MSc Artificial Intelligence
Master Thesis

Career planning with Reinforcement Learning

by
Spyros Avlonitis
12899283

November 1, 2022

48 Credits
December 2021 - September 2022

Supervisor:
Dr. Masoud Mansoury
Examiner:
Dr. Maarten Marx

Second reader:
Dor Lavi & Dr. David Graus
(Randstad)

Universiteit van Amsterdam

Contents

Abstract

This thesis explores how machine learning could help employees with their career planning. In recent years, machine learning has been used to predict employees' future development and recommend career paths with high potential. In this research, I aim to surpass the limitations of previous works and develop a system with more reasonable assumptions. To do so, I formulate the career planning process as a Markov Decision Process (MDP) and apply Reinforcement Learning (RL) methods. The goal is to find strategies employees can use to increase their long-term incomes. First, I train RL methods with employees and job datasets provided by Randstad NL. Then, I evaluate the recommended by the models paths against the actual observed paths and observe improvements of 5-6% in the mean accumulated incomes. Finally, I qualitatively evaluate the experiments' results, discuss the challenges I faced and provide suggestions about future work directions.

Chapter 1

Introduction

Motivation Career planning is the process of making decisions about what a person wants to do with their professional life and how they will accomplish it. Although not every person has the same goals and priorities, everyone could be benefited from thinking about their career proactively and making a plan for their next career moves. For example, if a person is interested in maximizing their lifetime income, they should take action towards this objective and not seek a job that only gives a higher wage in the short term. Of course, the reality is not always that simple and usually contains multiple objectives and constraints.

Objective Success in career planning depends on how much insight one has on the set of possible career paths and the expected payoff each one yields. The motivation of this thesis was to utilize artificial intelligence to provide employees with such insights. Therefore, in this thesis, I aim to propose a framework that will assist employees with their career planning by utilizing historical data and reinforcement learning. I worked together with Randstad NL company, a global leader in the HR services industry, to process a large amount of employees, job applications and salary data. Then, I utilized machine learning methods to simulate the Dutch job market and finally, I applied reinforcement learning to find strategies which will maximize employees' long-term incomes. As I mentioned earlier, income might not be the only objective of one's decision-making. However, for the scope of this research, I will assume that employees are only looking to optimize towards one objective. Furthermore, for us, the objective will be the total income acquired over a period of time. Of course, this research can be extended by replacing income with another objective, such as job satisfaction, or a combination of objectives, provided that the necessary data are available.

More precisely, my goal is to develop a system which takes as input the work experience of an employee and gives as an output a recommended career path, a sequence of occupations and industries, which will, on average, yield the highest income over a period of ten years. I have decided to set the time period to ten years because I assume that the job market dynamics are not predictable for extended periods. Thus, any recommendations beyond that timespan might be unreliable. It is also important to note that the career path recommendations should be feasible. This means that employees will likely be hired if they try to follow them.

Contributions This thesis is closely related to previous work of [Oentaryo et al., 2018b, Kokkodis and Ipeirotis, 2021, Guo et al., 2022]. However, in contrast to [Kokkodis and Ipeirotis, 2021] who experimented with online freelancers and projects, I focus on long-term employment relationships. In contrast to [Oentaryo et al., 2018b, Guo et al., 2022], which do not work with monetary rewards, I try to find the optimal path which will yield the highest long-term income for the employees. Additionally, [Guo et al., 2022] assumes every transition between jobs to be possible. In my case, I assume the transitions to be a stochastic process and use methods

that learn the probabilities from the data. Finally, [Oentaryo et al., 2018b] also assumes a stochastic process for transitions but a memoryless one. This means that a person's next job is independent of their previous experience, but only on their current job. In this work, I work with two settings, the naive one which also makes this assumption and the standard one which does not and utilize the employees' experience to predict their next move.

Chapter 2

Background

2.1 Career development and Artificial Intelligence

Research on workforce mobility and career development has been done for decades [Topel and Ward, 1992, Long and Ferrie, 2006, Moscarini and Thomsson, 2007, Fuller, 2008, Joseph et al., 2012, Oentaryo et al., 2018a]. Most of these studies rely on surveys, census, tax lists, and population registers to analyze various aspects of the workforce, such as age and wage growth. Due to the rise of online professional networks (OPN) and the data abundance during the last decade, many researchers have approached this area with data-hungry machine learning methods. A significant part of the research has recently focused on modelling career paths to predict future mobility or help candidates with their career development.

Predicting when employees will change jobs and who their next employer will be is valuable for both parties. Candidates can utilize this information to apply for vacancies with a high likelihood of being hired. On the other side, companies can use this information to retain their current employees with targeted promotions or to influence their decisions during the hiring process.

[Paparrizos et al., 2011] proposes a naive Bayes model to predict job transitions. Their technique exploits all past job transitions and the data associated with employees and institutions to predict an employee's next job transition (institution). Their experiments focus on the 100 companies and universities in their dataset and use the top 25 companies as classification labels. Their results show that job transitions can be accurately predicted, significantly improving over a baseline that always predicts the most frequent institution in the data.

[Wang et al., 2013] estimates the likelihood of an employee's decision to make a job transition at a particular time, which is denoted as the tenure-based decision probability. The authors propose using the proportional hazards model to tackle the problem and extend it with a hierarchical Bayesian framework. Then they use the predictions to find out when is the right time to make a job recommendation and how they use it to improve the utility of a job recommender system.

In [Liu et al., 2016], the authors study the feasibility of career path prediction from social network data. In particular, they fuse information from multiple social networks to comprehensively describe a user and characterize the progressive properties of his or her career path. In addition, their model can identify the influential factors of career paths. For their experiments, they collect data from LinkedIn, Facebook and Twitter, and they extract demographic, psychological and user topic features from them.

[Li et al., 2017] propose the NEMO model for predicting a person's next company and title. NEMO generates a contextual representation by aggregating all the profile information and explores the dependencies in the career paths through the Long Short-Term Memory (LSTM) networks. Extensive experiments on a large, real-world LinkedIn dataset show that NEMO

significantly outperforms strong baselines and reveals interesting micro-level labor mobility insights.

The authors of [Meng et al., 2019] propose a hierarchical neural network structure with an embedded attention mechanism to predict the next employer and the duration of the placement for employees.

[Xu et al., 2019] perform a talent flow analysis for analyzing and modelling the flows of employees into and out of targeted organizations, regions, or industries. The authors formalize the talent flow modelling problem to predict the increments of the edge weights in the dynamic job transition network. In this way, the problem is transformed into a multistep time series forecasting problem.

[Liu and Tan, 2020] proposes a logistic regression model to predict student STEM career choices. They propose a machine learning system that automatically reformats the dataset, generates new features and prunes redundant ones, and performs model and feature selection.

In [Al-Dossari et al., 2020], a recommendation system called CareerRec is proposed, which uses machine learning algorithms to help IT graduates select a career path based on their skills. Their main idea is to calculate skill similarities between the candidates and current employees in order to predict the candidate's career path.

[Yamashita et al., 2022] proposes NAOMI, a model that predicts the sequence of future job titles and companies for an employee. NAOMI uses (1) multi-view graph embeddings and BERT embeddings from job titles and companies extracted from resumes, (2) job duration masking to adjust the job experience, and (3) neural collaborative reasoning to represent the multi-factors available among jobseekers' resume graph.

Another part of the research is based on the assumption that the most frequent paths observed in the data are not always the most beneficial for the employees.

In [Lou et al., 2010], the authors are proposing a method that recommends the shortest possible career path given a person's current job (origin) and his/her dream job. In order to achieve this, they model people's career developments with Markov Chain, and they use Dijkstra's algorithm to find the shortest feasible path.

[Oentaryo et al., 2018b] key premise is that the observed career trajectories in online professional networks may not necessarily be optimal and can be improved by learning to maximize the sum of payoffs attainable by following a career path. Thus they propose a multi-objective learning procedure to achieve the best tradeoff among different payoff criteria in career path planning. The criteria they use are the transition duration cost, the level gain and the desirability gain.

[Shahbazi et al., 2019] introduced a method for job allocation in a construction company which, in contrast to conventional productivity-oriented workforce planning models, optimizes towards the career development of employees by allowing them to grow their skillset. While leading to a slight loss of productivity, their results show a significant improvement in the career development of employees with on average 8.6% improvement in employees' closeness to their ideal skill set.

In [Gugnani et al., 2019] and [Dawson et al., 2021] authors leverage the notion of skills to construct skill graphs. Then they use the graphs to measure the similarity between occupations using their underlying skills. Finally, they build recommendation systems for identifying optimal transition pathways between occupations.

