Agent Based Model Discovery with Reinforcement Learning

Recovering latent school-choice policies from summary statistics using RL for ABMs

Author:	Alexios Ioannis Carras[1]
Daily Supervisor:	Brian Logan[1]
1st Supervisor:	Michael Lees[2]
2nd Supervisor:	Eric Dignum[2]
Examiner:	Mehdi Dastani[1]

A thesis submitted in partial fulfilment of the requirements for the Masters of Science degree in Artificial Intelligence at Utrecht University

May, 2023

Utrecht University[1]
University of Amsterdam[2]

Abstract

Agent Based Modelling (ABM) is a powerful tool for modelling social systems. Generative runs simulate micro-level behaviours that give rise to emergent macro-level outcomes. To ensure the accuracy of those outcomes to the modelled process, behavioural rules are carefully implemented and their parameters calibrated. Recently, methods for the inverse generation of ABMs - from outcomes to behavioural rules - have received much attention. Most approaches aim at constructing parts of the ABM or require high-resolution data. In this thesis, we use Reinforcement Learning (RL) to learn the individual policies of a school choice model using only summary statistics of the reference process. A Deep Q-Network is used to learn and encode the recovered policy, which can then be used in simulations. We demonstrate the robustness of our method for the recovery of different latent behavioural rules using different reward functions. We find that our method is not very robust, although it shows signs of learning. In subsequent experiments, we show that the recovered policies generalise better than a baseline random agent, but the learned behaviour only partially matches the reference. We speculate on two critical obstacles to the performance that future research should address.

Keywords:

Reinforcement Learning, Agent-Based Models, Inverse Generative Social Science, School-choice models

List of Notation

Agent Based Modelling	
H	Set of households
M	Set of schools
i	Household agent
m	School agent
c_m	Composition of school m
d_{mi}	Euclidean distance from school m to agent i
t_i	Tolerance threshold of agent i
$\mathcal{M}$	Homogeneity preference factor
U_{mi}	Utility of school m for agent i
C_{mi}	Composition utility of school m for agent i
D_{mi}	Distance utility of school m for agent i
T_G	Maximum system segregation
T_{Li}	Local segregation (e.g., in a school)
T	Theil's index
Reinforcement Learning	
S	Set of states
A_s	Set of actions in state s
$R(\cdot)$	Reward function
γ	Discount factor
G_t	Return at time-step t
π	Policy
Our method	
S	Reference set of summary statistics
$\hat{S}$	Sample set of summary statistics
τ	Target (i.e., a reference measure we want to match)
$\mathbf{t}^{\tau}$	Target vector of statistic τ
$\hat{\mathbf{t}}^{\tau}$	Target vector of statistic τ
$\mathbf{s}$	Household state-vector
a	Household action
$\mathbf{Q}(\mathbf{s})$	DDQN's output action-value vector
$R^{\tau}(\cdot)$	Reward for target τ
$R^{\tau}_{gradient}(\cdot)$	Reward for change in dissimilarity of target τ

Contents

Chapter 1

Introduction

Agent Based Models (ABMs) are a popular tool for modelling complex systems where the interaction of individual agents at the micro-scale leads to emergent outcomes at the macro-scale. Agent interaction takes place in a computational environment according to predefined behavioural rules (Bonabeau, 2002). A limitation of ABMs and other modelling architectures is that designing the behavioural rules requires deep domain knowledge. Moreover, human designers may be biased or, due to the complexity of an agent, have to make assumptions resulting in the implementation of simplistic rules or the omission of unknown ones. These issues can limit the simulation's validity (i.e., how closely real-world behaviours are computationally reproduced) and, thus, how useful they are as a tool for scientists and policymakers.

Behavioural rules may be subject to parameters which affect the agent's behaviour and system outcome. In these cases, calibration methods can identify the parameters that lead to most accurate behaviours with the reference process (Cranmer et al., 2020). However, this is a computationally expensive process requiring thousands of generative simulations under different parameters to compare them with the macro-level summary statistics of the reference processes. Moreover, the outcome is still limited by the validity of the coded behavioural rules.

The new, upcoming field of inverse generative social science is concerned with automating the design of behavioural rules by inferring them using the system output data (Epstein, 2023). Learning methods from Artificial Intelligence (AI) - mainly supervised learning (SL) and genetic programming (GP) - have been at the forefront of this research (Lavin et al., 2021). However, SL is unsuitable for optimising sequences of actions and can thus lead to the accumulation of errors in simulation. GP is a promising approach but sample inefficient. Furthermore, both require high-resolution data for training, which is often unavailable, particularly in privacy-sensitive social processes.

Reinforcement Learning (RL) is a learning method that has the potential to address the aforementioned limitations of inverse ABM construction. RL is a family of AI methods for learning sequential decision-making in dynamic environments; by specifying a quantifiable objective, an RL agent can learn the optimal policy that maximises it (Sutton and Barto, 2018). RL implemented with deep neural networks (DNN) can learn to approximate functions of arbitrary complexity in high-dimensional state spaces to optimise long-term rewards. Consequently, it can model decision-making without accumulating errors in simulation. Moreover, it only requires sparse learning information as a scalar reward rather than high-resolution spatiotemporal information as in traditional supervised learning approaches. Finally, the learned policy can be naturally integrated into an existing ABM to model the agents' decision process. The ABM can then be used to forecast and understand the process dynamics.

Here we introduce RL as an automated discovery tool in the inverse generative process for school-choice ABMs. We leverage the generality of RL to learn latent policies (behavioural rules) for ABMs from scratch by using the dissimilarity of the RL system's macro-scale outcome summary statistics with that of a reference system as the reward signal. The problem of inversely recovering latent policies from summary statistics is especially hard as the reference processes are complex with non-linear dynamics sensitive to initial conditions and characterised by emergent outcomes.

1.1 COMPASS project

Our research is constrained to theoretical grids and is intended as a proof-of-concept for using RL in model discovery. We use the work of Dignum et al. (2022) to address the problem in a controlled, computational environment like the one depicted in Figure 1.1. The focus is on recovering the utility function[1] of a school-choice ABM. In the context of social segregation ABMs, this work is novel and part of a growing body of work on the intersection of complex systems and AI. Findings and intuition regarding the development of such a methodology may be relevant to other domains where only aggregated, summary statistic data of social processes are available, but modelling individual decision-making is desirable.

[1]A note on terminology: We refer to the terms policy, behavioural rule, decision process, and utility function interchangeably though they are subtly different. For example, in ABMs the decision process is implemented as a behavioural rule which in our case implements a ranking logic over utilities that are calculated using a utility function. In RL, policies refer to behaviours as manifested in sequences of actions. Finally, we refer to the reference policy as latent policy because in the formulation of our problem ,summary statistics are available but not the exact decision rules that lead to them. This terminological overlap is an unavoidable side-effect of interdisciplinary work on the intersection of complex systems and RL.

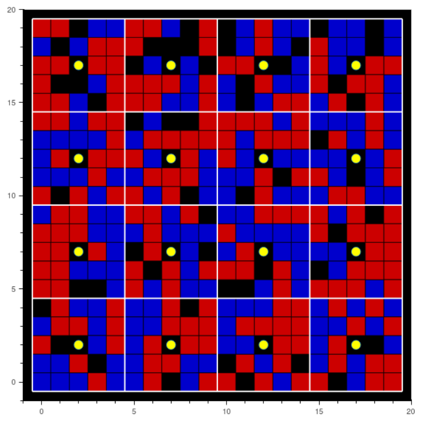


Figure 1.1: Grid environment of size 20 × 20, 16 neighbourhoods (white lines) and 16 schools (yellow circles). There are two types of household agents: blue (0) and red (1). Households make decisions about where to reside on the grid and which school to attend (Dignum et al., 2022)

.

The work for this thesis takes place in the context of a larger project titled Computational Modelling of Primary School Segregation (COMPASS) for the Dutch Inspectorate of Education and the City of Amsterdam. Therefore, the ultimate objective of this research is to develop a model that accurately captures parent decision-making and, thus, school segregation in Amsterdam.

The school-choice model is a suitable testbed, not only on account of how consequential it is as a societal process but because of its inherent characteristics: it involves multiple factors in the decision rule (e.g., distance, demographic, economic, and other) which allows for a diverse set of reference rules to be recovered and get the potency of our method evaluated. Moreover, the school process is an example domain where social data is highly sensitive and thereby, only aggregate statistics and information is usually available. Finally, school models have a spatial dimension which can influence the dynamics. This also means that the process can be broken down into spatially distinct units, e.g., households and schools, which can be encoded as vectors of statistical information and thus naturally compared in terms of their dissimilarity. These properties are common in utility-based spatial ABMs.

1.2 Research Questions

We focus on recovering a model of the household school choice policy exclusively using function approximation to learn a value map of the state-action space. Given the novelty of our problem, our primary research question is general:

Research question: What is the feasibility of using reinforcement learning (RL) to recover a valid school-choice policy for use in ABM of school segregation using only outcome summary statistics data?

To answer research question 1, we must examine the robustness, generalisation performance, and micro-scale prediction accuracy of the policies recovered using our method. Therefore, the following three sub-questions must be addressed:

Sub-research question 1: What is the robustness of the method's learning performance for different configurations of the reference system and reward function?

Sub-research question 2: What is the generalisation performance of the recovered policy on unseen instances, measured in terms of Mean Absolute Error, between the sample and reference segregation (Theil index)?

Sub-research question 3: What is the accuracy of the recovered policy in predicting individual household's school choices made by the reference policy?

Answers to these research questions will provide insight into the suitability of RL for inverse generative social science from outcomes and embolden interested researchers to tackle similar challenges by identifying the strengths and limitations of the methodology.

1.3 Structure

The thesis starts with a brief overview of the necessary background and work related to the intersection of AI, modelling, RL and ABMs. We then cover our methodology in Section 3. In Section 4 we give an overview of our experimental setup. In the following Section 5, we present our experimental results. We interpret and contextualise the results in the discussion Section 6, where we further attempt to draw general guidelines from our findings. The thesis concludes with Section 7 by providing a summary of our main findings, limitations of our work, and suggestions for further research.

Chapter 2

Background and Related Work

First we introduce two fundamental concepts that form the foundations of our work: Agent-Based Modelling and Reinforcement Learning. Subsequently, we summarise previous work on the intersection of learning and simulations in general, and learning in the inverse generative process in particular.

2.1 Agent Based Models

Simulations are a tool for exploring a system's dynamics and evolution. In computational contexts, one can design a model of some process grounded in real life to explore perturbations in the system's environment or counterfactual and hypothetical scenarios. Agent-Based Models (ABMs), a computational simulation architecture, are based on collections of autonomous interacting units through which complex dynamics and behaviours arise (Bonabeau, 2002). Agents can represent any entity, for example, a student or an organisation and implement the characteristics and behaviours of the entity they model that are necessary for the studied process to take place. For example, an agent representing a pedestrian can have characteristics such as speed, direction, and preferred walking paths. Moreover, ABMs used for prediction and explanation must be validated by evaluating their accuracy with respect to the real-life process being modelled (Macal and North, 2009).

Repeated agent interaction can lead to **self-organisation** and the **emergence** of structures or behavioural patterns that have not been explicitly programmed and are thus not part of the individual properties or policies. For example, birds flocking or neighbourhoods becoming segregated. In this research, we focus on a class of ABMs - namely, **random utility models** where at every time step, agents have to make a choice or take an action from a finite discrete set of choices which is evaluated using a **utility function**. An agent's choice can affect the environment and, thus, the choices of other agents. Choice modelling aims at designing formal models of a decision process that individuals follow in some phenomenon (Train, 2009).

ABMs are generative models. This means that they are used to generate data about a phenomenon to better understand and predict the underlying dynamics in a bottom-up fashion: implementing the basic individual rules which lead to the emergent outcome (Epstein, 1999). Traditionally, to use an ABM for forecasting it must be calibrated which entails identifying appropriate parameterisations of the utility function such that the emergent outcome and agent behaviours are as close as possible to the real process and behaviours. Herein lies the assumption that matching outcomes can imply matching individual behaviours. Calibrating ABMs requires running multiple generative runs with different parameter settings in the utility functions and comparing the emergent outcomes with real world data (O'Sullivan, 2004). This is a laborious process, not scalable for complex utility functions with many parameters, and the achieved validity is limited by the utility function design in the first place.

More recently, researchers have been exploring the inverse of this process, coined in the context of social science research by Vu et al. (2019) as **inverse generative social science** (IGSS). Here the agent rules are not the input but the output, and the goal is not to generate statistics about a process or give rise to an emergent outcome but to discover or recover the decision process that led to them.

2.1.1 ABMs of Segregation

ABMs have often been used in the context of social science research (Heath et al., 2009). In fact, one of the first instances of social science research being conducted by means of computational methods and what we would now call an ABM is the Schelling segregation model used to study patterns of residential segregation (Schelling, 1971). In this model, Schelling assumed the existence of two types of agents: blue and red households (A and B, respectively). These agents reside on an $N \times N$ grid in which each cell can be empty or occupied by either type of household. Each agent i has a tolerance value t_i, which corresponds to the proportion of neighbours of the opposite type of i that they are willing to tolerate. In terms of utility, this entails that the closer the ratio of neighbours to that tolerance factor, the higher the utility. For example, given $t_i = 0.5$ and neighbourhood composition of agent i $c_i = 0.5$ the agent is at the maximum utility. The behavioural rule is that if the household neighbour ratio $\frac{c_i}{t_i}$ is less than the tolerance factor, then the household moves to an empty spot on the grid. The key insight of the Schelling model is that even for simulations where households have only a moderately low tolerance of opposite-type neighbours, households form clusters on the grid resulting in spatial segregation. Therefore, individual preferences can be central to the residential segregation dynamics (Clark and Fossett, 2008). This example is an instance of emergent outcomes: moderate individual preferences (e.g., $t = 0.6$) can lead to extreme outcomes (e.g., only similar neighbours $c_i = 1$)

School segregation and modelling

Simulation models inspired by Schelling's work are still being implemented today to explore exactly this: how individual preferences lead to collective behavioural patterns and emergent segregation outcomes. An ongoing subject of research in many parts of the world, due to its role in inequality more broadly, is segregation in schools. School segregation can be defined as the skew in the distribution of students in schools with respect to some social or economic characteristic. For example, a school is segregated when it is predominantly attended by people of one ethnic group in an otherwise diverse city. Boterman et al. (2019) find that school segregation is widespread in cities and that the segregation can be with respect to social dimensions other than ethnicity, e.g., socioeconomic background. In the US school segregation has resulted unequal education and fewer opportunities for intergenerational social mobility (Creusere et al., 2019; Reardon and Owens, 2014).

Notwithstanding school segregation being a global phenomenon, different countries exhibit different patterns of school segregations as they are subject to different group distributions, residential segregation, spatial distribution of schools, and preferences. Additionally, countries follow different policies for the allocation of students to schools. For example, the US and the UK implement catchment areas, that is, geographical areas wherein households get prioritized allocation to specific schools or the existence of private schools. In contrast, systems like that of the Netherlands allow for more freedom of school choice and most schools are publicly funded (Boterman, 2019). The diversity in public policy juxtaposed with the universal outcome of segregation has led researchers to explore the decision process households follow when choosing a school. While there are multiple factors at play, researchers have identified household location and school composition as primary factors in this decision and have been studied in relation to residential segregation patterns (Wilson and Bridge, 2019; Boterman, 2019). Composition, like in the residential process, is important because of choice homophily (i.e., the tendency of people to be attracted to schools with more people of their own group attending) and distance for practical reasons.

Stoica and Flache (2014) augment the Schelling model into one for school choice, by incorporating two preferences in the parent's utility function: candidate school composition and candidate school distance to the household. They find that the existence of catchment areas or high importance to school proximity can abate the effect that the residential distribution has on school segregation and is usually observed. Dignum et al. (2022) built on that work and designed an ABM which models the process of both residential and school segregation to test an alternative hypothesis of why empirical research often finds that schools are more segregated compared to neighbourhoods. In contrast to previous literature they found that this gap could even arise in case of symmetric preferences and households don't need to be less tolerant for school compositions than for residential compositions. In their model they include neighbourhoods - subsets of the grid at which scale the residential segregation is calculated - and schools. Since our work is based on this work we elaborate on it further in Section 3.1.

Measures of segregation

For an extensive review, the work by Royuela et al. (2010) offers a survey of three different measures of segregation. Our experimental work uses the Theil index of segregation due to its decomposability. We therefore limit our discussion to it. Theil's index is a statistical, entropy-based measure of segregation across different locations. Segregation in this case could be with respect to income, ethnicity, or other social demographic characteristic and a location could be any spatial apportion, for example, a neighbourhood or a school. Theil's index ranges between 0 (complete integration) and 1 (complete segregation) and is measured by taking the difference of the maximum theoretical entropy and the observed entropy, normalized by the global entropy (Equations 2.1.1, 2.1.2, and 2.1.3, respectively).

$$T_G = xlog\frac{1}{x} + (1-x)log\frac{1}{(1-x)} \tag{2.1.1}$$

$$T_{Li} = x_i log\frac{1}{x_i} + (1-x_i)log\frac{1}{(1-x_i)} \tag{2.1.2}$$

where x and x_m are the global and local compositions (i.e., proportions) of the group, respectively in a system with $|M|$ spatial apportions, local population n_m, and total population N. In our case, spatial apportions correspond to schools $m \in M$ and therefore local populations are the students attending each school.

$$T = \sum_{i=1}^{|M|} \frac{n_i}{N} \frac{(T_G - T_{Li})}{T_G} \tag{2.1.3}$$

2.1.2 Properties of the considered ABMs

Although the school-choice process is complex and challenging to recover, it is not unique. It belongs to the class of ABMs that represent spatial patterns i.e., spatial ABMs (Manson et al., 2020). Specifically, ABMs where the behavioural rules are defined in terms Random Utility Models (RUM), similar to discrete choice models (Train, 2009). ABMs within this class share properties that can make latent policy recovery from macro-scale summary statistics particularly challenging. Here we list a few of these properties: ABMs are **non-linear** in terms of the outcome measures, **emergent** in terms of the macro-scale characteristics not necessarily being present in the micro-scale, and are **sensitive to initial conditions** (Bonabeau, 2002; Mitchell, 2009).

