Modelling individual routing of pedestrians as a utility maximization process using inverse reinforcement learning

van Hees, Stijn P.A.

Award date:
2022

Modelling individual routing of pedestrians as a utility maximization process using inverse reinforcement learning

S.P.A. van Hees

Student nr. 1022047
s.p.a.v.hees@student.tue.nl

TU/e supervisors:
Prof. dr. F. Toschi
dr. A. Corbetta
dr. A. Gabbana

DEPARTMENT OF APPLIED PHYSICS
CROWDFLOW RESEARCH GROUP

Eindhoven University of Technology
The Netherlands
July 22nd, 2022
R-2124-A

Abstract

Despite the long-standing interest in the motion of crowds, the problem of understanding its underlying dynamics is far from being solved. Prominent models use the concept that individuals minimize their travel time, but avoid highly crowded regions. An extension of this idea describes a cost function that pedestrians minimize along their trajectory. In this thesis we consider trajectories of pedestrians on a platform on the Eindhoven railway station. We investigate whether a cost function can be constructed from the utilities on the station in an intuitive way, so that the function gives insight by itself.

The platform is spatially discretized into a grid. The reward of a tile is presumed to depend on how many pedestrians occupy it and whether it is located at points of interest on the station, such as the escalator, stairs or boarding zone. We set out to quantify this dependency with maximum entropy reinforcement learning.

When considering a single person's trajectory, we found it was practical to only consider the environment up to a certain range from this person instead of the whole station. Long-range effects that stem from objects outside this range are accounted for by adding proxies of them in this smaller environment. The reward on a tile then also depends on the proxies present on it.

It was found that the tiles occupied by one pedestrian have lower reward on average than squares occupied by two or three. We theorize that for these utilities, information about the direction of the velocity is needed to further classify them. Tiles in the boarding zone were found to consistently have low reward. Between the escalator and the stairs, the reward was found approximately the same.

Trajectories across the platform are generated with a heuristically found reward that is similar to the obtained reward. The trajectories are similar in broad lines to real trajectories, but consist of straight segments instead of diagonals. Suggestions to improve the generation of these trajectories are given. To further increase accuracy and reliability, we suggest exploring inverse reinforcement learning frameworks that allow for non-linear cost functions.

Contents

Chapter 1

Introduction

1.1 Scope

Not only is our world population growing, our society is also becoming faster: the demand for quick access to services is on the rise, as illustrated by the booming industry of fast delivery services. When time margins become slimmer and places more crowded, it is up to public facilities to withstand the growing strain. Whether they can do so is not so clear: anyone who has been at a crowded event, be it a festival, an airport, or a train station knows how inefficient it already is to navigate through large groups of people. In the most dire scenarios, an uncoordinated crowd can even be dangerous; this is by no means a phenomenon of the past, as is illustrated by modern occurrences such as last year's tragic Astroworld incident [1]. In a faster society and a world with a growing population, it is vital to streamline public facilities to not only maintain efficiency, but also to guarantee the safety of their users.

In order to prescribe streamlining policies for crowded facilities, we believe we must first understand the underlying dynamics of large groups of people. Due to their inherent lack of coordination and the mixture of objectives among individuals within these large groups, though, the dynamics of such groups are intrinsically complex. The movement and behavior of crowds is commonly modelled as one would a complex fluid. Still, in spite of significant advancements in computer vision allowing accurate measurements of crowd motion, the underlying dynamics remain largely not understood, and this is further exacerbated by the innumerable variety in geometry.

1.2 State-of-the-art

1.2.1 Crowd as a fluid

Pedestrians in large groups have been long known to exhibit properties of fluids. As such, the modelling of naturally occurring flows of crowds has long been inspired by existing fluid models [2]. In addition, in numerous instances of pedestrian streams, self-organizing phenomena have been shown to occur [3], [4]. Prediction models based on behavioral heuristics have been extended to fluid dynamics, resulting in the development of macroscopic kinetic models that, at the basis, follow an optimization of velocity, constrained to the avoiding of collisions [5]. Still, a generalized, scalable model has not been developed yet, and doing so has been a prominent challenge in the field.

A common approach to model crowd dynamics comes from the minimization of the travel time of individual pedestrians. In 2003, a theoretical model prescribing basic governing equations for crowds as a continuum has been proposed by Hughes [6]. At the basis is the continuity equation representing the conservation of pedestrians. To form a closed governing equation, the report proceeds to prescribe equations supported by simple and reasonable hypotheses drawn from observations, the most important of which is that pedestrians aim to minimize travel time, but modulate this behavior to avoid high densities. In technical terms, the travel time is defined over the space as a function of the density, and pedestrians move along the negative gradient of this field.

In 2004, Hoogendoorn and Bovy [7] formulated a new approach to the modelling of route choice of individual pedestrians. At the basis lay the behavioral assumption that in deciding their route, pedestrians make a trade-off between the utility that they gain from performing activities at specific locations against the perceived cost of the route along these locations. This cost is an extended model of Hughes' travel time: not only the density field is factored in, but also other sources of discomfort from factors such as walking close to walls. A total cost function is built from these factors, and the pedestrians, again, walk along the negative gradient of this scalar field. In the report in question, such a scalar field built from cost functions was performed on the Schiphol Plaza, a mall on the eponymous airport. In Figure 1.1, the idea is represented.

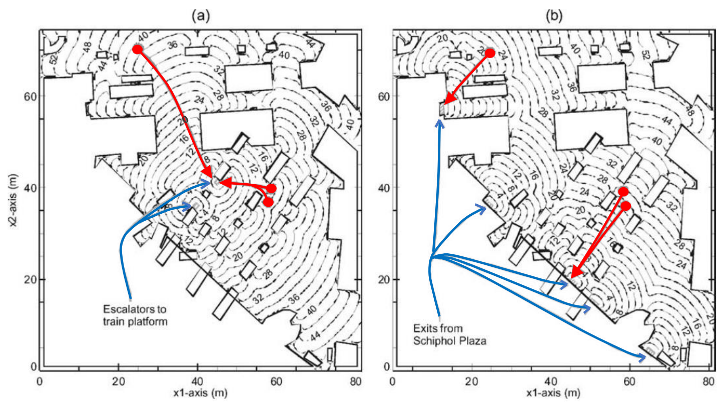


Figure 1.1: Expected minimum cost function across Schiphol Plaza for (a) leaving via the escalators and (b) leaving via either of the five exits. At any point, the discomfort of the route to the exits (in seconds) ensuing from optimal choices is given. Example routes are given in red. Reprinted from Hoogendoorn and Bovy, 2004 [7] (within STM permission guidelines; colors manually added).

Later, the concept of the minimization of such a cost function was used to program AI to walk in a simple environment [8]. In this case, the cost function is a function over a set of states that ascribes a 'cost' to being in the respective state. With this in mind, it could prove worthwhile to model individual routing of real pedestrians in a similar manner. If successful, insight into the very dynamics of pedestrian motion could be extracted from such a cost function itself. While a relatively simple effort-distance balance suffices to train AI in simple environments, a cost function similar to Hoogendoorn and Bovy's model that accurately describes crowd flow in more complex geometries could prove highly complicated.

1.2.2 Machine learning approaches

An emerging set of tools to help find a cost function for a complex system comes from the field of machine learning. Machine learning approaches have been attempted with various success rates to solve problems in chaotic systems, such as predicting the financial market [9], forecasting the weather [10], and have even been proposed to help streamline operations in healthcare [11]. When systems are of such high complexity that existing physical models fall short, machine learning provides a way to get around this complexity and still make predictions. Relating to our problem, machine learning techniques have been used to predict pedestrian trajectories in an office environment by way of learning a cost function [12].

1.2.3 Tracking pedestrian movement

In the last few years, the quality of tracking devices has seen a significant uptick in terms of both resolution and accuracy. Benchmark tests on modern commercial depth sensors such as Xovis^{TM} report that close to 100% accuracy in determining the number of pedestrians in their frame, with their locations being accurate up to a 4.4 cm standard deviation. Full benchmark test results can be found in the report by Pouw et al[13]. The ability to measure trajectories up to high accuracy opens the possibility of employing machine learning.

1.3 Research question

The mentioned approaches of learning a cost function had the goal of making accurate predictions in mind. So while learning a cost function is helpful for that goal, it is not necessarily helpful to understand the underlying dynamics of crowds. This understanding must be provided by the cost function itself, so we must operate under the constraint that the cost function that we would like to learn be intuitive, i.e. the cost function must provide insight by itself.

With the mentioned objective in mind, we want to pose individual routing of pedestrians as an optimization problem: conversely to the cost function, we may suppose there is a certain quantity that people individually *maximize* when moving through crowds. The goal, then, would be to see if we can reverse engineer this quantity. For this reason, we looked for machine learning methods that seemed compatible with our task of reverse engineering the utility maximization process that people solve while moving through crowds.

In short, we want to investigate whether a quantity exists that is built from the utilities within a system of pedestrian dynamics that accurately predicts pedestrian trajectories. This brings about the research question:

Can the individual routing of people in crowded environments be modelled as a utility maximization problem?

1.4 Outline

In Chapter 2, the means to answer the problem are presented, as well as their technical implementation. The reason why these tools were chosen is also given. Then, in Chapter 3 a simplistic toy problem about a binary choice scenario is presented to investigate the validity of the proposed implementation, and then we attempt to answer the real-life parallel to this toy problem. The results of both the toy problem and the real-life implementation are summarized and discussed.

Thereafter, in Chapter 4 a more complex toy problem about a pedestrian walking to an exit through a crowded environment is presented and a new approach to the same framework is proposed to solve it. The outcome is summarized and discussed.

In Chapter 5, data of the pedestrian trajectories of people on a platform of the Eindhoven railway station is presented, and we attempt to combine the previous approaches to understand the dynamics of the pedestrians. We lay out our findings and discuss them.

In the end, the results are summarized in Chapter 6 and a general conclusion is drawn. The main points of discussion are summarized and suggestions for future work are given.

Chapter 2

Scientific and technical background

In this chapter, we first lay out the similarity from the field of physics to the cost function appraoch. We then consider machine learning tools, and point to the option that we find most suited for our goal. An illustrative scenario is outlined in which the necessity and core idea of the proposed method is explained. Next, some fundamental frameworks and their applications are laid out, and common algorithms are given.

2.1 Cost function

In formulating a utility maximization approach, we can borrow from the concept of the action in Lagrangian mechanics, which is a functional defined as

$$S[L, t_0, t_f] = \int_{t_0}^{t_f} L\left(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), t\right)\ \mathrm{d}t, \tag{2.1}$$

with L being the Lagrangian of the system. The equations of motion then follow from the requirement that $\delta S = 0$ along a trajectory in the configuration space: the action is a local minimum. Our suggestion is to consider a similar functional. Similar to trajectories emerging from the requirement that the action be a local minimum, for this functional, it is assumed that emerging trajectories maximize this functional. Precisely, in a configuration space described by the generalized coordinates $\boldsymbol{x} = \{x_0, ..., x_N\}$, trajectories of the form $\zeta = \{\boldsymbol{x}(t_0), ..., \boldsymbol{x}(t_f)\}$ emerge from the maximization of

$$V[R, t_0, t_f] = \int_{t_0}^{t_f} R\left(\boldsymbol{x}(t)\right)\ \mathrm{d}t, \tag{2.2}$$

where, instead of a cost function representing the *cost* of a state $\boldsymbol{x}$, $R(\boldsymbol{x})$ is the opposite; say it gives the *reward* of the state, making it a reward function.

The act of finding globally optimal trajectories or distributions in stochastic systems using cost minimization methods has been employed in various applications [14]–[18], from vehicle control during rally driving [17] to the dynamics of arm movements [18], and a utility maximization approach seems like a logical extension. Instead of finding globally optimal trajectories given a cost functional, we can take a more passive role and do the inverse: we can use real trajectories obtained from measurements to find the cost function, or equivalently, the reward function instead.

2.2 Machine learning tools

When one has access to sufficiently many demonstrations within a system, such as trajectories, one can leverage them to bypass the complexities of the system through machine learning. The most common machine learning tool that operates this way is the artificial neural network, which is an input-output device whose structure is complex; for all intents and purposes, a neural network is a black box; its usefulness lies in its ability to predict outcomes of very complex processes, and the reason it is able to do this is because neural networks are trained by real input-output pairs. We could model a reward function as a neural network to get around the inherent complexity of the dynamics: we can carry out measurements and train the neural network with them. Let us imagine we do this, then after enough training, the neural network will provide us with very accurate predictions of the positions and instantaneous velocities, perhaps also even the accelerations of all people in any crowd of our choice. At that point, however, it is important to realize that we set out to *understand* the underlying dynamics, not to emulate them. Being a black box, a neural network fundamentally provides no insight. Since we want to find an intuitive reward function that by itself provides insight to crowd dynamics, a neural network is unfit for our outline.

A method that is fit for our outline of reverse engineering a reward function is Inverse Reinforcement Learning (IRL). This is another machine learning method that sets out to find a reward function that is optimal when evaluated along observed behavior. By letting this reward function be parameterized by the utilities of a system, we can employ IRL to help us answer the utility maximization problem.

2.3 Introductory example

The simplest example of a utility maximization process is a system of two paths of different lengths $\{L_A, L_B\}$ and different occupations $\{N_A, N_B\}$ that go to the same destination: the utilities at play are path length and path occupation.

Now, suppose we want to train an AI to make sensible, human-like choices in this system. Without thinking twice, we may make the AI prefer short and unoccupied paths. As such, without knowing anything about the exact lengths or the occupation levels of the paths, we may a priori prescribe it to make choices according to Table 2.1.

	$N_A = N_B$	$N_A > N_B$	$N_A < N_B$
$L_A = L_B$	50/50	B	A
$L_A > L_B$	B	B	undetermined
$L_A < L_B$	A	undetermined	A

Table 2.1: Decision table for the two route system.

Indeed, there are two scenarios in which the choice is unclear. The human choice would be to choose the path that has the lowest travel time. A generalized way to couple the travel time to the occupation and length of a path is the following:

$$t(L, N) = \frac{L}{v(N)}.$$

Given L and N, the choice now depends on the function $v(N)$, so we need this function to be accurate for human behavior.