[Kokkodis and Ipeirotis, 2021] proposes a career development framework that combines reinforcement learning, Bayesian inference, and gradient boosting to provide recommendations on how contractors should behave when choosing new skills to learn. Their framework uses market information to estimate current and future wages for different skills, while it uses observed skill sets to identify the feasible actions, a contractor can take to learn new skills. The proposed

approach relies on a Markov decision process (MDP) to dynamically recommend actionable career paths. At each period, the MDP provides recommendations on how contractors should spend their next period: (1) learn a new set of skills, (2) work using their current set of skills, or (3) do both and learn while working.

Recently, [Guo et al., 2022] proposed a variant of RL method with a stochastic subsampling mechanism for career path searching. They tried to optimize people's career path by recommending a sequence of companies and corresponding staying durations, which will result in the highest accumulative reward to an individual. For their experiments, they used data from an online professional network.

2.2 Reinforcement Learning

Navigation of the job market is a sequential decision-making task with delayed rewards. That is because the decisions a person makes, for example, whether to take a job or not, affect not only their immediate returns (income, happiness, etc) but also the long-term ones. This kind of task finds a very natural fit within the Reinforcement Learning paradigm. Unlike supervised learning, RL does not require a knowledgeable external supervisor to provide labelled examples. And unlike unsupervised learning, RL's objective is not to uncover a structure in the data.

As defined by [Sutton and Barto, 2018], reinforcement learning is learning what to do, how to map situations to actions, so as to maximize a numerical reward signal. Therefore, reinforcement learning is happening between an agent, who acts as a learner and decision-maker, and the environment in which the agent lives. The learner is not told which actions to take but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the following situation and, through that, all subsequent rewards. These two characteristics, trial and error search and delayed reward, are the two most important distinguishing features of reinforcement learning. Beyond the agent and the environment, one can identify four main subelements of a reinforcement learning system: a *policy*, a *reward signal*, a *value function*, and, optionally, a *model of the environment*. A *policy* defines the agent's way of behaving at a given time. Roughly speaking, a policy is a mapping from perceived states of the environment to actions to be taken when in those states. A *reward signal* defines the goal of a reinforcement learning problem. On each time step, the environment sends to the reinforcement learning agent a single number called the reward. The agent's sole objective is to maximize the total reward it receives over the long run. Whereas the reward signal indicates what is good in an immediate sense, a *value function* specifies what is good in the long run. Roughly speaking, the value of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state. The fourth and final element of some reinforcement learning systems is a *model of the environment*. This is something that mimics the behavior of the environment, or more generally, that allows inferences to be made about how the environment will behave. For example, given a state and action, the model might predict the resultant next state and next reward. Models are used for planning, by which I mean any way of deciding on a course of action by considering possible future situations before they are actually experienced. Methods for solving reinforcement learning problems that use models and planning are called model-based methods, as opposed to simpler model-free methods that are explicitly trial-and-error learners-viewed as almost the opposite of planning.

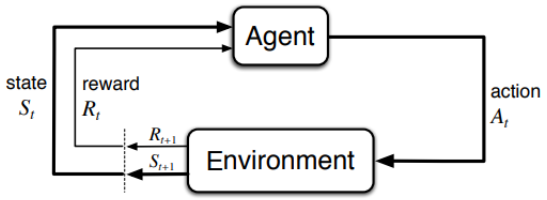


Figure 2.1: The agent-environment interaction in a Markov decision process.

2.2.1 Markov Decision Process

In order to approach the career planning problem with reinforcement learning, we need to model it as a Markov Decision Process (MDP). This is necessary in order to take advantage of the majority of the RL research that has be been done so far. MDPs are a mathematically idealized form of the reinforcement learning problem for which precise theoretical statements can be made. Therefore, most of the existing RL algorithms assume an underlying MDP.
As defined in [Sutton and Barto, 2018], MDPs are a classical formalization of sequential decision-making, where actions influence not just immediate rewards, but also subsequent situations, or states, and through those future rewards. Thus, MDPs involve delayed reward and the need to trade of immediate and delayed reward. MDPs are meant to be a straightforward framing of the problem of learning from interaction to achieve a goal. The learner and decision-maker is called the agent. The thing it interacts with, comprising everything outside the agent, is called the environment. These interact continually, the agent selecting actions and the environment responding to these actions and presenting new situations to the agent. The environment also gives rise to rewards, special numerical values that the agent seeks to maximize over time through its choice of actions. If the state and action spaces of and MDP are finite, then the MDP is called finite.
In addition, we have to formally define the objective of learning. Usually, reinforcement learning tasks can be broken into two categories, one in which there is an end (episodic tasks), and one in which the process goes on forever (continuous tasks). An episodic task could be, for example, a football match, where each team tries to maximize their score before the game ends. In episodic tasks, the agent aims to maximize the expected sum of rewards before the episode ends:

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T \tag{2.1}$$

However, in continuous tasks, there is no foreseeable end. In this case, maximizing the sum of rewards would be problematic, because the sum could easily be infinite. To tackle this problem in continuous tasks, the concept of discounting has been used. According to this approach, the agent tries to select actions so that the sum of the discounted rewards it receives over the future is maximized. Now the agent tries to maximize the expected discounted return:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.2}$$

where γ is a parameter, $0 \leq \gamma \leq 1$, called the discount rate. If $\gamma < 1$, the infinite sum in 2.1 has a finite value as long as the reward sequence R_k is bounded.
Later below, we will see how the navigation of the job market by employees can be formalized as Markov Decision Process.

The Markov property

In Markov Decision Processes, the probability that the process moves into its new state $\mathbf{S}'$ is influenced by the chosen action. Specifically, it is given by the state transition function $P_a(\mathbf{S}, \mathbf{S}')$. Thus, the next state $\mathbf{S}'$ depends on the current state $\mathbf{S}$ and the decision maker's action $\mathbf{a}$. But given $\mathbf{S}$ and $\mathbf{a}$, it is conditionally independent of all previous states and actions; in other words, the state transitions of an MDP satisfy the Markov property.

2.2.2 Dynamic Programming

A family of algorithms which can be used to compute optimal policies for MDPs is the dynamic programming (DP) family. As noted by [Sutton and Barto, 2018], compared with other methods for solving MDPs, DP methods are actually quite efficient. If we ignore a few technical details, then, in the worst case, the time that DP methods take to find an optimal policy is polynomial in the number of states and actions. In addition, DP methods are guaranteed to f ind an optimal policy. Linear programming methods can also be used to solve MDPs, and in some cases their worst-case convergence guarantees are better than those of DP methods. But linear programming methods become impractical at a much smaller number of states than do DP methods (by a factor of about 100). For the largest problems, only DP methods are feasible.

However, DP methods require a perfect model of the MDP to be available. In our case, the true transition probabilities and rewards of the MDP are not known, and therefore we do not have access to the true model of the MDP. For this reason, I will only focus on RL which do not require a model of the environment.

2.2.3 Offline reinforcement learning

Another significant limitation of our use case, is the lack of online interaction with the environment. The typical paradigm of RL allows the agent to access the environment and interact with it. That is, to perform actions and get back the consequences of them (new state and reward). In our use case, while training, the agent can not interact with the environment. This would require the training system to start making job applications on behalf of candidates and wait for their outcomes. Therefore, we have to learn from offline historical data. This approach is called offline reinforcement learning or batch RL. As summarized by [Levine et al.,], offline reinforcement learning, in essence, requires the learning algorithm to derive a sufficient understanding of the dynamical system underlying the MDP entirely from a fixed dataset, and then construct a policy that attains the largest possible cumulative reward when it is actually used to interact with the MDP.

The most obvious reason why offline RL is difficult is it inability to interact with the environment, explore new states and experiment with new actions to find high reward regions. If the captured dataset, does not have transitions to high reward regions, then the offline RL agent might never be able to explore them. In addition, if the agent eventually finds a novel state outside the training distribution, it will make bigger mistakes that compound until the policy diverges wildly from the one it was trained on. This behavior is a type of distributional shift.

In a recent survey on offline RL, [Levine et al.,] provide a helpful overview of the most common approaches for offline RL. A graphical presentation can be found in figure 2.3.