ABMs model **heterogeneity** in the system units, which requires modelling behavioural rules along various **choice attributes** - for example, in the school choice model, composition and distance characteristics and **agent attributes**, archetypes blue and red (Train, 2009). From an inverse generative perspective, this entails that the latent policy depends on multiple factors to unknown degrees. **Path dependence** implies that decisions in one-time step can have long-lasting effects, making optimal sequences of actions necessary in complex systems (Thurner et al., 2018). In the theoretical system we address - namely, the school-choice model, agent actions can have long-lasting effects, for example, by moving to a school and therefore deterring opposite-type agents from moving there. However, they base their decisions on the current state only; that is, their decision is not subject to any previous moves they have made. Hence, the Markov assumption is satisfied as multiple RL algorithms require (Sutton and Barto, 2018).

The considered systems may have **equifinal** outcomes. Equifinality is the presence of multiple explanations for a single outcome. In the context of complex systems, this may mean that multiple policies are equivalent with respect to the outcome, but, possibly, qualitatively very different (Williams et al., 2020).

We take these properties into account when designing our methodology. However, they make the predictability of complex systems outcomes difficult and, thus, the recovery of micro conditions from those outcomes challenging.

2.2 Reinforcement Learning

Reinforcement Learning (RL) is a family of AI methods for learning sequential decision-making in dynamic environments, that is, learning complex behavioural policies where the outcome depends on a series of decisions. In RL, an agent is situated in an interactive environment *Env* and receives reward signals (e.g., in the form of a scalar) through a reward function based on the outcome of the actions it has taken. (Sutton and Barto, 2018).

Formally, this process is defined as a **Markov Decision Process** (MDP) 5-tuple $\langle S, A, p, R, \gamma \rangle$.

- S is the set of observable **states** containing all possible environment representations. In a fully observable environment this is the true set of states.
- A_S is the set of **actions** the RL agent can choose from in state s
- $p : S \times A \times S \rightarrow [0, 1]$ is a model of the state transition probabilities $P(s_{t+1} = s' | s_t = s, a_t = a)$. The probability of transitioning to state s' in time-step $t + 1$ from previous state s having taken action a in time-step t is given the by this model.
- $R : S \times A \times S' \rightarrow \mathbb{R}$ is the **reward function** which returns a scalar reward $r_{t+1} = R(s_t = s, a_t = a, s_{t+1} = s')$
- $\gamma \in [0, 1]$ is the **discount factor** which determines the discount in the valuation of rewards over time (i.e., rewards received later in time are valued less). For example, $\gamma = 0$ means that the agent only cares about immediate rewards, as future rewards are completely discounted.

Importantly, environments modelled as MDPs satisfy the Markov condition, which requires that a state transition is dependent only on the current state and action and no previous states. Given an MDP and an agent that repeatedly interacts with and explores the environment, we generate a sequence of 4-tuples - namely a **trajectory** of state, actions, rewards, and next states $\tau = \{(s_0, a_0, r_1, s_1), ..., (s_T, a_T, r_T, s_T)\}$.

RL provides a collection of optimisation processes for solving MDPs: An agent learns a policy π through iterated interaction with the environment such that $\pi \approx \pi*$, i.e., the policy approximates the optimal policy. Optimality in a MDP is the maximisation of **return** or the cumulative future discounted reward of a trajectory τ starting at time $t = 0$ defined as:

$$R(\tau) = \sum_{t=0}^{T} \gamma^t r_t \tag{2.2.1}$$

where T is the final time step. In continuous tasks $T = \infty$ but in episodic tasks that we consider T is an integer value. Note that for infinitely long episodes (i.e., continuous episodes), a discount rate $\gamma < 1$ is used to avoid infinite rewards.

A **policy** in the context of reinforcement learning is a decision-making strategy in which $\pi(a|s)$ for $a \in A$, i.e., the probability of taking an action given the current state. The goal of RL, and hence a policy π, is the maximisation of the expected return. Formally, given that the probability of a trajectory τ depends on the policy, the **expected return** is:

$$P(\tau|\pi) = P(s_0) \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t) = \mathbb{E}_{\tau\sim\pi}[R(\tau)] \tag{2.2.2}$$

In considering different approaches to RL, we distinguish between **model-based** and **model-free** approaches. The former are methods of optimising a policy given a state transition probability function. This function may not always be available and must be learned through exploration of the environment. The latter do not require this information and do not use an explicit MDP structure but learn the policy from experience. Therefore, The latter methods can be more flexible and practical if an environment's MDP is unknown or too complex to map accurately. In learning a policy, there are two main methods **value-based** and **policy gradient** methods.In our research we focus on the former.

2.2.1 Value-based methods

Value-based methods learn a value estimate of taking an action at a given state, expressed as the expected return. The goal is to learn a value function $Q(s, a)$ that maps a state and action to an expected return. Through repeated interaction with the environment, the value estimates can be updated in two ways: **Monte Carlo** (MC) and **Temporal Difference** (TD). The former is based on full episode returns, while the latter relies on bootstrapping (i.e., estimates of state-action values). TD has been empirically found to be more sample efficient, although they suffer from overestimating state-action pairs, i.e., maximisation bias (Sutton and Barto, 2018).

An example value-based, TD algorithm is **Q-learning**. The state-action value function is updated using an estimate of the next state value instead of a full return. Since we use the next state's action value estimate, this is referred to as **one-step TD**. Equation 2.2.3 shows the temporal difference equation where a_{t+1} is the action taken in s_{t+1} using the greedy policy i.e., $a_{t+1} = \text{argmax}_{a \in A_{s_{t+1}}} Q(s_{t+1}, a)$

$$\delta = r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s, a) \tag{2.2.3}$$

The value function is then updated according Equation 2.2.4, where α is the learning rate hyper-parameter.

$$Q(s, a) \leftarrow Q(s, a) + \alpha\delta \tag{2.2.4}$$

There exist more distinctions within value-based methods. For example, an approach can be **on-policy** or **off-policy**. On-policy methods update the value function using the same policy with which they interact with the environment, whereas off-policy methods use a different policy. Q-learning (see update rule Equation 2.2.4) interacts with the environment using the ϵ-greedy policy but updates using a greedy policy and is therefore an off-policy algorithm. ϵ-greedy is a method of exploration where the agent explores the state action space by taking a random (as opposed to optimal) action with a probability ϵ.

Another relevant distinction is that between **tabular** and **function approximation** methods. For cases with small state and action sets the q-function can be represented as a table. However, as the size of the state representation increases the number of states increases exponentially. For large state and action spaces the table can be represented by a function approximator such as a neural network. This leads to much better memory efficiency and state space generalisation as similar states return similar values for the same actions (Sutton and Barto, 2018).

2.2.2 Neural Networks

In this project, we use artificial neural networks (ANNs) as function approximators. ANNs are loosely inspired by biological neural networks and based on the perceptron algorithm. In their simplest form, they are implemented as a feedforward multi-layer perceptron (MLP) (Rumelhart et al., 1986). The feedforward architecture means that there are no cycles in the network. The connections between the units are real-valued parameters (also referred to as weights). Following the analogy with biological neural networks, their magnitude can be understood as the connection's strength, excitatory if positive or inhibitory if negative. A unit's activity at layer l is the weighted sum of the previous layer's $l-1$ activities (Equation 2.2.5. This sum can be applied to an **activation function** $g(\cdot)$, which can be non-linear. Networks with hidden layers (i.e., more than one layer) and non-linear activation functions are called **deep neural networks** (DNNs).

$$z_i^l = g(\sum_j w_{ij}^l h_j^{l-1}) \tag{2.2.5}$$

Non-linearity is a critical factor in neural networks which allows them to approximate any function, otherwise, their approximation power is limited to that of linear functions. Furthermore, the layered architecture of such networks is considered to result in hierarchical compositions of the input at different levels of abstractions (Bengio et al., 2009). This means that by using hidden layers, a DNN automatically learns useful features or representations of the input in the first few layers of the network, removing the need for manual feature engineering or in RL constructing state feature vectors.

The final output layer of an MLP can be a value or a vector of values (e.g., action values). ANNs and, by extension DNNs are fitted using gradient descent optimisation methods, which ultimately involve updating the parameters such that some objective is minimised. In the context of RL, the objective is reward maximisation and, therefore, gradient ascent. This process requires defining three components: the objective function, the derivative calculation, and the update method. The first can be computed using a **loss function** (e.g., mean squared error). The loss has to be calculated against some ground truth. In RL, specifically Q-learning, this can be the TD (Equation 2.2.3) which should be minimised. The derivatives are calculated using backpropagation. This algorithm first computes a forward pass of the network (i.e., all the unit activations) and then, through a backward pass, the partial derivative of each parameter with respect to the objective. That is, the effect of each parameter on the final loss is computed. Since the partial derivatives point to the objective's maximisation, the weights are then adjusted by them multiplied by a negative learning rate value α such that the loss is minimised.

We have reviewed the foundational concepts behind our work. In the following subsection we cover work similar to ours in spirit: attempts to integrate AI methods in modelling and recovering decision rules for ABMs using AI. There exists a growing body of research on the combination of AI methods with simulation and modelling in general and RL and ABMs in particular. In the following sections we cover work that falls within two categories: first, work that generally combines learning systems with computational modelling and second, that uses learning systems to recover decision rules and policies — in other words, focusing on inverse generative process.

2.3 Learning and generative models

One of the primary ways to combine RL with ABMs is by replacing or augmenting hardcoded logics or behavioural rules with learned ones. This combination maintains the generative nature of ABMs as the learning process involves optimising for some internal ABM objective, e.g., the utility of individual agents rather than accuracy to the real world process.

An example is the research of Zheng et al. (2020), who leverage RL methods to learn optimal taxation policies in a simulated environment where human-designed agents who are part of a virtual economy interact by producing, selling and buying goods. The RL system uses the monetary health of the simulated economy as feedback to adjust its taxation policy, finally learning the optimal taxation strategy for different economic agents. The objective is not to recover the real-life system's decision process or match some target outcome but rather the optimisation of critical metrics in the virtual economy and insight into what an optimal policy may look like.

The work of Radovic et al. (2022) is another example of multi-agent RL being integrated in a wargame between international oil companies. In this instance, companies observe energy futures and demand/supply metrics and make decisions on energy-type production levels and investments. By playing against hard-coded adversaries that exploit strategies or behave in a business-as-usual manner they identify a series of robust individual strategies.

Sert et al. (2020) focus on Schelling's residential segregation model. Using the DQN algorithm, a DNN based Q-learning implementation, they learn two neural networks corresponding to the two types of agents (red and blue) located on a grid. They use a deep-convolutional neural network to encode the state of an agent with respect to their surrounding neighbours and train the two controllers to make decisions for the agents on the grid - for example, whether to move or stay in their current location, using a neural-network as opposed to codified rules. By introducing different kinds of rewards that resemble societal incentives or agent utility functions in a traditional ABM, they explore the segregation dynamics of the trained agents. They find that despite segregation rewards, integration can be achieved by establishing interdependencies between different agent types. They also find that younger agents are attracted to diverse neighbourhoods more than older ones.

The work of Jäger (2019) is another example of this methodology also applied to a Schelling model of residential segregation. Instead of defining the decision-making logic, they define a utility function which rewards the learning agent for its actions. They show that by using Deep RL they can match the segregation dynamics of ABMs with manually defined rules. Additionally, they show that the learning capacity of those models facilitate the expansion of the models to new environments and therefore research of different segregation dynamics.

Osoba et al. (2020) use RL in ABMs to explore policy decisions and compare them to heuristic models. They demonstrate that RL is a suitable abstraction for behavioural ABMs and outperforms non-learning agents in all experiments, with better generalisation in multi-agent cases. The RL formalism provides a natural abstraction for the reward seeking behaviour of agents allowing researchers to focus on designing sound counterfactual and predictive experiments instead of the internal decision-making structure or parameters. However, their models do not satisfy the Markov condition, which can limit some RL algorithms' performance. Furthermore, the learned policies solely aim to maximise the ABM utility objective without ensuring their real-world validity.

The reviewed work illustrates that RL and ABMs are naturally compatible in terms of formalism: definition of an agent and environment, mappings of states to actions, and presence of a temporal scale and objectives. However, the above approaches pay little attention to the validity of the learned models, focusing instead on the traditional bottom-up approach, the flexibility of the agent, the achievement of the environment objective, and the emergent outcome. Moreover, while these approaches are meant to bypass the complicated process of constructing utility functions, the burden is, in fact, shifted to constructing appropriate reward functions and imputing the latent decision process.

2.4 Learning and inverse generative models

Work in this subsection leverages data-driven AI techniques to improve or ensure the validity of the created models. That means that the ultimate goal of using the model for explanation or prediction using the statistics it generates is preceded by automatically learning or generating policies through minimising the difference between the generated statistics and those of the reference system. Much of this work focuses on the programmatic generation of the agents and models. This kind of work leverages techniques from evolutionary computing and more recently RL to synthesise agent programs that are then run in the simulation, offloading from human researchers the testing of multiple variations and hyperparameters of their programs (Lavin et al., 2021).

One of the first instances of such data-driven ABM work is the development of ABMs for the study of walking and moving patterns. Here Torrens et al. (2011) develop a method for learning action models of movement behaviour given trajectory data. The method involves collecting samples of real-world movement trajectories to train a linear regression and using k-means clustering and nearest-neighbour to integrate into a variety of action models. The goal is for agents to output plausible moving patterns. Results of the learned model are quantitatively compared with the reference real-world trajectory data on different tasks (children walking, adults walking, and cycling patterns). They show that the learned action models are qualitatively and quantitatively similar and generalise to non-sampled situations.

Wunder et al. (2013) collect data on participants who played a series of public goods games where they are required to contribute part of an endowment to a public pool. They fit and compare 1) a small deterministic model and 2) a stochastic model where both predict the a) average participant contribution and b) full distribution of participant contributions. Unlike the stochastic model, the former model does well in the average contribution but fails to fit the entire distribution. This is an important point because it shows that using coarser statistical information may be enough to recover the average behaviour, but not accurate to capture the full diversity of latent policies. Furthermore, they show how the stochastic model generalises to extrapolations of simulation settings. In sum, they show the advantages of learning and validating models on empirical data.

Zhang et al. (2016) follow a similar methodology to the work above and highlight the importance of empirically assessing the model's validity. They develop a framework for learning individual behaviours using machine learning, deploying the learned models in a simulation, and then evaluating the model's predictive performance on a test set. They specifically showcase their framework on a simulation of solar-panel adoption. They compare their model to an ABM, parameterised such that the mean-squared error of its simulated behaviour against the ground truth is minimised. However, they show that their maximum-likelihood model outperforms it. In this work, the decision model is based on the probabilities of solar panel adoption estimated by two logistic regression classifiers which are shared among all deciding agents.

An alternative approach to recovering acting models or decision rules with learning methods is based on program synthesis. As the nomenclature suggests, it is the process of automatically synthesising programs. The synthesiser can be an optimisable module that iteratively improves the output code quality as measured via some objective. An example of such an approach which uses RL to synthesise differential equation models (therefore not ABMs) is the work of Bassenne and Lozano-Durán (2019). Differential equations are used for modelling which, unlike ABMs, are expressed in analytical form and evaluated using a computational solver. The differential equation of choice is a fluid-dynamics equation of which they try to generate a missing term.

Bassenne and Lozano-Durán (2019) formulate the problem as a multi-armed bandit problem (i.e., a category of RL algorithms for single-state MDPs) where the agent consists of a model generator so that its actions are formulations of differentiable expressions. The action is a sampling from a domain-specific language (DSL). Given an action and thus the equation's completion, the equation is passed into a computational solver to run and compare it to a reference expression (the entire ground-truth equation). The reward is calculated based on the difference in the exact solution of both equations, including a complexity penalisation (where more terms lead to a smaller reward). Given this reward signal, they show that the agent can eventually learn to substitute the missing terms of the fluid-mechanics equation correctly.

Previous work focusing on ABMs has used evolutionary methods to evolve logic for a crowd simulation model (Junges and Klügl, 2011). However, the resulting program was meant to be used as inspiration for a modeller rather than a ready-to-use agent. Greig and Arranz (2021) use program synthesis to generate the action logic for two ABMs: an opinion dynamics model and a bird flocking model. They develop a broad methodology for evolving agent logics for ABMs given a DSL. Similar to extensive hypothesis testing, their approach leverages evolutionary computation by using a fitness metric to converge to an agent logic. The fitness is calculated by comparing the candidate agent's behaviour against an output reference model, specifically by calculating the mean-squared error between the two. Behaviour, in this case, is represented as a set of state and action trajectories. Thus, given a state, the mean-squared error is calculated over the difference in actions of the candidate and reference models. The reference systems, as in our case, are hand-designed ABM models which can be run extensively to obtain large amounts of data. While they provide examples of the qualitative resemblance of the learned model and how it generalises, the most significant advantage of this approach is that the output can be human-readable, albeit requiring some modification. However, such genetic and evolutionary approaches tend to be computationally inefficient and prone to overfitting target data. In both cases, it is unclear how the approach would perform for larger systems with agent states of higher dimensions.