We set out to use IRL to find this function. To do so, we have to first define a utility. A straightforward utility would be the inverse of the travel time. We might then consider the quantity

$$R(L, N) = \frac{1}{t(L, N)} = \frac{v(N)}{L},$$

which will be referred to as the *reward* of choosing a path with length L and occupation N. The term 'reward' is interchangeable with 'combined utility' throughout this report. The next step is to make an estimate for $v(N)$, so we may use $v(N) = \alpha_1 - \alpha_2 N$. Here, α_1 and α_2 are constants. This gives

$$R(L, N) = \alpha_1 \frac{1}{L} - \alpha_2 \frac{N}{L}.$$

Notice that the reward increases with both decreasing route length L and occupation level N. Now, the choice can be prescribed by picking the route whose length and occupation yield the highest reward. Of course, the values of α_1 and α_2 entirely dictate which route is chosen, so we need to somehow find good values for them.

IRL uses real observations to determine the constants. By looking at what people do in the described scenario, we can infer what range of possible values for α_1 and α_2 might have brought them to their respective decisions. Within this range, we pick an α_1 and α_2 of our liking[1] to obtain the reward function. Finding a reward function that explains observed behavior is the core outset of IRL. The method hinges on the assumption that demonstrated behavior follows a maximization of the reward.

Equipped with this reward function, one can present the AI with scenarios and let it make choices. We can then judge whether these choices are similar to what a real human would do. If there is a clear mismatch, one may amend the reward function to include more terms. I.e., it may occur that the AI's choices do not pass the eye test with the current obtained values for α_1 and α_2, so then we have reason to believe that there is actually another utility at play. For example, we suspect that next to the the inverse of the travel time, the pedestrian also considers the path width W. We then modify the reward function to include the path width W of the paths:

$$R(L, N, W) = \alpha_1 \frac{1}{L} - \alpha_2 \frac{N}{L} + \alpha_3 W.$$

We would then proceed to find α_1, α_2 and α_3 and let the AI make choices with the new reward. This process can be repeated until we are satisfied with the AI's behavior. When the AI shows behavior that is indistinguishable from real people, the reward function tells us which utilities are taken into account in peoples' decisions, and the respective α's tell us the relative proportionality of these utilities. For example, if the function above with the obtained coefficients can accurately mimic human choices, we know that the model for the velocity $v(N)$ was accurate given that the path width W was also taken in to account.

2.4 Frameworks

IRL frameworks can be model-based or model-free. A model-based framework is used when there is full knowledge of the dynamics of the system, which allows for evaluation of a strategy because it is exactly known what will happen if this strategy is carried out. In model-free frameworks, demonstrations are used to learn the strategy directly.

2.4.1 Markov decision processes

Markov decision processes, also called Markov Decision Problems (MDPs) form a well studied class of model-based frameworks. Any problem that can be described as an MDP lends itself to IRL. An MDP is a stochastic control process described by

- The set of **states** $S = \{s_1, ..., s_N\}$;

[1]In practice, algorithms are used to pick 'the best' combination of these coefficients, this is elaborated on in section 2.6.

- The set of **actions** $A = \{a_1, ..., a_k\}$;
- The **state transition functions** $\{P_a(s, s')\}$ giving the probability of moving from state s to state s' with action a. A state transition function is usually represented by an $N \times N$-shaped stochastic matrix;
- The **reward** $R(s)$ serving as a motivation to propagate the decision making process. This may be trivially extended to include the actions in the manner of $R(s, a)$;
- The **discount factor** $\delta \in [0, 1)$ which is a measure of how much emphasis is put on immediate reward in comparison to future reward. This value is typically 0.9 or higher.

Further, a **policy** π is a mapping of the states onto the actions. Then, starting from any state $s = s_0$, we can create an infinite sequence of states $(s_0, s_1, s_2, ...)$ by taking the actions prescribed by the policy. A policy can also be stochastic, which is to say that it allows for stochastic selection of actions: instead of prescribing an action for each state, it gives for each state the probabilities of the actions being chosen. Either way, the **value** of a state under a policy, denoted by $V^\pi(s)$, can then be evaluated by considering the reward on the state itself, as well as the expected future-discounted rewards on the states that occur later in the sequence. Te be precise:

$$V^\pi(s_0) = \mathbb{E}\left[\sum_{i=0}^{\infty} \delta^i R(s_i) | \pi\right], \tag{2.3}$$

which always converges since $\delta < 1$ by definition. A policy is considered *optimal* if it simultaneously maximizes $V^\pi(s)$ for all s. Such a policy is denoted by π^*. Succinctly, for any policy π,

$$V^{\pi^*}(s) \geq V^\pi(s) \;\; \forall \;\; s \in S. \tag{2.4}$$

This equation is the centerpiece of IRL, as observations are assumed to exhibit an optimal policy. Also, notice the parallel between above two equations and the description of Equation 2.2.

Now that we understand the value of policies, we can talk about applying IRL to the given MDP. The first ingredient is the assumption that observed behavior displays an expert policy, which is another way to say that we would like to learn the decision making strategy from real data. Secondly, it is postulated that the reward function $R(s)$ is a linear combination of utility functions, or *features*:

$$\begin{aligned} R(s) &= \alpha_1 \phi_1(s) + \alpha_2 \phi_2(s) + ... + \alpha_N \phi_N(s) \\ &= \boldsymbol{\alpha}^T \boldsymbol{\phi}(s) \end{aligned} \tag{2.5}$$

where $\alpha_1, ..., \alpha_N$ are free constants. We can choose these features ϕ_i ourselves. Then, upon inferring π^* from the observations, we can obtain the coefficients $\alpha_1, ..., \alpha_N$ so that Equation 2.4 holds.

2.5 Objectworld

A general idea to describe motion around a physical environment as an MDP is to discretize the space into a grid. On the tiles of this grid, objects can be placed that might or might not influence the pedestrian. Consider the square space subdivided into an $N \times N$ grid with an actor moving through it. The states are described by the x, y-coordinates of the tiles in the grid, and the actions in this MDP are to move up, move down, move left, move right, and remain in place. Levine et al. [19] used the description of an *objectworld*, where there are objects placed around this grid on random tiles. In their description, the objects had an inner and outer color, both chosen from an ensemble of C colors. There are $2C$ features, each being the Euclidean distance to the nearest object with a specific inner or outer color:

$$\phi_{\text{inner } i}(s) = \text{dist}(s,\ \text{nearest object of inner color } i), \tag{2.6}$$

and, mutatis mutandis, the same for the inner colors.

When one has access to trajectories in this grid, one can solve the reward that is parameterized by the mentioned features. When a reward is obtained, the coefficients of the features will then reveal whether the different inner and outer colors each had a repulsive or attractive effect on the pedestrians. To see that a repulsive effect corresponds to a positive constant and vice versa, one needs to realize that increasing the distance to an object also increases the reward if the base function corresponding to that object has a positive coefficient.

2.6 Solving principles

Given any MDP, the problem of finding a reward function that upholds Equation 2.4 has many degenerate solutions [20], among which the trivial zero reward: $R(s) = 0$. To resolve this ambiguity, there exists a multitude of principles to select one of these solutions, each based on a different rule.

2.6.1 Maximum difference

A principle proposed by Ng et al. [20] that infers a policy from actual trajectories is what we refer to as the **maximum difference** algorithm. At the basis of it are expert demonstrations on the one hand and nonexpert demonstrations on the other. For example, Muelling et al. [21] used one skilled table tennis player and a group of five novices. From these demonstrations, a presumed expert policy π^* and a set of nonexpert policies $\{\pi_1, ...\pi_5\}$ were inferred. The point is to find the reward that leads to a highest possible difference in value functions V^{π^*} and V^{π_i} simultaneously for $i = 1, .., 5$. More precisely, one looks for the set of coefficients $\boldsymbol{\alpha}^*$ such that

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \sum_i p_m \left(V^{\pi^*}(s_0) - V^{\pi_i}(s_0) \right) \tag{2.7}$$

where

$$p_m(x) = \begin{cases} mx & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

is a penalty function that deters a violation of Equation 2.4. Indeed, the higher m, the larger the penalty for a violation. Typically $m = 2$ is enough to converge to satisfactory solutions [20].

When there is no access to nonexpert demonstrations, one can make these demonstrations by modifying the expert policy. Since the idea behind the max-difference algorithm is to make single-step deviations from the observed policy as costly as possible, this is done by changing the policy at just one state.

A typical maximum difference algorithm first infers the assumed optimal policy π^* from observed demonstrations, then creates the set $\{\pi_1, ...\pi_k\}$ of k nonexpert policies by one-step deviations from the optimal policy. Then, it calculates the reward with Equation 2.7. With that, new policy π_{k+1} is found that is optimal under the calculated reward and added to the set of nonexpert policies, after which the reward is updated. This process repeats until a satisfactory reward is converged on.

2.6.2 Maximum entropy

Unlike the previous principle, the **maximum entropy** or MAXENT principle first suggested by Jaynes [22] and implemented in IRL by Ziebart et al. [23] proposes to look for the solution that out of all possible rewards has the highest information entropy. In other words, the actor aims to maximize the expected reward by acting as randomly as possible.

The parallel with statistical thermodynamics is clear: in Boltzmann's formulation of the entropy of a system, it is given by $S = k_B \ln \Omega$; it scales logarithmically with the number of microstates in a system Ω. Maximizing the entropy then entails finding a macrostate of the system that comprises the highest possible number of microstates. For example, in the system of 100 flipped coins, maximizing the entropy means looking for the combination of total heads and total tails that allows for the highest possible number of ways these totals could have come about. The lowest entropy is the case of 100 heads or 100 tails as there is only one way to achieve these, so we already fully know the microstate. Meanwhile, the highest entropy is the case of 50 heads and 50 tails, as there are $\binom{100}{50} \approx 10^{29}$ ways to get this. Then, despite knowing the macrostate, we know practically nothing about the microstate that brought it about. In this sense, the number of microstates Ω represents the lack of knowledge of the system. By extension, the maximum entropy state then makes the least number of assumptions about the system, which is the thrust behind the principle.

The basic Maximum entropy reinforcement learning description concerns solving a Markov Decision Problem, making it model-based. Although modern model-free maximum entropy IRL models have been developed [24], we will only consider the model-based description in this paper.

Pivotal in the MAXENT description is the feature count of a trajectory. We remind the reader that features are the base functions that parameterize the reward and are represented by the vector $\boldsymbol{\phi}(s)$, see Equation 2.5. A trajectory $\zeta = (s_0, ..., s_f)$ has total feature count $\boldsymbol{\phi}_\zeta = \sum_{s_i \in \zeta} \boldsymbol{\phi}(s_i)$. Assuming access to a number m of demonstrated trajectories $\{\zeta_1, ..., \zeta_m\}$ through the state space, the algorithm considers the expected accumulated feature counts along these trajectories, i.e. the *feature expectations*:

$$\tilde{\boldsymbol{\phi}} = \frac{1}{m} \sum_{j=0}^{m} \boldsymbol{\phi}_{\zeta_j} = \frac{1}{m} \sum_{j=0}^{m} \sum_{s_i \in \zeta_j}^{f} \boldsymbol{\phi}(s_i). \tag{2.8}$$

On the one hand, there are the real trajectories, and on the other hand there is the learner, whose behavior we would like to match with the real trajectories. In MAXENT, this goal is prescribed as merely matching the feature expectations of the learner's trajectories with those of the real trajectories:

$$\sum_{\text{Path } \zeta_i} P(\zeta_i) \boldsymbol{\phi}_{\zeta_i} = \tilde{\boldsymbol{\phi}}_i \tag{2.9}$$

which implies that we need a way to calculate the probability of any trajectory ζ_i. In order to do so, MAXENT models the distribution of trajectories as a function of the feature coefficients $\boldsymbol{\alpha}$:

$$P(\zeta_i | \boldsymbol{\alpha}) = \frac{1}{Z(\boldsymbol{\alpha})} e^{\boldsymbol{\alpha}^T \boldsymbol{\phi}_{\zeta_i}} \tag{2.10}$$

and it can already be inferred that trajectories with high cumulative reward are the most common, though trajectories with lower reward still have a probability of occurring.

Now, like the maximum difference algorithm, there are multiple distributions that satisfy the matching of feature counts (Equation 2.9). To resolve this ambiguity, MAXENT chooses the feature coefficients that maximize the entropy of the distribution; that is, it looks for

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}}{\text{argmax}} \sum_{\text{examples}} \log P(\tilde{\zeta} | \boldsymbol{\alpha}). \tag{2.11}$$

This function is convex and can be optimized by taking the difference in feature counts as gradient; these feature counts are obtained by calculating the expected visitation frequency of each state under the policy of the current iteration. A general algorithm that implements this principle is given in Ziebart et al.'s paper [23].

A great advantage of MAXENT is that per Equation 2.10, emerging trajectories can still be suboptimal with respect to the current iteration of the reward function. Therefore, if in observed data, such supposedly suboptimal trajectories occur, the algorithm finds this acceptable and does not need to majorly change the coefficients of the features. This makes MAXENT robust to statistical noise in observations, as is shown in [25], where the reward was modelled as a linear combination of features plus actor-specific statistical noise. These properties make MAXENT today's preferred framework for most IRL applications.

Chapter 3

Binary choice

In the simplest case, utility maximization involves a binary choice. Maximizing the utility then simply means choosing the option that, out of the two, has a higher utility. An instructive example of a binary choice scenario with people balancing the utilities of path length and path occupation is the Glow 2019 setting, of which the routing choices of pedestrians has been studied before [26], [27]. Glow is a yearly event in which people walk through the city of Eindhoven along a route decorated with light exhibitions. At a certain point, the route went under the Philips Stadium, where a pillar split up the route into two paths, one being shorter than the other. See Figure 3.1.

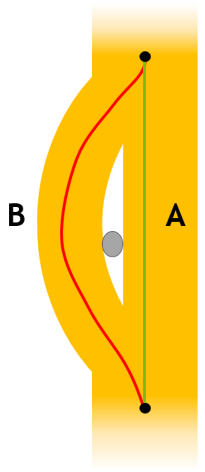


Figure 3.1: Visualization of the Glow scenario. People enter from below and are subjected to a binary choice between path A or path B. Typical trajectories along path A and path B are respectively drawn in green and red.

With no people crowding either of the paths, the most sensible choice is to take path A, as it is shorter. However, as more people occupied path A, path B comparatively became a more attractive choice until enough of the crowd in path A had passed by.

In this chapter, we first present a toy problem based on this event before considering real measurements. In this toy problem, artificial observations are created according to a ground truth reward, and it is investigated how accurately this reward can be reobtained through inverse reinforcement learning. The second part of this chapter involves performing IRL on real observations that were carried out by A. Gabbana, F. Toschi, P. Ross, A. Haans and A. Corbetta [26].