2.2.4 On-policy and Off-policy RL

Reinforcement learning methods trying to learn a policy face a dilemma: They seek to find the optimal actions, but to do so they need to behave non-optimally in order to explore all

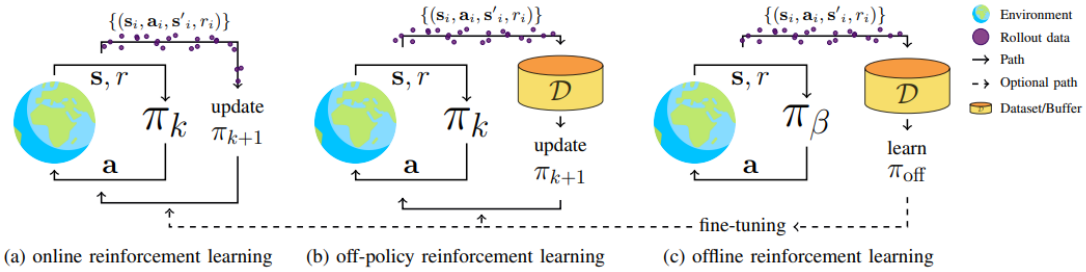


Figure 2.2: Illustration of the different reinforcement learning paradigms including (a) online RL, (b) off-policy RL (discussed below), and (c) offline RL taken from [Prudencio et al., 2022]. In online RL, one must collect new experiences with the latest policy before updating. In off-policy RL, an agent interacts with the environment and appends its new experiences to a replay buffer, which can then be sampled to perform a policy update. We can reuse previous experiences in this paradigm but still rely on a continuous collection of new experiences. In offline RL, a behavior policy is used to collect experiences that are stored in a static dataset D. We then learn a policy $_{off}$ without further interactions with the environment. After learning $_{off}$, one can opt to fine-tune their policy using either online or off-policy RL methods. This image is largely based on [Levine et al.,] pictorial illustration of RL paradigms presented in their tutorial article. Earth image made by Freepik from flaticon.com.

actions. This dilemma is being approached by two families of methods, on-policy and off-policy methods. On-policy methods, learn action values for a near-optimal policy that they also use to explore. On the other hand, off-policy methods use two policies. The policy which is being learned is called the target policy and the policy which is used to explore is called the behaviour policy. In off-policy methods, the learning is happening from data which are "off" the target policy, which we are trying to learn, and therefore they are called "off-policy".

In online RL, when the agent can interact with the environment, both approaches can be used. However, in offline RL, where we are trying to learn a target policy, with offline data generated from a sub-optimal behaviour policy, we have to utilize off-policy methods.

2.2.5 Tabular and Approximate Solution Methods

Another decision that someone needs to make when applying RL methods is the selection between tabular and approximate solution methods. Tabular methods can be used for environments with small state and action spaces, where the value functions can be presented as arrays or tables. These methods can often find the optimal value function and the optimal policy. However, for larger environments approximate solution methods are needed. For example, if we want to apply RL to a problem where each state is an image; the number of possible states could be larger than the number of atoms in the universe. In such cases, we can not expect to learn the optimal value function or the optimal policy and therefore our goal is to find an approximate solution. As noted by [Sutton and Barto, 2018], the problem with large state and action spaces, is not just the memory needed for large arrays and the time and data to fill them accurately. For many tasks, the space is so large that every state encountered may have never been seen before. Therefore, we need to generalize from previous encounters with different states that are similar to the current one. This kind of generalization is an instance of supervised learning and is often called function approximation. In recent years, many state of the art RL methods have been using artificial neural networks for function approximation. This area has become popular under the name deep reinforcement learning.

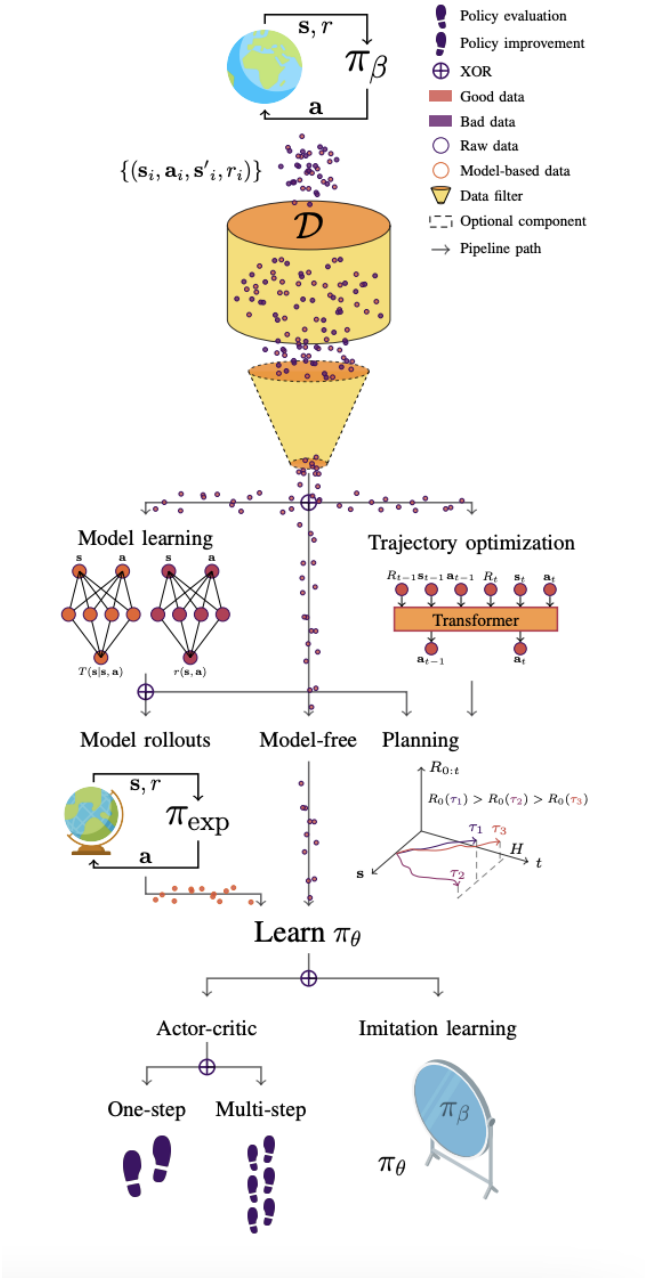


Figure 2.3: Illustration (by [Levine et al.,]) of the general structure of an offline RL algorithm, where different paths represent the decisions one can make when designing an algorithm. Initially, the behavior policy π_β interacts with the environment to collect experiences and store them in static dataset D. Then, the data is optionally filtered (e.g., using some heuristic or value-based approach) to retain only experiences from high-return trajectories. The remaining samples can then be used to either directly learn a policy π_θ, learn a dynamics model or a model of the trajectory distribution. If we opt to learn a trajectory distribution, we can use it for planning, which is interested in selecting the trajectories with the highest return. If we learn a dynamics model, we can choose whether to use our model for planning or to roll out our model and generate synthetic samples to learn a policy. Then, to learn π_θ, we can opt between actor-critic and imitation learning methods, where the latter typically relies on a good filtering process or an appropriate outcome to condition the learned policy. Finally, in actor-critic methods, one can opt to use single or multiple steps of policy evaluation and policy improvement, in which one-step methods allow us to avoid distributional shift entirely. Earth and globe images made by Freepik from flaticon.com

2.2.6 Sarsa

Sarsa is an on-policy and tabular temporal difference (TD) method. TD learning is a combination of Monte Carlo and dynamic-programming ideas. Like Monte Carlo methods, TD methods can learn directly from raw experience without a model of the environment's dynamics. Like DP, TD methods update estimates based in part on other learned estimates without waiting for a final outcome (they bootstrap). *Sarsa* algorithm aims to learn an action-value function $q_\pi(s, a)$ that gives the expected reward starting from the state s, taking action a and following the policy π. The general form of *Sarsa* can be found in the Algorithm 1.

Due to the random selection of A_{t+1}, Sarsa is usually suffering from high variance. To eliminate this issue, another version of Sarsa has been proposed. It is called Expected Sarsa and it replaces the $Q(S, A)$ update step with:

$$Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + \gamma \sum_a \pi(a|S') Q\left(S', a\right) - Q(S, A) \right]$$

where $\pi(a|S)$ is the probability of taking the action a under the current policy. As noted by [Sutton and Barto, 2018], *Expected Sarsa* usually performs slightly better than *Sarsa*.

Algorithm 1 Sarsa general form

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{ terminal}, \cdot) = 0$
for each episode **do**
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  for each step of episode **do**
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + \gamma Q\left(S', A'\right) - Q(S, A) \right]$
    $S \leftarrow S'; A \leftarrow A'$;
    Break if S is terminal
  end for
end for

2.2.7 Q-Learning

Q-Learning was introduced by [Watkins and Dayan, 1992] as another tabular TD method. However, Q-learning is an off-policy method. In this case, the learned action-value function, Q, directly approximates q_*, the optimal action-value function, independent of the policy being followed (behavior policy). The general form of the Q-Learning algorithm can be found in the Algorithm 2.

Double Q-Learning

Double Q-Learning ([Van Hasselt,]) is an extension of Q-Learning which makes the learning more stable. The max operator in standard Q-learning uses the same values both to select and to evaluate an action. This makes it more likely to select overestimated values, resulting in overoptimistic value estimates. To prevent this, Double Q-Learning decouples the selection from the evaluation. *Double Q-learning* stores two Q functions: Q_A and Q_B. Each Q function is updated with a value from the other Q function for the next state. In this thesis, I have implemented and experimented with *Double Q-Learning*.

