ to S_i.

The second function $\texttt{MayBeValid} : \mathfrak{S} \mapsto \{true, false\}$ is then even simpler: $\texttt{MayBeValid}(S) = true \equiv S \neq \{\}$. It works because the only procedure that adds items to the Earley sets is the scanner. The scanner will only add an item if it finds a dot-production that has the read terminal right before the dot position. This means that a production sequence exists (following the dot-productions backwards in the set sequence) that could generate the current sequence, so the word may still be valid. If this is not true, then no dot-production will be added to the next set (the other procedures work only on the current set).

A.3.4 Context-Free Grammars: Earley Parser Complexity

This modified Earley parser has the same complexity of the original and for each function its complexity is given. It is assumed that the word to parse is much larger than the amount of productions in the given grammar.

Scanner $O(n)$ because it only analyses the dot-productions on the current set and since in the worst case $|S_i| = k|x|$[5] and the procedure only depends on the amount of items in the set, so its complexity is $O(n)$.

Predictor $O(n)$ for the same reason of the previous procedure.

Completer $O(n^2)$ because it is the only procedure that does a backtracking, searching in all previous sets (contribution with $O(n)$, since a parsing process has $|x| + 1$ sets) and in the worst case it may need to analyse kn productions, then its complexity is $O(n^2)$ – actually, $kn(0 + 1 + 2 + \cdots + n)$, being k the amount of productions in the grammar and n the size of the word.

This means that the complexity of the parsing process for a single terminal can be said $O(n^2)$ (in the worst case). For the proof of correctness please refer to [Ear68, Ear70, AH02].

[4] For the proof of validity please refer to [AH02].

[5] Notice that since the grammar production amount is fixed and assumed smaller than the word size, then in the worst case when a terminal is read all productions are added, so for each state a maximum of $|P_G|$ productions are added, which is still proportional to $|x|$.

A.4 Modified Dijkstra

For the path finding algorithms a shortest-path method is necessary. For this task the Dijkstra Algo. [Dij55] was chosen and the original algorithm is here modified to include the label parsing procedure explained before – the Earley parser. It is assumed that exactly one label is associated with each edge in a super-network, then the function $\texttt{GetLabel} : E_{\mathcal{S}} \mapsto \mathcal{L}$ returns this label. This function can be seen as $\texttt{GetLabel}(e \in E_{\mathcal{S}}) = \texttt{GetAny}(\texttt{Labels}(e))$, where $\texttt{GetAny}(\bullet)$ returns an element from a set and because in the super-network the label set associated with any edge has exactly one element it returns the correct element.

Before presenting the modified version, the standard Dijkstra algorithm (only modified to support multiple edges per vertex pair) is shown in Algo. A.11. However, to completely understand the algorithm, some structures and functions must be defined. A Fibonacci Heap [FT87] is the basic heap structure used here, which is modified to store a tuple instead of a pair. The tuple is as follow: $t : \mathbb{R}^+ \times V \times H \times E \times \{true, false\}$. Where the arguments are: weight $w \geq 0$; corresponding vertex; previous tuple entry that lead to this entry; used edge to reach the vertex; and a boolean value to flag the entry as the true minimum. The tuple can be seen as a structure with the following field names: $t : (weight, vertex, previous, edge, minimun)$. To facilitate the manipulation, some auxiliary functions are defined as $\texttt{Weight}(t)$ which returns the corresponding field on the structure and $\texttt{Weight}(t, w)$ which adjust the field *weight* in t to w. The other functions are $\texttt{Vertex}(\bullet)$ for manipulating the *vertex* field; $\texttt{Edge}(\bullet)$ for the *edge* field manipulation; and so on.

In this algorithm the function $\texttt{Min}(H)$ supposes that H is a Fibonacci Heap [FT87] and it returns the tuple with the lowest weight, assuming the values in the field *weight*. Another function that manipulates the heap is $\texttt{Decrease}(H, t, w)$, which decreases the key/weight for entry t to w for the heap H. The function $\texttt{EdgeWeight}(\bullet) : E \to \mathbb{R}^+$ returns the corresponding weight for the edge informed as argument. The Algo. A.11 preserves the original optimal complexity $O(m+n\ log(n))$. The proof will be omitted here because it can be derived from the proof for the modified version. To keep the explanations simple the operation $\texttt{SaveTuple}(\bullet)$ stores the corresponding tuple on the corresponding vertex. This is latter changed and the complexity impact analysed as well.