3.1 System description

Consider a person walking at a constant speed V_0 encountering the Glow scenario where they must choose between path A and path B. Let the length of path A and B be respectively L_A and L_B. Upon arrival, the person sees the number of people in either path, denoted as N_A and N_B. The person considers the combined utility of each path and makes their choice, assumedly the path with the highest reward. An observation in this system would entail seeing the velocity of the actor, the lengths and occupation of the paths, and the choice of the actor given these factors.

An MDP description is redundant due to the simplistic nature of this system; we simply need to prescribe how the reward for path A and path B is constructed. The utilities at play are clearly V_0, N and L, so we are looking for

$$R(V_0, N, L) = \sum_i \alpha_i \phi_i(V_0, N, L). \tag{3.1}$$

In addition, the notion of a value of a policy in this system is redundant; it suffices to say that a pedestrian takes the path with the highest reward.

3.2 Toy problem

To perform IRL, we need to evaluate the value function on all states. To do this, we must first infer the policy from choices displayed in our observations. It was mentioned that given the velocity, the path lengths, and path occupations, a person makes a choice based on whether path A or path B yields a higher reward. In this section, we consider 10 000 artificial persons, and present each of them with a different realization of the Glow system, i.e., each person encounters the system with a unique combination of velocity and path occupations. The path length is let equal for all persons. Then, the choices of the persons are made according to a *ground truth reward* $R_{gt}(V_0, N, L)$, that is, they choose the path that yields them the highest reward. We want to find out whether we can recover this ground truth reward with the maximum difference IRL algorithm.

3.2.1 Creating observations

An existing model that combines path length and path occupation is the fundamental diagram, where the velocity of a pedestrian is reduced linearly with the number of people N in their path. The speed V is then given by

$$V = V_0 - \gamma N, \tag{3.2}$$

where V_0 is the unobstructed speed and γ is the speed penalty per pedestrian in the path. This model has been shown to be accurate for large crowd flows in urban areas [28]. Within the system at hand, calculating the travel time for a person at speed V_0 along a path of length L and occupation N is as follows:

$$t_{\text{fd}} = \frac{L}{V_0 - \gamma N}. \tag{3.3}$$

We want to make our artificial pedestrians to behave ideally, so we should make them choose the path that minimizes their travel time. A viable ground truth reward function would be the *inverse* of the travel time, since then, this reward is maximized:

$$R_{gt}(V_0, N, L) = \frac{1}{t_{\text{fd}}} = \frac{V_0}{L} - \gamma\frac{N}{L}. \tag{3.4}$$

So, 10 000 pedestrians are initialized, each with a different combination of V_0, N_A and N_B. Each of them looks at which of $R_{gt}(V_0, N_A, L_A)$ and $R_{gt}(V_0, N_B, L_B)$ is higher and chooses the corresponding path. In the end, the observations look as provided in Table 3.1:

ID	V_0	N_A	N_B	choice
0	1.49	7	2	B
1	1.39	3	5	A
2	1.58	2	1	A
3	1.50	2	8	A
...				
9999	1.14	8	6	B

Table 3.1: Observations of the artificial pedestrians. Here, γ is set to 0.09.

3.2.2 Performing IRL and results

Now that we have observations to work with, IRL can be performed. The way we provide the observations to the system is to convert the set of 10 000 observations to a concatenation of choices. For instance, the concatenation corresponding to Table 3.1 would look like $(B, A, A, A, ..., B)$. We consider this sequence the optimal policy π^* for the given set of observations, and this sequence counts as one singular demonstration. Suboptimal demonstrations are created with the one-step deviation rule, i.e. we permute one of the choices in the optimal sequence to create a suboptimal demonstration, see Figure 3.2.

We chose to start out with $k = 3$ suboptimal demonstrations. Now that we have an optimal demonstration and a set of suboptimal demonstrations, we are ready to perform the maximum difference IRL algorithm. To get a sense of how consistent the algorithm is, the algorithm is performed 50 times, each time on a unique realization of the observations.

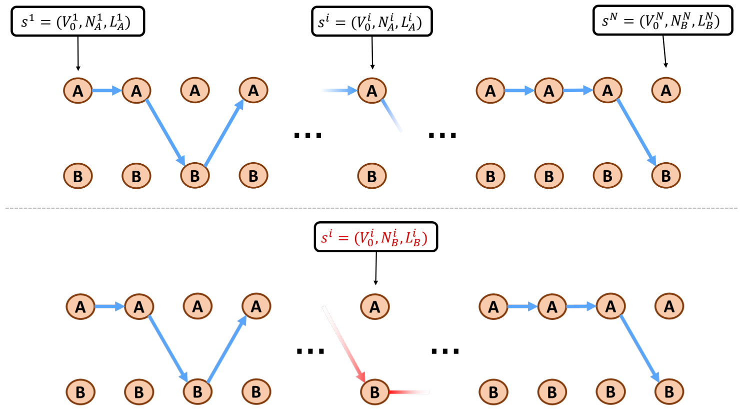

$s^1 = (V_0^1, N_A^1, L_A^1)$
$s^i = (V_0^i, N_A^i, L_A^i)$
$s^N = (V_0^N, N_B^N, L_B^N)$
$s^i = (V_0^i, N_B^i, L_B^i)$
A
B
...

Figure 3.2: Concatenated sequence of choices. A one-step permutation is made to the sequence at the top so that state s_i changes from (V_0^i, N_A^i, L_A^i) to (V_0^i, N_B^i, L_B^i).

We assume a reward of the form of Equation 3.1:

$$R(V_0, N, L) = \alpha_1 \frac{V_0}{L} + \alpha_2 \frac{N}{L}. \tag{3.5}$$

Since the goal is to recover $R = R_{gt}$, we expect $(\alpha_1, \alpha_2) = (1, -\gamma)$. Using the maximum difference algorithm, we get using discount $\delta = 0.95$ and penalty factor $m = 2$ that $\alpha_1 = 1$ without fail, and for α_2 we see the following convergence as more suboptimal policies get added with every iteration:

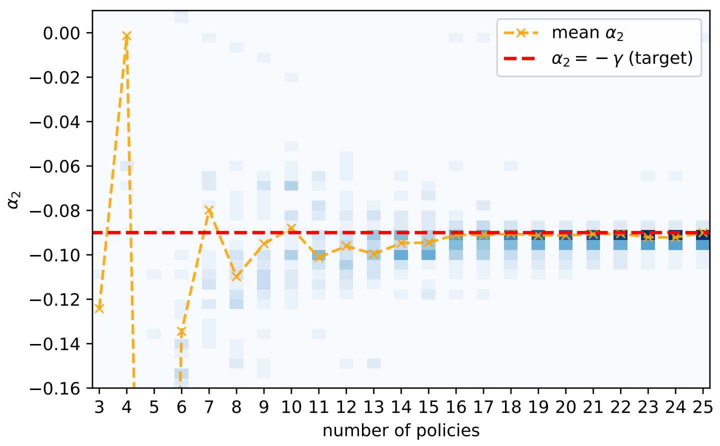


Figure 3.3: Progress of α_2 optimization. The blue squares represent individual measurements while the dashed line goes through their means. After 23 iterations, at which point there are 25 total nonexpert policies, the average value of the coefficient α_2 reaches -0.0929 $\pm$ 0.005. This shows that for this toy problem which featured ideal behavior, the fundamental diagram coefficient can be recovered.

After 23 iterations, at which point there are 25 total nonexpert demonstrations, the algorithm arrives at mean $\alpha_2 = -0.0929 \pm 0.005$.

With the recovered reward at each iteration, we can let the pedestrians make choices in the same way as originally was done with the ground truth reward. By comparing these choices in our 'observations', we can then get a sense of how accurate the obtained reward is. In Figure 3.7, the fraction of choices that are identical to the observed choices is set out against the number of policies in the optimization (which increases with every iteration). After 22 iterations, the accuracy is 0.988 on average.

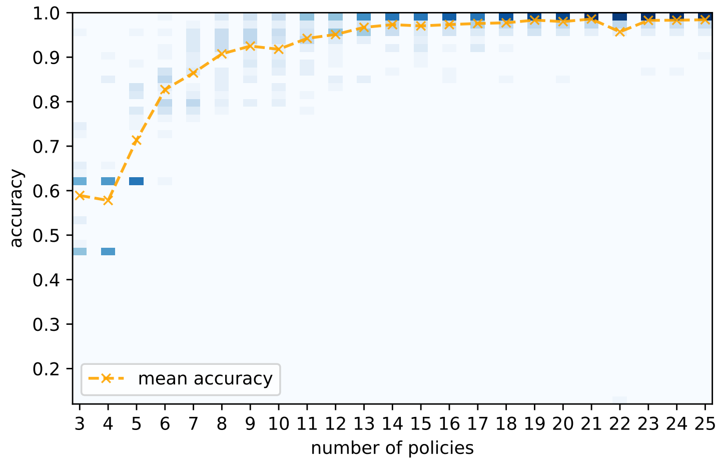


Figure 3.4: Accuracy of predicted choices of the obtained reward vs the number of policies in the optimization. The blue squares represent individual measurements while the dashed line goes through their means. An average accuracy of 0.988 is obtained after 22 iterations.

3.2.3 Bias against one path

In an earlier analysis of the glow bifurcation, it was shown that for low populations of path B (that is, at low values of $\frac{N_B}{N_A+N_B} \equiv \frac{N_B}{N_{tot}}$), pedestrians did not take path B even though this would minimize their travel time [26]. Hence, specifically for path B, there is a penalty to the reward that dissipates at high values of $\frac{N_B}{N_{tot}}$. In our analysis, we want to simulate this effect by using the ground truth reward function

$$\begin{aligned} R_{gt}(V_0, N, L, N_{tot}) &= \frac{1}{t_{\text{fd}}} - p\,\phi_3\left(\frac{N}{N_{tot}}\right) \\ &= \frac{V_0}{L} - \gamma\frac{N}{L} - p\frac{\left(1-\frac{N}{N_{tot}}\right)}{L}\delta_{L,L_B} \end{aligned} \tag{3.6}$$

where the Kronecker delta is employed to only incur the penalty if path B is chosen. Note that this is equivalent to saying that the perceived velocity in path B is given by

$$V = V_0 - \gamma N - p\left(1 - \frac{N}{N_{tot}}\right) \tag{3.7}$$

while in path A, the perceived velocity is given by just the fundamental diagram. Again, dummy observations are simulated and treated as observations for the maximum difference algorithm. We use the same settings, with $p = 0.3$. We use the following linear combination as supposed reward:

$$R(V_0, N, L, N_{tot}) = \frac{V_0}{L} - \gamma\frac{N}{L} + \alpha_3 \frac{\left(1 - \frac{N}{N_{tot}}\right)}{L}\delta_{L,L_B}, \tag{3.8}$$

expecting that $\alpha_3 = -p$. We obtain the following result for α_3:

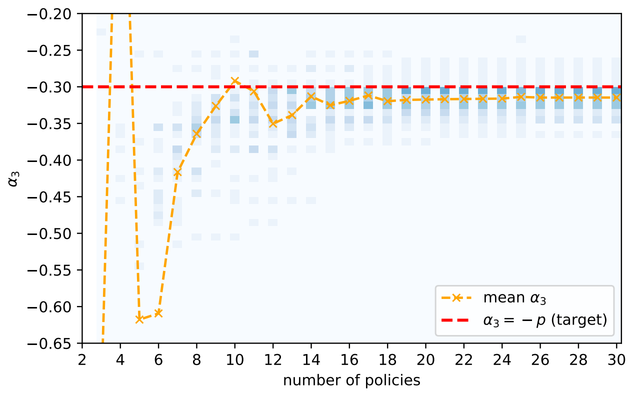


Figure 3.5: Result for α_3 during the iterations of the maximum difference algorithm. This figure is analogous to Figure 3.3, this time for α_3, which is the coefficient in fron tof the penalty function. After 28 iterations, it finds $\alpha_3 = -0.315 \pm 0.019$ on average.

After 28 iterations, at which point there are 30 total nonexpert demonstrations, the algorithm arrives at mean $\alpha_3 = -0.315 \pm 0.019$. The predicting accuracy is given in Figure 3.6.

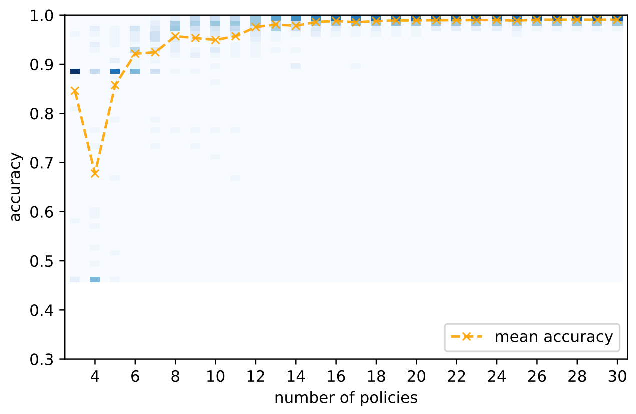


Figure 3.6: Accuracy of predicted choices of the obtained reward with the obtained coefficient α_3 during the iterations of the maximum difference algorithm. We see that high accuracy can already be achieved when the obtained coefficient is only somewhat near the ground truth value.

On average, an accuracy of 0.990 is attained after 28 iterations. This shows that there is barely any room for improvement, even if the obtained α_3 is not exactly equal to its ground truth value. An explanation is that the coefficient cannot be improved if the policy that is obtained from the reward with this coefficient is already the same as the observed policy. This means that any coefficient that can be obtained with the maximum difference algorithm is bounded by a range where the resulting policy is the same as the observed policy, although this range is likely small, as is the case in this exercise.

3.3 IRL on real Glow data

Now it is time to perform IRL on real data. With sensors, the positions of all pedestrians are tracked in real time with high precision. Let us look at the heatmap of these pedestrians to get a sense of which of them we are concerned with:

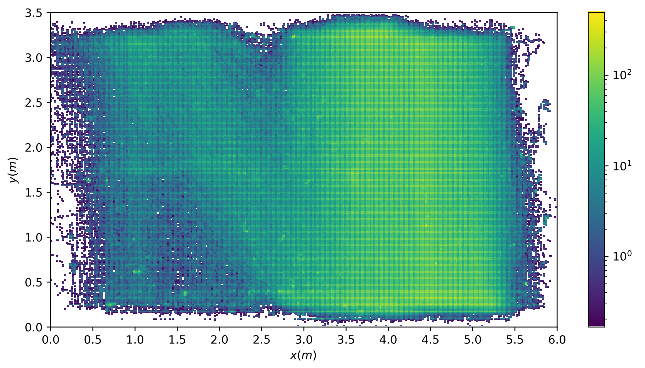


Figure 3.7: Heatmap of pedestrian locations. We see path A on the right, which is more popular than path B, the diagonal stream originating from the bottom right, going towards the left. A small influx into path B from the lower left is also present, we know that these are cyclists.