Algorithm 2 Q-Learning general form

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{ terminal},\cdot) = 0$
for each episode **do**
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  for each step of episode **do**
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + \gamma \max_a Q\left(S', a\right) - Q(S, A)\right]$
    $S \leftarrow S'$
    Break if S is terminal
  end for
end for

2.2.8 Deep Q-Learning Network (DQN)

Sarsa and Q-Learning discussed above are tabular methods. Therefore, they are not effective for environments with large state spaces. For this reason, below we will introduce two approximate solution methods which will be used for the standard environment. DQN was first introduced by [Mnih et al., 2013] and it is an off-policy approximate solution method. It is the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards. [Mnih et al., 2013] applied DQN to seven Atari 2600 games and found that it outperforms all previous approaches on six of the games and surpasses a human expert on three of them.

2.2.9 Advantage Actor Critic (A2C)

A2C is another approximate solution RL method which utilizes deep reinforcement for function approximation. In contrast to DQN A2C is an on-policy method. A2C was introduced by [Mnih et al., 2016] and it is an actor-critic method. Actor-critic methods are a subset of temporal difference (TD) learning methods that represent the policy function independent of the value function. The "critic" model estimates the value function and the "actor" learns the target policy. Both the Critic and Actor functions are parameterized with neural networks. As explained in [Mnih et al., 2016], the main advantage of compared DQN is that it can be trained significantly faster. That is the reason, I selected this method to experiment with, in addition to DQN.

Chapter 3

Methodology

3.1 Datasets

My experiments were performed on datasets provided by Randstad Netherlands. For this work, I utilized the datasets described below.

3.1.1 Work experience dataset

The work experience dataset is a tabular dataset and consists of work experience items that employees submit to Randstad through its website or its consultants. More precisely, the useful attributes for this research in this dataset are: *1) employee ID 2) Job start and end date 3) ISCO code* [1] *(used as occupation identifier) 4) SBI code* [2] *(used as industry identifier).* While reprocessing this dataset, I filter out employees with missing data. In addition, I filter out jobs with a duration of less than a week and employees with more than fifty work experience items. Finally, the dataset consists of 200K employees with 400K work experience items.

It is important to mention that most of the work experience items (99.99%) are about Randstad placements. These are jobs which the employees found through Randstad. This is the case because essential attributes are missing for most of the items coming from employees' previous experience (before using Randstad services).

Furthermore, as expected from Randstad's business model, most of the jobs are short-term placements. Therefore, the mean job duration is 161 days, and the median is 95 days. The distribution of durations can be found in Figure 3.1.

Finally, in Figure 3.2, the distributions of career path's lengths and durations can be found. A career path is the sequence of jobs an employee has listed. As can be seen, the majority of employees have less than 15 jobs listed and have worked, with Randstad, for less than six years.

3.1.2 Vacancies dataset

The vacancies dataset contains structured information, including offered salaries, for six million vacancies posted on various websites in the Netherlands. This dataset is used for estimating the expected salaries for each occupation.

The mean yearly salary is found to be 42K euros and the median 38K euros. In addition, the mean salary for four different levels of experience is shown in Figure 3.3.

[1] ISCO Wikipedia page : https://en.wikipedia.org/wiki/International$_{S}tandard_{C}lassification_{o}f_{O}ccupations$

[2] SBI official website: https://www.kvk.nl/overzicht-standaard-bedrijfsindeling/

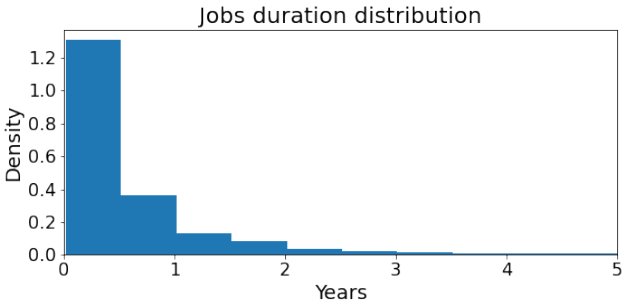


Figure 3.1: Work experience dataset: Jobs duration histogram with density.

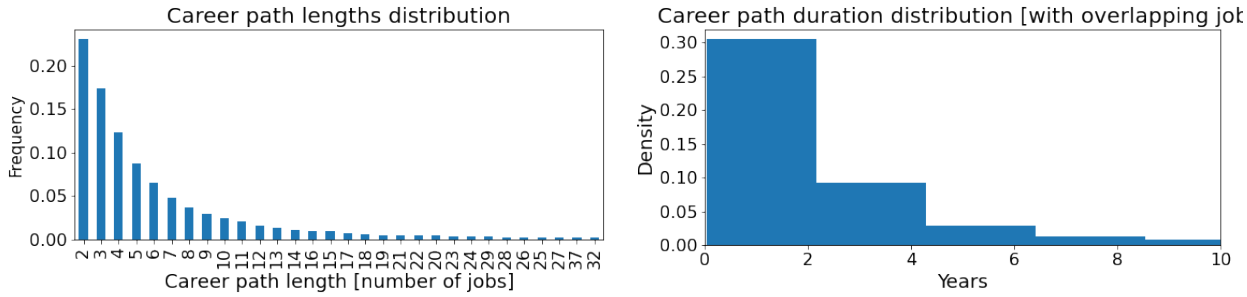


Figure 3.2: Work experience dataset: Career paths length and duration histogram with density.

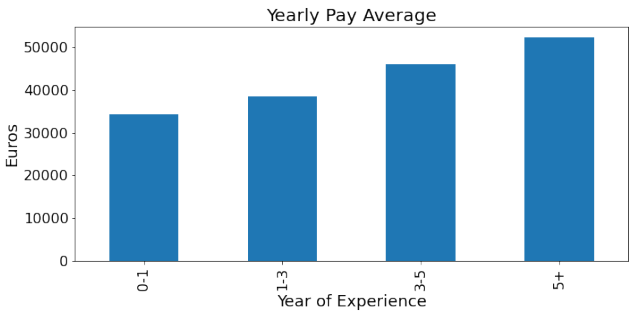


Figure 3.3: Vacancies dataset: Mean salary for different years of experience in the Netherlands.

3.1.3 Randstad's vacancies dataset

This dataset is similar to the *Vacancies dataset* but contains information about vacancies managed by Randstad Netherlands.

3.1.4 Job applications dataset

This dataset contains information about applications made by candidates for Randstad's vacancies. For each application there is an outcome available (rejected or hired).

3.2 Markov Decision Process

To start experimenting with Reinforcement Learning we need to formulate career paths as a Markov Decision Process. In the scope of this thesis, I experimented with two different MDP formulations. The *Naive* and the *Standard* environments. The *Naive* environment is a more simplified version of the *Standard* one. And the *Standard* one is, of course, a simplification of the real job market. Below, I will describe the two environments.

3.2.1 Naive environment

States

At every given time, an employee has a specific job. A job is defined as the combination of an occupation and an industry. For example, this can be a software engineer (occupation) in banking (industry). Therefore, a state $\mathbf{s}$ of the MDP is defined to be a job. However, in reality there could be up to 13000 possible jobs (130 ISCO3 occupations and 100 SBI2 industries). In order to keep the environment in a computationally tractable size, I only keep the 142 most common jobs (in the Work Experience dataset). Consequently, the career path of a person can be through as a sequence of states:

$$\mathbf{S_0}, \mathbf{S_1}, ..., \mathbf{S_N}$$

Actions

Starting from every state $\mathbf{S}$, the agent has the ability to apply for any other job. An application is formulated as the action $\mathbf{a}$. In each time step, the agent has the option to apply for a new job or stay in their current job.

Rewards

At each time step, the agent will earn a reward $\mathbf{r}$. Informally, the agent's goal is to maximize the total amount of reward it receives. This means maximizing not the immediate reward, but cumulative reward in the long run. For this research, the reward is the salary earned by the employee over the time period of a time-step. Therefore, the reward can be defined as a function of the state. More precisely, I define the reward as:

$$r = R(s)$$

Transitions

Applying to a job does not have a deterministic outcome. Therefore, the actions that the agent takes should not have a deterministic outcome. When the agent selects an action $\mathbf{a}$) and succeeds to be hired, moves from the current state $\mathbf{s}$ to a new state $\mathbf{s'}$. If not, stays in the previous state $\mathbf{s}$. This transition is happening with probability $P(\mathbf{S}, a, \mathbf{S'})$.

Learning the environment dynamics

To simulate the environment, we need to learn the transition probabilities $P(s'|s,a)$ and reward function $r(s,a)$ from the the data. First, the reward function $r(s,a)$ is learned from the Vacancies dataset using a Random Forest regression model. For the training, I use the *Vacancies* dataset which contains Dutch vacancies posted on various websites. The model learns to predict the yearly salary of an employee, given their occupation (ISCO3 code) and industry (SBI2 code). The mean absolute error of the learned model is approximately 6000. Given the average yearly is approximately 36000, I consider this error acceptable for my experiments. Obviously, further effort can be made to develop more accurate salary prediction models; but this is not the focus of this thesis.