A.4.1 Modified relaxation function

The modifications to include the Earley parser are shown in Algo. A.13, which replaces the original function in Algo. A.12. The stored tuple for the Fibonacci Heap must be expanded, it must include an entry for the corresponding Earley set, letting $t : (weight, vertex, previous, edge, minimun, state)$. Then additionally to the already existing functions the $\texttt{Parse} : \mathfrak{E} \times \Sigma \mapsto \mathfrak{E}$ that corresponds to parsing a single terminal (second argument) and generating the next Earley set based on the current set (first argument), where $\mathfrak{E}$ is the set of all Earley sets.

A.4.2 Algorithmic Complexity

The difference between the relaxation functions in Algo. A.12 and A.13 is first in the weighting operation in line 2 of Algo. A.12 that expands to lines 2 through 7 in Algo. A.13. This operation corresponds to a single terminal parsing, whose complexity is $O(n^2)$ – see Sec. A.3.4. Following the edge terminal parsing operation the used edge weight is adjusted. The adjustment lets the weight unchanged if the corresponding Earley state "may be valid" and infinity otherwise, which invalidates the weight/edge. The second modification is the Earley set storage in the main **if** in Algo. A.13 (in line 11), which is not present in Algo. A.12.

The complexities of the used functions are expressed in Tab. A.2. The functions of kind $\texttt{Set}(\bullet)$ and $\texttt{Get}(\bullet)$ have complexity $O(1)$ because it is just an access to an item of a structure. It is assumed that the $\forall v \in V \mid \texttt{Out}(v) \ll |V|$ and for this reason $\texttt{Out}(\bullet)$ has complexity $O(1)$. The $\texttt{MayBeValid}(\bullet)$ function is just a simple test: if the Earley set is empty or not, so $O(1)$. Please refer to [FT87] for the complexity of the functions related to the heap manipulation as $\texttt{Min}(\bullet)$, $\texttt{Decrease}(\bullet)$, and tuple inclusion, whose complexity is $O(1)$. The n and m refer to the amount

Algorithm A.11: Standard Dijkstra algorithm with modified heap

Data: A weighted digraph $G(V, E)$
Data: a source vertex $o \in V$
Result: Shortest-Path spanning tree

```
   /* H is an initially empty Fibonacci heap */
 1 H ← {};
   /* initialisation:  insert the initial tuple, where ? means undefined */
 2 H ← (0.0, o, ?, ?, false);
   /* resets the vertex stored tuple */
 3 forall v ∈ V do
 4 |  SaveTuple(v, ?);
 5 end
   /* the main loop.  */
 6 while H ≠ {} do
   |  /* removes the entry with the lowest weight in H */
 7 |  u ← Min(H);
 8 |  Minimum(u, true);
 9 |  v ← Vertex(u);
10 |  if Tuple(v) =? then
11 |  |  SaveTuple(v, u);
12 |  |  forall e ∈ Out(v) do
   |  |  |  /* relaxation, see Algo. A.12 */
13 |  |  |  H ← Relax(H, u, Target(e), e);
14 |  |  end
15 |  end
16 end
```

Algorithm A.12: `Relax(•)`

Data: H tuple heap
Data: t the target vertex
Data: u the source tuple
Data: e the edge being evaluated that connects u to t
Result: H updated

```
   /* Relaxation function */
 1 m ← Tuple(t);
 2 w ← Weight(u) + EdgeWeight(e);
 3 if m ≠?∧ Minimum(m) = false∧ Weight(m) > w then
 4 |  Weight(m, w);
 5 |  Edge(m, e);
 6 |  Decrease(H, m, w);
 7 else
 8 |  H ← (w, t, u, e, false);
 9 end
10 return H;
```