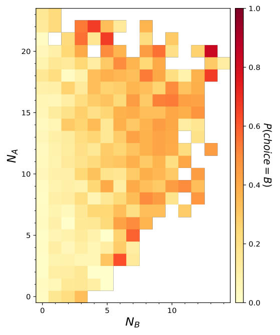
It is clear that most pedestrians take path A. In fact, from our analysis, 89.5% of pedestrians remained on path A. It should be noted that the small stream that originates from the bottom left seems to occupy path B; those are in fact cyclists.

Within the IRL description, the utility of path length should be based on the a priori estimate of this length. An earlier analysis of these data has shown that typically, a walk that goes along path B is 1.2 times longer than a walk that goes along path A [26]. In this analysis, it was shown that a fundamental diagram is present in both paths of the bifurcation, with free stream velocity $V_0 = 1.012$ and fundamental diagram constant $\gamma = 0.017$.

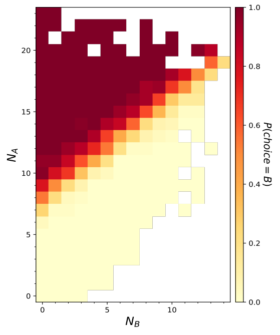
In an ideal world, the pedestrians choose the path that gives them the lowest travel time according to this fundamental diagram, like in the toy problem. However, the pedestrians at large did not do this - they had a bias for path A. To model this, the *perceived* travel time for path B was described as a functional of the fundamental diagram travel time. For our purpose, this means that the base functions in Equation 3.5 will not suffice. Instead, a reward function of the form

$$\begin{aligned} R(V_0, N, L, N_{tot}) &= \frac{1}{t_{\text{fd}}} + \alpha_3 \Pi(V_0, N, L, N_{tot}) \\ &= \frac{V_0}{L} - \gamma \frac{N}{L} + \alpha_3 \Pi(V_0, N, L, N_{tot}) \end{aligned} \tag{3.9}$$

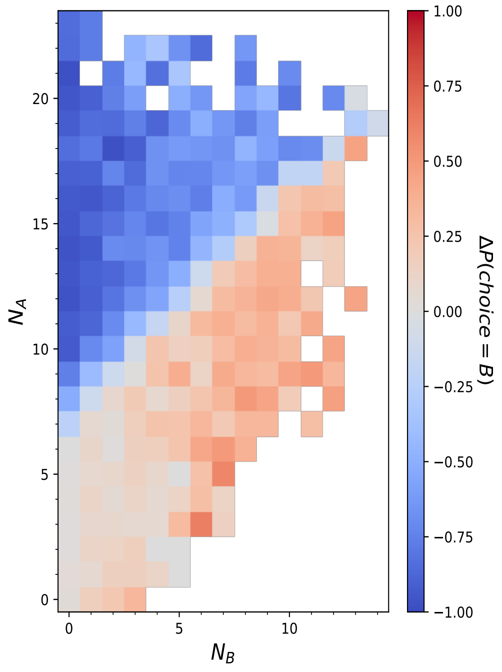
should be tried with the mentioned values for V_0 and γ, where $\Pi(V_0, N, L)$ is a penalty that should encode the bias against path B. To get a feeling for what this penalty should look like, the fractions of pedestrians that chose path B given N_A and N_B are laid out in Figure 3.8. The most egregious differences between the pedestrians' choices and the optimal choices occur at low N_B and high N_A; At low populations of path B, pedestrians are unlikely to choose the path even when path A becomes crowded. Earlier work has shown success by increasing the perceived travel time of path B at low relative populations of path B, with this penalty dissipating at higher populations [27]. We may try $\Pi(V_0, N, L, N_{tot}) = \phi_3(\frac{N}{N_{tot}})$ from equation 3.6.



(a) Actual choices



(b) Ideal choices



(c) Difference in choices

Figure 3.8: In the left figures, the probability of choosing path B for each combination of N_A and N_B are laid out, (a) being the real choices from the data and (b) being the choices in the ideal case where people choose the shortest path according to their fundamental diagram-based travel time. White space means that there is not enough data to infer the probability. In (a), we see that path B is rarely chosen when it has a low population. In (b), we see a red region where people who behave ideally always choose path B, and the white region is where they always choose path A. Inbetween, there is a region where the choice can be either, depending on the person's velocity. In image (c) on the right, the ideal choice probabilities are subtracted from the actual probabilities to show in what regime people act the least rationally. The largest discrepancy is when N_B is low and N_A is relatively high.

3.3.1 Results

With the proposed reward

$$R(V_0, N, L, N_{tot}) = \frac{V_0}{L} - \gamma\frac{N}{L} - \alpha_3\frac{\left(1 - \frac{N}{N_{tot}}\right)}{L}\delta_{L,L_B} \tag{3.10}$$

with $V_0 = 1.012$ and $\gamma = 0.017$, we get the following result for α_3:

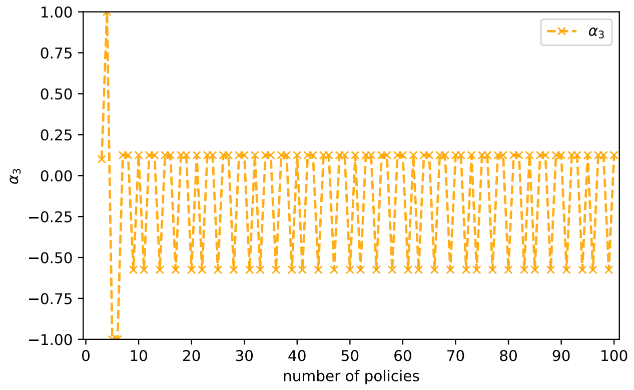


Figure 3.9: Result for α_3 during the iterations of the maximum difference algorithm. No convergence is reached even after 100 policies, meaning the algorithm cannot find a valid linear reward function of the proposed form. The best it can do is $\alpha_3 = -0.575$, which enters the regime where path A always yields a higher reward.

from which it is clear that the algorithm does not know what to do - the obtained α_3 bounces between the values $\alpha_3 = -0.575$ and $\alpha_3 = 0.125$. In fact, when we look at the accuracy in Figure 3.10, it becomes apparent that the value $\alpha_3 = -0.575$ enters the regime where the penalty is large enough to always let the reward be greater when path A is chosen: the accuracy is then 0.895, which is the fraction of pedestrians that chose path A. The other value gives very poor accuracy, the reason for which is unclear.

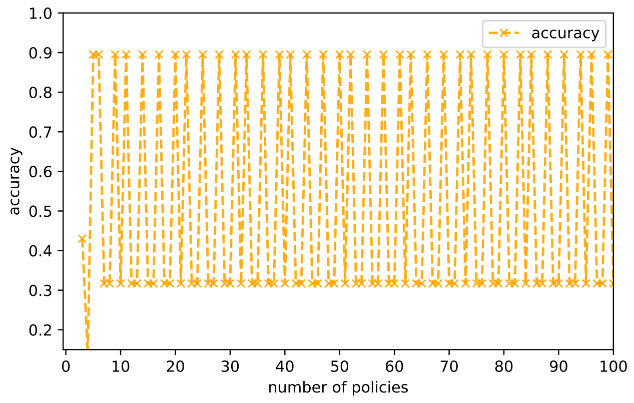


Figure 3.10: Prediction accuracy for the obtained α_3 values at each iteration. We see that an accuracy of 0.895 is reached when α_3 is negative enough to always suggest taking path A, which happened 89.5% of the time in the data. The other value for α_3 yields very low accuracy for unclear reasons.

Modeling the reward in this linear way with the proposed function for $\Pi(V_0, N, L, N_{tot})$ is clearly not possible. We could try other functions for $\Pi(V_0, N, L)$, but at that point it becomes a guessing game: we must conclude that attempting to model this scenario as a linear utility maximization process is not successful. Rather than a linear combination, the perceived travel time was modelled as a function of the fundamental diagram-based travel time in [26], which means that IRL frameworks that allow for non-linear reward functions should be considered in the future.

Chapter 4

Toy problem: routing toward an exit in a crowded environment

Eventually, we will want to move toward the environment of a train station platform. Before we go there, as an intermediate step, we will consider a square environment where an actor wants to reach the exit, and the space between the starting position of the actor and the exit is occupied by crowds at random locations. These crowds have varying densities. In this chapter, we consider a toy problem where we create artificial observations by simulating many realizations of these environments, each time letting the actor walk to the exit through the crowded space. As with Section 3.2, the trajectory of this walk will be dictated by a ground truth reward, and the goal is to see if we can recover this reward.

4.1 Description of features

In our problem, we use an extension of the objectworld description from Section 2.5. Instead of the objects having an inner and outer color, they have just one color, all still chosen from an ensemble of C colors. In addition, there is an object that acts as the exit, and this object is otherwise colorless. This means there are $C + 1$ features. We use a 10×10 grid and the (non-exit) objects can have one of three colors: yellow, orange or red. Instead of the features being the Euclidean distance to the nearest object of each color, the square root of the distance is taken, because this led to more intuitive trajectories; this is an otherwise arbitrary choice. A state $s = (x_s, y_s)$ in the grid then has the following feature vector:

$$\boldsymbol{\phi}(s) = \begin{pmatrix} \sqrt{\text{dist}(s,\ \text{nearest yellow object})} \\ \sqrt{\text{dist}(s,\ \text{nearest orange object})} \\ \sqrt{\text{dist}(s,\ \text{nearest red object})} \\ \sqrt{\text{dist}(s,\ \text{exit})} \end{pmatrix} \tag{4.1}$$

and the ground truth reward will be made from these features, by way of Equation 2.5. We are free to choose the ground truth coefficients $\boldsymbol{\alpha}_{gt}$ as we like. In this case, we want the different colors to represent different crowd densities, i.e. the yellow objects are low-density crowds, the orange objects are medium-density crowds and the red objects are high-density crowds. We want to make it so that higher density crowds have a stronger repulsive effect than low density crowds, and that the exit has an attractive effect. We remind the reader that a repulsive effect corresponds to a positive constant and vice versa. To this end, we chose $\boldsymbol{\alpha}_{gt} = (1, 3, 5, -8)$. For the reward, we assume the same features as 4.1.

An environment is randomly generated, with the exit being placed in the top row of the grid and the crowd objects randomly placed between the top and bottom rows. With the ground truth reward 10 trajectories are generated, each with a different starting square on the bottom row of the grid. On these trajectories, the MAXENT algorithm is performed to obtain the feature coefficients $\boldsymbol{\alpha}$ of the different object classes. A sketch of the procedure is given in Figure 4.1.

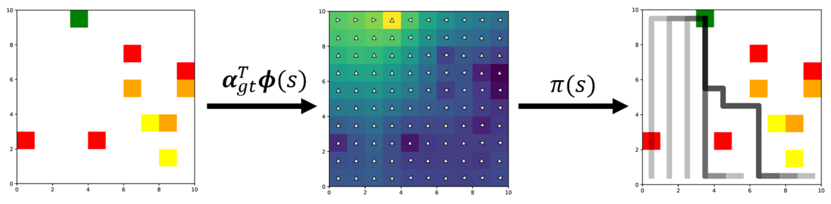
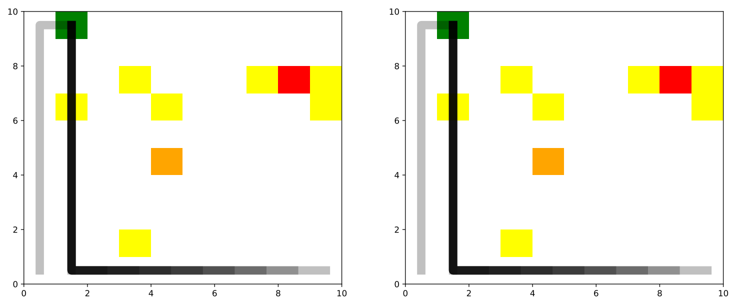


Figure 4.1: Process of generating trajectories through the objectworld environment. First, an objectworld is initialized (left). Then, based on the features of Equation 4.1 and the ground truth coefficients $\boldsymbol{\alpha}_{gt}$, the ground truth reward is constructed (middle). The optimal policy for this ground truth reward is also calculated and indicated by the triangles. Then, with this policy, 10 demonstrations are are generated, each being a trajectory starting from a different tile in the bottom row of the grid (right).

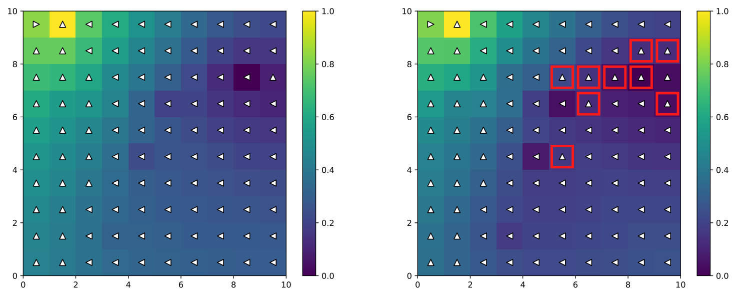
4.2 Results and discussion

The result heavily depends on the environment. In some environments, the trajectories created with the ground truth reward follow the same path regardless of their starting position. If this occurs, many states remain unexplored, and the MAXENT algorithm cannot find what the policy should be in these states. Many policies lead to the same trajectory, and this also means that there are many reward functions that explain the trajectories. The result is that the coefficients that the algorithm finds can differ significantly from the ground truth coefficients. In such cases, generating trajectories with the obtained reward does still lead to trajectories that are very similar to the original trajectories. An example of an environment that has a ground truth reward that leads to trajectories that leave many states unexplored is given in Figure 4.2.

Generally, the more states are explored by the ground truth trajectories, the more restricted the reward and the better the algorithm is at finding the proper coefficients. An example of a ground truth reward that leads to trajectories that explore many states in the grid is given in Figure 4.3.