Latent policy recovery offloads human labour, can improve a model's accuracy, and may improve the computational efficiency of a model as in Novati et al. (2021) in turbulence models, for example. The most accurate (with respect to spatial resolution) way to model turbulence flows is direct numerical simulation (DNS). However, this requires solving equations at extremely high resolutions (e.g., trillions of points in a flow field's grid). Instead, researchers use large eddy simulation to resolve the flow dynamics at larger scales and use heuristics or statistical models for finer scale points and their interaction with the larger scale. (Novati et al., 2021) replace those lower scale models with a trained controller which models the finer scale flow fields. The controller is trained to maximise the similarity of the statistics of the model it controls to that of a reference DNS model at different parts of the volumetric grid. This approach leads to much more efficient fine-scale modelling, shown to generalise better than approaches that solely use supervised learning. The authors hypothesise that this is due to RL optimising for the long-term reward of its actions which a traditional, single-step supervised learning approach does not, leading to an accumulation of errors in the model. Despite learning a single controller, they refer to their method as a case of multi-agent RL, given that the single controller has different effects on different parts of the environment simultaneously. In a more recent version they show their methodology can be extended to simulate the near-wall (i.e., close to a surface) turbulence dynamics (Bae and Koumoutsakos, 2022).

The most recent example of IGSS in the context of ABMs is the work of Chopra et al. (2022). The authors use DNNs with ABMs to create an end-to-end differentiable epidemiological model. In place of the traditional object-oriented design they use two parameterised models, for computing the probability of transmission between agents and a disease progression model. These models are used to simulate at the level of individual contact networks. The error is computed between the simulation's and real world macro statistics and the models are optimized via gradient descent. One of the key advantages of this process is the efficiency of running this simulation which is effectively reduced to a matrix multiplication between the two models and agent states. Additionally, this model is meant to inversely infer the mechanisms of the disease spread and progression without relying on handmade rules as in traditional ABMs or omitting the individual and network effects as in differential equation models.

The final set of approaches to the inverse generation of models involves two subdomains of RL: inverse RL (IRL) and imitation learning (IL). For a more extensive review of such methods, consult (Torabi et al., 2019). Imitation learning involves learning the state-action value function or policy in a domain by minimising the difference in the behaviour (i.e., action selection) of the learning and expert policies. Imitation learning can be used to improve sample efficiency by first training a model and then optimising the learned parameters through interaction Hester et al. (2018). More advanced methods use even less information (i.e., only state and next-state pairs), making use of causality to infer the latent expert policy (Edwards et al., 2019).

IRL is another methodology used to recover the behavioural objectives of agents given demonstrations of their behaviour by finding the appropriate reward function. In IRL the reward function is modelled as a linear combination of weights and features such that the observed trajectory behaviour maximises the received reward (Abbeel and Ng, 2010). Given the recovered reward function, one can train agents in parts of the state-space of the domain that may not be present in the given trajectories. Lee et al. (2017) present a methodology for using IRL to automatically learn the ABM policy of a system. The steps considered involve the collection of trajectory data, an MDP design, clustering of agent trajectories across behavioural patterns and candidate reward functions, estimating the transition probabilities for agents in clusters, extracting behavioural rules that maximise the expectation of the reward functions, and finally constructing the ABM from the extracted rules. They test their methodology on a simple segregation ABM where agents, split along a 2-level characteristic, can decide whether to move and whether to have a conversation with other proximal agents. In a very similar manner to us, they generate synthetic data using a known ABM and quantitatively and qualitatively compare the extracted behaviours with the reference ones.

While the objectives of the reviewed work resemble ours, we strive to recover a policy given very coarse data about the target process. Approaches like IRL and IL require high-quality, high-resolution (temporal) scale trajectory data which is not always available. For example, in the case of school segregation, the process may take multiple decades to stabilise. Data about school choices may not be consistently available over long periods and at the individual level, which is rarely available due to privacy concerns.

In this inverse generative work, we have seen that most approaches require and exploit trajectory data: fine-scale temporal and individual actions of the reference system. In such circumstances, simple ML methods like logistic regression, supervised-learning-based imitation learning or more advanced and general like IRL may be used. However, we take an approach where the reward signal is modelled on the final state or high-level statistical description of the system (as in Novati et al. (2021)).

Chapter 3

Methods

The following section covers our work's most important methodological details. First, in Section 3.1 we introduce the reference model and policy used in this thesis, which is a computational grid-based ABM of residential and school segregation processes. In Section 3.2, we introduce the implementation of the baseline and learning controllers for the RL process. In Section 3.3, we cover the process of collecting the summary statistic from the reference model and how they are used to run the recovery process. Section 3.4 describes our implementation of the MDP formalism. Section 3.5 covers the reward function implementation separately. Finally, we put all of the above together in Section 3.6 and close with some technical details regarding the implementation in Section 3.7.

3.1 School choice model

This section discusses the ABM used in this project - namely, the COMPASS model. In the context of the inverse latent policy recovery with RL, the ABM serves three functions: First, it implements the reference model(s), which we recover via our method. Its second purpose is to serve as the school-choice environment simulator for the RL controller. Finally, it serves as the ABM in which we incorporate the recovered policies for testing. The ABM is implemented by Dignum et al. (2022) based on the work of Stoica and Flache (2014), Sage and Flache (2020), and Schelling (1971). The purpose of the model is to help study the emergent dynamics of residential and school choice and the emergent segregation outcomes.

The processes can be simulated on grid environments and real city geographical data. It is assumed that residential choice precedes school choice and thus runs the two processes in sequence, with the former preceding the latter. Both processes include a random initialisation schedule, i.e., the allocation of agents to random residential locations and schools in the first step. In the following steps until the preset maximum or convergence (defined as a minimal change in average agent utility) a subset of agents is chosen to act using their behavioural rules. In the residential process, that action may be to change neighbourhoods and, in the school process, to change schools. Numerous parameters can be used to configure the model. Dignum et al. (2022) extensively analyses the available simulation parameters and their influence on the residential and school process outcomes. The parameters that affect our experiments are listed in Table A.1 of the Appendix A.1.

Our RL pipeline relies on a set of summary statistics S which is also obtained using methods of the reference model. We refer to each summary statistic as a target vector $\mathbf{t}$. Those are the school composition $\mathbf{t^c}$, average household distance $\mathbf{t^d}$, and attendance proportion $\mathbf{t^a}$ vectors. All target vectors have a norm equal to $|M|$. That is, they contain the relevant statistic for each school. We elaborate more on those targets in 3.5. The emergent outcome in both processes is segregation or lack thereof. The simulator measures the level of segregation according to Theil's index (see subsection 2.1.1) either in its decomposed or global form.

The COMPASS model implements a random utility model policy for every household. Since this utility function U, as implemented for the school choice model, is the latent policy we attempt to recover using RL, we discuss it in detail here. For each household $i \in H$, the utility of a school is calculated based on a combination of two factors. First, the utility from that school m composition c_{mi} i.e., the proportion of same-as-i-type agents over all attending agents. Second, the distance d_{mi} from household i to the school m. The two are linearly weighted using the $0 \leq \alpha \leq 1$ parameter. The utility of a school m for a household i is defined as per Equation 3.1.1:

$$U_{mi} = \alpha C_{mi} + (1 - \alpha) D_{mi} \tag{3.1.1}$$

The distance utility factor is linear and defined as :

$$D_{mi} = 1 - \frac{d_{mi} - d_{min,i}}{d_{max,i} - d_{min,i}} \tag{3.1.2}$$

where d_{mi} is the Euclidean distance from i to school m and $d_{min,i}$, $d_{max,i}$ is the distance to the closest and furtherst school to i, respectively. The composition utility factor defined using a single-peaked utility function:

$$C_{mi} = \begin{cases} \frac{x_{mi}}{t_i} & \text{if } x_{mi} \leq t_i \\ \mathcal{M} + \frac{(1-x_{mi})(1-\mathcal{M})}{1-t_i} & \text{if } x_{mi} > t_i \end{cases} \tag{3.1.3}$$

Here x_{mi} is the proportion of agents of the same type as i in school m. M is a factor which affects the utility to agents when the proportion is larger than the optimal fraction. An $\mathcal{M} = 1$ would signal indifference, but one may want to model agents seeking diversity.

Note that with the α parameter, we can alter the structure of the utility function: for $\alpha = 0$, the utility function is equivalent to Equation 3.1.2 while for $\alpha = 1$ it is equivalent to Equation 3.1.3. Other parameters relevant to the school-choice process but independent of the reference policy itself include the existence of capacity constraints in the schools, both in terms of the minimum number of students (which implies that attending students cannot leave) and maximum (which means no new students can attend). These can have a substantial effect on the processes' dynamics as they may block agent actions.

3.1.1 Model assumptions

There are numerous assumptions in the design of the school choice model. For a better understanding, the reader is referred to the original work of Dignum et al. (2022). For the purposes of our research, we further limit the operating parameters of the model to consider simple cases. For example, we allow schools to be empty (i.e., minimum capacity of 0) such that households can freely move around while maintaining a sizeable maximum capacity. We also consider identical behavioural rules for both agent archetypes (i.e., blue and red) parameterised with an optimal fraction $t_i = 0.6 \; \forall \; i \in H$. The utility functions we consider are also non-peaked ($\mathcal{M} = 1$). This means that if the composition of a school is higher than the optimal fraction that still leads to maximum utility. In turn, this entails that the process will likely lead to complete segregation. Finally, we randomly distribute the agent types such that their proportions are close to 0.5. Table A.1 in Appendix A.1 lists all parameters used in our method. The only parameters we vary in our experiments is the α value and the number of households $|H|$. We elaborate on those in the experimental setup Section 4. In general, we consider the extreme cases to be the simplest to recover as they are likely to converge to extreme outcomes and hence clear differences in the outcome summary statistics.

3.2 Controller implementations

This subsection describes the implementation of the controllers used in the RL environment. We implement two types of controllers: non-learning and learning ones. The former are used as baselines or benchmarks for the latter. The learning controller is a typical RL algorithm used in discrete-action spaces - a DDQN.

3.2.1 Baselines

Oracle model

The Oracle model implements the utility function in 3.1.1. It takes the α parameter as input to compute a value ranking according to the state vector. For example, in a scenario where reference policy is parameterised with an $\alpha = 0$, the Oracle model will output a school ranking in increasing order of the school distances (i.e., closest school first and furthest school last).

Even though the Oracle implements the reference behavioural rule, which is what the training process aims to recover, it is not guaranteed to receive maximum reward. The reward function is built on certain assumptions regarding the system's stochasticity and dissimilarity computations with the reference. There is no established reward function design for our task therefore it is subject of research and experimentation in itself. Basically, there may be a discrepancy between what we want the reward function to measure and what it actually measures. Consequently, the Oracle plays a crucial role in the design of the reward function during preliminary experiments. Ideally, a good reward function would be optimised under the Oracle's policy control.

Random choice model

As the name suggests, outputs a random ranking of schools, regardless of the household that is being moved. This sets a baseline against which we can compare the learning model.

3.2.2 Learning algorithm

Our RL agent is an implementation based on the Double DQN (DDQN), the algorithm behind many recent RL breakthroughs (Mnih et al., 2013). DQN is based on the Q-learning algorithm introduced in Section 2.2 but uses a deep neural network for function approximation. The key idea behind the DQN is to use the Q-learning update step 2.2.4 as the loss for stochastic gradient ascent.

We use experience replay for training. Experience replay introduces a memory buffer, a store of episode transitions. During training, transitions are pushed to the buffer. During optimisation, those transitions are uniformly sampled to compute the loss and update the DQN's parameters. One of the advantages of this process is that it maintains the i.i.d. assumption required for ML algorithms like backpropagation in DNNs since uniform sampling breaks any correlations between states in the gradient update (Sutton and Barto, 2018).

Van Hasselt et al. (2016) proposed DDQN, which is the algorithm we implement in this project. DDQN was proposed to address the maximisation bias present in Q-learning due to the use of the maximum expected action value in the temporal-difference computation. In DDQN, we maintain two DNNs, the online and target networks. These are used as separate estimates of the action value function Q^o and Q^t parameterised as θ and θ', respectively. The online network is used in the evaluation of the greedy policy. However, the target network is used for the estimation of its value as expressed in the formulation 3.2.1. The target network's parameters θ' are replaced with those of the online network θ every n steps.

$$Q^o(s,a) \leftarrow Q^o(s,a) + \alpha[r + \gamma Q^t(s', argmax_a Q^o(s',a)) - Q^o(s,a)] \tag{3.2.1}$$

The entire DDQN algorithm can be read in pseudocode 1, adapted from the work of Mnih et al. (2013).

Algorithm 1 Double Deep Q-Network (DDQN) with Experience Replay

```
Require: Environment with state $S$, action space $A$, reward function $R(\cdot)$,
    discount factor $\gamma$, and terminal state $T$.
Require: Q-networks $Q^o(s, a; \theta)$ and $Q^t(s, a; \theta')$ with parameters $\theta$ and $\theta'$, respectively.
Require: Replay memory $D$ with capacity $C$.
Require: Exploration policy $\epsilon$-greedy with decay schedule.
 1: Initialize Q-network $Q^o$ and target network $Q^t$ with random weights $\theta$ and
    $\theta'$, respectively.
 2: Initialize replay memory $D$ to capacity $N$.
 3: Initialize time step $t = 0$.
 4: for episode do
 5:     while $t < T$ do
 6:         Observe state $s_t$.
 7:         Choose action $a_t$ using $\epsilon$-greedy policy.
 8:         Take action $a_t$ and observe reward $r_t$ and next state $s_{t+1}$.
 9:         Store transition $(s_t, a_t, r_t, s_{t+1})$ in replay memory $D$.
10:         Sample mini-batch of transitions $(s_i, a_i, r_i, s_{i+1})$ from $D$.
11:         Compute target Q-value for each transition:
                $$\hat{y_t} = r_t + \gamma Q^t(s_{i+1}, \operatorname*{argmax}_a Q^o(s_{t+1}, a; \theta); \theta^{'}).$$
12:         Update Q-network parameters by minimising the loss on all the sam-
    pled transitions:
                $$\mathcal{L}(\theta_i) = \mathbb{E}(s_i, a_i, r_i, si + 1) \sim D\left[(\hat{y_i} - Q(s_i, a_i; \theta_i))^2\right].$$
13:         Every $C$ steps, update target network parameters: $\theta^- \leftarrow \theta$.
14:         Increment time step $t$.
15:     end while
16: end for
```

In contrast to early claims in the literature, recent work has shown that RL algorithms, in general, and DQN approaches in particular, are very sensitive to hyperparameters such as the learning rate, target network update frequency n, ϵ decay and more (Henderson et al., 2018). We observed this sensitivity to hyperparameters in our runs as well. Therefore we run a hyperparameter search based on a random search for multiple seeds and multiple environment setups to settle on the final parameters reported in Table A.2.

The DDQN gives us the expected reward over schools. For action selection we use the ϵ-greedy algorithm, where the threshold ϵ decays exponentially with the number of training steps. Equation A.2.1 in Appendix A.2.1 formulates our decay implementation and more details on the process.

3.3 Data

The reward signal depends on the outcome summary statistics of the reference system. Thus we require these summary statistics along with information about the system's setup, such as, for example, the geolocations of the households (which is relevant to the school process). This project is limited to computational, grid-structure reference models based on the work of Dignum et al. (2022), which we use as a synthetic data generator. We have complete control over the reference model and can use it to explore how our learning pipeline performs for different reference conditions.

To make the data collection and experimental process easy to understand, we define two terms: **scenario** and **instances**. A scenario is a particular configuration of reference system, including the reference policy. An example of a real-life scenario is the city of Amsterdam and the school-choice policies of its different household types. We can be more precise in a computational model over which we have complete control: A scenario is a particular system configuration, e.g., $\alpha = 0.2$, 100 households, and a low tolerance $t = 0.2$.

An instance is an instance of that scenario. While the scenario defines the process, there are still some random factors, for example, the exact distribution of different types of agents, the residential segregation pattern, initial random school allocations, and the sequence in which agents move. Thus a scenario can give rise to a variety of instances, and depending on the scenario, these differences may be substantial.

To clarify the distinction between instances and scenarios, we provide an analogy with the city of Amsterdam. Any neighbourhood of Amsterdam may be considered an instance because we assume the same individual policies and school constraints apply to the households, but there may be different residential patterns or initial school allocations - for example, a new neighbourhood, recently populated, where all schools are new. In the context of the school-choice simulator, instances are unique instantiations of the environment, distinguished by different random seeds and therefore residential and initial school allocations. Nonetheless, all instances are run with the same utility function and number of agents.

Consider the scenario where $\alpha = 0$ and thus the reference policy (utility function) is an inversely proportional, linear preference over distances. In such a case, the school process is entirely determined by the residential process and thus mirrors the residential segregation (provided there are no strict school capacity constraints). Households will always choose their local school. Different instances of this scenario will therefore be completely different: an instance with low residential segregation will result in low school segregation, whereas a highly segregated grid will lead to a high school segregation.

Obtaining the training data

The training data consists of the reference outcome summary statistics generated under the parameters listed in Table A.1. Complex system outcomes are sensitive to initial conditions. Therefore, we store information such as the exact distributions of agent archetypes and converged residential patterns of the reference run to make the sample as similar to the reference one, with respect to the initial conditions. We configure the school-choice model to the desired scenario to obtain the training data and run it to record the final residential pattern and school outcome summary statistics. We do this 25 times with different random seeds to generate more instances for the same scenarios. Note that the initial allocation of households to random schools is also part of the initial conditions. However, we omit this information and therefore the sample system does not have the same initial school distributions as the reference.

We make the decision to discard the initial school allocations because we assume that a feasible inverse generative method should work in the absence of this data. That is, we consider this data to be entirely unavailable in the real process because is unclear what the 'start' of the real system would be and how this data would could be estimated. In contrast the residential distributions can be estimated through aggregated data on a neighbourhood scale. This data, for the city of Amsterdam, is publically available (e.g., in (Municipality of Amsterdam, 2023)).

Once we generate the data, we can load it to train and evaluate our RL method. We implement a module that loads this data and reconstructs the grid by assigning the reference model's converged residential geolocations to the appropriate households. This makes the training process more versatile as we do not have to rerun the generative reference model when running the inverse process. In the loaded grid, a household i will reside at the exact same x, y coordinates in the reference and sample models. We can then assign each household to a random school and use the RL controller as the school-choice policy to run the school process. The sample model is run, and rewards are computed based on the dissimilarity with the recorded outcome statistics between the reference and the sampled learning model.