Algorithm A.13: `ModRelax(•)`

Data: H tuple heap
Data: t the target vertex
Data: u the source tuple
Data: e the edge being evaluated that connects u to t
Result: H updated

```
/* Modified relaxation function.  */
```

```
1  m ← Tuple(t);
2  s ← Parse(State(u), GetLabel(e));
3  if MayBeValid(s) then
4  |  w ← Weight(u) + EdgeWeight(e);
5  else
6  |  w ← ∞;
7  end
8  if m ≠?∧ Minimum(m) = false∧ Weight(m) > w then
9  |  Weight(m, w);
10 |  Edge(m, e);
11 |  State(m, s);
12 |  Decrease(H, m, w);
13 else
14 |  H ← (w, t, u, e, false, s);
15 end
16 return H;
```

of vertices and edges of the graph and it is assumed that $m > n$. The edge data access functions `Weight(•)`, `GetLabel(•)` and `Target(•)` have complexity $O(1)$ because it is supposed to be an access to the data structure already present in the edge.

Table A.2: Function complexity table

Function	Complexity	Function	Complexity	Function	Complexity
`Minimum(•)`	$O(1)$	`GetLabel(•)`	$O(1)$	`Tuple(•)`	$O(1)$
`Weight(•)`	$O(1)$	`Target(•)`	$O(1)$	`MayBeValid(•)`	$O(1)$
`State(•)`	$O(1)$	`Out(•)`	$O(1)$	`Parse(•)`	$O(m^2)$
`Edge(•)`	$O(1)$	`Weight(•)`	$O(1)$	`Min(•)`	$O(log(n))$
`Vertex(•)`	$O(1)$	`SaveTuple(•)`	$O(1)$	`Decrease(•)`	$O(1)$

The complexity of the initialisation process in the Algo. A.11 is dominated by the tuple initialisation for all vertices (between lines 3 and 5, which is $O(n)$). Then for the modified relaxation function in Algo. A.13 the most time consuming operation is the `Parse(•)` function, whose complexity is $O(m^2)$. The **forall** loop (line 12 in Algo. A.11) combined with the **while** (line 6 in Algo. A.11) will run at most over all edges, this means that the `Relax(•)` function will be executed m times. Therefore the relaxation function has complexity $O(m^3)$. This allows the exclusion of the **forall** (line 12 in Algo. A.11) loop from the remainder complexity calculation. This simplified remaining **while** loop (line 6 in Algo. A.11) has its complexity dominated by the `Min(•)` function, which is executed for all vertices. This means that it contributes with $O(n*log(n))$, then summing up all terms leaves a final complexity of $O(m^3 + n * log(n))$, where $m = |E|$ and $n = |V|$.

This is however the theoretical complexity. This standard approach is not appropriated because it modifies the input data (graph and starting vertex). The modifications are related to the function `SaveTuple(•)`, which alters the vertex structure to include the corresponding tuple. This makes this algorithm not optimal for an environment where the Dijkstra is in constant use and may have

concurrent access. Therefore it is modified to let the input data intact. This requires an additional data structure to store the tuples corresponding to the vertices and it may imply in complexity increase.

The adopted data structure is a hash table [Knu98], which has an access time ranging from $O(1)$ to $O(n)$. For further information as also complexity proof please refer to [Knu98]. The complexity increase, that may occur, is not in the relaxation function from Algo. A.15 (which is still dominated by the `Parse`($\bullet$) function); but in the main **while** loop (line 4 in Algo. A.14). The complexity change may occur if the hash table degrades to $O(n)$ for insertion. This increases the simplified inner loop complexity from $O(log(n))$ (the complexity of `Min`($\bullet$)) to $O(n)$. For the worst case the final complexity will be $O(m^3 + n^2)$. To prevent this problem the hash table is initialised with a capacity much higher than the amount of vertices. The hash table resize is supposed not necessary as well (it is made beforehand). The complexity of the implemented Dijkstra is then $O(m^3 + n * log(n))$.