Secondly, the transition probability $T(s'|s,a)$ depends on the job of the source s and target s' states. For the naive environment, P_{Naive} is calculated by counting observations in the Work Experience dataset.

$$P_{Naive} = P(NextState = A|PreviousState = B) = \frac{\text{\# People worked in A and came from B}}{\text{\# People worked in B}}$$

3.2.2 Standard environment

States

In the Standard environment in addition to their job, employees may have a work experience from their previous jobs. A person's current job **j** and their past work experience **w** form their state. Therefore, a state **s** of the MDP is defined as:

$$\mathbf{s} = (j, \mathbf{w})$$

Where **j** is the job and **w** the work experience of an employee. Consequently, the career path of a person can be represented as a sequence of states:

$$\mathbf{S_0}, \mathbf{S_1}, ..., \mathbf{S_N}$$

w is a vector with 130 dimensions. Each dimension represents a specific occupation (ISCO3 code) and its value represents the months of experience the employee has in this occupation.

$$\mathbf{w} = (d_{0,i}, d_{1,i}, ..., d_{129,i})$$

Where $d_{k,i}$ is the months of experience an employee has in occupation k in time step i. To make the work experience information more accurate, we could also add industries in the experience vector. However, I avoided doing so because this will increase the number of dimensions of **w** by 100 times (100 SBI2 industries). This addition would significantly increase the state space of the environment and would make policy learning a lot more challenging. Finally, similarly to the Naive environment, I only keep the 142 most common jobs.

Actions

The actions are defined similarly to the Naive environment.

Rewards

The rewards are defined similarly to the Naive environment.

Transitions

Similarly to the Naive environment, applying to a job does not have a deterministic outcome. Therefore, the actions that the agent takes should not have a deterministic outcome. When the agent selects an action $\mathbf{a} = (\mathbf{j}')$ and succeeds to be hired, moves from the current state $\mathbf{S} = (\mathbf{j}, \mathbf{w})$ to a new state $\mathbf{S}' = (\mathbf{j}', \mathbf{w}')$. If not, moves to a new state $\mathbf{S}$" $= (\mathbf{j}, \mathbf{w}$") , where only the work experience has changed. This transition is happening with probability $P(\mathbf{S}, a, \mathbf{S}')$.

Learning the environment dynamics

To simulate the environment, we need to learn the transition probabilities $P(s'|s, a)$ and reward function $r(s, a)$ from the the data.

For the Standard environment, similarly to the Naive, the reward function $r(s, a)$ is learned from the Vacancies dataset using a Random Forest regression model.

However, the transition probabilities differ. In *NaiveJobMarketEnv* the transition probability $T(s'|s, a)$ depends only on the occupation and the industry of the source s and target s' states. Therefore, in contrast to the *StandardJobMarketEnv* it is independent of the past experiences of the agent. Another difference from the *NaiveJobMarketEnv*, is that I estimate and use the probability of being hired (making the transition) given that the worker applied for the job:

$$P_{standard} = P(hiredInStateA = True|CurrentState = B, AppliedToA = True)$$

. $P_{standard}$ is estimated by training a binary Random Forest classifier on the Job Applications dataset.

One important note here is that the state space of the *NaiveJobMarketEnv* is significantly smaller than the one of the *StandardJobMarketEnv*. This is because, in the naive environment, the transition probabilities are independent of the candidates' past experiences W. Therefore, W does not need to be in the state and therefore, the number of possible states is significantly smaller.

3.3 Offline Reinforcement Learning

As discussed in Section 2.2.3, there multiple approaches for applying RL on offline data. In this thesis, I make the assumption that an accurate model of the job market is not easy to be created. Therefore planning is not an option and I have to proceed with using methods to learn a policy π_θ. In addition, I avoided proceeding with a Model-free approach, because of evaluation difficulties that arise. Offline policy evaluation, without a model of the environment, is an active area of research in reinforcement learning. The main difficulty here is the counterfactual evaluation (as I propose in Section 3.4) without the access to an environment.

Therefore, in this thesis I simulate the job market environment by learning a model with supervised learning. Then, use the learned model to make rollouts and learn a policy with online RL methods. The high level overview of this process can be found in Figure 3.4. The important assumption here is that, although online interaction is not available in the scope of this thesis, it might be in the future if a recommendation system with RL becomes available on production.

Continuous task

In reality a career path always has an end, due to death or retirement, so the career planning process is naturally an episodic task. However, in my case I do not have any information about

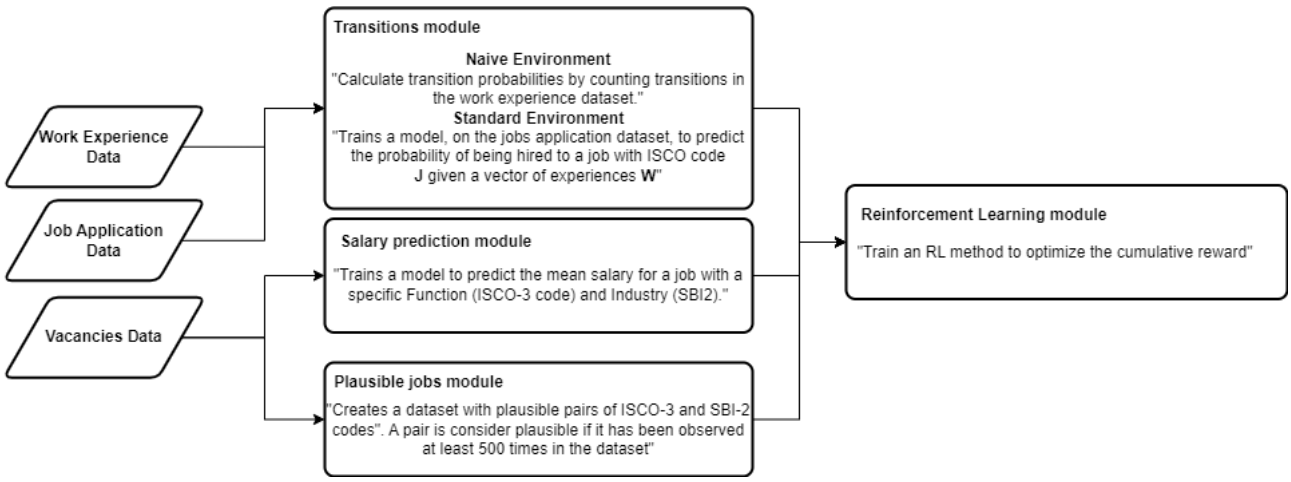


Figure 3.4: A high-level diagram of the various modules which are used for the experiments.

employees' age in my dataset, and therefore I do not know when the "episodes" should end. Therefore, in this research I will formulate the career planning process as a continuous task. As mentioned in the previous chapter (2.2), in continuous tasks, a discount rate γ is being used for rewards. In my case, this discount factor accounts for inflation. Finally, for my simulations, I assume that the agent will make a decision (to apply for a new job or stay in their current job) every 3 months. Therefore, I work with 3-months time steps.

3.4 Metrics and baselines

The purpose of the methods discussed is to recommend feasible career paths which yield high incomes for the worker who will follow them. Therefore, to evaluate these methods, I measure the difference between the cumulative income that the observed paths (factuals) and the recommended paths (counterfactuals) yield.

3.4.1 Observed career paths

I use the Work Experience dataset to construct a list of observed career paths and the income that each one produced. However, workers often have more than one job in parallel or have periods of unemployment. Therefore, given that in my MDPs the agent can not be unemployed or work in multiple jobs in parallel, I have to process the observed paths to make them comparable with the recommended ones. If a worker has multiple jobs in parallel, then I would estimate the monthly salary of each job (40 hours per week) and then assume they made the mean of them. For periods of unemployment, I assume the worker kept on making the salary from their last job.

Finally, due to the lack of quality salary data for the observed paths, I use the Regression Model from Section 3.2.1 to estimate the mean salary of each job based on its function and industry.

3.4.2 Counterfactual career paths

After training a model with each RL method discussed before, I sample a number of observed career paths and generate their counterfactuals. As a counterfactual path, I define the path

that each model recommends starting from the same initial job as the observed one and having the same duration.

In addition, to the trained RL models, I introduce two simple baseline methods which generate counterfactual career paths. I follow the same process to evaluate the baseline methods against the observed career paths.

3.4.3 Baseline 1: Greedy most common transition

In this method, the agent always selects to apply to the job with the highest transition probability. If there are multiple jobs ranking at the top, then one of them is selected randomly every time.

3.4.4 Baseline 2: Greedy highest expected reward

In this method, the agent always selects to apply to the job with the highest expected reward. The expected reward for each job is defined as the product of the transition probability and the immediate reward (salary) the agent will receive after the transition. If there are multiple jobs ranking at the top, then one of them is selected randomly every time.