3.4 MDP design

Since we only use model-free approaches, the exact specification of the MDP dynamics is not useful. Generally, the state transitions happen with probability 1 unless some capacity constraint is encountered, in which case the transition does not happen. Notwithstanding, we elaborate on the design of the state, observations, and other MDP factors while we expand on the reward function in the following subsection 3.5

Note that the agent does not fully receive information on all the summary statistics. That may tempt one to consider our problem formally as a Partially Observable MDP. However, that is not the case, as POMDPs imply noisy observations of system states or the collapse of multiple states into identical observations. Both issues can lead to the Markov assumption not holding (Spaan, 2012). Although not all summary statistic targets are part of the state space, both POMDP characteristics are not present in our system. Some targets are a ommitted on the grounds that they are irrelevant or inaccessible to the individual agents and therefore not specified in the agent interface. Thus, the state contains all the information that is given to the reference policy (utility function).

- **Environment:** The learning and simulation environments are the same and consist of a set of spatial units which can hold a school m, a household agent i, or nothing (i.e., an empty location). The areas are cells in an $H \times W$ grid. In every environment, M and H are the set of schools and households, respectively. Each school $m \in M$ can be described by various features including, but not limited to, its location x, y coordinates, composition $c_m \in [0,1]$ and the Euclidean distance to a household i, $d_m i \in [0, H \times W]$. D is the $|H| \times |M|$ matrix of school to households distances. Unlike grids in previous research in the school process, we assume non-periodic boundaries.

- **State s**: The state vector must be carefully designed to include all relevant information to the households' decision process. It must also match all information available to the reference decision-makers. Of course, in our case, that is trivial to obtain, but in a real-world application, this decision must be informed by prior research. For a system with $|M|$ we have a state vector $\mathbf{s} \in \mathbb{R}^{2|M|+2}$. Each state vector includes the agent's type, the agent's current school, and for every school, its distance from the agent and its composition. The school's composition c_m is the number of agents enrolled in it of the same type as the agent whose state is being computed, divided by the total number of students in that school. The composition and distances are real-valued, so the state space is continuous.

- **Action** $a \in A$: The action of the controller represents the choice to move to a school m. For a given household, the controller can choose to move to the household's current school and therefore not move. Therefore, the number of actions $|A| = |M|$.

Note that in our experiments, the type (blue vs red) of an agent is obsolete with respect to the latent policy though relevant in discriminating among choices. That is, it is a factor in the agent actions but there we consider scenarios where all agents follow the same policy. The compositions in the state vector are computed such that they always reflect the composition as a ratio of $c_m = \frac{\text{self-type}}{\text{total}}$ and thus is implicit in the state vector information.

As aforementioned, there can be constraints to the execution of actions, for example, if a school is at maximum capacity. We briefly illustrate how we handle such action blocks. While algorithms like ϵ-greedy prescribe a single action to take, we use the full action set A in the simulator. Recall that the output of the controller in value-based methods is the expected reward, and thus we interpret the full output $\mathbf{Q(s)}$ where $\mathbf{Q(s)} = [Q(s,a_1), Q(s,a_2), \ldots, Q(s,a_n)]$ as the value of moving to every school $m \in M$. The simulator takes as input the full action-value vector.

In the context of ϵ-greed: If the random action is triggered then we shuffle the action-value output. If the optimal action is to be taken then we sort them from highest to lowest expected value. The simulator receives as input this ranking and moves the agent to the first school. If the move to is blocked, then a move to the second school in the ranking is attempted. This process continuous until the agent is assigned to a school.

How this is handled during training is important for learning. The simulator returns the chosen action (i.e., the school to which the agent was moved), and it is this action that is inserted into the replay buffer for training. Thus, if the optimal move is not allowed from the perspective of the learning controller, this is equivalent to an exploration move and should theoretically lead to an appropriate valuation of all schools, given sufficient training.

We set reward discount factor $\gamma = 0.995$. This is a typical, although marginally large value for this factor. With a high gamma we increase the value of longer term rewards, which makes sense in a sparse reward environment. The exact details of the reward function are critical. We elaborate on them in the following section.

3.5 Reward function

The reward function outputs a scalar which aggregates the dissimilarity computation between the reference and sample model summary statistics S and $\hat{S}$. The reference output is the terminated ground-truth policy, while the sample model is the one controlled by the RL agent. The reward is computed for different summary statistics, which we refer to as targets for short. There are three targets τ: the school composition vector, the average household distance vector, and the attendance proportion vector. Note that we use the hat notation (e.g., $\hat{\mathbf{t^c}}$) to refer to target vectors of the sample system.

- The school composition vector $\mathbf{t^c}$ contains, for each school, the number of type 0 (blue) agents over the number of agents that attend that school.
- The average household distance vector $\mathbf{t^d}$ contains, for each school, the average distance to all the attending households from taht school.
- The attendance proportion vector $\mathbf{t^a}$ contains, for each school, the number of households that are enrolled to that school over all households in the system.

We chose these targets as our summary statistics based on preliminary experiments and the following observations: We require that the targets change linearly to the controller actions. For example, using Theil's decomposed index which is based on the non-linear entropy calculations would change differently at different value ranges producing an unclear dissimilarity signal. Instead we use averages and proportions. Moreover, the summary statistics should convey information that is causally related to the latent reference policy. All our targets change subject to agent choices. An example of a summary statistic independent to the policy would be the average distance between schools. Furthermore, due to equifinality (see 2.1) having multiple targets facilitates discrimination between latent policies. To illustrate, consider two latent policies: one parameterised with an $\alpha = 0$ - where only distance is taken into account and another parameterised with $\alpha = 1$ where only the school compositions are taken into account. Given an environment that is residentially highly segregated both outcomes lead to high school segregations. Consequentially, the composition target vectors describing both systems show extreme composition values. Both policies are equifinal with respect to composition. However, if we include another target, for example the average distances, we observe a difference. The system subject to the former policy has lower values of all school. Hence the two policies can be discriminated.

There are two more reasons for which including multiple targets is beneficial for learning: First, we want to demonstrate that the learning is robust to reward signals dependent on information that is not exclusively used in the latent decision rule. The second reason is that in preliminary experiments we identified some mode-collapses where the network pushed the system to extreme states (e.g., all students in one school) which would maximise reward but lead to inaccurate policies. This is a case of shortcut learning (Geirhos et al., 2020). Including information such as attendance leads to optimal rewards only when a plausible system is reached.

The sparseness of the reward signal is another critical design choice. Given that we are computing the dissimilarity between outcome summary statistics which are often describing emergent and non-linear processes, we provide a reward signal at the end of every episode, making this a sparse reward task. Potential-based reward shaping approaches discussed by Ng et al. (1999) are not compatible precisely due to the non-linear dynamics of complex systems: a reward signal at each step would require the learning controller to get to the outcome summary statistics at each step which is not what the reference policy is doing. One of the implication of non-linear dynamics is that the dissimilarity between some of the reference and sample targets cannot monotonically decrease, even under reference policy control. The random sampling of moving agents also plays a role: an agent may be moved to the correct school, but because not all agents have moved this may worsen one of the other target metrics e.g., the attendance proportions. Hence we can only compare the reference outcome summary statistics with the sample outcome ones.

The reward function is implemented as follows. We measure the dissimilarity D between reference outcomes and sample summary statistics, at the episode's initialisation ($t = 0$) and at the episode's termination ($t = T$). If the dissimilarity at termination is less than at initialisation (i.e., a negative gradient) and the magnitude of the change is above a certain sensitivity threshold ϵ then a reward of $+1$ is given. If the gradient is positive and the magnitude larger than the ϵ threshold then a -2 reward is given. The asymmetry between reinforcement and punishment empirically found to work better, as it helps the DDQN discriminate between good and bad outcomes. To balance this asymmetry we introduce an accuracy threshold α. That is, if the dissimilarity gradient is negative (i.e., dissimilarity has decreased) and the termination dissimilarity is below α a positive reinforcement of $+2$ is given. This helps the DDQN discriminate between sequences of actions that lead to the right direction of change and those that lead to the actual correct outcomes. This process is calculated for every target and for a predefined dissimilarity metric.

There are two important design decisions here, and we elaborate on the motivation behind them: the use of gradients and thresholds. The motivation for the former was to avoid rewarding 'lucky' initialisations but focus on the controller's improvement. The use of thresholds has two advantages: first, it only considers larger changes and, therefore, better sequences of actions, which can expedite learning. In the case of the accuracy threshold, it accounts for minor noise in system outcomes and dissimilarity calculations. Second and related, it stabilises the reward signal by allowing mapping to integer scalars. Using the dissimilarity values directly was attempted but led to very noisy reward signals. A limitation of this decision is that we introduce reward function hyperparameters that have to be set for every measure of dissimilarity and every target. Here, it is important to note that the exact values of the thresholds were set empirically but not as a result of systematic testing.

Furthermore, the exact choice of reward values was empirically motivated. In particular we noticed that the learned DDQN's expected values of most schools for numerous distinct household states were all positive. Generally, even random policies where able to achieve positive rewards. We therefore make the reward for negative similarity gradients that are larger than *epsilon* to have an equal value to the positive and accurate gradients and facilitate the discrimination between good and bad action sequences.

The dissimilarity function D is a critical component and has important implications for what is learned. D takes as input the reference sampled targets in the summary statistics and applies a dissimilarity measure to each summary statistic. The resulting dissimilarity values are used by $R_{gradient}$ to determine the reward. The reward function design is formally defined in the following equations: Equation 3.5.1 defines the computation of maximal reward given to accurate outcomes for a target τ. Equation 3.5.2 defines the gradient reward computation.

$$R^{\tau}(\hat{S_0}, \hat{S_T}, S_T) = \begin{cases} +2 & \text{if } D_T^{\tau} \leq D_{\alpha}^{\tau} \\ R_{gradient}^{\tau}(D_0^{\tau}, D_T^{\tau}) & \text{else} \end{cases} \tag{3.5.1}$$

In Equation 3.5.1 D_t^{τ} is the shorthand notation for $D_t^{\tau}(\hat{\mathbf{t}_\mathbf{t}^{\tau}}, \mathbf{t}_\mathbf{T}^{\tau})$ i.e., the dissimilarity of the sample system's target τ at time t with the reference system's outcome target τ.

$$R_{gradient}^{\tau}(D_0^{\tau}, D_T^{\tau}) = \begin{cases} -2, & \text{if } D_T^{\tau} - D_0^{\tau} \geq +\epsilon \\ +1, & \text{if } D_T^{\tau} - D_0^{\tau} \leq -\epsilon \\ 0, & \text{else} \end{cases} \tag{3.5.2}$$

At the last time step, the target rewards are computed by applying the dissimilarity measures to each target and summing the resulting values. These target rewards are then aggregated using a summation and normalised, as shown by the following equation:

$$R(\hat{S_0}, \hat{S_T}, S_T) = \frac{R^c(\hat{S_0}, \hat{S_T}, S_T) + R^d(\hat{S_0}, \hat{S_T}, S_T) + R^a(\hat{S_0}, \hat{S_T}, S_T)}{6} \quad (3.5.3)$$

Here, R^c, R^d, and R^a represent the target rewards computed using the dissimilarity measures for the summary statistics related to the system's composition, spatial distribution, and attendance distribution, respectively. Note that only the outcome summary statistics of the reference system are available. This is one of the key challenges faced by our approach, due to the lack of high resolution data. We notate the reference outcome S_T i.e., termination though of course a real reference system cannot be said to terminate.

Given a maximum reward for each target summary statistic the total maximum is $(3 \times 2 =)+6$ and in the minimum cases $(3 \times -2 =)-6$. By normalising, we get $+1$ and -1. The resulting reward scalar $R(\cdot)$ is used to update the RL agent's policy parameters in order to maximise the expected cumulative reward over time. Normalising the reward is useful because it leads to smaller gradients and a more stable gradient descent process and is common practice in RL.

Note that different targets in the summary statistics have different values and vary differently with the controller actions. Therefore, the sensitivity gradient threshold ϵ and the threshold accuracy α are defined differently for each. The gradient thresholds also depend on the number of moving agents, as moving more agents at a time can change the sampled summary statistics more. Table A.3 in the Appendix A.2 overviews the reward function parameterisation for the single-moving agent cases. Another implementation detail which we found to be critical for learning is the handling of undefined values. An example is the computation of the average household distance for an empty school. While these are generally rare cases, in the extreme scenarios that we run they can occur frequently. In the Appendix A.2 we cover in detail the handling of undefined values in the summary statistics.

Since computing a reward from summary statistics is a critical component of an inverse generative pipeline, the definition of the dissimilarity function D is the subject of experimental investigation. In our experiments, we consider two different implementations of the dissimilarity function, each having different implications for the inductive bias of the agent. Those are the Wasserstein distance and Mean Squared Error (MSE).

3.5.1 Wasserstein distance

The Wasserstein distance, based on the idea of optimal transport, computes the distance between distributions. Thus in our case the target vectors are treated as empirical distributions. Formally, given two probability distributions $\mathbf{t}$ and $\hat{\mathbf{t}}$ the Wasserstein distance is:

$$W(\mathbf{t}, \hat{\mathbf{t}}) = \inf_{\pi \in \Gamma(\mathbf{t}, \hat{\mathbf{t}})} \int_{\mathbb{R}\times\mathbb{R}} |x - y| d\pi(x, y) \tag{3.5.4}$$

where $\Gamma(\mathbf{t}, \hat{\mathbf{t}})$ is the set of all joint distributions over $\mathbb{R}^2$ whose marginals are $\mathbf{t}$ and $\hat{\mathbf{t}}$, x and y are the elements of the target vectors, and $d(x, y)$ is the Euclidean distance metric over $\mathbb{R}$. The Wasserstein distance measures the minimum amount of 'work' necessary to transform one distribution to the other. The cost is proportional to this work. Since this is a distribution distance metric the minimum is 0 but there is no upper bound.

In Wasserstein distance, each school is interpreted as an observation and the target vectors as random variables. We choose the Wasserstein distance specifically because of the following three properties: First, it considers the probability and the distance between two outcomes implying that it respects the geometry of the distributions. Second, it can handle continuous values (Panaretos and Zemel, 2019). Finally, by treating the target vectors as empirical distributions this measure does not assign importance to the order of the vector elements. It considers matching distributions of outcomes and not exact outcomes. In our case, this entails that it does not matter which exact school is described by a particular value in the reference-system. Effectively, the spatial information between the reference and sample systems is lost. This is important in complex systems where initial conditions and path dependency can lead to different outcomes, but a similar distributions of outcomes. For example, a latent policy parameterised by $\alpha = 1$ entails that high composition values in $\hat{\mathbf{t}}^{\mathbf{c}}$ are probable but not in the exact same schools in every run. Which schools will have high type-0 compositions partly depends on initial conditions.

3.5.2 Mean-Squared Error

Given two target vectors $\mathbf{t}$ and $\hat{\mathbf{t}}$ of length $|M|$, containing summary statistic information for each school, the Mean Squared Error (MSE) between them is defined as:

$$MSE(\mathbf{t}, \hat{\mathbf{t}}) = \frac{1}{|M|} \sum_{i=1}^{|M|} (\mathbf{t}[i] - \hat{\mathbf{t}}[i])^2 \tag{3.5.5}$$

where $\mathbf{t}[i]$ and $\mathbf{t}[i]$ denote the i^{th} elements of the vectors. The MSE measures the average squared difference between the two vectors and is calculated for each reference-sample pair of target vectors. A smaller MSE implies higher similarity.

MSE has three properties that are of interest to our application: Firstly, it is exponential in nature, which implies that high divergence between two particular schools will substantially impact the error. Second, the error is calculated between the two specific schools in the reference and sample systems. That is, the exact difference between a school (which is associated with specific spatial coordinates) is computed. This retains information regarding location, which is important in cases where the reference policy depends on spatial information, such as distance. Finally, as the error is an average, the MSE's range remains theoretically independent of the number of schools. This eliminates the need for a new reward function hyperparameterisation of the various thresholds for more complex scenarios with more schools. That said, in practice, changing the number of schools can affect the learning dynamics and lead to higher average errors than one may usually get in cases with fewer schools.

3.6 Training and evaluation

The goal of the RL controller is to maximise the discounted return in expectation. Given that our reward is computed on the similarity of the reference and sample statistics S and $\hat{S}$, respectively, the assumption is that the optimal policy π^* (which should be the reference, latent policy) is the one that leads to an outcome most similar to S. The problem of recovering the school-choice policy is implemented as an episodic task. The terminal episode is determined empirically based on the number of steps rather than being determined by a specific terminal system state due to the absence of any convergence measures.

On a high level, the training process involves iterating over three steps, comprising of three key modules. Figure 3.1 offers a schematic overview of this process. First, an instance is loaded from the scenario to be recovered, along with all relevant summary statistics. As described in 3.3 the actual reference process is not run during training. This comprises the reference system (light blue). Second, the school choice model is run using the policy of the RL learning controller. This is the sample-system (green-grey). Third, the reference statistics S are compared with the sample statistics $\hat{S}$, and a reward is computed (orange). The policy is then improved based on that reward. The process is repeated, starting at step two on the same instance, but with a new random school initialisation, i.e., households are assigned to random schools at the start of each episode. Pseudocode 3.6 provides a detailed overview of the training and evaluation loop i.e., the sample system process.

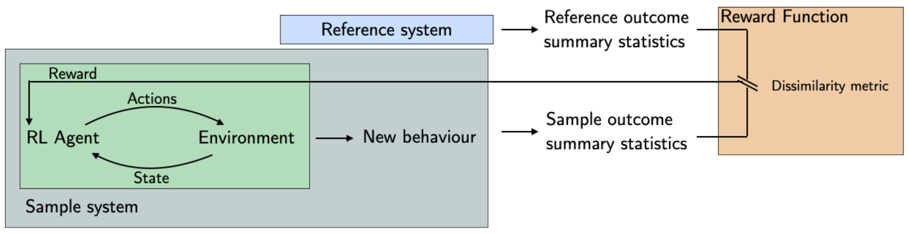


Figure 3.1: A schematic illustration of the proposed RL pipeline for discrete-choice model discovery.