Algorithm A.14: Implemented Dijkstra algorithm

Data: A weighted digraph $G(V, E)$
Data: a source vertex $o \in V$
Data: a starting Earley state s
Result: Shortest-Path spanning tree

```
   /* H is an initially empty Fibonacci heap */
1  H ← {};
   /* initialisation:  insert the initial tuple, where ? means undefined */
   /* notice that the initial Earley state s is included in the tuple */
2  H ← (0.0, o, ?, ?, false, s);
   /* SPT is a hash table, which has a vertex as key and a tuple as value */
3  SPT ← {};
   /* the main loop.  */
4  while H ≠ {} do
       /* removes the entry with the lowest weight in H */
5      u ← Min(H);
6      Minimum(u, true);
7      v ← Vertex(u);
8      if Tuple(v) =? then
9          SPT ← u;
10         forall e ∈ Out(v) do
               /* see Algo. A.15 */
11             H ← ImpRelax(H, u, Target(e), e);
12         end
13     end
14 end
15 return SPT;
```

The complexity contribution of the relaxation function is said to be $O(m^3)$ but not completely discussed. The interpretation is that each edge, adopting the super-network, has exactly one label. On the worst case all edges are "parsed", i.e. their terminals/labels are parsed. In this situation the parsing process will have the complexity of parsing a word with size m and therefore $O(m^3)$.

What was not explained is that the final result is the table SPT associated with the graph G and source vertex o. With SPT is possible to reconstruct any path, remember that in the tuple the leading edge is stored. The validity criterion was yet left open because just the "may be valid" state is tested by the Dijkstra. The validity is implicit on the grammar and on the super-network structure. Recall that all paths must start and end on the access layer and the special edges from and to this layer have special labels. These labels are latter used to "encapsulate" the grammar

Algorithm A.15: `ImpRelax(•)`

Data: H tuple heap
Data: t the target vertex
Data: u the source tuple
Data: e the edge being evaluated that connects u to t
Result: H updated

```
/* Implemented relaxation function */
/* the entry is NOT removed from the hash table */
```
$m \leftarrow$ `Get` (SPT, t);
$s \leftarrow$ `Parse(State(`u`), GetLabel(`e`))`;
`/* Algo. A.16 */`
$w \leftarrow$ `ModifiedEdgeWeight(`s, u, e`)`;
`/* Minimum(`m`)` $= false$ `means that the entry still exists in` H`, i.e. it was not removed by Min() and the functions inside the if change the entry in` H `and` SPT `as well */`
if $m \neq ?\wedge$ `Minimum(`m`)` $= false \wedge$ `Weight(`m`)` $> w$ **then**
  `Weight(`m, w`)`;
  `Edge(`m, e`)`;
  `State(`m, s`)`;
  `Decrease(`H, m, w`)`;
else
  $H \leftarrow (w, t, u, e, false, s)$;
end
return H;

Algorithm A.16: `ModifiedEdgeWeight(•)`

Data: s resulting Earley state from parsing the edge e
Data: u the source tuple
Data: e the edge to be evaluated
Result: modified weight for the edge e

```
/* Modified grammar aware edge weighting function */
```
$w \leftarrow 0$;
if `MayBeValid(`s`)` $= true$ **then**
  $w \leftarrow$ `Weight(`u`) + EdgeWeight(`e`)`;
else
  $w \leftarrow \infty$;
end
return w;

start symbol. Then just valid paths reach the access layer and "may be valid" states that are also "invalid" will never be "valid" by adding the final edge (back to the access layer). This guarantees that paths searched within the access layer using the implemented Dijkstra are always grammatically correct, if they exist.

A.5 Dijkstra Based Path Generators

In the previous section a modified version of the Dijkstra algorithm was presented that incorporates the Earley parser. A shortest-path (SP) algorithm can nevertheless give only one path for a given OD pair. To overcome this shortage in paths some algorithms were proposed and among them are: k-shortest-path in [Epp98, JM03] or in [EH99, HMS03, HSB07] (k-SP); Dial in [Dia71]; Constrained k-SP in [vdZC05]; SP with several metrics in [BBAR06]; Way-finding in [NCK+06] or in [BS90, RW02, Hoc05], which is a SP with a different metric. All algorithms cited above are based on a SP algorithm to generate their multitude of paths. For some of them the use of the modified Dijkstra is transparent, i.e. no additional change in the algorithm in necessary. This is the case of SP with several metrics and Way-finding. For the others the algorithm changes are necessary to take into account the use of the grammar. For example the Dial algorithm needs the inverse of the grammar (it can be automatic generated) to calculate reverse shortest-paths (from a destination to an origin).