Chapter 4

Experiments

In this chapter, I present the results from the experiments I ran using the two environments described in Section 3.2. Namely, the Naive and the Standard environments. For each of the environments, I report the results from the two baseline methods described in Section 3.4. Then, I report the results from the methods I applied in order to learn a competent policy. More specifically for each environment-method pair I ran and report and the following experiments:

Starting vs Final Function/Industry distribution For this experiment, I run 1000 episodes for 40 time steps (10 years), using a specific model, and report the starting and the final functions and industries distributions using histograms. As a reference point, in Figure 4.1 I present the *Starting vs Final Function/Industry* histograms for the real observed career paths in my dataset. These histograms are useful in order to detect and understand the model's preferences towards specific functions and industries.

Transitions Graph Similarly to the previous experiment, I run 1000 episodes for 40 timesteps (10 years), with a specific model, and reported the frequency of transitions between functions and industries. However, in contrast with the previous experiment, I report all the transitions which are happening during the episodes. Again, as a reference point in Figure 4.2, I present the *Transitions Graph* for the observed career paths in my dataset. The width of the edges represents the frequency of the transition between the nodes; the higher the frequency the bigger the edge.

Counterfactual evaluation Finally, as described in Section 3.4 I sample 20000 employees and I generate their counterfactual paths. Then I compare the mean reward (in Euros) accumulated in the real and the counterfactual world. If there is a significant difference (p-value ≤ 0.05) I also sample, present and qualitatively compare the career paths of some of the employees whose income increased (*Gainers*) and some whose income decreased (*Losers*).

The reported metrics can be show in Table 4.1 for the Naive environment and in Table 4.7 for the Standard one. For each model, I present the *Mean Factual* accumulated reward and the *Mean Counterfactual* one. That is the mean income employees accumulated in the real world and the one that they would have accumulated in the counterfactual world. Next, I present the *Change %* between the two means and *p-value* from the two sided permutation test I perform in order to understand if there is statistically significant difference between the two. Finally, I report the percentage of employees who saw an positive change in the counterfactual world (*Gainers*) and the mean of this changes. I do the same for those who saw a negative change (*Losers*).

As mentioned above, I also present some samples of factual and counterfactual paths. For example in Table 4.3 I present a sample of employees whose income would have been increased

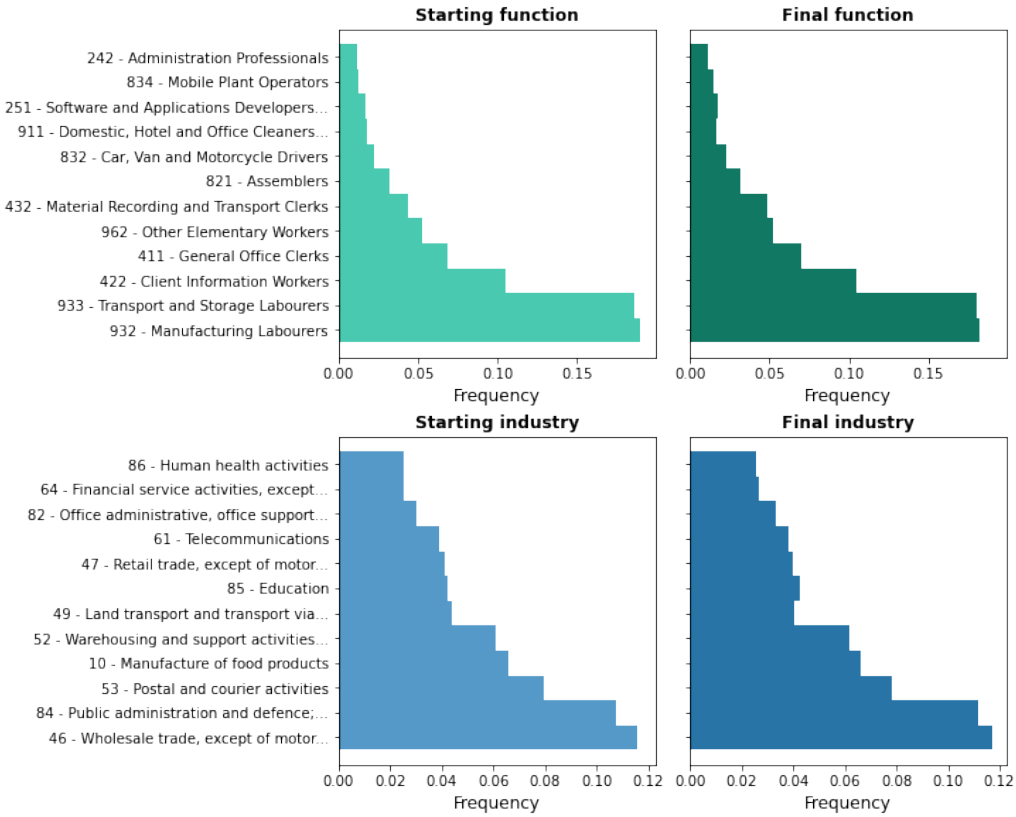


Figure 4.1: **Observed career paths:** Starting and Final distributions for the 12 most common functions and industries.

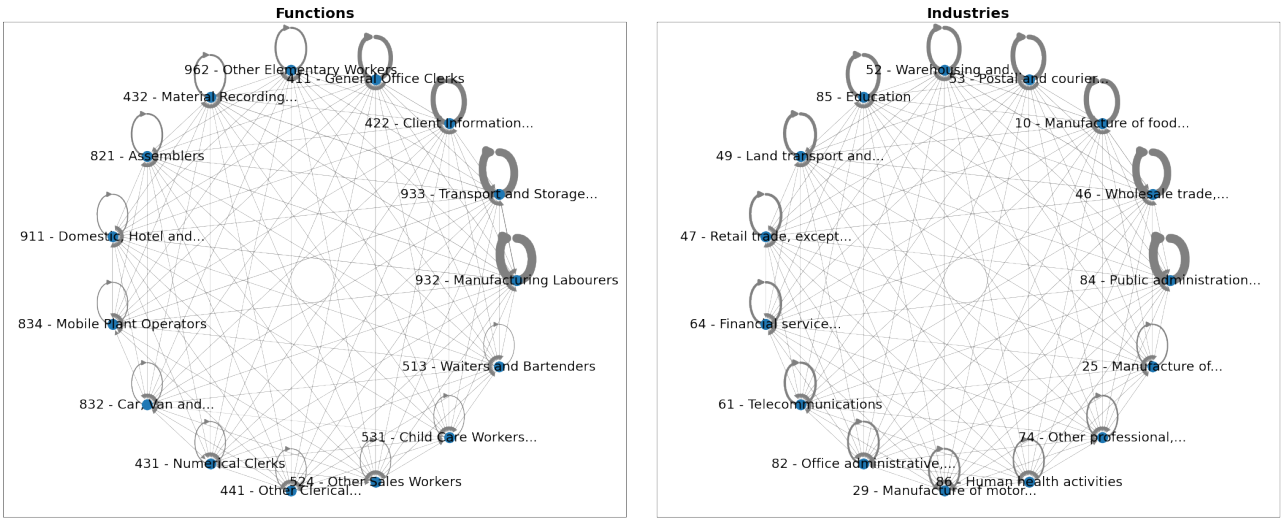


Figure 4.2: **Observed career paths:** A directed graph, showing the transitions between the 15 most common functions and industries.

Model	Mean Factual €	Mean CF €	Change %	p-value	Gainers %	Mean Gain %	Losers %	Mean Loss %
Baseline: Most Common	90283.42	89644.81	-0.7	0.69	8.85	8.17	11.62	-8.40
Baseline: Highest Exp. Reward	90283.42	89434.75	-0.94	0.59	8.39	7.50	12.52	-8.48
Double Q-Learning	90283.42	95077.13	**5.3**	0.01	27.53	13.81	12.56	-7.63
Expected Sarsa	90283.42	94836.08	**5.04**	0.01	32.84	11.50	10.95	-7.46

Table 4.1: **Naive Environment**: Factual vs Counterfactual career paths. The metrics reported are described in Section 3.4.

if they had followed the policy suggested by the Sarsa model. The same I do for people whose income would have been decreased in Table 4.4. The first two columns consist of the employee identifier and the uplift (positive or negative) that the employee would have gained in the counterfactual world. Then, each sub-row for the rest of the columns is a time step (with duration reported in the last column) and shows the jobs that the employee had in the two worlds (factual and counterfactual). Lastly, the columns *Factual €* and *CF €* show the income the person accumulated in the two worlds during this period of time. One important note is that in the factual world a employee can have more than one job in parallel. If that is the case, then their income is calculated by averaging the salaries of the different jobs. In the counterfactual world, a person can have only one job at a time.