The evaluation loop is the same with the omission of the optimisation step and use of the optimal policy with no exploration. The evaluation loop is ran to track the learning progress and evaluate the learning optimal policy using the evaluation loop.

Each episode represents a simulation and each step is a controller's decision(s) for the moving agents. Given a sampled household's state, the output of the DDQN is the value of moving to each of the $|M|$ schools. We treat the full output as a ranking over schools. The simulator assigns the household to a school according to the logic described in 3.4. We found that using simple random sampling to select the acting agents without a really high number of steps lead to very unequal number of moves per households.

Algorithm 2 Training loop

```
1: Simulator($H, M, \cdot$)    ▷ Simulator with a set of households, schools, and other scenario-defining information
2: policy $\bar{\pi}_\theta(\cdot)$ initialised with random weights $\theta$
3: $R(\cdot, \cdot)$    ▷ The reward function
4: $T \leftarrow$ maximum number of steps
5: households $\leftarrow$ shuffled household array
6: for episode do
7:     $G_t \leftarrow 0$
8:     initialise the simulator based on an instance (randomly allocate households to school)
9:     load reference statistics $S = \{\mathbf{t}^c, \mathbf{t}^d, \mathbf{t}^a\}$ from instance
10:    collect sample summary statistics $\hat{S_0} = \{\mathbf{t}^c, \mathbf{t}^d, \mathbf{t}^a\}$
11:    while step $< T$ do
12:        move_batch $\leftarrow f$ agents from households based on sliding window
13:        for agent $\in$ move_batch do
14:            a, school_ranking $\leftarrow \bar{\pi}_\theta$(agent.state)
15:        end for
16:        Simulator.update(a, school_ranking)
17:        collect sample summary statistics $\hat{S_T} = \{\mathbf{t}^c, \mathbf{t}^d, \mathbf{t}^a\}$
18:        $reward \leftarrow \begin{cases} R(\cdot, \cdot) & \text{if } t == T \\ 0 & \text{otherwise} \end{cases}$
19:        append all relevant information to buffers for training
20:        $G_t \leftarrow G_t + reward$
21:        update weights $\theta$ using gradient-based optimization
22:
23:    end while
24: end for
```

The selection of moving agents, a random sampling process taking place in line 11 and implemented as follows: First, the array of households is shuffled. Throughout an episode, we iterate over the shuffled array and exclusively select as many agents as we want to move in each step (defined by the `max_move_frac` parameter). For all experiments, we use the singleton moving agent batch. This decision is empirically informed based on evidence from our preliminary runs, although not systematically evaluated. However, using a larger moving-agent batch size, the system will resemble the real processes' dynamics better (as agents change schools simultaneously) and reduce the number of training steps (as we move all agents in fewer steps).

Note how, between training and evaluation episodes, the scenario, i.e., latent reference policy, summary statistics, position of schools, and residential choices of households remain constant. Only the initial school allocation of households changes. This is a complicated choice as different reference policies are sensitive to initial conditions in different ways. For example, a reference policy implemented as a distance-only utility function is deterministic given the residential decisions of households. On the other hand, a composition-utility policy is very sensitive to initial conditions, such as the initial household allocation to schools and random batching of moving agents. The data available from the reference system in the sample system initialisation was discussed in 3.3.

3.7 Software, hardware, and implementation

The code for this project is available at COMPASS repository under the *policy-learning* branch (https://gitlab.computationalscience.nl/edignum/compassproject/-/tree/policy-learning). Following standard practice in AI and the existing ABM, the language of choice was Python (specifically, version 3.9), and the code is dependent on the usual scientific programming stack for Python used for statistics, vectorised calculations, visualisations, and plotting. PyTorch was used to implement neural networks and automatic differentiation modules. Particular attention was paid to logging the performance and behaviour of the RL controllers in the environment to facilitate debugging and experimental analysis. For this purpose, the Tensorboard package was used. Finally, the ABM is implemented in the Mesa Python library. The experiments were run on a 36 core Intel Xeon Platinum 8360Y CPU and Nvidia A100 GPU, provided by Snellius, the Dutch national supercomputer cluster.

The codebase follows an object-oriented design by implementing various modules, such as the learning agents and reward functions, as objects. This architecture helps keep the code modular, intuitive, and consistent with the existing ABM architecture even as the codebase grows. We also ensure consistent documentation and fully use Python's support for soft typing. Generally, changes to the existing ABM code were kept to a minimum, although some additional functionalities were added. Functional capabilities of Python were also used to implement functions responsible for training, evaluation, and various logs statistical logs. The combination of simulation and deep learning can be computationally expensive, so decisions to optimise the code by reducing the number of computations inside the step-loop and allowing for parallelisation of matrix multiplication calculations using PyTorch's CUDA GPU API and tools such as CPython profiler were used to identify and eliminate performance bottlenecks. Finally, RayTune was used for parallelising the hyperparameter sweep. Generally, the computational complexity scales linearly with the number of households, schools, and episode / simulation time-steps: $O(|H| \times |M| \times T)$, but the Neural Network related computations such as inference and backpropagation are the most expensive operations.

Chapter 4

Experimental Setup

This section provides an overview of our experimental setup and process. We describe our experiments in detail and the evaluation measures used within them. We justify their design and execution with respect to the research question.

4.1 School-choice model configuration

We attempt to recover the latent policy from the school-choice model by Dignum et al. (2022). We do this using only outcome summary statistics on the reference system and RL. Our experiments address the primary and sub-research questions posed in Section 1. Given three sub-questions, we focus on three sets of experiments. Regarding problem complexity, we consider the six scenarios: latent policies where $\alpha \in \{0, 0.5, 1\}$ and for each α, $|H| \in \{12, 96\}$. In all scenarios $|M| = 4$, and households reside on a 15×15 grid. We use the same tolerance threshold $t = 0.6$ and $\mathcal{M} = 1$ for both archetypes in all experiments.

The choice of the α parameter entails different sensitivity of the outcome to the initial conditions: In the first case ($\alpha = 0$), the school process mirrors the residential distribution, whereas in the latter cases, the initial random distribution of households to schools matters. This particular set of parameters was chosen in our experiments because they lead to extreme outcomes (i.e., highly segregated schools), which means that the summary statistics are more distinct (e.g., t^{τ} is often a binary vector in the $\alpha = 1$ case). In general, experimenting with different latent reference policies helps evaluate the robustness of our method and, therefore, the feasibility of its application.

The choice of households is based on the approximate initial household-to-school density (i.e., $\frac{|H|}{|M|}$). That is, for the first case, we expect approximately three agents per school and for the second case, an initial school density multiplied by a factor of eight i.e., 24. We consider the sparser cases to be simpler because, at the start there is much less diversity in the composition values. A school can take four composition values for each archetype: $0, 0.333, 0.667, 1$. Sparser scenarios are simpler from a combinatorial perspective as well: If we consider the possible set of solutions (i.e., allocations of households to school that perfectly match the summary statistics). The combinatorial implication of the problem guided the decision to test on systems where $|M| = 4$. Finally, fewer households allow for shorter episodes and therefore more experiments to be run in the context of the project's time and computational constraints.

Different reference policies entail different outcome sensitivity to initial conditions (see Section 3.5). Hence we consider reward functions that compute the reward using identical logic but on different dissimilarity metrics of the reference and sample summary statistics. The dissimilarity metrics are the Wasserstein and MSE criteria. In total, we evaluate the recovery performance in terms of reward in twelve separate cases.

4.2 Experiments

Learning performance experiment setup

In our methodology, recovering the latent policy amounts to training the learning RL controller - namely, the DDQN. Our first set of experiments focuses on evaluating the learning performance of the DDQN during training. We measure this by running 15 evaluation episodes every 60 training episodes and plotting the average reward received. This is done for all three scenarios, and both reward function dissimilarity criteria.

Training requires selecting the appropriate amount of episodes and episode steps. Both choices are scenario dependent since higher household density requires more moving steps and implies a more complex problem which requires more episodes. The first case is run for 5000 episodes of 40 steps each, and the second for 6000 episodes and 290 steps each. As we use singleton batches of moving agents, each step corresponds to a single agent move. We choose the number of steps by multiplying by 3 the number of households and ceiling-rounding to a multiple of 10. This was done to ensure that the controller has sampled and moved all agents at least 3 times, again, a decision made on empirical evidence from preliminary experiments. More challenging tasks also require more exploration.

The same DDQN hyperparameter choice was made for all experiments. The hyperparameters were chosen based on a random grid search. This was done over three scenarios, two instances, and two random seeds to identify the best hyperparameters for most tasks. We chose the hyperparameters based on the controller's sample efficiency (inversely weighing the reward by the number of time-step), average reward, maximum reward, and minimum standard deviation of evaluation rewards. The set of hyperparameters used can be found in Table A.2 in Appendix A.2.1. Similarly, small grid sizes lead to faster convergence in the residential process, which substantially sped up the data-generation process. We kept the number of exploration steps (random actions) as a constant proportion of total training steps i.e., 14%.

In Section 5, we present the results for both the reward function criteria

(Wasserstein and MSE). However, the two cannot be directly compared in experiment 1. That is because the various reward thresholds were not systematically set. For instance, a theoretical maximum can be computed for the MSE criterion. Thresholds could be set as percentages of that maximum, but the Wasserstein criterion has no such maximum.

Following the field's best practices (Henderson et al., 2018), we run our training process for five random seeds. In experiment 1 we present aggregated results. For experiments 2 and 3, we pick one random seed run and use the weight checkpoint of the best-performing model for each scenario and reward function criterion combination. Even if the model finishes training at a lower performance, we select an earlier weight checkpoint where performance was maximum. The corresponding details on the seed and maximum evaluation reward can be found in Table A.4 in Appendix A.3.

Generalisation experiment setup

In the second set of experiments, we explore the intra-scenario generalisation of the recovered (learned) policy. Specifically, we evaluate the zero-shot generalisation of our agent - how well our agent performs to other unseen instances without any training on them (Kirk et al., 2023). Overfitting is a pertinent issue with traditional ML techniques, including RL, where agents specialise in the training environment instance and fail to perform adequately in novel instances (Cobbe et al., 2019). In our case generalisation is important because we want to use the recovered ABM in counterfactual settings, for example, what happens in the event of a school closure.

We collect 25 instances per scenario and train on only one. We then use the learned controller (taking the weights at the highest performance checkpoint) and run the school-choice model on all 25 instances. The evaluation is done with respect to Theil's segregation index (see Section 2.1). Specifically, we compute the mean absolute error (MAE) between the reference and sample outcome segregation indices. Note that when loading the scenario instances we maintain the same residential distribution of agents as in the reference run, but not the same random allocation of households to schools.

Ranking score experiment setup

The third set of experiments leverages the fact that our reference model is computational and fully available: The trained controller is supposed to recover and encode the household decision utility function by outputting a school ranking based on a sampled household's observations. By comparing the full school ranking of the reference and sample model outputs for different, randomly sampled households, we gain insight into the extent of the accuracy of the policy recovered on a micro-scale for different scenarios. That is, whether the reference utility function has been recovered or whether the summary statistics were matched through some other behavioural policy. Additionally, learning shortcuts is commonly present in Deep RL (Geirhos et al., 2020). By comparing the reference and recovered policies for identical inputs, we may ascertain that a behaviourally similar policy has been recovered with respect to the reference policy and scenario.

The exact reference and sample policy outputs do not represent the same quantities. The reference policy, implemented as a behavioural rule on a utility function, relies on utility computations. On the other hand, the DDQN outputs the expected value (i.e., discounted return) of taking an action in a particular state. That is, we did not compare the exact valuation of schools between reference and recovered policies but how they are ranked. This is in line with how the simulator handles the utility and value outputs (see 3.4 for more details on the handling of actions). The underlying assumption is that the reward depends on the outcomes. Thus, a high expected reward will be estimated for a move to a school that will lead to a similar system outcome and, therefore, a school that the reference behavioural rule (utility function in this case) would have chosen.

Similarly to Section 3.3, we collect 1250 observations and corresponding rankings made by the reference model decision process in each scenario. We use those observations to infer a ranking with our learned model. Since we do have lax capacity constraints in schools, the households are most often assigned to first school choice. We measure top-1 accuracy, top-2 accuracy of the reference and recovered school rankings. Additionally, we measure the top-half accuracy and the average Kendall-Tau rank correlation and standard deviation. The theoretical random baseline scores are 25%, 50%, 83%, and 0, respectively. Top-1 accuracy is the proportion highest-ranked element by the recovered policy that matches the highest-ranked element in the reference policy list. Top-2 is the same, but the second-highest-ranked elements of the reference policy are also counted as hits. We additionally calculate whether there is a significant difference in the Top-1 accuracy between the DDQN and Random policies. Significance was calculated with the Wilcoxon signed test because we cannot assume that performance is normally distributed.

We have introduced two hitherto undefined evaluation metrics: Top-half accuracy and Kendall-Tau correlation. Top-half accuracy was designed post-hoc, in light of our results, specifically for the scenarios where $\alpha = 1$. Top-half accuracy measures whether any same schools exist in the top-half of the reference and sample rankings. We implement this measure to account for the high randomness of this particular scenario. Because of the extreme configuration of the simulation, it is subject to extreme, discrete outcomes. This entails that the composition target vector (the relevant part of the household state, in this case) only contains fully segregated schools (i.e., values of 0 or 1). Thus, the' true' reference ranking expresses indifference between all the fully segregated schools. The exact ranking (assessed via the other measures) is subject to the random tie-breaking of the simulator's scheduler. Accordingly, we expect and observe all performance measures to be very low, even for the Oracle model. Our new measure brings the Oracle score close to 100 but offers a minimal difference to the random performance. The random baseline performance for this new measure is 83%.

Kendall-τ correlation is an established measure in the field of information retrieval for comparing rankings (Sanderson and Soboroff, 2007). The coefficient τ measures the correlation between two rankings. A correlation of $\tau = 1$ indicates that the two rankings are in the same order, $\tau = -1$ that they are the reverse of each other and $\tau = 0$ that there is no association (Kendall, 1938). The definition of Kendall-τ is given in Equation 4.2.1.

$$\tau = \frac{C - D}{|M|} \tag{4.2.1}$$

Here C, D, and $|M|$ are the number of concordant, discordant pairs, and the number of samples (in this case, the number of schools), respectively. Pairs between the two ranks are considered concordant if they both have the same ranking relative to another pair. Discordant are pairs that have different rankings relative to another pair. A limitation of this measure is that it can be sensitive to the range of values (Sanderson and Soboroff, 2007). However, that is not a problem since the reward is bounded between 1 and -1, as is the reference utility function output. Note that Kendall-τ coefficient was computed for each ranking individually. The average and standard deviation for the coefficient are shown in the respective table.

Chapter 5

Results

In this section we present our experimental results. First, in experiment 1 (5.1), we show the differences in learning performance for different scenario and reward function criteria. We find that, despite indications of learning (performance improvement), the RL controller fails to converge to the optimal policy. Next, in experiment 2 (5.2), we evaluate how well our method generalises when used as a forecasting tool. The generalisation is better than random, but poor when compared with the Oracle policy. Finally, in experiment 3 (5.3), we assess the learned-recovered policy at a micro, individual household level. We find that the recovered policy matches the top-school significantly better than random, but fails to match the school-ranking of the reference utility policy.

5.1 Learning Performance comparisons

Figure 5.1 shows the performance achieved in terms of reward and performance improvement. Due to the different distance metrics and unsystematic use of reward thresholds, the rewards are different for similar policies. For example, one can see that the DDQN$_{MSE}$ starts at approximately half as low a reward as the DDQN$_{Wass}$, whereas both are randomly parameterised at the start.