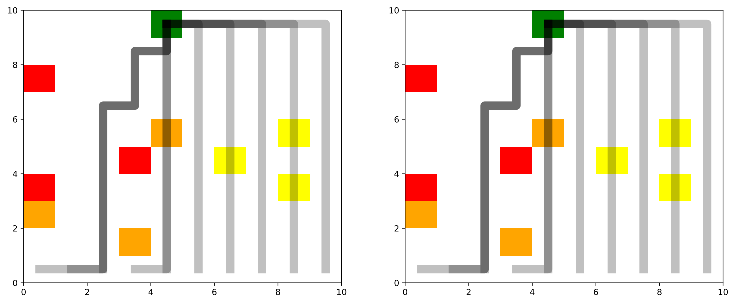


(a) Both images display a 10×10 grid environment with objects placed on random tiles. Yellow squares represent a low density crowd, orange tiles a medium density crowd and red tiles a high density crowd. The green tile is the exit. Trajectories starting from the bottom row are made to walk to the exit, optimizing a reward. On the left image, they walk while optimizing a ground truth reward and on the right, they walk while optimizing the reward that is recovered by our algorithm. The trajectories are identical.

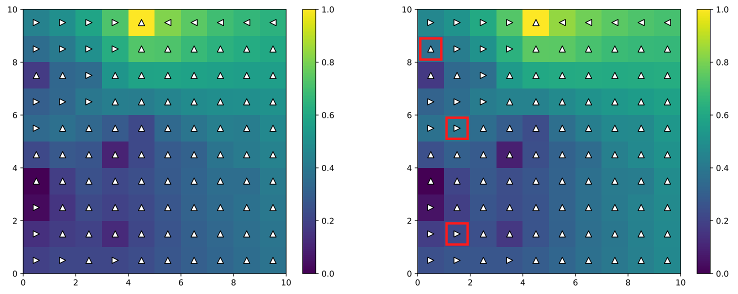


(b) On the left, the ground truth reward is given by the colors, see the colorbar. The policy that indicates the optimal action is given by the directional triangles. On the right, the obtained reward and policy are shown. The red frames highlight tiles where the obtained policy differs from the ground truth policy. The obtained coefficients are $\boldsymbol{\alpha} = (1.87, 4.48, 3.21, -7.88)$, which is significantly different from the ground truth coefficients $\boldsymbol{\alpha}_{gt} = (1, 3, 5, -8)$.

Figure 4.2: An example of an environment that leads to an obtained reward that is different from the ground truth reward. Many states in the grid are unexplored, and the policy in the unexplored region has relatively many differences with the original policy, but this is inconsequential for the simulated trajectories, as is indicated by the top right image.



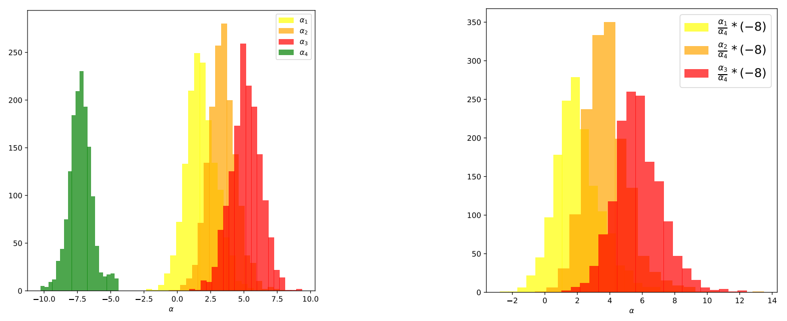
(a) Another example of an environment. Refer to Figure 4.2a for a description.



(b) Left: ground truth reward and policy. Right: recovered reward and policy. Refer to Figure **??** for a description. The obtained coefficients are $\boldsymbol{\alpha} = (0.97, 3.11, 5.77, -7.20)$, which is similar to the ground truth coefficients $\boldsymbol{\alpha}_{gt} = (1, 3, 5, -8)$.

Figure 4.3: Sometimes, the environment leads to a good recovery of the reward. These figures are analogous to Figure 4.2.

By and large, the ground truth reward at hand leads to trajectories that explore sufficiently enough states to retrieve that $\alpha_1 < \alpha_2 < \alpha_3$. To confirm this, 1500 environments were simulated, and the statistics of the 1500 sets of coefficients $\boldsymbol{\alpha}$ are laid out in Figure 5.13. The average coefficients of the features is $\tilde{\boldsymbol{\alpha}} = (1.80, 3.38, 5.14, -7.26)$. Since the algorithm calculates the coefficients unconstrained, we can get a meaningful result by looking at them in relation to each other. To this end, since we are interested in the coefficients of the crowd classes, we scaled the coefficients $\alpha_1, \alpha_2, \alpha_3$ by the environment-respective α_4 and multiplied that by -8 since that was the ground truth value for α_4. This gives the average result $(\alpha_1, \alpha_2, \alpha_3) * \frac{-8}{\alpha_4} = (2.06, 3.81, 5.74)$.



(a) (b)

Figure 4.4: Histograms of the obtained α's for 2000 different environments. In (a), all α's are plotted, while in (b), the α's belonging to the crowd classes are scaled with $-8/\alpha_4$. With this scaling, the α_4 have been scaled to -8, which is the ground truth value.

When compared individually to the ground truth coefficients, the obtained coefficients are often much different. However, averaged over many environments, the coefficients resemble their ground truth counterparts relatively well. What we conclude from this is that coefficients for a single environment are not reliable, but if multiple environments are considered, the average coefficients should give a good indication of reality.

What does hold, however, is that the prediction of new trajectories in any environment is always accurate, no matter how close the obtained coefficients are to their ground truth counterpart. An idea that comes to mind is to build a database of environments. If the database contains sufficiently many environments, then when a new environment is presented, instead of recalculating the coefficients of this environment with the IRL procedure, we can point to the most similar environment in the database and use the coefficients for that one to make predictions. At the same time, the average value of the coefficients in the database does provide information on the relative importance of the utilities that the features respective to the coefficients represent. A caveat with this procedure is that in order to make accurate predictions, the set of environments in the database must cover a substantial fraction of all possible configurations of the environment.

Chapter 5

IRL applied to a train station platform

In this chapter, we first describe general behavior of pedestrians leaving a train and walking toward the escalator or staircase at the exit of the platform. Then, the reward is proposed as a combination of the static environment such as the staircase, the escalator and the boarding zone, and the dynamic environment which consists of the moving pedestrians. The dynamic environment is presented as a frequently updating objectworld. Finally, a way to predict how people handle a bifurcation is presented.

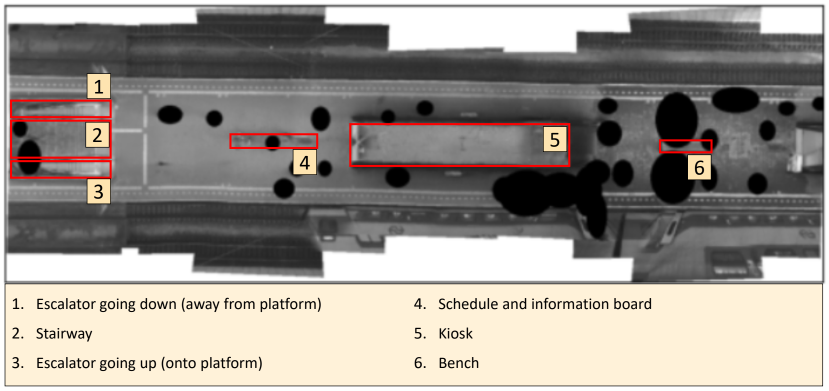

1
2
3
4
5
6
1. Escalator going down (away from platform)
2. Stairway
3. Escalator going up (onto platform)
4. Schedule and information board
5. Kiosk
6. Bench

Figure 5.1: Overview of platform and objects. The black circles are artifacts to hide pedestrians present during the snapshot.

5.1 Exploring the data

Every day, anywhere from 20 000 to 50 000 people walk across platform 2 of the Eindhoven railway station. The platform and its peripherals are described in Figure 5.1. The platform is 14 m wide and, stretches about 70 m long, measured from the beginning of the stairs to the end of the figure. With a sensor tracking setup, the x- and y-positions of all pedestrians on the platform are continuously and accurately tracked. Measurements are carried out by Prorail under the tightest privacy constraints, and this is compliant with the ethical authorization we received. Measurements from October 2021 to March 2022 are considered.

The pedestrians can be divided into two groups: people who leave the train and enter the platform, and people who leave the platform and enter the train. In this paper, the people who leave the train and walk toward the exit of the platform, eventually taking the escalator or stairway are studied.

For people stepping out of a cabin far from the exit, the kiosk in the center has a role similar to the stadium pillar in the Glow scenario: it creates a bifurcation in which one path to the exit is shorter than the other. For convenience, the shorter path is said to go 'over' the kiosk and the longer path to go 'under' it. We will focus on the pedestrians who leave the train far enough to be presented with the choice between these paths.

We will first consider the positions of other people on the station at the moment a pedestrian leaves their cabin and enters the platform in tandem with their eventual trajectory. In Figure 5.2, two cases of an actor choosing to go under the kiosk is presented, one of which initially has a considerable crowd in the path over the kiosk and one where there are barely any people on the station, and two cases of an actor choosing to go over the kiosk. We see that the actors who go under the kiosk initially do not plan to do so; they remain undecided until after a few steps. Presumably, especially during busy hours, this is because there is an initial group of people lining up at the cabin door, waiting to enter after everyone has stepped out of the cabin. This wait-and-see-type behavior is common and invalidates any model that predicts the choice of going under or over the kiosk from the moment the actors leave their cabin.

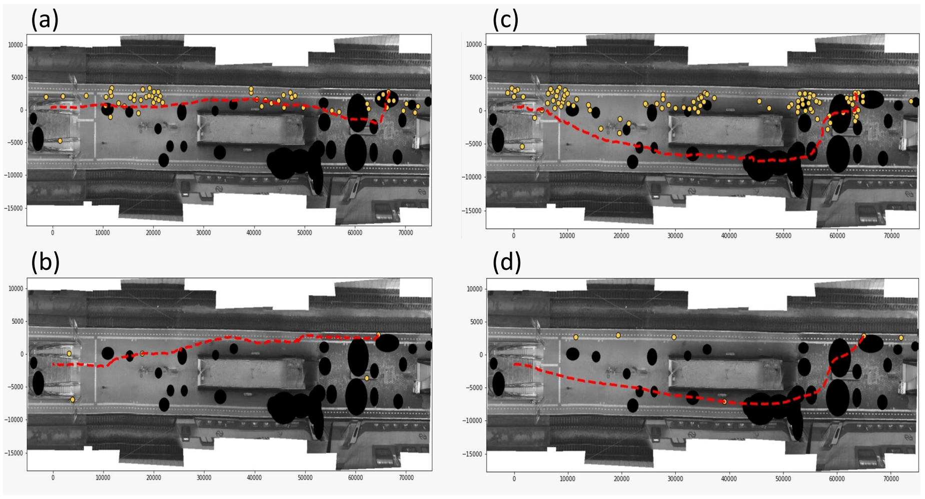


Figure 5.2: Four examples of actor trajectories. While both (a) and (b) are common, it is by far more common to see (c) than (d). However, the existence of (d) implies that it only takes very low occupation of the area above the kiosk to make pedestrians walk under the kiosk.

Questions

The first question we ask is whether we can understand and predict the close-range interactions between an actor and the pedestrians around them. Can we distinguish whether dense spots have a different utility to the actor than less dense spots?

Another proposition is to understand how we can predict whether a person goes over or under the kiosk. Do they immediately know which choice they will make? If so, the first few steps they make should provide enough information to infer their choice. However, to remain in line with our analysis of the Glow scenario, we want to use the populations in the paths over and under the kiosk in our prediction.

5.1.1 Populations under and over the kiosk

Similar to the Glow scenario, it is assumed that people base their decision on whether to go over or under the kiosk on the populations of these regions. The group of people who start their walk from the top-right region of the platform have a good idea of how many people there are in the region above the kiosk, since they can fully see this region. However, they cannot fully see the region below the kiosk, so they have limited information about how many people occupy this region. Therefore, we assume they base their decision on the difference between the visible population above and under the kiosk. We count the population in the to the actor visible region over and under the kiosk, where we assumed the visible regions are as given in Figure 5.3.

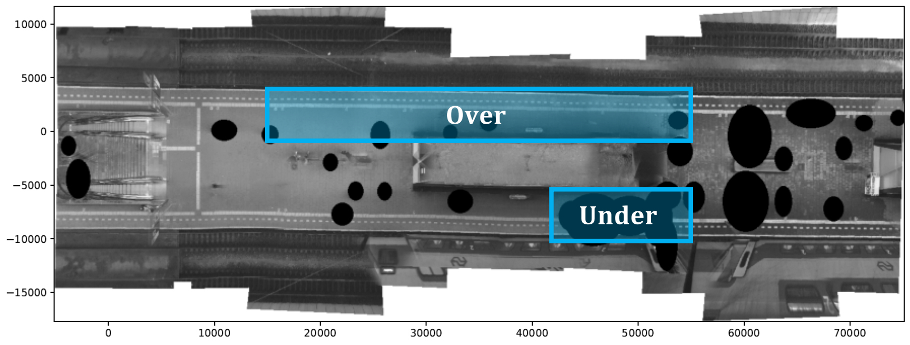


Figure 5.3: The regions where the population under N_{under} and over N_{over} are counted.

We track the total population inside both zones during the month of October 2021 to get a general idea of what these are. We are interested in the difference between these populations $N_{\text{over}} - N_{\text{under}}$, so the distribution of this quantity is laid out in Figure 5.4.

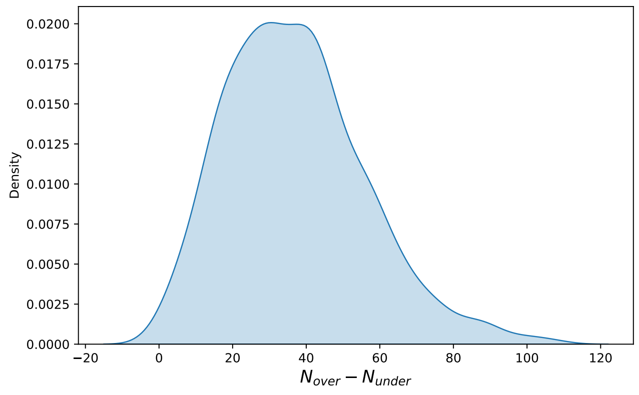


Figure 5.4: Distribution of the population differences $N_{\text{over}} - N_{\text{under}}$ in the visible regions of Figure 5.3.

We will later use this distribution to threshold it into equally populated quantiles for the population differences, i.e. we will distinguish different cases of $N_{\text{over}} - N_{\text{under}} > 0$.