4.1 Naive Environment

Starting with the Naive Environment, I implemented and trained two models with Double Q-Learning and Expected Sarsa methods. The hyperparameters used for training were selected using grid search and can be found in Table 4.2. Then, I evaluated the learned policies against the baselines and reported the result below. In Table 4.1, I present the results from the Counterfactual evaluation for each method on the Naive Environment. Subsequently, in the following subsections, I present more insights into each experiment described above.

4.1.1 Baselines

As can be observed from Table 4.1, the baseline methods do not yield any significant changes compared to the factual career paths. In addition, by looking into the Starting vs Final distribution figures (Figures 4.3 and 4.5) we can see that there are not significant differences compared to the observed career paths (Figure 4.1). That means that the baseline models preferences match those of the real world. Finally, the Transitions graphs (Figures 4.4 and 4.6) show that the two models work as expected. First, the *Greedy most common*, always selects the action with the highest transition probability, and almost always, that is to stay in the current function/industry. This is validated by Figure 4.4. Secondly, the *Greedy highest expected reward* baseline, follows a similar behavior. This is expected because almost always the transition probability for staying in the current function/industry is significantly larger than the second most common. This means that all the transitions which offer higher rewards would, almost never, be selected as their expected reward ($P(s'|s,a) * R(s')$) would be significantly smaller.

Greedy most common

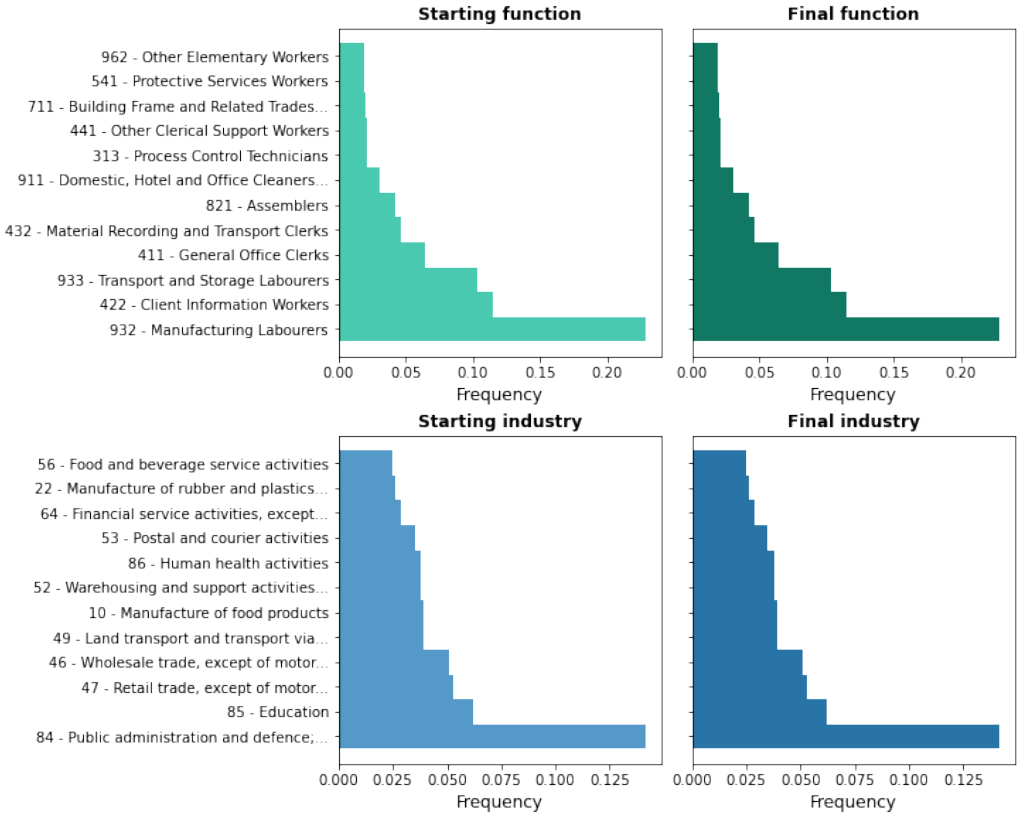


Figure 4.3: **Naive Environment - Greedy Most Common Baseline:** Starting and Final distributions for the 12 most common functions and industries. The data were generated by running 1000 episodes of 40 time steps (10 years) each.

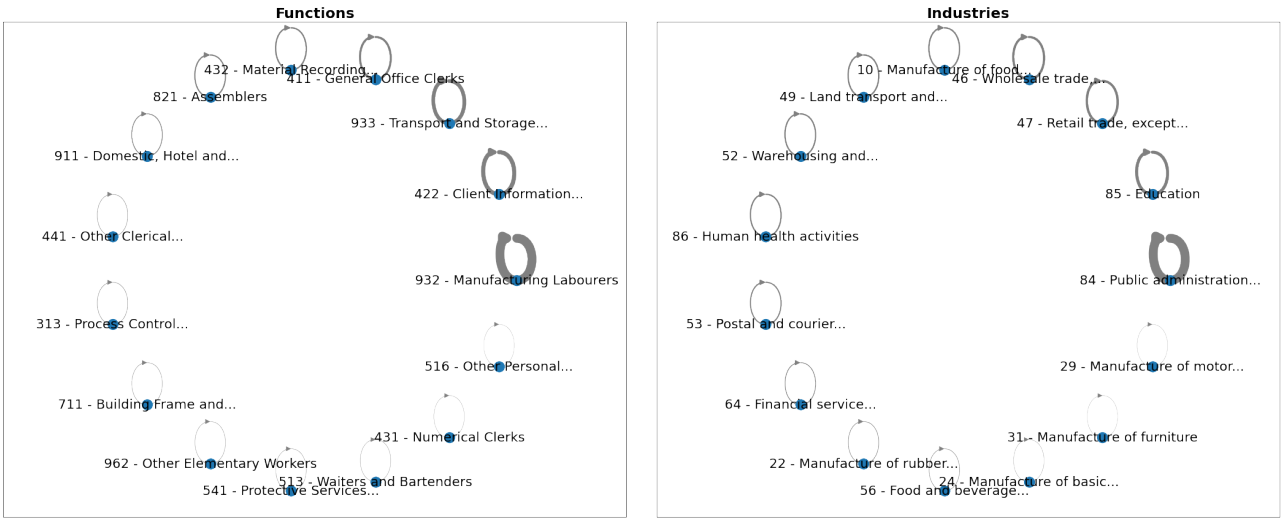


Figure 4.4: **Naive Environment - Greedy Most Common Baseline:** A directed graph, showing the transitions between the 15 most common functions and industries. The data were generated by running 1000 episodes of 40 time steps (10 years) each.

Greedy highest expected reward

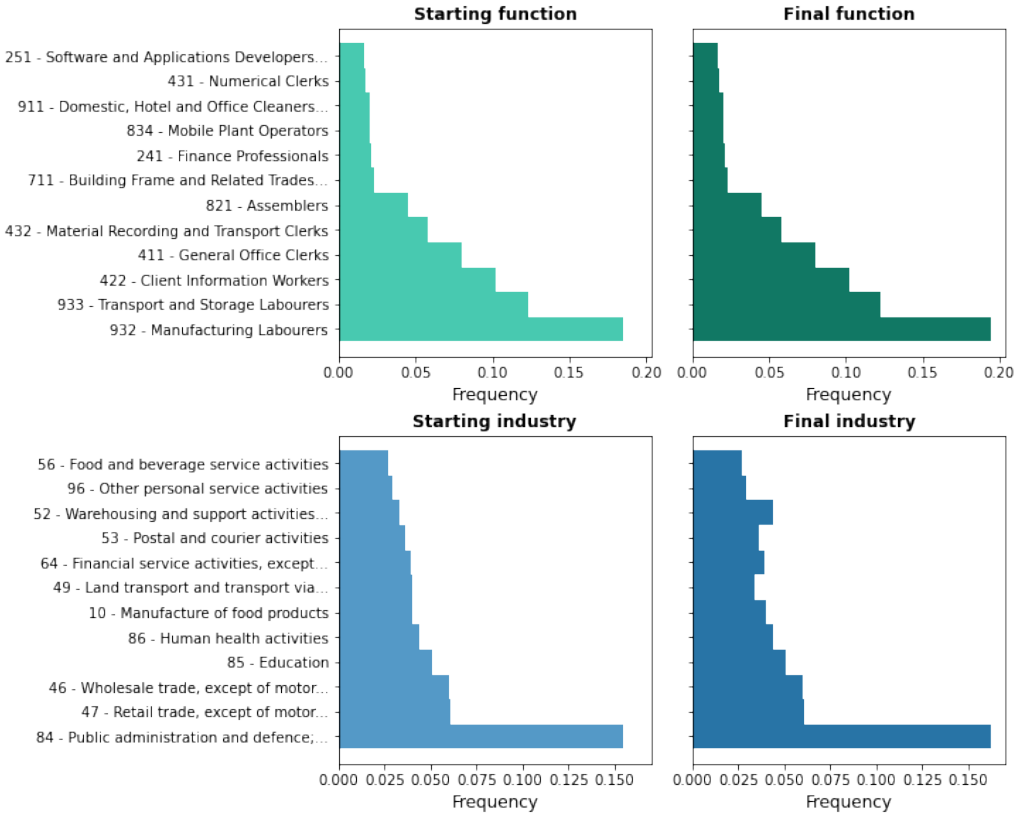


Figure 4.5: **Naive Environment - Greedy Highest Expected Reward Baseline:** Starting and final distributions for the 12 most common functions and industries. The data were generated by running 1000 episodes of 40 time steps (10 years) each.