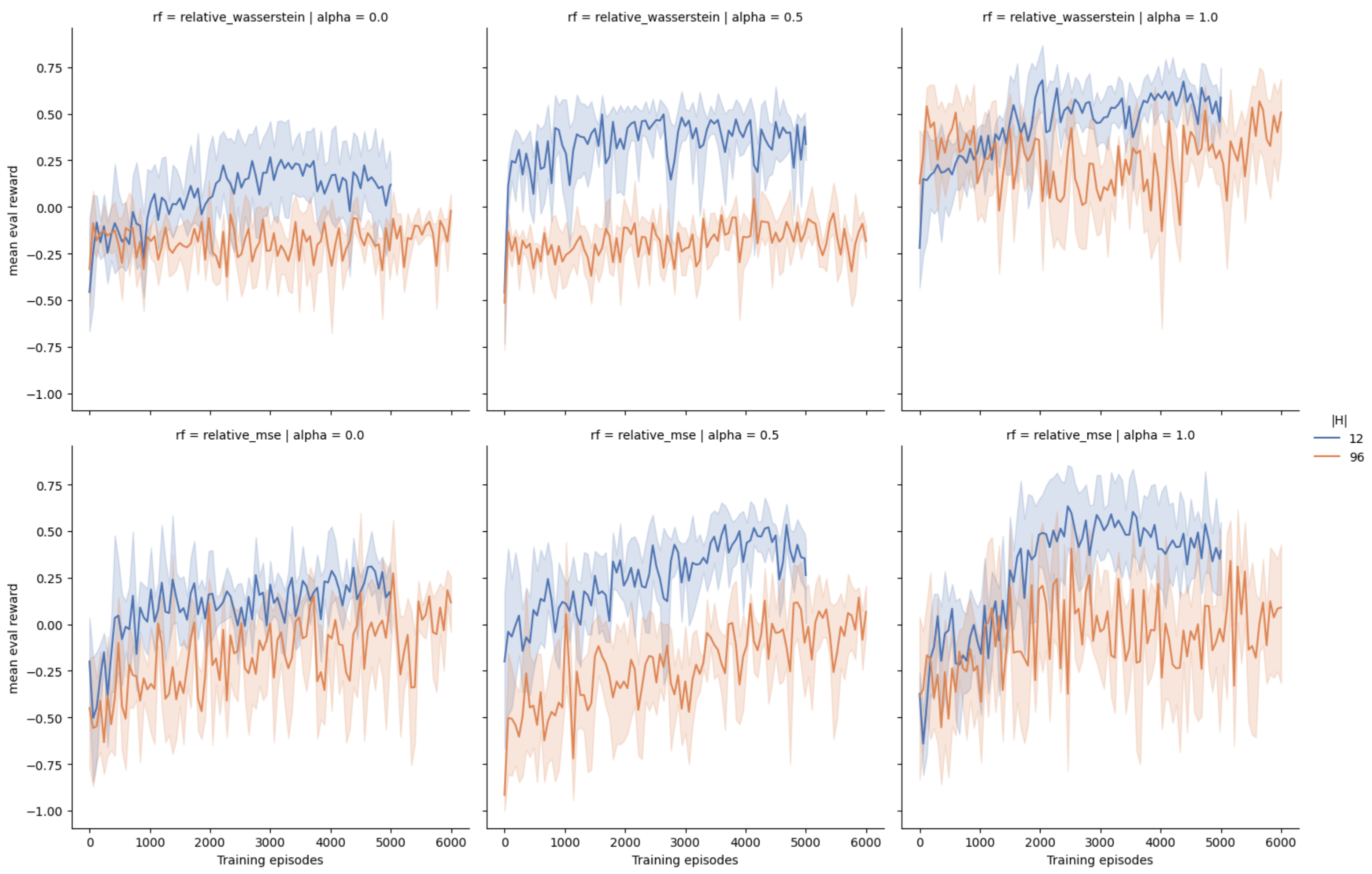


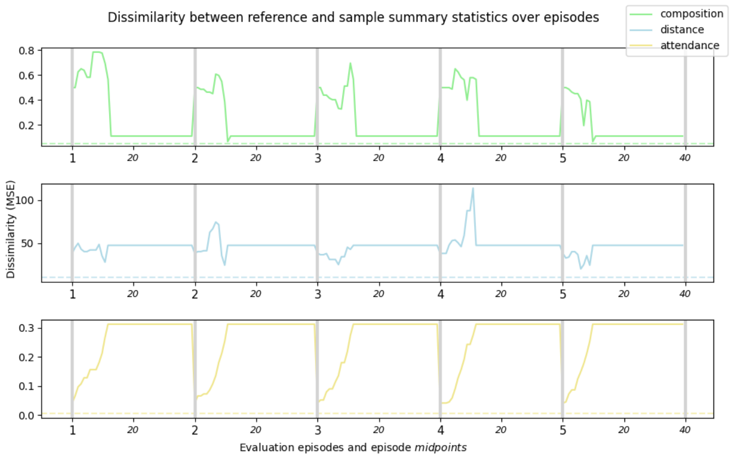
Figure 5.1: Average reward received over 15 evaluation episodes plotted at intervals of 60 training episodes of the DDQN controller in different scenarios (columns) trained and evaluated using different reward function criteria (rows). This is the average over five random seed trainings with the 95% confidence interval visible as the faded colour. The first, second, and third column correspond to a latent reference policy parameterised by an $\alpha = 0$, $\alpha = 0.5$, and $\alpha = 1$, respectively. The top and bottom rows show the $\text{DDQN}_{\text{Wass}}$ and DDQN_{MSE}, respectively. The x-axis marks the number of training episodes. The maximum and minimum score in all plots is +1 and -1, respectively.

It is noticeable that the $\text{DDQN}_{\text{Wass}}$ does not show any performance improvement in scenarios with higher household density (orange plots). That contrasts with the DDQN_{MSE} for the dense scenario. When $\alpha = 0$, both DDQNs rapidly improve initially, but the learning slows down after ≈ 1000 episodes. Given sufficient training, the DDQN_{MSE} in denser cases reaches the same reward. While generally, there appears to be an upward trend; there seems to be a slight decrease in four out of twelve scenarios towards the end of the training. This may be a case of catastrophic forgetting due to a limited replay memory size (see hyperparameter Table A.2) (Kirkpatrick et al., 2017).

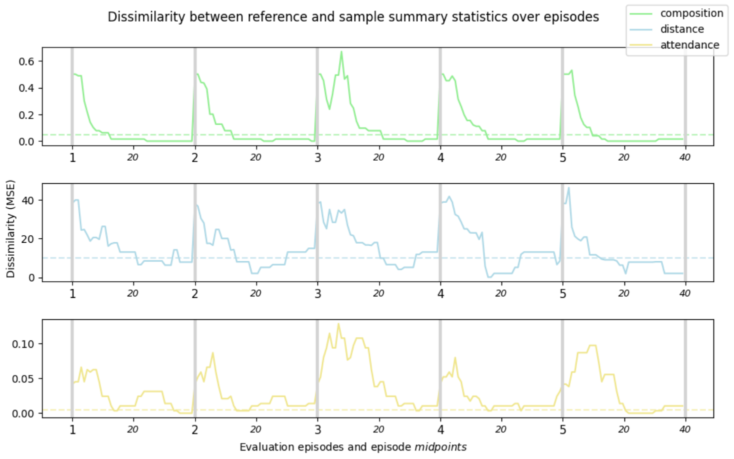
Both $\text{DDQN}_{\text{Wass}}$ and DDQN_{MSE} fail to receive maximum reward consistently in all scenarios. They tend to stabilise between a 0.25 and 0.55 reward. However, the DDQN_{MSE} consistently starts at a lower reward than $\text{DDQN}_{\text{Wass}}$, indicating that the MSE reward function produces a better reward signal for learning, possibly due to its thresholds being better calibrated. We note that, generally, in scenarios where the composition is part of the latent reference policy (i.e., $\alpha > 0$), the DDQNs, are better at maximising the reward for both reward criteria. For example, the learning performance of DDQN_{MSE} in the $\alpha = 1$ scenario which shows a rapid performance improvement. This is surprising as such scenarios are typically more complex due to the presence of interaction, which entails a higher sensitivity to initial conditions.

The learning process appears to be unstable in all scenario and reward function dissimilarity criteria combinations. Instability is typical in the DQN learning algorithm. However, high spikes (not visible due to data averaging) indicate that the model has not converged and therefore not robustly recovered the reference policy.

We have found that the reward is not maximised, which raises a question regarding the exact trade-off between the three targets and their dissimilarity with the reference outcomes. To better understand this, Figures 5.2 and 5.3 show the dissimilarity over the steps of five evaluation episodes, for all three target summary statistics under the control of DDQN_{MSE}. That is we plot, for each time-step, how far the sample RL-controlled model is from the reference model's outcome summary statistics. For completeness we include the respective plots for the Wasserstein dissimilarity in the Appendix A.4.



(a) Dissimilarity trajectory using the initial random weights



(b) Dissimilarity trajectory of the recovered policy (DDQN$_{\mathrm{MSE}}$)

Figure 5.2: Change of dissimilarity (MSE) between the reference outcome S_T and current sample $\hat{S}_t$ over the steps t of five evaluation episodes (grey lines); each episode is independent of the other and, therefore, subject to different random household-to-school distributions on initialisation of the environment. The reward is calculated based on the gradient between the two grey lines (start and termination of episodes) and whether the samples are sufficiently accurate, i.e., below the accuracy threshold (dotted lines). This is the scenario where $|H| = 12$ and $\alpha = 0$. a) shows $D(\mathbf{t}_{\mathbf{T}}^{\tau}, \hat{\mathbf{t}}_{\mathbf{t}}^{\tau})$ for the DDQN with randomly initialised weights. The average reward received over these five episodes is -0.5. b) shows $D(\mathbf{t}_{\mathbf{T}}^{\tau}, \hat{\mathbf{t}}_{\mathbf{t}}^{\tau})$ of the policy at the weight checkpoint where the DDQN$_{\mathrm{MSE}}$ achieves the maximum reward. The average reward received over these five episodes is 0.8. Note that the values are not normalised; the errors are not comparable across target statistics as absolute values.

Figure 5.2a shows the dissimilarity as a result of control with the random initial weights of the DDQN. This is a subset of the evaluation episodes of the very first point in the bottom left plot in Figure 5.1. After all (12) agents are moved, the dissimilarity no longer changes: the randomly initialised DDQN moves the households when sampled the first time but did not react to changes to the updated system state when it can move the agents for a second or third time because the random decision space is so coarsely separated. In this case, the random weights happen to lead to a decrease of the composition target dissimilarity though outside the accuracy threshold.

In 5.2b, we see that the controller has learned to decrease the dissimilarity of the composition and distance vectors to reach the accuracy threshold in most episodes. The attendance target is also closely matching the reference one. This is always the case at the start because of the uniform initial distribution of agents to schools. The dissimilarities in 5.2b indicate that the policy learned is close to optimal i.e., maximising the reward function.

An important observation is that the dissimilarities do not stabilise. The composition dissimilarity remains within the accuracy thresholds, but distance and attendance summary statistics fluctuate until the end of the episode. This implies that the DDQN has not reached a converged state and possibly more steps were necessary. Nonetheless, it is important to take into account that in this scenario the reference policy can be matched in a single move per agent (i.e., move to the closest school).

The dissimilarity measure appears very volatile to the controller's actions (i.e., households moving). We hypothesise that this is due to the low number of agents. The summary statistics are much more sensitive to individual household decisions, as a single agent can substantially change the statistics of each school. Moreover, the changes are very large: consider a school composition $c_m = 1$ as a result of only two type-0 households attending m. If one agent of the opposite type moves to that school, the composition will change to 0.66, which can substantially affect the error. While this can make the reward signal volatile, it also reduces the importance of the credit assignment problem: the RL controller receives a clear, positive reinforcement when taking the correct sequence of actions because those actions will have a larger effect on the dissimilarity at the end of the episode. The difference in the volatility of the dissimilarity can be compared with the dissimilarity measures over steps of the more dense scenario, depicted in Figure 5.3.

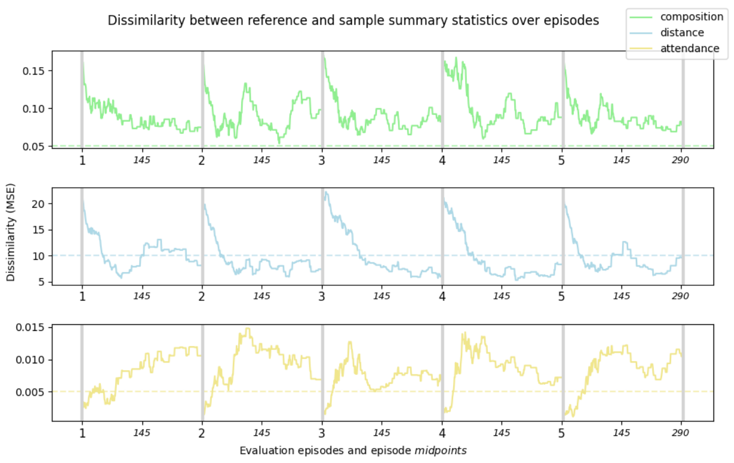


Figure 5.3: Change of dissimilarity between the reference outcome S_T and current sample $\hat{S}_t$ across the steps t of five evaluation episodes (grey lines); each episode is independent of the other and, therefore, subject to different random initialisations of the environment. The scenario plotted here is $|H| = 96$, $\alpha = 0$ under control of the DDQN_{MSE}. The average reward received over these five episodes is 0.167.

Comparing Figure 5.2b with Figure 5.3 we notice three interesting differences. First in the composition target dissimilarity $D(\mathbf{t}_T^c, \hat{\mathbf{t}}_t^c)$ specifically, the dense case (5.3) the DDQN does not reach the accuracy threshold and it appears to oscillate rather than stabilise. As aforementioned the more dense case leads to more diverse composition vectors (in terms of values). This means that it is harder for the DDQN_{MSE} to match the exact reference vector. The distance target can still be consistently matched: In all episodes reducing the average distance of attending households will decrease dissimilarity, as the reference vector consistently contains small values. The DDQN appears to have learned that. In this case, the policy recovered is not optimal because it fails to decrease the dissimilarity along the attendance target. While the attendance target dissimilarity is increasing, the MSE value in 5.3 is an order of magnitude less than the untrained controller in 5.2a.

The second observation is that the oscillation of the composition suggests that the DDQN_{MSE} keeps reacting to its previous actions without stabilising. Particularly we notice that the composition fluctuates until the end of the episode, which may imply that the DDQN_{MSE} is reacting to composition changes in the input states. However, under the reference policy these should be ignored. The third observation is that the dissimilarity changes more smoothly, since each agent move has a smaller impact. This is due to the effect of each agent on the summary statistics being smaller as the number of agents increases.

5.2 Generalisation

The second set of experiments evaluate the generalisation of the recovered policies for each scenario. The results for those runs are shown in Table 5.1.

			MAE			
$\|H\|$	$\|M\|$	α	Oracle	DDQN$_{Wass}$	DDQN$_{MSE}$	RandomPolicy
12	4	0.0	0.000	0.353	0.422	0.637
		0.5	0.142	0.432	0.425	0.544
		1.0	0.361	0.473	0.552	0.759
96	4	0.0	0.000	0.315	0.175	0.435
		0.5	0.307	0.679	0.402	0.745
		1.0	0.025	0.883	0.828	0.948

Table 5.1: Generalisation to unseen instances for DDQN$_{\text{Wass}}$ and DDQN$_{\text{MSE}}$, compared against the Oracle and Random Policy performance. Note that the two baselines are not learned and therefore we do not distinguish them between reward function criteria. The scores are the MAE between the global Theil index outcome of the reference policy and the controller indicated by the column name.

We find that the errors are lower for the more densely populated grids in the $\alpha = 0$ scenario. Strikingly, that is also the case for the random policy, suggesting that this is a simulator/environment artefact as opposed to something related to the learning process. A possible explanation is that having few agents on the grid leads to very unstable processes and, therefore, highly varied outcomes (despite the same latent policy), which are harder to predict. A dense grid will be subject to more complex dynamics but more consistent outcomes. Paying attention to the relative decreases of error between the random and DDQN controllers, we find that they are not equal. The DDQN$_{\text{MSE}}$ error is lower in the dense $\alpha = 0$ scenario by 0.247, whereas the random is lower by 0.202. We attribute this difference in decrease to the fact that the weight checkpoint used for this particular scenario is optimal (with respect to the reward function, see Table A.4) policy and better than the checkpoint used for the less dense case.

Another finding is that the results contrast the learning performance plots in Figure 5.1. The recovered policies where the composition was part of the reference policy (i.e., $\alpha > 0$) perform very poorly on unseen instances. In contrast, these were the policies that obtained higher rewards. Possibly, the learned models maximise reward by memorising a system configuration that maximises the similarity of reference - sample outcomes instead of recovering a valid policy.

On average, the difference between the DDQN's errors to the random policy errors (Wasserstein: 0.16; MSE: 0.217 - larger is better) is smaller than to the Oracle errors (Wasserstein: -0.383; MSE: -0.328 - closer to zero is better).

This implies that the generalisation performance is closer to random than the reference policy.

The Oracle also has an above-zero error in scenarios where $\alpha > 0$. This can be attributed to our assumption regarding the unavailability of the initial random allocation of households to schools when loading the sample runs and, therefore, a mismatch in initial conditions between the reference and sample models. Consequently, the system starts from different initial conditions, which in the $\alpha = 1$ scenario where only compositions matter, they are detrimental to the process. In contrast, the error is 0 in the $\alpha = 0$ scenario where the relevant initial conditions (i.e., the residential distributions) are preserved.

There is a discrepancy in the sparser and denser cases controlled by the Oracle: the former has a larger error (0.361) than the latter (0.025). We suspect this is due to the noisier outcomes of sparser cases: fewer agents can lead to unequal agent-type distributions (due to the random process of generating them), which can affect the global Theil measure. Furthermore, in the sparse scenario where $\alpha = 1$ and distance is not a factor, we expect at least one empty school. This again affects the global Theil measure making the outcome on which the MAE is computed more sensitive to the initial conditions than the denser case (where we expect the Theil measure to yield a segregation of ≈ 1). This does not entail that the dense case is any less affected by the initial conditions. However, it does show how different summary measures reflect these effects differently.

5.3 Ranking score

Experiments 1 and 2 focused on aggregate performance in terms of reward and recovered summary statistics. In the ranking-score experiment 3 we compare and evaluate the school rankings of the reference and recovered policies, for arbitrary households. The results from this experiment can be found in Table 5.3.

First, we focus on the top-1 accuracy, which is the most important metric in our case due to the relaxed school capacity constraints. Relaxed constraints entail that the top school is often chosen and, therefore, sufficient for accurate outcomes. Interestingly, the learned model is significantly better compared to the random baseline. Exceptions include the sparse $\alpha = 1.0$, where both DDQNs behave randomly and the dense $\alpha = 1$ case, where they are significantly worse. In three out of four cases where the DDQN is better than random, the MSE criterion leads to a more accurate policy than the Wasserstein criterion, except for the sparse $\alpha = 0.5$ case where the Wasserstein criterion led to a worse than random accuracy. In the dense $\alpha = 0.5$ the DDQN$_{Wass}$ performs substantially better than the DDQN$_{MSE}$, nearly double the random one.