The most simple of these multi-path generators are the ones using different metrics. These are followed by the Dial, which was implemented but showed itself too time consuming for a realistic use and therefore not adopted. The algorithms based on the k-SP are even more complex and have a major drawback: they generate paths that are too similar to each other. This would require the generation of many paths and to discard some of them (implying in a post-generation filtering procedure), which increases even more the complexity.

The drawbacks of the multi-metric SP is that the amount of paths generated are restricted to the amount of metrics used. Therefore the adopted algorithm is a Link-Elimination Shortest-Path (LESP) that is simple and generates several paths on-demand. This algorithm is based on [ACaERSMM93] but here just one link is eliminated by each run.

The basic algorithm is in Algo. A.17. There, the function `Dijkstra`(•) (Algo. A.14) is modified to returns only the shortest-path between o and d. Besides this, it also checks if the edge being evaluated is in W (the last argument that works as a taboo list) and if it is then it invalidates the edge (`EdgeWeight`$(e) = \infty$). The complexity is not affected by the use of W because the inclusion has complexity ranging from $O(1)$ to $O(m)$ if implemented as a hash-table. Then the path generation is still dominated by the `Dijkstra`(•) call. The function `Random`(•) has also no impact because its complexity is at most $O(m)$ – it selects an edge among the edges in the already calculated paths, excluding the ones in W. The last issue is the query of W inside the function `ModifiedEdgeWeight`(•) (Algo. A.16).

This complexity could degenerate to $O(m)$ for consulting W, therefore the complexity of `ModifiedEdgeWeight`(•) increases from $O(1)$ to $O(n)$. However, this is a sub-step in the `Relax`(•) (Algo. A.15) that is still dominated by the `Parse`(•) function (whose complexity is $O(m^2)$), therefore no complexity increase is made by the inclusion of these extra steps. The final complexity of the LESP algorithm is $kO(m^3 + n\ log(n))$, being: k the amount of paths desired; m the amount of edges in the super-network; and n the amount of vertices.

Algorithm A.17: Link-Elimination Shortest-Path

Data: A weighted digraph $G(V, E)$
Data: source vertex $o \in V$
Data: target vertex $d \in V$
Data: k the amount of paths
Result: path set P

```
1 P ← {} ; /* path set */
2 W ← {} ; /* an edge set */
3 forall i ← 1 to k do
4 |   sp ← Dijkstra(G, o, d, W) ; /* generates a new path */
5 |   P ← P ∪ {sp};
  |   /* selects a random edge among all edges in P, excluding the edges
  |      present in W */
6 |   e ← Random(E_P, W);
7 |   W ← W ∪ {e};
8 end
9 return P;
```

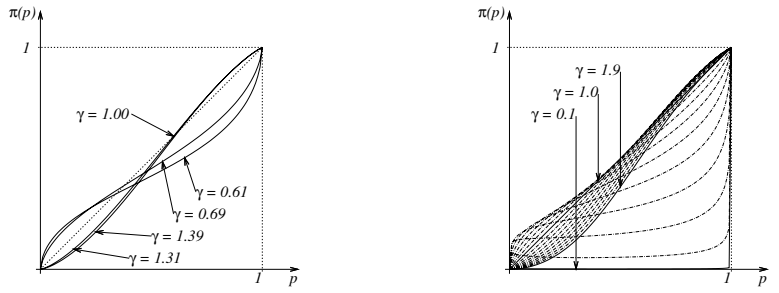
Appendix B

Extra Figures

In this appendix are included the graphics that would be too cumbersome to be included in the main text or that give extra understanding but are not closely related with the text in question.

B.1 Prospect Theory $\pi(\bullet)$ Function

In Fig. B.1 are depicted the behaviour of the $\pi(\bullet)$ function with different γ parameters, as mentioned in Sec. 8.11.



(a) Function $\pi(\bullet)$ for the used γ values

(b) Function $\pi(\bullet)$ for several γ values

Figure B.1: Function $\pi(\bullet)$ for different γ values

B.2 Exponential Learning Parameter α_0 Comparison

Here the comparison between two different initial exponential learning parameter α_0 is depicted. The Fig. B.2 shows the α_n evolution for $1,000,000$ steps, has commented in Sec. 5.4.1.