5.2 IRL implementation

We set out to learn a reward function that is a linear combination of utilities and extract information from the coefficients. From what we see, it is clear that the pedestrians in question consider the following utilities when choosing their path: the escalator, the stairs, the boarding zone, and the other pedestrians. The escalator and stairs serve the same function, but could mutually differ in utility. A starting point would be to use a linear reward built from features similar to 4.1, with the objects now being the escalator, stairs, boarding zone and pedestrians. The kiosk and bench simply serve as impassable obstacles. However, as explained in Section 2.3, we might set out to amend the reward to include more utilities if the results are unsatisfactory.

If we were to use this approach, we must translate the platform and its pedestrians to an objectworld environment: this entails discretizing the space into a grid and then placing objects on the grid tiles. For reasons that become clear later, it is convenient to ensure the tiles are squares. Since the dimensions of the station are 14×70m, we can use a 20×100 grid; the tiles are then 70×70 cm in real measurements. The proposed grid is given in Figure 5.5 and shows us in which tiles to place our objects.

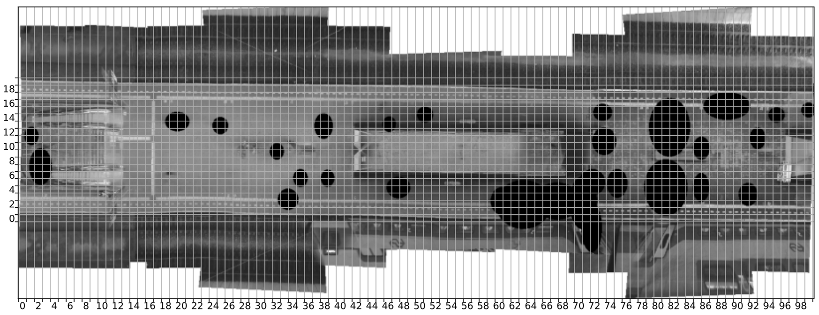


Figure 5.5: Discretization of the platform environment. From this, we read on which tiles we should place the objects in our objectworld environment.

Object colors and features

A 70×70 cm tile is likely to only be occupied by one or zero pedestrians, but it does occur that two or even three pedestrians occupy it. For this reason, we choose three different colors for these occupation levels: a yellow object represents an occupation of 1 pedestrian, orange represents 2 pedestrians on the tile, and red represents 3. Moreover, we need three colors for the static objects - the boarding zone, the escalator and the stairs. This leaves us with 6 colors:

static environment colors:	yellow, orange, **red**
dynamic environment colors:	**boarding zone**, **escalator**, stairs

and therefore 6 features. The words 'yellow', 'orange' and 'red' refer to the cases of respectively 1, 2 or ≥ 3 pedestrians on a tile. The text colors represent how the objects will be displayed in the coming graphs. The features will be similar to Equation 4.1, although we will later discuss their exact form. Important to note is that one object can be of multiple colors; for example a tile on the boarding zone on which one person is standing has both the yellow and the boarding zone color.

In our discussed version of objectworld, we need an MDP description. The allowed actions in the MDP are to go up, down, left, right, and to remain in place. we need to discretize the trajectories of the actors as well. In the discretized version of the trajectories, it may occur that an actor makes a diagonal transition from one tile to another - we must then break it up in two steps, e.g., a step diagonally upwards and to the left is broken up into two steps: one upwards and one to the left. In Figure 5.6, an example is given of an objectworld with a discretized trajectory through it.

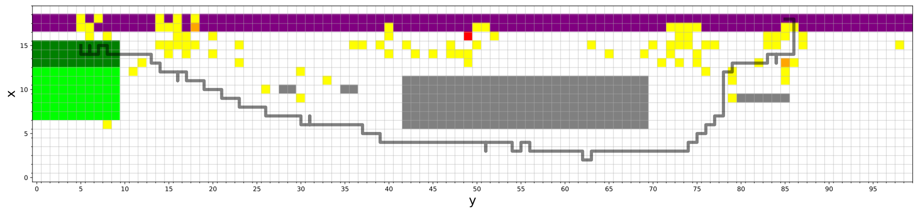


Figure 5.6: Proposed objectworld representation of the top image in Figure 5.2. Similarly to Section 4, the different colors are different object classes, the crowd being classified as either low density (1 person per square; yellow), medium density (2 persons per square; orange), or high density (3+ persons per square; red).

As found in the previous chapter, the coefficients that will be obtained from any given single objectworld will not provide a good insight individually. However, when coefficients for many different objectworlds are obtained, we can still gain insight by looking at their average. As mentioned, to make predictions for a new environment using this database, we will look at the closest match for the new environment within the database, and choose the coefficients of the objectworld respective to that environment. It was postulated that for this to work well, the database must contain a significant fraction of the possible configurations. This is where we see an issue with our current idea for the objectworld:

the number of possible configurations is astronomically high, primarily due to how many tiles we have. To given an idea, we have $O(10^3)$ tiles, each of them can be either empty or occupied by one of three object classes (yellow, orange or red). This amounts to $4^{10^3} \approx O(10^{600})$ different configurations, and this is without even considering the placement of objects with the colors of the static environment.

In addition, there is one obvious and major difference with the scenario in Chapter 4. While in that section, the crowd objects were modelled as stationary during the actor's trajectory, in this case, the crowd is time-dependent: in the next step of our actor, the environment has already changed completely. We need to find a way to get around this time dependence. The next section will propose a way to deal with these two problems simultaneously.

5.2.1 The box approach

To address the problem of the platform objectworld having an enormous number of possible configurations, we must lower the number of tiles. This can be done by changing the resolution of the grid. Using a coarse grid means that we cannot model the close-range interactions with pedestrians, which is what we are interested in, so doing this is not compatible with our goal. Another way to lower the number of tiles is to only consider ('zoom in' on) a certain part of the environment. If we consider an actor at time t at a certain tile x, y on the grid, we can draw a square box of size $B \times B$ centered around this tile and disregard anything that is outside of this box. This new environment has only B^2 tiles. We know the crowd at time t, so we can already place the crowd objects at the proper locations in the box. We can also place the static objects in the box. Naturally we can place any static objects in the box. We know the trajectory of the actor while they are inside this box. If we can capture all utilities relevant to the actor inside this box, we can find the coefficients of the features respective to the utilities with the usual objectworld IRL scheme.

The second obstacle for the objectworld approach was the time dependency on the crowd. We will explain how the box idea can help us take on this problem. Consider the following (Figure 5.7 will illustrate the concept). At time t_1, we draw a box around the actor who is located on the tile at x_1, y_1. We track the tiles the actor visits to get their trajectory inside this box. The actor crosses the edge of this box at time t_2, at which point they are at location x_2, y_2. Now, we draw another box around the tile at x_2, y_2 and load the crowd inside this box at time t_2. We repeat this process until the entire trajectory of the actor has completed. So, the crowd is updated every time the actor leaves a box, retaining the time dependency of the crowd. This way, we get for every pedestrian on the station a set of boxes and their trajectory in these boxes. For every box, we can carry out the objectworld IRL procedure and get the coefficients of the features corresponding to the colors of the objects in the box.

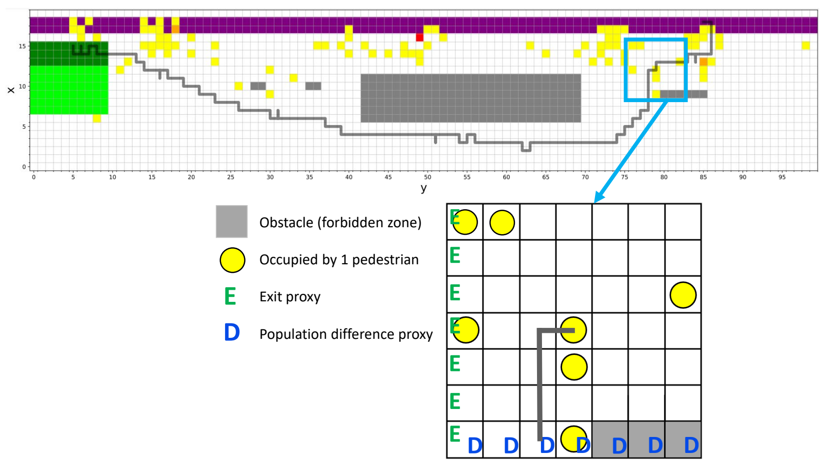


Figure 5.7: Box around an actor at an arbitrary point x, y in their trajectory. The time is t; at this time, the locations of the other pedestrians are read out and the objectworld on the platform is built. A box is (in this example, of size 7×7) drawn around the actor at x, y. For the colors in this box, the coefficients will be calculated with MAXENT IRL on the trajectory segment in this box. In this case, it is the 14^{th} step of their trajectory, so population difference proxies are placed in the box. We can learn the coefficients for the the yellow color and the two proxy colors (we should note that the grey objects are just obstacles, not utilities).

On the one hand, there are time-dependent objects, which are colored yellow, orange and red. On the other hand are the objects colored with the static environment colors: the boarding zone, stairs and escalator colors. The box will be filled with objects with objects of these colors. However, in any box that is not near the stairs or escalators, there is no sense of direction because there is no object that represents the exit. In order to still account for the fact that the actor walks toward their long-range goal of the exit, proxies of the exit are placed in the box if the escalator or stairs are not in it. These proxies are placed in the B closest tiles to the escalator or stairs.

Further, the long-range effect of choosing to go under or over the kiosk has to also be represented in the box. As mentioned, we see people generally make their choice to go either under or over the kiosk between their 5^{th} and 15^{th} step. To this end, when the box is loaded at one of these steps, proxies are placed on the bottom row of the box so they give direction. We would like to make it so the difference between population above and under the kiosk $N_{\text{over}} - N_{\text{under}}$, determines the color of these proxies. So, we create three colors that each correspond to a level of population difference. We only want to place the proxies when $N_{\text{over}} > N_{\text{under}}$, so we divide the positive part of the distribution in Figure 5.3 into tertiles T1, T2 and T3. See Figure 5.8.

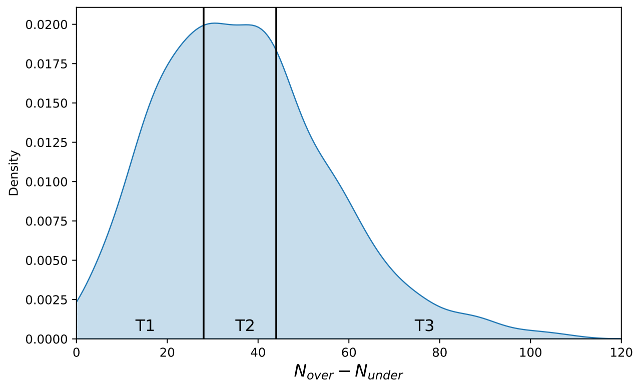


Figure 5.8: The division of the positive part of the distribution in Figure 5.3 into tertiles. We have the tertiles T1: $[0, 27]$, T2: $[28, 44]$ and T3: $[44, \infty]$.

The three colors associated with the population difference levels will be dubbed T1, T2, and T3.

We propose to split up the reward in four parts: one part that originates from the static objects, one part that encapsulates the crowd effects, one part for the exit proxies and one part for the population difference proxies:

$$R(s) = R_{\text{static}}(s) + R_{\text{crowd}}(s) + R_{\text{exitproxy}}(s) + R_{\text{popdiffproxy}}(s). \tag{5.1}$$

For reasons that will become later, the features from which the reward is built are chosen as follows:

$$\begin{aligned} R_{\text{static}}(s) &= \boldsymbol{\alpha}_{\text{static}}{}^T \boldsymbol{\phi}_{\text{static}}(s) \\ &= \begin{pmatrix} \alpha_{\text{escalator}} & \alpha_{\text{stairs}} & \alpha_{\text{boardingzone}} \end{pmatrix} \begin{pmatrix} \delta_{s,\text{escalator}} \\ \delta_{s,\text{stairs}} \\ \delta_{s,\text{boardingzone}} \end{pmatrix}, \end{aligned} \tag{5.2}$$

$$\begin{aligned} R_{\text{crowd}}(s) &= \boldsymbol{\alpha}_{\text{crowd}}{}^T \boldsymbol{\phi}_{\text{crowd}}(s) \\ &= \begin{pmatrix} \alpha_{\text{yellow}} & \alpha_{\text{orange}} & \alpha_{\text{red}} \end{pmatrix} \begin{pmatrix} \delta_{s,\text{yellow}} \\ \delta_{s,\text{orange}} \\ \delta_{s,\text{red}} \end{pmatrix}, \end{aligned} \tag{5.3}$$

where *yellow*, *orange* and *red* refer to the three possible density levels of a tile; the exit proxy

$$R_{\text{exitproxy}}(s) = \alpha_{\text{exitproxy}} \delta_{s,\text{exitproxy}}, \tag{5.4}$$

and

$$\begin{aligned} R_{\text{popdiffproxy}}(s) &= \boldsymbol{\alpha}_{\text{popdiffproxy}}{}^{T} \boldsymbol{\phi}_{\text{popdiffproxy}}(s) \\ &= \begin{pmatrix} \alpha_{\text{T1}} & \alpha_{\text{T2}} & \alpha_{\text{T3}} \end{pmatrix} \begin{pmatrix} \delta_{s,\text{T1}} \\ \delta_{s,\text{T2}} \\ \delta_{s,\text{T3}} \end{pmatrix}. \end{aligned} \tag{5.5}$$

The square of the distance to the object of each color in Equation 4.1 is replaced with a Kronecker delta on the object of the color, because reward is a very local concept in this case: e.g., it is only important whether the actor is in the boarding zone or not; their distance to the boarding zone is presumably not relevant. To compare the coefficients that we will obtain, it is helpful if their respective features are identical in form, so we decided that all features be Kronecker delta functions.

5.3 Results and discussion

We choose a box size of $B = 5$ and carry out the MAXENT IRL scheme with discount δ equal to 0.9. Trajectories from October 2021 and December 2021 are taken as input. We made sure these trajectories start on the top right and reach the exit, filtering out any trajectories that get cut off halfway through due to sensor limitations. In total, this amounts to 1419 trajectories. We remind the reader that every trajectory provides multiple box environments, and each environment gives the coefficients of the colors present in the each box. The trajectories make up 29 800 boxes. Distributions of the obtained coefficients are given in Appendix B.