Policy	Top1 (%)	Top2 (%)	Top-half (%)	$\overline{KT}$	σKT
12 households, $\alpha = 0.0$					
Oracle	**100.0**	100.0	100.0	1.000	0.000
RandomPolicy	23.6	49.5	81.1	-0.047	0.493
DoubleDQN$_{wasserstein}$	**35.2**	68.5	90.4	0.121	0.425
DoubleDQN$_{MSE}$	**50.4**	71.9	93.7	0.211	0.511
12 households, $\alpha = 0.5$					
Oracle	**79.6**	95.3	99.9	0.528	0.553
RandomPolicy	21.6	47.3	82.2	-0.0212	0.485
DoubleDQN$_{wasserstein}$	**15.8**	54.6	85.1	-0.075	0.447
DoubleDQN$_{MSE}$	**31.2**	56.7	89.4	0.152	0.520
12 households, $\alpha = 1.0$					
Oracle	**55.8**	87.6	97.1	0.241	0.591
RandomPolicy	24.7	51.6	82.8	0.024	0.470
DoubleDQN$_{wasserstein}$	26.3	46.5	81.8	0.032	0.484
DoubleDQN$_{MSE}$	25.7	48.8	85.6	0.026	0.479
96 households, $\alpha = 0.0$					
Oracle	**100.0**	100.0	100.0	1.000	0.000
RandomPolicy	24.7	50.3	81.9	-0.021	0.499
DoubleDQN$_{wasserstein}$	**28.2**	51.3	76.5	-0.082	0.527
DoubleDQN$_{MSE}$	**38.6**	53.7	89.0	0.234	0.442
96 households, $\alpha = 0.5$					
Oracle	**90.8**	99.1	100.0	0.463	0.563
RandomPolicy	24.5	49.2	83.1	0.007	0.488
DoubleDQN$_{wasserstein}$	**46.0**	73.4	93.3	0.126	0.484
DoubleDQN$_{MSE}$	**39.1**	72.3	90.4	0.058	0.502
96 households, $\alpha = 1.0$					
Oracle	**44.0**	84.5	100.0	0.425	0.560
RandomPolicy	25.5	50.4	82.9	-0.007	0.483
DoubleDQN$_{wasserstein}$	**21.3**	43.5	80.6	-0.008	0.477
DoubleDQN$_{MSE}$	**20.4**	45.4	82.1	0.003	0.473

Table 5.2: Scores from the comparison of the school rankings between the reference and recovered policies, for individual households. The scores are computed on 1250 observation-ranking pairs per scenario. In the Top1 column, bold values indicate a statistically significant difference with the random policy.

The result trends of the top-1 accuracy are not reflected in the other performance measures. For example, the difference to the random controller for the top-2 accuracy is almost always smaller than the difference in the top-1. Even though the random baseline performance in top-2 accuracy doubles, the top-2 for the recovered DDQN policies less than double. We find that the DDQNs that are significantly better than random in the top-1 accuracy are also substantially better than random in the top-half measures, though all other cases match the random performance.

A higher than random top-school accuracy is a promising sign. However, we are interested in the school ranking made by the recovered policy compared with the reference's. The Kendall-τ correlation gives us a measure of similarity for the entire ranking. The correlation measure between rankings should be close to 1 for a good approximation. Despite the DDQN$_{MSE}$ always getting a positive correlation, the scores are low and not near those of the Oracle. This finding suggests that our method can learn the expected reward of the best school but has not converged to the true expected reward of all actions. In turn, this entails that the full reference utility function has not been recovered.

In Section 4, we justified why we expect low accuracy scores of the Oracle policy. The same reason applies to the low Kendall-τ score: the discrete nature of the system leads to binary representations of school compositions in the household states and thus the exact ranking is random. However, the argument developed in 5.2 does not apply here because we are no longer dealing with outcomes but individual states.

Chapter 6

Discussion

In view of our results, we contextualise them and interpret their significance with respect to our research question(s). Furthermore, we offer recommendations and cautionary advice regarding the use and extension of our methodology.

Sub-research question 1 concerns the robustness of our method's learning performance in different scenarios and reward function criteria. Interestingly, the combination can have detrimental effects on learning. For example, high household-to-school density scenarios show no learning when using the Wasserstein criterion in the reward function and, thus, no progress in the latent policy recovery. On the other hand, DDQN$_{MSE}$ improves its performance even in dense scenarios though it also fails to converge to the optimal reward. However, we have noted that the two dissimilarity criteria cannot be compared directly, and therefore we cannot conclude whether one metric is better than the other for learning.

There are two factors that render the comparison of the two reward functions unfair: First, unsystematic configuration of the sensitivity and accuracy thresholds of the reward function. A more systematic calibration of the thresholds may have yielded better results for both reward functions. The second factor is the size of the target vectors and its effect on the dissimilarity computation. In scenarios with few schools, as in our experiments, each target (in the context of Wasserstein, random variable) only contains 4 observations. Wasserstein, being a distribution distance measure, is sensitive to the number of observations, which may make its dissimilarity signal unreliable. In other words small number of schools lead to a higher sampling error. In contrast MSE, which measures an average, is not as sensitive to vector sizes.

Both DDQN$_{\mathrm{Wass}}$ and DDQN$_{\mathrm{MSE}}$ are able to recover scenarios where $\alpha > 0$. In these scenarios the initial random allocation of household to schools greatly influences the outcome because they define the initial compositions that in turn dictate the reference policy's initial actions. Since we did not guarantee that the initial conditions were identical between the reference and sample runs, we cannot expect identical outcomes to be reached for the same schools even under the reference policy's control. Instead we expect the distribution of outcomes to be similar. This suggests that even though both DDQNs learn and therefore recover a policy, the $\text{DDQN}_{\text{Wass}}$ may have recovered a more accurate one and DDQN_{MSE} overfitted to the set of summary statistics used for training. That is, the DDQN_{MSE} have memorised a configuration that matches the summary statistics of the training instance without recovering a valid school-choice policy. However, in the second set of experiments we find evidence that both recovered policies fail to generalise.

In the generalisation experiment 5.2, we evaluated the robustness of our recovered policies for forecasting the school segregation in unseen instances. The answer to sub-research question 2, concerning the generalisation of the recovered policy, is that despite outperforming a simple random baseline controller, the MAE errors of both the DDQNs are large. In our real-world analogy, this implies that the DDQN would fail to generalise to arbitrary neighbourhoods in Amsterdam when trained on a (singleton) subset of them. We again observe that performance depends on the scenario being recovered, particularly the household-to-school density. It appears that the recovered policy is not robust enough and for scenarios where $\alpha > 0$ the generalisation is even poorer. Since the $\text{DDQN}_{\text{Wass}}$ also generalises poorly we cannot ascertain that Wasserstein, in the tested configuration, is a more suitable dissimilarity criterion for these scenarios.

In this set of experiments we additionally found that in scenarios where $\alpha > 1$, the Oracle fails to match the policy it implements in terms of the outcome Theil segregation. While the number of households does seem to also affect the stability of outcomes, this is due to the initial conditions that are relevant to this process not being identical with the reference S that we use as targets.

The third set of experiments addressed sub-research question 3, on the micro-level school-ranking comparison of the reference and recovered policies. Our results show that for the average household, the school-ranking of the recovered policy does not correlate highly to that of the utility function implementing the latent reference policy. However, the optimal actions of the DDQN are significantly more accurate than the random baseline in most cases where $\alpha < 1$.

In hindsight, it is perhaps unreasonable to expect the RL controller to learn the full function, especially in the absence of constraints in the simulator. For a full ranking to be learned, this would require multiple blocked actions to be encountered during training a sufficient number of times such that the network learns an appropriate evaluation for all schools for a household's state. Alternatively, a much higher exploration rate would be required as, currently, the controller receives a very sparse reward on the tail of its rankings. This would also require many more training steps. Moreover, DQNs offer no theoretical guarantees of convergence, so getting the exact expected reward values for all the schools for one given household is not guaranteed.

We again found that the Oracle performs worse when $\alpha = 1$. However, this was for a different reason to the previous experiment where we were focusing on outcomes and not the individual school-choices. In the ranking scores, the discrete nature of the household states as a result of the extreme scenarios tested, imply that the exact school choices are subject to random tie-breaking by the reference environment's scheduler and therefore so are the exact outcomes. In support of this hypothesis we devised the top-half accuracy to show that the Oracle ranking scores were high when the order of the first two schools is not considered.

In response to the main research question, the methodology and RL-based pipeline proposed in this thesis for the school-choice model, is not a feasible solution to latent policy recovery using outcome summary statistics. The method is brittle and the recovered policy does not fully match the behaviour of the latent one on a micro-scale. Moreover, the reference policies considered were simple and uniform across agent archetypes. We have no evidence to suggest that this method is precise and robust enough to discriminate and recover more complex utility functions with even subtler differences. For example, utility functions with different tolerance thresholds or preference profiles for different agent types. Even such scenarios would only scratch the surface of the complexity of the decision process of real households in real cities, and noisy data.

On a positive note, our method is often (significantly) better than a random policy and shows promising learning trajectories (Figure 5.1). Furthermore, it effectively reduces the research overhead required for designing the ABM's behavioural rule and allows for arbitrary policies to be recovered, albeit at an increased computational cost. This indicates that future work may improve our method, even if constrained to theoretical and controlled contexts.

In the following subsections we diverge from our specific results and offer more speculative explanations about what made our method work to the extend that it did and what limited its capacity to recover the latent reference policies. In an effort to make these points more general, we relate them to the properties of the class of problems the school choice belongs (2.1). We do this to facilitate further work on our method or its application on different domains.

Recommendations:

We identify two key design choices that led to the above-random performance discussed earlier. Those are the choice of summary statistics and the reward sparsity:

Choice and combination of summary statistics: Not any set of summary statistics S can be used in the inverse generative process. $\mathbf{t}^{\tau} \in S$ must change linearly to the controller's actions in all system phases. The need for linear statistics is why we do not use the decomposed (vector-form) Theil segregation. We chose linear measures with respect to the controller's action selection (e.g., proportions and averages). Furthermore, we found it useful to include a target for each statistic that may be part of the decision-making process and thus may reflect helpful decision-making information. Specifically, since we knew that we would be recovering policies incorporating some (possibly 0) weighting of either composition or distance characteristics, we included both as summary statistics. Finally, we include a 'plausibility' target statistic, such as attendance statistics, to address mode collapses, for example, the controller emptying most schools (see A.2). Plausibility targets incorporate additional information in the reward signal, which does not refer to the hypothesised target measures, but helps by enforcing plausible outcomes in optimal rewards. However, whether such 'plausibility' targets would be necessary for a more realistic system where constraints are imposed as part of the system design to ensure plausible, non-extreme outcomes is unclear. Finally, in applications where the latent reference policy is entirely unknown, we can ensure that we have sufficient relevant reference statistical information to recover a valid policy.

Frequency: Complex, emergent systems are non-linear. Providing a frequent reward signal based on dissimilarity between reference and sample outcome summary statistics at each step will reward monotonic improvement for all targets. However, non-linear dynamics imply that the summary outcomes, e.g., segregation and, thus, composition, may emerge after sequences of interactions. Moreover, due to the random batching of agents, in an arbitrary step, one target metric dissimilarity must be sacrificed to improve on another, as is visible in Figure 5.2. Therefore, while sparse rewards can complicate learning due to the credit-assignment-problem, when rewarding based on outcome summary statistics, there can only be terminal rewards to allow for non-linear dynamics with respect to the reference outcome throughout an episode. Of course, that would not be the case if the reference summary statistics were available at a higher temporal resolution.

This list is not exhaustive, and there are many other factors related to the reward function implementation (see Appendix A.2) and other, e.g., the use non-linear function approximators, that were crucial in achieving the reported performance.

Considerations

The key considerations also relate to the reward function. Specifically, how the reference targets should be interpreted in the computation of dissimilarity with respect to the consistency of initial conditions between the reference and sample models.

In complex systems, differences in initial conditions can lead to substantial differences in the outcomes (Mitchell, 2009). In our case, we have considered the variation of two initial conditions: 1. the residential distribution and 2. the initial random allocation of households to schools. We have maintained an identical residential distribution between the reference and sample runs in training and experiments. However, the initial allocations of households to schools are not kept the same. We made this decision on the basis that the initial allocations of students to schools cannot be obtained. Results showing an imperfect Oracle policy generalisation where $\alpha > 0$ imply that without both initial conditions, it may be impossible to match the exact reference outcomes even under the same reference policy. We have argued that this is the case when the reference rule is affected by the initial conditions. This finding has important implications for the choice of the reward function dissimilarity criterion - if the outcomes cannot be exactly the same spatially, then they must be interpreted differently and not as point-wise targets. A solution is to use our approach and violate the assumption regarding data to train on initial conditions identical to the reference model. Alternatively, this assumption may not be required in other applications where initialisation data is fully available.

In case the the assumption regarding unavailability of initial conditions for the choice process cannot be violated, a solution may be to approximate the outcome conditions as distributions rather than point estimates which are imposed by use of vector-based dissimilarities. Dissimilarity metrics on distributions, like the Wasserstein distance, represent the set of summary statistics as a distribution of statistical outcomes. An important point to consider is that larger sample sizes and therefore target vectors are needed. Since $|\mathbf{t}^{\tau}| = |M|$, scenarios with more schools will be necessary. Alternatively one can attempt to use the reference training data differently, for example by aggregating all summary statistics for all instances in one scenario to produce a distribution of that scenario. Training will then involve rewarding based on the decrease of dissimilarity to the average instance and therefore not depend on any one set of initial conditions. This approach seems more promising as it is more general, allowing for less dependence on initial condition data of the reference process.

Chapter 7

Conclusion

In conclusion, our method is brittle and, without addressing its limitations, shows no prospects of working in the variety of complex conditions present in real-world scenarios. We have reached this conclusion by testing our method on three sets of experiments: 1. on the learning performance, 2. on the generalisation, and 3. on the micro-level behavioural accuracy. The results show sensitivity to the scenario being recovered, and even the best models score far from the Oracle policy, albeit significantly better than the Random baseline. Furthermore, the Oracle policy scores have led us to challenge some of the assumptions of our work regarding the initial conditions stored in the training data. In an effort to provide actionable insights on the extension of our method, we have made some general recommendations and considerations for the inverse generative model construction using our method. Using outcomes as the only ground-truth signal requires making important assumptions about the availability of the system's initial conditions and the interpretation of the summary statistic data. Therefore, further research with the appropriate dissimilarity metrics and training data may yield more promising results. There are, however, limitations to our work that hinder the value and generality of our findings, recommendations, and considerations. We conclude our research by listing those limitations and offering precise directions for future work that extends our method or related but alternative approaches that rely on higher-resolution data.

7.1 Limitations

- We use the random school-choice policy as a baseline for our learning method. However, there are better, more challenging baselines that can be used and which may have rendered the DDQN results insignificantly better than the baseline. For example, a better alternative would have been a neural network optimised via random parameter search. Literature has shown that even such simple approaches can match the performance of state-of-the-art RL algorithms (Mania et al., 2018).

- A premise of our work is that extreme scenarios and system configurations have led to more discrete and thus discernible summary statistics and, therefore, clearer learning signals. However, while it is true that system statistics take extreme values, we have found that this may actually make the task more challenging. Running our experiments using more realistic assumptions (e.g., capacity constraints) may have led to improved and more stable learning dynamics.
- Our reward functions rely on certain thresholds to magnify and resolve the reward signal. However, those thresholds were not systematically set. This makes comparing the reward functions harder and leaves unused potential in their suitability for the task.
- Using the Wasserstein distance as the reward function's dissimilarity criterion may be too noisy when working with such a small number of schools. As discussed in the considerations subsection 6 attempting to use this criterion in a larger system with more schools or with a different representation of the reference data may reveal the metric's advantages.
- We show results for only one RL algorithm. Other approaches, including policy gradients which can approximate stochastic policies or actor-critic methods that can include information about the summary statistic dissimilarity in the critic module, may have led to improved performance.
- The interpretation of our results has been, to a great extent, speculative due to the black-box nature of the DQN. Explainability methods (see (Heuillet et al., 2021) for examples) may have allowed us to offer more concrete interpretations of our results and understanding of the recovered policy.

7.2 Further work

We believe further research on this topic should take either of the following two approaches:

Refining and expanding the proposed method

Future work can address the limitations of our approach outlined in 7.1. This will allow more general claims to be drawn and shed light to fundamental shortcomings of the inverse generative process using summary statistics and RL. Such research may, for example, explore reward shaping techniques, the use of more sophisticated RL algorithms, and explainability methods for more insight. For example, actor-critic controller architectures may be used to incorporate system-wide summary statistic information in the critic module. A crucial limitation to address is to systematically set and understand the effect of the reward function parameters in the recovery process.

Any extensions to our work should pay attention to the considerations discussed in Section 6. If the assumption regarding the initial conditions cannot be violated then approaches should focus on distributional dissimilarity metrics like Wasserstein and ensuring they are properly applied. The research of Novati et al. (2021); Bae and Koumoutsakos (2022) are recent examples that use distributional dissimilarities to recover models with RL.

Alternative methods

In recent years there have been many promising approaches on IGSS leveraging AI methods as reviewed in Section 2. A promising example is the work of Greig and Arranz (2021) which was recently expanded by Greig et al. (2023) which relies on evolutionary computing for program synthesis. Though sample inefficient, the recovered policies are symbolic and therefore interpretable. The approach by Chopra et al. (2022) is a promising alternative. By implementing an end-to-end differentiable ABM they achieve good forecasting performance with efficient computation. Although RL approaches are often sample inefficient, in the presence of fine-resolution data, methodologies based on imitation learning (Edwards et al., 2019) or inverse RL (Lee et al., 2017) show promising results.

Any future work should also address more realistic models and a real-world application of the method. For example, modelling the task as a POMDP to model partial, wrong, or unknown information to which real-world decision-makers are subject and use of geo-location data of Amsterdam.

Bibliography

Abbeel, P. and Ng, A. Y. (2010). Inverse reinforcement learning.

Bae, H. J. and Koumoutsakos, P. (2022). Scientific multi-agent reinforcement learning for wall-models of turbulent flows. *Nature Communications*, 13(1):1443.

Bassenne, M. and Lozano-Durán, A. (2019). Computational model discovery with reinforcement learning. *arXiv preprint arXiv:2001.00008*.

Bengio, Y. et al. (2009). Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the national academy of sciences*, 99(suppl_3):7280–7287.

Boterman, W., Musterd, S., Pacchi, C., and Ranci, C. (2019). School segregation in contemporary cities: Socio-spatial dynamics, institutional context and urban outcomes. *Urban Studies*, 56(15):3055–3073.