B.3 Richard's Function With Different Parameters

It was mentioned in Sec. 5.4.1 that the parameters in the Richard's function (Eq. 5.5) are flexible. Therefore, in Fig. B.3 this function is depicted varying each of the parameters: Q, B, M, and ν.

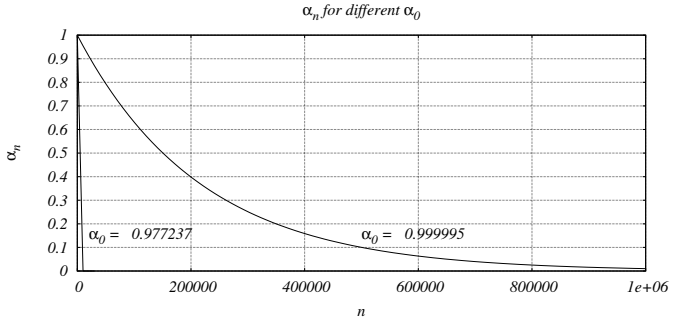
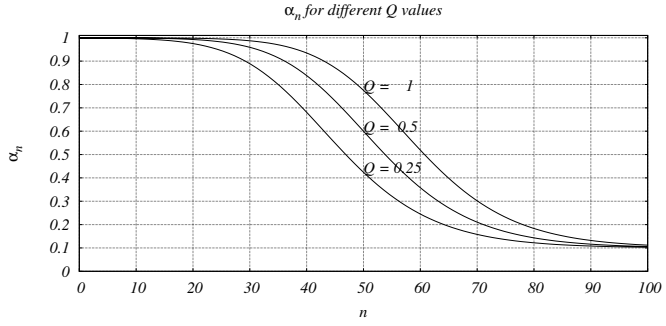
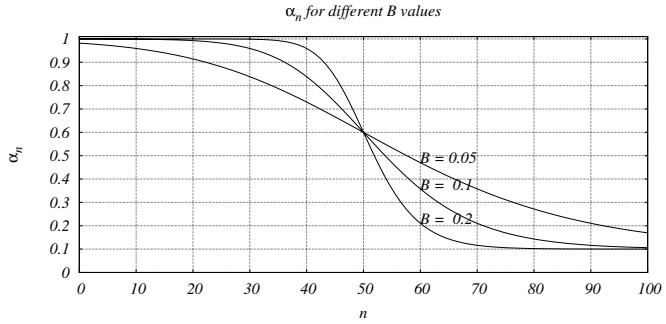


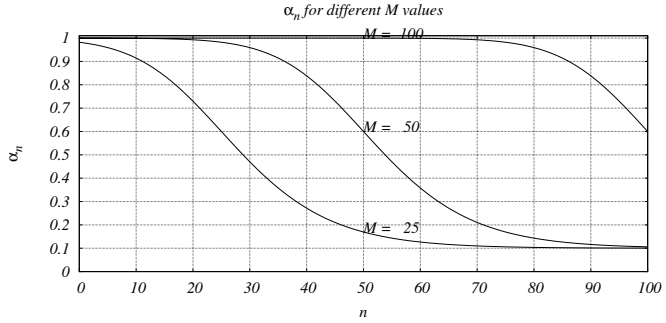
Figure B.2: Exponential $\alpha(i)$ for a horizon of 1000000



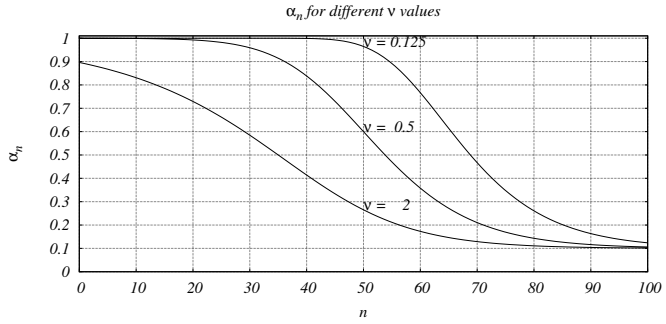
(a) Different Q values



(b) Different B values



(c) Different M values



(d) Different ν values

Figure B.3: Richards' function with different parameter value