The average values of the coefficients are as follows (for exact values, see Appendix B):

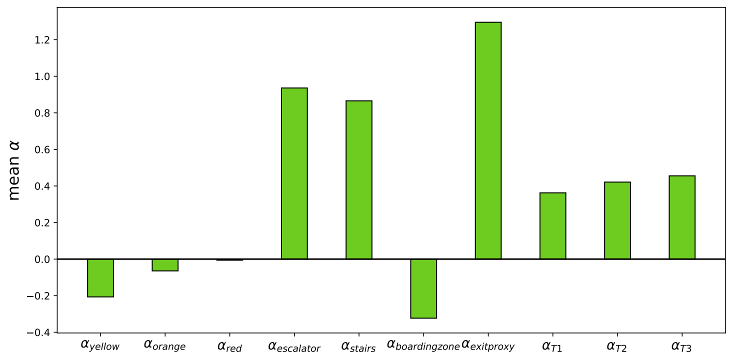


Figure 5.9: Mean values of the coefficients of the colors.

The average coefficients tell us the average reward of a tile occupied by an object of the corresponding color. For example, a tile in the boarding zone has a lower average reward than a tile occupied by one person, and a tile on the escalator has a higher average reward than a tile on the stairs if neither are occupied. However, a tile on the escalator that also has a person occupying it has lower reward than an unoccupied tile on the stairs, and so on.

5.3.1 Crowd object coefficients

We see that $\tilde{\alpha}_{\text{yellow}} < \tilde{\alpha}_{\text{orange}} < \tilde{\alpha}_{\text{red}}$, suggesting that on average, a tile occupied by one person has lower average reward than tiles occupied by 2 or more persons. This is not intuitive. The distribution of α_{yellow} tells a more complete story, see Figure 5.10.

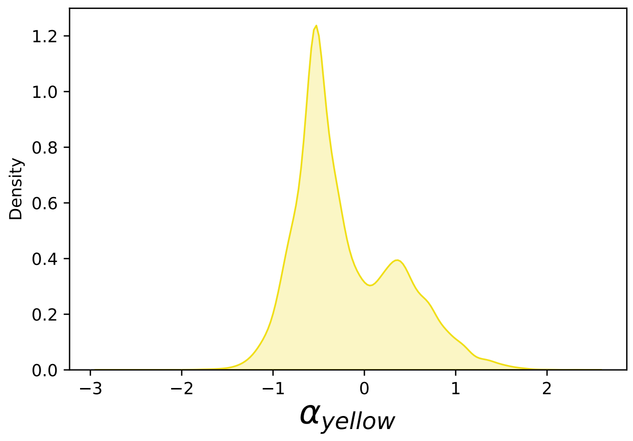


Figure 5.10: Distribution of the obtained values of α_{yellow}. The distribution is seemingly built from two smaller distributions, one of which has a negative mean and one a positive. The former suggests a set of cases where the actor avoided the yellow object, i.e. the object was a likely a person who was standing still. The latter suggests a set of cases where the actor followed the object, so likely these objects were persons who were walking in the same direction.

The distribution of α_{yellow} is comprised of two smaller distributions, one of which has a negative mean and the other a positive. The former suggests a group of boxes where the actor avoided the tiles with a yellow object on them, i.e. tiles occupied by one person. The latter represents a group where the actor instead pursued such tiles. An explanation for this is that in the first case, the person on the tile was standing still, forcing the actor to avoid them. In the latter case, the person was walking in the same direction as the actor - toward the exit - and the actor followed the person, walking over the tile that is vacated by the time they reach it. It appears that the first case occurs more often than the second case, hence the negative average value of α_{yellow}. This average value falls between two distinct cases and is in general far off from the true coefficient of any box.

We see a similar structure for α_{orange} and α_{red}, though in their case, the negative peaks are close to 0, α_{red} more so than α_{orange}. This means that in boxes where an orange or red colored object is present, the algorithm thinks this object has low impact on the actor's trajectory. An explanation for this is that yellow objects dominate the interaction, as in nearly all boxes with an orange object in them, multiple yellow objects are present. The collective of the yellow objects has a greater influence than the lone orange object. For red objects, this is even more pronounced since red objects are scarcer. The positive peaks of the distributions of α_{orange} and α_{red} are low, which makes sense since the more persons occupy a tile, the less likely that tile will be vacated for the actor.

5.3.2 Static object coefficients

We see a slightly higher average reward for tiles on the escalator than on the stairs. The distributions of $\alpha_{\text{escalator}}$ and α_{stairs} both have a negative and positive peak, see Figure 5.11. The negative peak in the distribution for $\alpha_{\text{escalator}}$ represents boxes where the actor chose the stairs and the positive represents boxes where they took the escalator, and vice versa for α_{stairs}. The escalator was simply chosen more often than the stairs, hence the higher average coefficient.

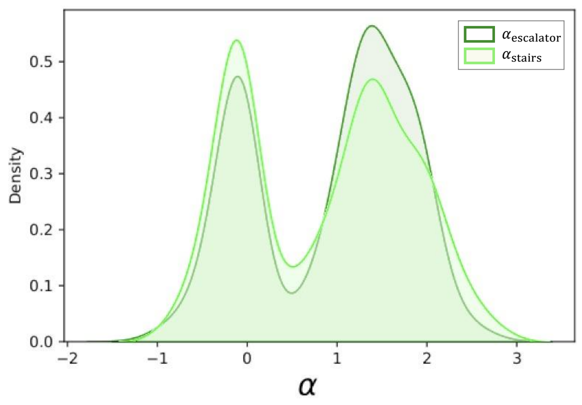


Figure 5.11: Distribution of the obtained values of $\alpha_{\text{escalator}}$ and α_{stairs}. The presence of the positive and negative peak for both distributions points to boxes where if one of the two is chosen, the other is not, leading to a positive coefficient for the former and a negative for the latter.

Save for a few cases, $\alpha_{\text{boardingzone}}$ is always negative, even more so than the coefficients for the crowd colors. This is an intuitive result.

5.3.3 Proxy object coefficients

The average value of $\alpha_{\text{exitproxy}}$ is highest of all, which is expected. If there were a tile on the boarding zone that is occupied by one person, and this was the only square that brings the actor closer to the exit, it would still have a positive reward on average. There are rare cases where it is negative, meaning that in such boxes, the actor does not exit in one of the 5 squares closes to the exit. We can think of two examples where this occurs. First are the boxes at the beginning of the actor's trajectory, when they just leave their cabin. They have to walk downward because crowd waiting to enter the cabin is waiting besides the exit. Second would be the boxes where the actor walks downward to get in the region under the kiosk.

The coefficients of T1, T2 and T3 colored proxies have the clearest dichotomy in their distribution: a sharp peak around zero and a spread out peak across relatively high positive values. It is insightful to look at them side by side, see Figure 5.12.

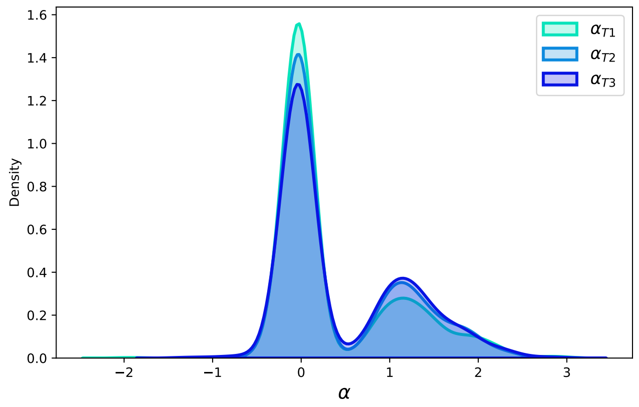


Figure 5.12: Distribution of the obtained values of α_{T1}, α_{T2} and α_{T3}. Each consists of a clear positive and a part centered around 0, the former representing cases where the actor decided to walk under the kiosk, and the latter where they walk above the kiosk. The height of the peaks represents the frequency of the respective cases, and we see that when the population difference $N_{\mathrm{over}} - N_{\mathrm{under}}$ is in the third tertile, the pedestrians walk under the station most often, and the least often for the first tertile.

The height of the peaks around $\alpha = 0$ represents how often the actor chose to go over the kiosk. The reason it is centered around $\alpha = 0$ is because the exit proxies already have a positive enough reward so that it is sufficient to let the population difference proxies have zero reward. When the actor chooses to walk under the kiosk, the reward at the population difference proxies must be greater than at the exit proxies, and this is represented by the highly positive peaks. At the third tertile of $N_{\mathrm{over}} - N_{\mathrm{under}}$, it occurs most often that the actor chooses to walk under the kiosk, and at the first tertile it occurs the least often. This is directly reflected in the average values of the coefficients.

5.3.4 Generating trajectories

As mentioned, the way to generate trajectories would be to use the database of boxes and their coefficients. If we were to generate a trajectory starting at time

t at location x_0, y_0, we would do the inverse: draw a 5×5 box around x_0, y_0, load the environment at time t and fill the box with objects of the applicable colors. This box might not exist in our database, so we will need some fitting procedure to find the most similar box in our database, and use the coefficients of that box to get the reward function (Equation 5.1). In the box, with its respective reward function, the trajectory that is optimal with respect to the reward in this box is carried out. When the actor exits this box, the crowd is updated. To do so, we need to know the time and load the crowd at this time. The time can be inferred by knowing that each tile is 70×70 cm, so we can assume some average walk speed ($\tilde{1}.3$ m/s) and calculate the expected residence time of a tile. Here, it is convenient that we used square tiles, so we assumed a residence time of $t_{res} = 0.7/1.3$ seconds. This way, by concatenating trajectories in boxes, a complete trajectory from the cabin to the exit can be built. However, conceptualizing and implementing a fitting procedure to get coefficients for any thinkable box is a monumental task and goes beyond the scope of this work.

Instead, one might consider generating trajectories with the average obtained coefficients, but we can dismiss this idea on the basis that the average coefficients are never a good representation of the true coefficients of any box, as we discussed. What we can do, however, is to search for some coefficients that lead to trajectories that 'look' good compared to a real trajectory that started at the same time and at the same location. Thanks to our distributions, we know what the approximate relative proportionalities of the coefficients should be.

Using the following coefficients, we got trajectories that along broad lines follow a similar path as their real counterparts:

α_{yellow}	-1
α_{orange}	-0.5
α_{red}	-0.5
$\alpha_{\text{escalator}}$	+1.5
α_{stairs}	+1.5
$\alpha_{\text{boardingzone}}$	-1.2
$\alpha_{\text{exitproxy}}$	+1.5
α_{T1}	+1.3
α_{T2}	+1.5
α_{T3}	+1.7

Table 5.1: Heuristically found coefficients that lead to acceptable trajectories.

These coefficients have similar proportionality to the average obtained coefficients (Fig 5.9) and we will show some trajectories that were generated with the resulting reward function. In the following figures, the blue trajectory is generated, starting at the same time and location as the real trajectory which is drawn in black.

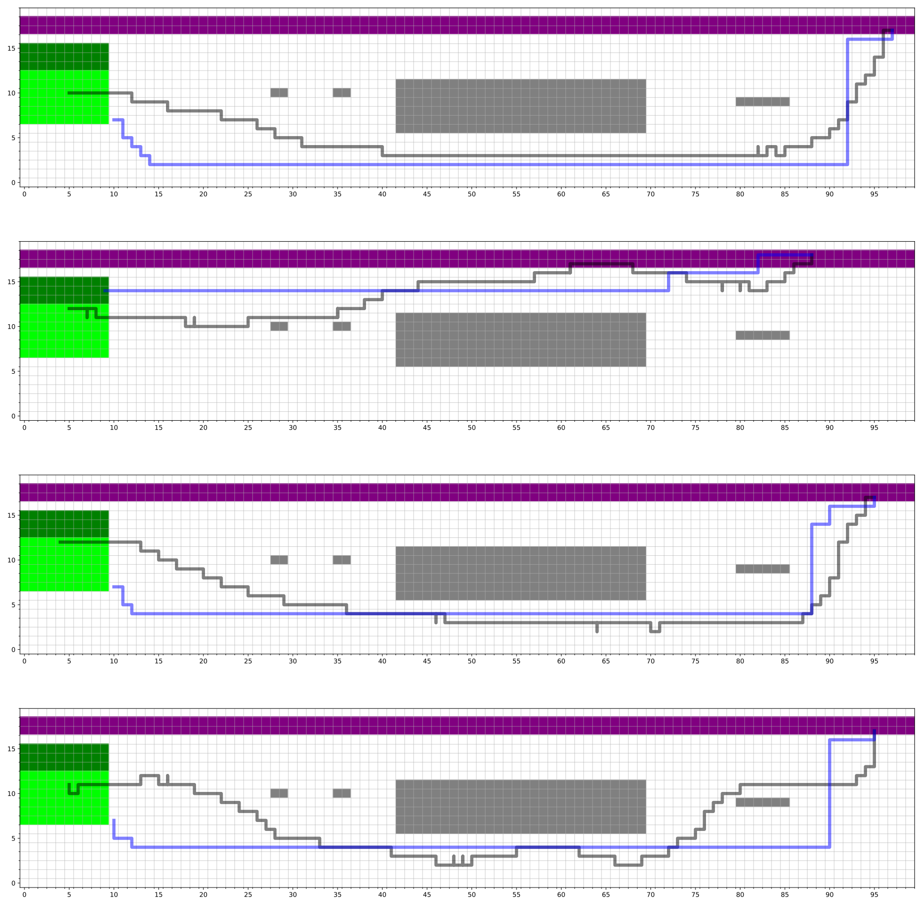


Figure 5.13: Trajectories generated with the coefficients in Table 5.1. The most striking differences are the fact that the generated trajectories consist of straight segments, which is an artefact of the fact that going diagonally is not a permitted action in the MDP.

Remarkably, the generated trajectories consist of straight segments. The reason for this is that a diagonal step is not permitted, so the actors of the generated trajectories do not bother to. The actors of the real trajectories do take diagonal steps, which appear as two steps in their discretized version, because this is quicker than walking in two straight segments.

Another artefact is the fact that the generated trajectories that terminate at the stairs, do so by taking the lowest possible point on those stairs. In this section of the stairs, people commonly walk up, onto the platform - they would be oncoming traffic. The real trajectories terminate at the highest section, since that is where people commonly walk down. The stairs object does not make this distinction, so this is why the generated trajectories take the first accessible tile with the stairs.

Chapter 6

Concluding remarks and future work

It has been attempted to find a quantity that is optimized by pedestrians in their routing using inverse reinforcement learning. The quantity in question must be a linear combination of functions of the utilities in the system. The coefficients that are obtained then give a sense of proportionality between the utilities.