Boterman, W. R. (2019). The role of geography in school segregation in the free parental choice context of dutch cities. *Urban Studies*, 56(15):3074–3094.

Chopra, A., Rodríguez, A., Subramanian, J., Krishnamurthy, B., Prakash, B. A., and Raskar, R. (2022). Differentiable agent-based epidemiology. *arXiv preprint arXiv:2207.09714*.

Clark, W. A. and Fossett, M. (2008). Understanding the social context of the schelling segregation model. *Proceedings of the National Academy of Sciences*, 105(11):4109–4114.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. (2019). Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR.

Cranmer, K., Brehmer, J., and Louppe, G. (2020). The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062.

Creusere, M., Zhao, H., Bond Huie, S., and Troutman, D. R. (2019). Postsecondary education impact on intergenerational income mobility: Differences by completion status, gender, race/ethnicity, and type of major. *The Journal of Higher Education*, 90(6):915–939.

Dignum, E., Athieniti, E., Boterman, W., Flache, A., and Lees, M. (2022). Mechanisms for increased school segregation relative to residential segregation: a model-based analysis. *Computers, Environment and Urban Systems*, 93:101772.

Edwards, A., Sahni, H., Schroecker, Y., and Isbell, C. (2019). Imitating latent policies from observation. In *International conference on machine learning*, pages 1755–1763. PMLR.

Epstein, J. M. (1999). Agent-based computational models and generative social science. *Complexity*, 4(5):41–60.

Epstein, J. M. (2023). Inverse generative social science: Backward to the future. *Journal of Artificial Societies and Social Simulation*, 26(2):1–9.

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., and Wichmann, F. A. (2020). Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673.

Greig, R. and Arranz, J. (2021). Generating agent based models from scratch with genetic programming. In *ALIFE 2021: The 2021 Conference on Artificial Life*. MIT Press.

Greig, R., Major, C., Pacholska, M., Bending, S., and Arranz, J. (2023). Learning interpretable logic for agent-based models from domain independent primitives. *Journal of Artificial Societies and Social Simulation*, 26(2):1–12.

Heath, B., Hill, R., and Ciarallo, F. (2009). A survey of agent-based modeling practices (january 1998 to july 2008). *Journal of Artificial Societies and Social Simulation*, 12(4):9.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2018). Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., et al. (2018). Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Heuillet, A., Couthouis, F., and Díaz-Rodríguez, N. (2021). Explainability in deep reinforcement learning. *Knowledge-Based Systems*, 214:106685.

Jäger, G. (2019). Replacing rules by neural networks a framework for agent-based modelling. *Big Data and Cognitive Computing*, 3(4):51.

Junges, R. and Klügl, F. (2011). Evolution for modeling: a genetic programming framework for sesam. In *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, pages 551–558.

Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

Kirk, R., Zhang, A., Grefenstette, E., and Rocktäschel, T. (2023). A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76:201–264.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Lavin, A., Zenil, H., Paige, B., Krakauer, D., Gottschlich, J., Mattson, T., Anandkumar, A., Choudry, S., Rocki, K., Baydin, A. G., et al. (2021). Simulation intelligence: Towards a new generation of scientific methods. *arXiv preprint arXiv:2112.03235*.

Lee, K., Rucker, M., Scherer, W. T., Beling, P. A., Gerber, M. S., and Kang, H. (2017). Agent-based model construction using inverse reinforcement learning. In *2017 Winter Simulation Conference (WSC)*, pages 1264–1275. IEEE.

Macal, C. and North, M. (2009). Agent-based modeling and simulation.

Mania, H., Guy, A., and Recht, B. (2018). Simple random search of static linear policies is competitive for reinforcement learning. *Advances in Neural Information Processing Systems*, 31.

Manson, S., An, L., Clarke, K. C., Heppenstall, A., Koch, J., Krzyzanowski, B., Morgan, F., O'Sullivan, D., Runck, B. C., Shook, E., et al. (2020). Methodological issues of spatial agent-based models. *Journal of Artificial Societies and Social Simulation*, 23(1).

Mitchell, M. (2009). *Complexity: A guided tour*. Oxford university press.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Municipality of Amsterdam (2023). Basisbestand gebieden amsterdam (bbga). `https://onderzoek.amsterdam.nl/dataset/basisbestand-gebieden-amsterdam-bbga`. Accessed February 2023.

Ng, A. Y., Harada, D., and Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287. Citeseer.

Novati, G., de Laroussilhe, H. L., and Koumoutsakos, P. (2021). Automating turbulence modelling by multi-agent reinforcement learning. *Nature Machine Intelligence*, 3(1):87–96.

Osoba, O. A., Vardavas, R., Grana, J., Zutshi, R., and Jaycocks, A. (2020). Modeling agent behaviors for policy analysis via reinforcement learning. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 213–219. IEEE.

O'Sullivan, D. (2004). Complexity science and human geography. *Transactions of the Institute of British Geographers*, 29(3):282–295.

Panaretos, V. M. and Zemel, Y. (2019). Statistical aspects of wasserstein distances. *Annual review of statistics and its application*, 6:405–431.

Radovic, D., Kruitwagen, L., de Witt, C. S., Caldecott, B., Tomlinson, S., and Workman, M. (2022). Revealing robust oil and gas company macro-strategies using deep multi-agent reinforcement learning. *arXiv preprint arXiv:2211.11043*.

Reardon, S. F. and Owens, A. (2014). 60 years after brown: Trends and consequences of school segregation. *Annual Review of Sociology*, 40(1):199–218.

Royuela, V., Vargas, M., et al. (2010). Residential segregation: A literature review. *Universidad Diego Portales: Facultad de Economía Y Empresas*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Sage, L. and Flache, A. (2020). Can ethnic tolerance curb self-reinforcing school segregation? a theoretical agent based model. *arXiv preprint arXiv:2006.13531*.

Sanderson, M. and Soboroff, I. (2007). Problems with kendall's tau. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 839–840.

Schelling, T. C. (1971). Dynamic models of segregation. *The Journal of Mathematical Sociology*, 1(2):143–186.

Sert, E., Bar-Yam, Y., and Morales, A. J. (2020). Segregation dynamics with reinforcement learning and agent based modeling. *Scientific Reports*, 10(1):1–12. Number: 1 Publisher: Nature Publishing Group.

Spaan, M. T. (2012). Partially observable markov decision processes. *Reinforcement learning: State-of-the-art*, pages 387–414.

Stoica, V. I. and Flache, A. (2014). From schelling to schools: A comparison of a model of residential segregation with a model of school segregation. *Journal of Artificial Societies and Social Simulation*, 17(1):5.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Thurner, S., Hanel, R., and Klimek, P. (2018). *Introduction to the theory of complex systems*. Oxford University Press.

Torabi, F., Warnell, G., and Stone, P. (2019). Recent advances in imitation learning from observation. *arXiv preprint arXiv:1905.13566*.

Torrens, P., Li, X., and Griffin, W. A. (2011). Building agent-based walking models by machine-learning on diverse databases of space-time trajectory samples. *Transactions in GIS*, 15:67–94.

Train, K. E. (2009). *Discrete choice methods with simulation*. Cambridge university press.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Vu, T. M., Probst, C., Epstein, J. M., Brennan, A., Strong, M., and Purshouse, R. C. (2019). Toward inverse generative social science using multi-objective genetic programming. In *Proceedings of the genetic and evolutionary computation conference*, pages 1356–1363.

Williams, T. G., Guikema, S. D., Brown, D. G., and Agrawal, A. (2020). Assessing model equifinality for robust policy analysis in complex socio-environmental systems. *Environmental Modelling & Software*, 134:104831.

Wilson, D. and Bridge, G. (2019). School choice and the city: Geographies of allocation and segregation. *Urban Studies*, 56(15):3198–3215.

Wunder, M., Suri, S., and Watts, D. J. (2013). Empirical agent based models of cooperation in public goods games. In *Proceedings of the fourteenth ACM conference on electronic commerce*, pages 891–908.

Zhang, H., Vorobeychik, Y., Letchford, J., and Lakkaraju, K. (2016). Data-driven agent-based modeling, with application to rooftop solar adoption. *Autonomous Agents and Multi-Agent Systems*, 30(6):1023–1049.

Zheng, S., Trott, A., Srinivasa, S., Naik, N., Gruesbeck, M., Parkes, D. C., and Socher, R. (2020). The AI Economist: Improving Equality and Productivity with AI-Driven Tax Policies. Number: arXiv:2004.13332 arXiv:2004.13332 [cs, econ, q-fin, stat].

Appendix A

Appendix

A.1 COMPASS model

Parameter	**Range**
Environment	
Size (N)	15
Type distribution	50%
Number of households	{12, 96}
Number of schools/neighbourhoods (m)	4
Households	
Archetypes	{0, 1}
Optimal fraction (t)	0.6
Composition/distance trade-off (α)	{0, 0.5, 1}
Radius (r)	3
Schools	
School capacity (c)	$2 \times \frac{\lvert H \rvert}{\lvert M \rvert}$
Minimum attendance	0
Simulation	
Max number of residential steps	500
Max number of school choice steps	500
Convergence threshold (based on utility)	0.01
Number of agents moved per step(f)	1
Choice randomness i.e., temperature T	50

Table A.1: The configuration parameters of the COMPASS model that is used as the reference-system for the RL process. Note that the values were used to run the reference models, that are meant to be recovered with RL. The parameters stay the same for the inverse runs, but some are obviously ignored (e.g., α, convergence threshold, number of school steps). Notice how even though we have two agent archetypes we use the same utility function parameters for both, for simplicity. For a detailed description of the parameters see Dignum et al. (2022).

A.2 Extended Methods

A.2.1 DDQN details

Hyperparameters

Hyperparameters are critical to the DDQN's performance. Table A.2 lists the hyperparameters chosen for all experiments. Note that we use an exponential decay function for exploration. As we use a different number of training steps for different scenarios we report the approximate number of exploratory steps taken in training. Of the listed parameters only the learning rate, batch size, target update frequency were determined using a hyperparameter grid search.

Hyperparameter	Value
Discount factor γ	0.99
Learning rate α	1E-5
Optimizer	Adam
Batch size	64
Replay buffer size (state transitions)	7.5E5
Target update frequency (training steps)	500
Exploration ϵ steps $\approx$	14%
Decay	exponential
ϵ start	100%
ϵ end	5%
Number of hidden layers	2
Size of hidden layers	32
Activation function	ReLU

Table A.2: Hyperparameters for DDQN with Experience Replay

ϵ-greedy

Given three factors E_{start}, E_{end}, and E_{decay} which correspond to the start, end, and decay rate values, respectively we can compute the ϵ threshold for each training time-step n_training_steps. To take an action we sample a random value from the range of 0 and 1, uniformly. If the value is above the threshold we take the optimal action. Otherwise we take a random action. In evaluation mode we only take optimal actions. Our exponential decay threshold function is defined in A.2.1.

$$\epsilon = E_{end} + (E_{start} - E_{end}) \times a^{\frac{1}{\frac{\text{n\_training\_steps}}{E_{decay}}}} \tag{A.2.1}$$

A.2.2 Reward Function

Threshold parameters

Target	Threshold	Criterion	
		MSE	Wasserstein
Compositions	α	0.05	0.05
	ϵ	0.01	0.01
Distances	α	2	10
	ϵ	1	5
Attendance	α	0.01	0.005
	ϵ	0.05	0.0005

Table A.3: The reward hyperparameters for the accuracy (α) and sensitivity (ϵ) thresholds, for the different dissimilarity criteria.

Handling undefined values

In our preliminary experiments, we observed a mode collapse in training where the agent would choose one school for all households, leading to overpopulating one school and emptying most other schools. This is a consequence of having very lax constraints, such as school capacities. Upon investigation, we found that this strategy did indeed consist of a local optimum. Emptying schools would render some of the summary statistics undefined (e.g., the average distance to attending households) and due to an implementational decision of setting such NaN values to zero. This would mean that the empty schools would be described by near-zero average distance vectors. Due to the exponential nature of MSE, this yielded a much better (lower) error as the targets had low values in scenarios where distance mattered (households acted to minimise the distance to their schools in the reference).

To eliminate this mode collapse, we designed the following method of dealing with undefined values. The handling depends on the presence of undefined values in the reference model (i.e., whether there is an empty school), which is rare but a possible occurrence. Undefined values of the reference vector would be assigned a special value for each target. This value has to be carefully chosen as it is relative to the value ranges of the target and may overshadow errors in different school observations. For such schools, the reference-sample dissimilarity would be minimised when the learning agent empties the appropriate school. If the learning agents empties a school (by moving households away from it) that is not empty in the reference model, then that school's value is dropped by both reference and sample vectors in the dissimilarity calculation. This is a natural choice for undefined values, but it serves to magnify the errors of the other schools. Hence, instead of minimising the difference with an 0 reference vector, the error now highlights the remaining schools' mismatches.

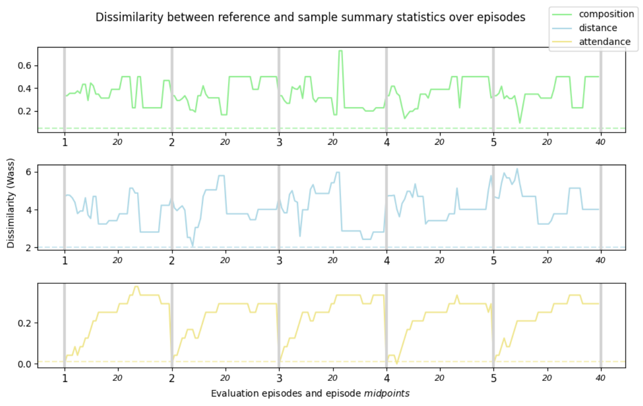
A.3 Experimental Details

$\lvert H \rvert$	$\lvert M \rvert$	Seed	α	Reward Function	Maximum reward
12	4	6	0.0	Wasserstein	0.65
				MSE	0.88
		3	0.5	Wasserstein	0.77
		1		MSE	0.80
		6	1.0	Wasserstein	1.00
		3		MSE	1.00
96		6	0.0	Wasserstein	0.33
				MSE	1.00
		1	0.5	Wasserstein	0.33
		42		MSE	0.77
		3	1.0	Wasserstein	0.83
		42		MSE	0.92

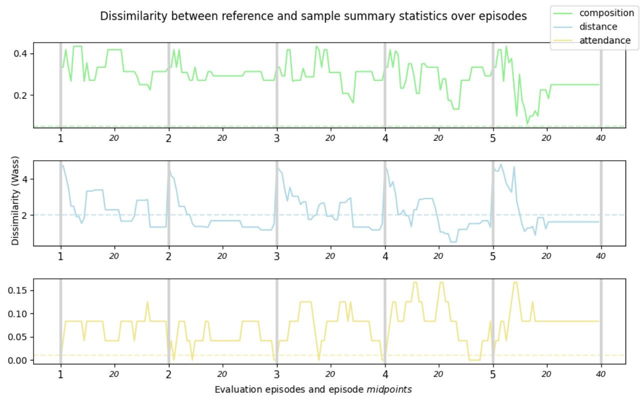
Table A.4: Maximum evaluation reward of the different scenarios run, and the corresponding run seed. In experiments 2 and 3 we use the DQN weights at those checkpoints.

A.4 Extended results

For completeness we show the change of dissimilarity over evaluation episode steps for the $\text{DDQN}_{\text{Wass}}$.



(a) Dissimilarity trajectory using the initial random weights



(b) Dissimilarity trajectory of the recovered policy (DDQN$_{\text{Wass}}$, $|H| = 12$)

Figure A.1: Change of dissimilarity (Wasserstein) between the reference outcome S_T and current sample $\hat{S}_t$ over the steps t of five evaluation episodes (grey lines); each episode is independent of the other and, therefore, subject to different random initialisations of the environment. The reward is calculated based on the gradient between the two grey lines (start and termination of episodes) and whether the samples are sufficiently accurate, i.e., below the accuracy threshold (dotted lines). This is the scenario where $|H| = 12$ and $\alpha = 0$. a) shows $D(\mathbf{t}^\tau, \hat{\mathbf{t}}^\tau)$ for the DDQN with randomly initialised weights. The average reward received over these five episodes is -0.6. b) shows $D(\mathbf{t}^\tau, \hat{\mathbf{t}}^\tau)$ of the policy at the weight checkpoint where the DDQN$_{\text{Wasas}}$ achieves the maximum reward. The average reward received over these five episodes is 0.43. Note that the values are not normalised; the errors are not comparable across target statistics as absolute values.

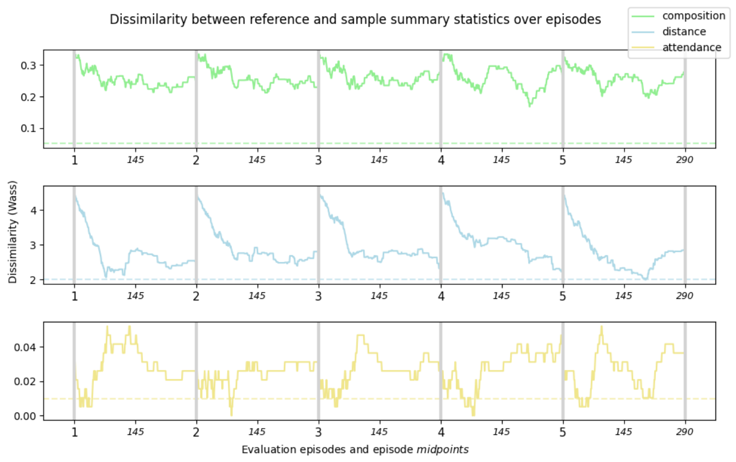


Figure A.2: Change of dissimilarity between the reference outcome S_T and current sample $\hat{S}_t$ across the steps t of five evaluation episodes (grey lines); each episode is independent of the other and, therefore, subject to different random initialisations of the environment. The scenario plotted here is $|H| = 96$, $\alpha = 0$ under control of the $\text{DDQN}_{\text{Wass}}$. The average reward received over these five episodes is 0.