In the case of a binary choice, a toy problem was considered where pedestrians choose one of two possible paths of different lengths and occupations. We created artificial observations where the pedestrians choose the path that yields a higher reward under a known reward function. Two functions have been tested, and both have been recovered with good accuracy. A next step could be to introduce statistical noise to the observations, i.e, to let each person have a 10% chance to make their choice randomly. If we fail to converge with the present algorithm, a maximum entropy algorithm should be tried instead, because such algorithms are robust to statistical noise. This is not done in the thesis due to programming technicalities - we were not able to extend the maximum difference algorithm to a maximum entropy model within the time frame of this thesis.

However, we did not succeed in finding a reward function for real data in a binary choice scenario. This can be ascribed to the fact that there are a significant number of instances where, pedestrians make different choices in otherwise identical scenarios. This too could be accounted for by a maximum entropy algorithm. Moreover, frameworks that allow for non-linear solutions should be considered.

Routing of pedestrians toward an exit in a crowded environment was investigated as a toy problem where people walked to an exit through a space where crowds of three different density levels were randomly placed. Many of such environments were considered, and the artificial pedestrians walk to the exit optimizing a reward function that is a linear combination with predetermined coefficients. For each environment, it was attempted to recover the coefficients with inverse reinforcement learning. It was found that individually, the obtained coefficients are not consistently faithful to the original, although the mean coefficients give a good indication of the proportionality of the predetermined coefficients.

Trajectories on platform 2.1 of the Eindhoven central station were discretized into a grid and broken down into segments. In each segment, the direct environment of the pedestrian up a short distance was considered. In total, 28 900 of such environments were obtained from the segmentation of real trajectories. On each tile, the coefficients of the objects in the tile, such as one or more pedestrians, the escalator, stairs, and the boarding zone. It was found that on average, tiles with one person have lower reward than tiles with more persons. Further, tiles on the escalator exit have higher reward on average than tiles on the stairs exit. Tiles on the boarding zone had very low reward.

Trajectories were generated across the station with a reward function built from coefficients similar to the obtained average values. They were similar to real trajectories along broad lines, but unlike them, the generated trajectories consisted of long, straight segments.

The fact that the generated trajectories on the station consisted of few linear segments can be ascribed to the fact that diagonal steps were not allowed, but had to be broken up into the allowed steps: left, right, up and down. In the future, a model that allows diagonal steps should be considered. This could lead to entirely new, likely more accurate coefficient distributions. In addition, generated trajectories could better resemble real trajectories.

To answer the research question, individual routing on a discretized version of the platform can be modelled as a utility maximization process, where on the tiles up to a short distance away from the individual in question, the utility is represented by a linear combination of delta functions, and the coefficients of these delta functions are environment-specific.

Bibliography

[1] R. Treisman, “8 people were killed at houston’s astroworld festival after crowd rushed the stage,” *NPR*, Jun. 11, 2021.

[2] L. Henderson, “On the fluid mechanics of human crowd motion,” *Transportation research*, vol. 8, no. 6, pp. 509–515, 1974.

[3] D. Helbing, L. Buzna, A. Johansson, and T. Werner, “Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions,” *Transportation science*, vol. 39, no. 1, pp. 1–24, 2005.

[4] W. Daamen and S. Hoogendoorn, “Experimental research of pedestrian walking behavior,” *Transportation research record*, vol. 1828, no. 1, pp. 20–30, 2003.

[5] P. Degond, C. Appert-Rolland, M. Moussaid, J. Pettré, and G. Theraulaz, “A hierarchy of heuristic-based models of crowd dynamics,” *Journal of Statistical Physics*, vol. 152, no. 6, pp. 1033–1068, 2013.

[6] R. Hughes, “A continuum theory for the flow of pedestrians,” *Transportation Research Part B: Methodological*, vol. 36, no. 6, pp. 507–535, 2002.

[7] S. Hoogendoorn and P. Bovy, “Pedestrian route-choice and activity scheduling theory and models,” *Transportation Research Part B: Methodological*, vol. 38, no. 2, pp. 169–190, 2004.

[8] M. Garcia, O. Montiel, O. Castillo, R. Sepulveda, and P. Melin, “Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation,” *Applied Soft Computing*, vol. 9, no. 3, pp. 1102–1110, 2009.

[9] B. Henrique, V. Sobreiro, and H Kimura, “Literature review: Machine learning techniques applied to financial market prediction,” *Expert Systems with Applications*, vol. 124, pp. 226–251, 2019, ISSN: 0957-4174. DOI: `https://doi.org/10.1016/j.eswa.2019.01.012`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S095741741930017X`.

[10] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems*, C Cortes, N Lawrence, D Lee, M Sugiyama, and R Garnett, Eds., vol. 28, Curran Associates, Inc., 2015. [Online]. Available: `https://proceedings.neurips.cc/paper/2015/file/07563a3fe3bbe7e3ba84431ad9d055af-Paper.pdf`.

[11] A Callahan and N. Shah, "Chapter 19 - machine learning in healthcare," in *Key Advances in Clinical Informatics*, A. Sheikh, K. Cresswell, A. Wright, and D. Bates, Eds., Academic Press, 2017, pp. 279–291, ISBN: 978-0-12-809523-2. DOI: `https://doi.org/10.1016/B978-0-12-809523-2.00019-4`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/B9780128095232000194`.

[12] B. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. Bagnell, M. Hebert, A. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2009, pp. 3931–3936.

[13] C. Pouw, J. Willems, F. van Schadewijk, J. Thurau, F. Toschi, and A. Corbetta, "Benchmarking high-fidelity pedestrian tracking systems for research, real-time monitoring and crowd control," *arXiv preprint arXiv:2108.11719*, 2021.

[14] J. Tsitsiklis, "Efficient algorithms for globally optimal trajectories," *IEEE Transactions on Automatic Control*, vol. 40, no. 9, pp. 1528–1538, 1995. DOI: `10.1109/9.412624`.

[15] G. Eyink, "Action principle in nonequilibrium statistical dynamics," *Physical Review E*, vol. 54, no. 4, p. 3419, 1996.

[16] Q. Wang, S. Bangoup, F. Dzangue, A. Jeatsa, F. Tsobnang, and A. Le Méhauté, "Reformulation of a stochastic action principle for irregular dynamics," *Chaos, Solitons & Fractals*, vol. 40, no. 5, pp. 2550–2556, 2009.

[17] G. Williams, P. Drews, B. Goldfain, J. Rehg, and E. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1433–1440. DOI: `10.1109/ICRA.2016.7487277`.

[18] D. Wolpert, Z. Ghahramani, and M. Jordan, "Are arm trajectories planned in kinematic or dynamic coordinates? An adaptation study," *Experimental Brain Research*, vol. 103, no. 3, pp. 460–470, Nov. 1995, ISSN: 1432-1106. DOI: `10.1007/BF00241505`. [Online]. Available: `https://doi.org/10.1007/BF00241505`.

[19] S. Levine, Z. Popovic, and V. Koltun, "Nonlinear inverse reinforcement learning with gaussian processes," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., vol. 24, Curran Associates, Inc., 2011. [Online].

Available: `https://proceedings.neurips.cc/paper/2011/file/c51ce410c124a10e0db5e4b97fc2af39-Paper.pdf`.

[20] A. Ng, S. Russell, *et al.*, "Algorithms for inverse reinforcement learning," in *Icml*, vol. 1, 2000, p. 2.

[21] K. Muelling, A. Boularias, B. Mohler, B. Schölkopf, and J. Peters, "Learning strategies in table tennis using inverse reinforcement learning," *Biological cybernetics*, vol. 108, no. 5, pp. 603–619, 2014.

[22] E. Jaynes, "On the rationale of maximum-entropy methods," *Proceedings of the IEEE*, vol. 70, no. 9, pp. 939–952, 1982.

[23] B. Ziebart, A. Maas, J. Bagnell, A. Dey, *et al.*, "Maximum entropy inverse reinforcement learning," in *Aaai*, Chicago, IL, USA, vol. 8, 2008, pp. 1433–1438.

[24] B. O'Donoghue, R. Munos, K. Kavukcuoglu, and V. Mnih, "Combining policy gradient and q-learning," *arXiv preprint arXiv:1611.01626*, 2016.

[25] A. Pitombeira-Neto, H. Santos, T. da Silva, and J. de Macedo, "Trajectory modeling via random utility inverse reinforcement learning," *arXiv preprint arXiv:2105.12092*, 2021.

[26] A. Gabbana, F. Toschi, P. Ross, A. Haans, and A. Corbetta, "Fluctuations in pedestrian dynamics routing choices," *arXiv preprint arXiv:2202.02108*, Feb. 2022.

[27] T. Lazar, F. Toschi, A. Corbetta, and A. Gabbana, "Modeling of the emergent behavior for a unidirectional pedestrian flow at a bifurcation point," 2021.

[28] N. Geroliminis and C. Daganzo, "Existence of urban-scale macroscopic fundamental diagrams: Some experimental findings," *Transportation Research Part B: Methodological*, vol. 42, no. 9, pp. 759–770, 2008, ISSN: 0191-2615. DOI: `https://doi.org/10.1016/j.trb.2008.02.002`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0191261508000180`.

[29] M. Alger, *Inverse reinforcement learning*, 2016. DOI: `10.5281/zenodo.555999`. [Online]. Available: `https://doi.org/10.5281/zenodo.555999`.

Appendix A

Code used for results

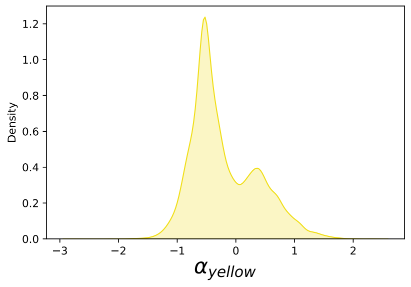
The python scripts and notebooks used to create results and figures is available on this webpage:

`https://git.phys.tue.nl/MSc/hees/-/tree/main/Thesis_files`

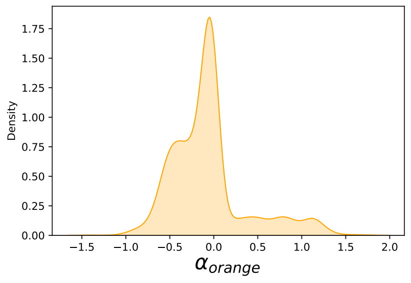
The objectworld codes are a modified and extended version of code originally developed by Matthew J. Alger [29].

Appendix B

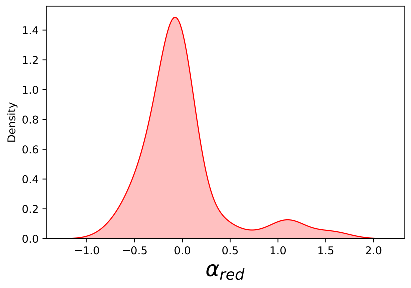
Coefficient distributions



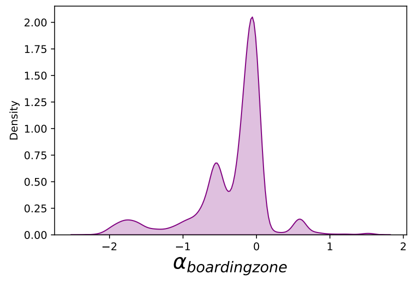
(a) N = 29 656, $\tilde{\alpha}_{\mathrm{yellow}} = -0.207$



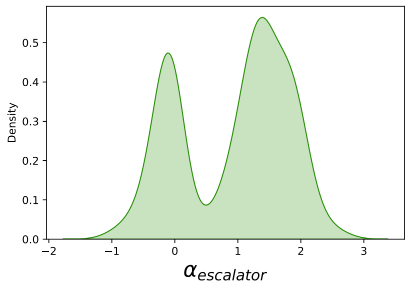
(b) N = 2 710, $\tilde{\alpha}_{\mathrm{orange}} = -0.064$



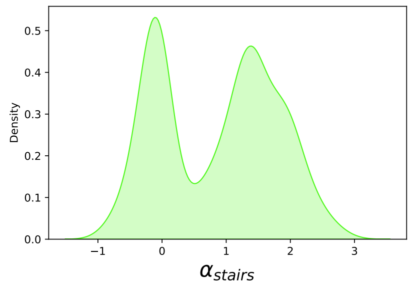
(c) N = 91, $\tilde{\alpha}_{\mathrm{red}} = -0.006$



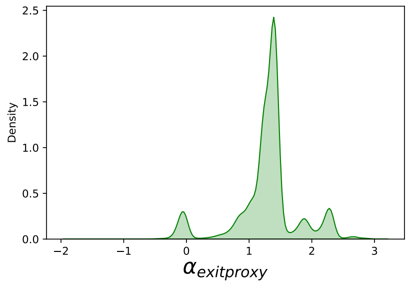
(d) N = 10 678, $\tilde{\alpha}_{\mathrm{boardingzone}} = -0.323$



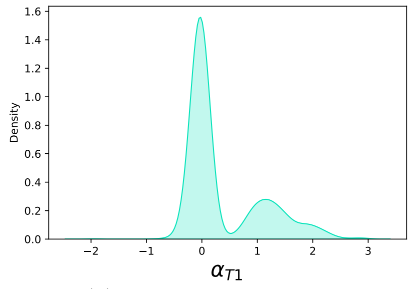
(e) N = 778, $\tilde{\alpha}_{\mathrm{escalator}} = 0.936$



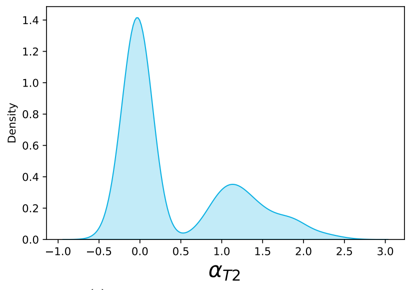
(f) N = 958, $\tilde{\alpha}_{\mathrm{stairs}} = 0.866$



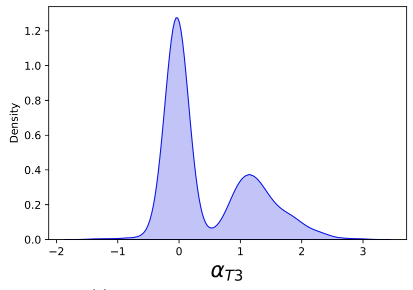
(g) N = 28 631, $\tilde{\alpha}_{\mathrm{exitproxy}} = 1.295$



(h) N = 893, $\tilde{\alpha}_{\mathrm{T1}} = 0.362$



(i) N = 751, $\tilde{\alpha}_{\mathrm{T2}} = 0.421$



(j) N = 728, $\tilde{\alpha}_{\mathrm{T3}} = 0.455$

Figure B.1: Density plots of the coefficients. Their means are given in the caption, as well as the number of boxes the color was present.