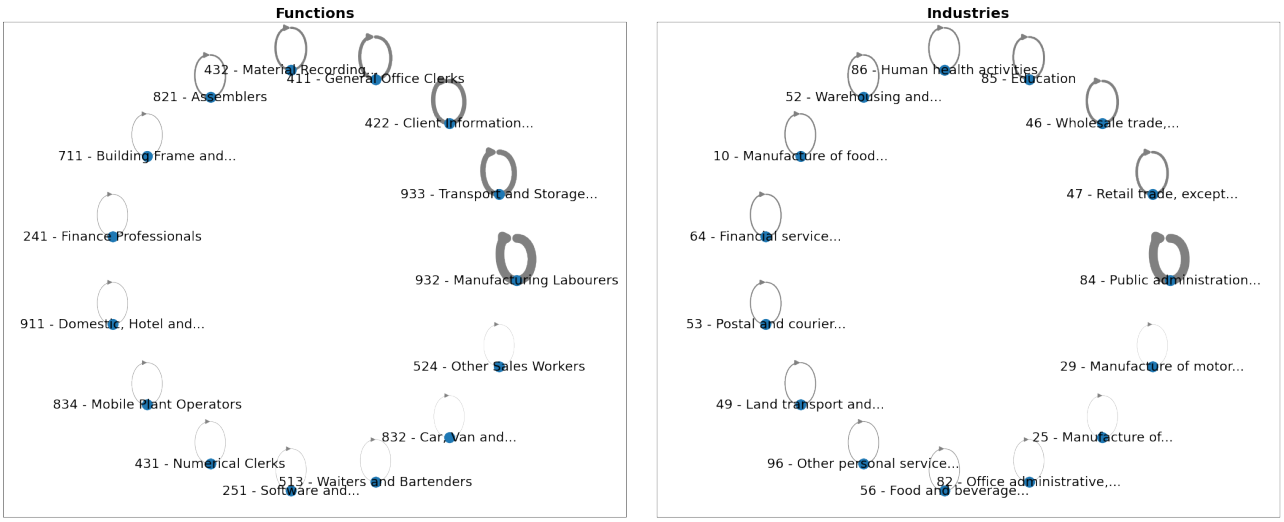


Figure 4.6: **Naive Environment - Greedy Highest Expected Reward Baseline:** A directed graph, showing the transitions between the 15 most common functions and industries. The data were generated by running 1000 episodes of 40 timesteps (10 years) each.

Method	Episodes	Learning rate	γ	Init e	e discount rate	min e
Double Q-Learning	100M	0.05	0.99	0.9	10e-8	0.1
Exp Sarsa	100M	0.05	0.99	0.9	10e-8	0.1

Table 4.2: Hyperparameters for Q-Learning and Sarsa training on the Naive environment.

4.1.2 Expected Sarsa

After training a model with the *Expected Sarsa* algorithm and with the hyperparameters displayed in Table 4.2, I performed the aforementioned experiments. As can be seen from Figure 4.7 and Figure 4.8 the model prefers certain functions (such as finance and software) and industries (such as the public administration industry). From Table 4.1 we can see that the mean counterfactual income has increased by 5.04% compared to the mean factual income.

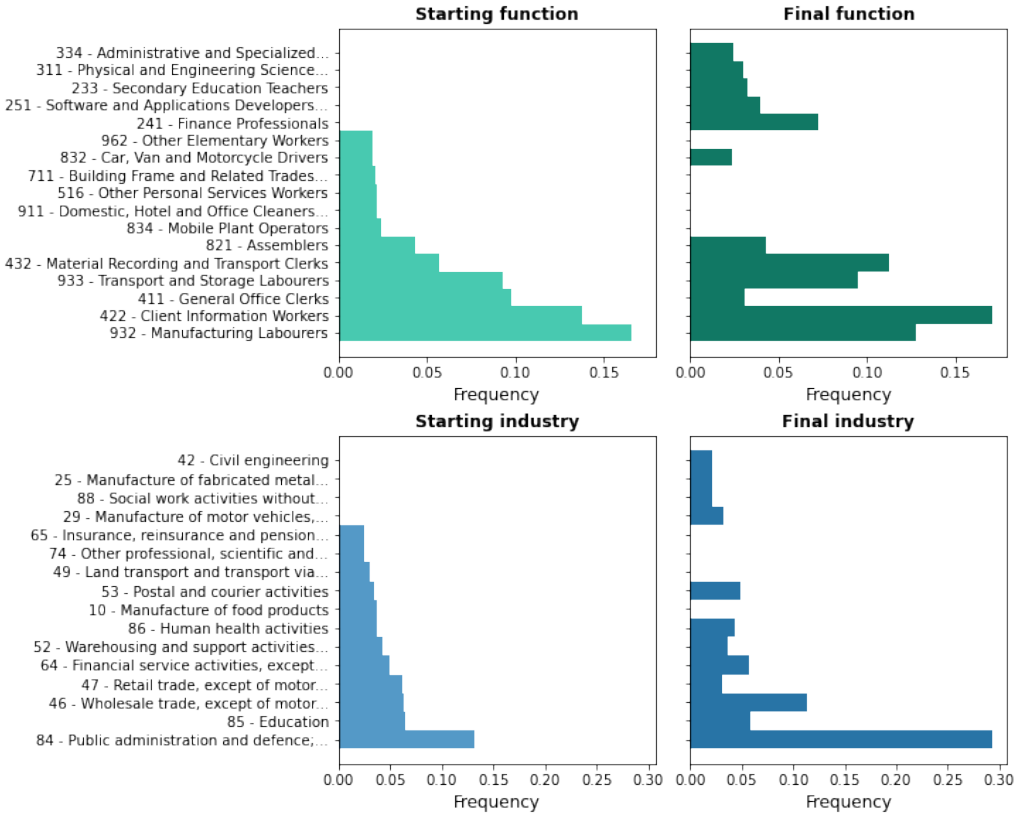
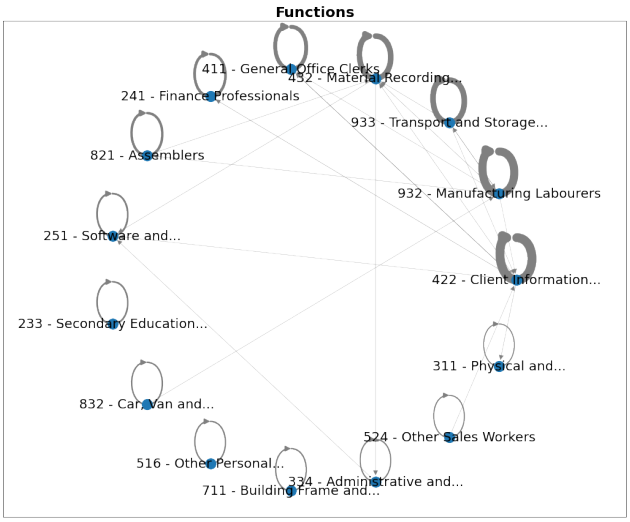


Figure 4.7: **Naive Environment - Expected Sarsa:** Starting and final distributions for the 12 most common functions and industries. The data were generated by running 1000 episodes of 40 time steps (10 years) each.



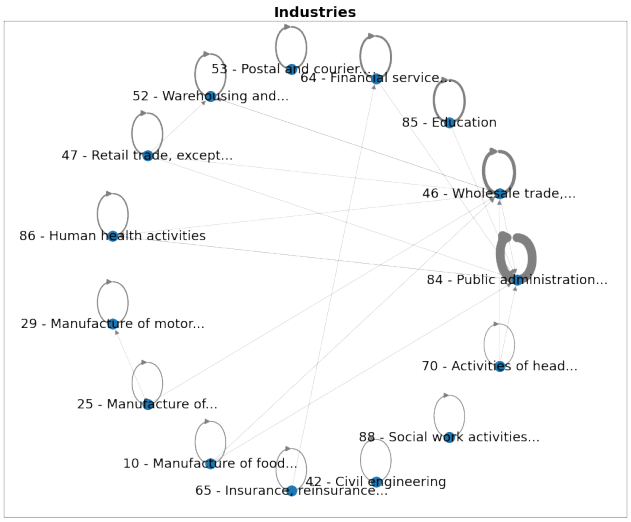


Figure 4.8: **Naive Environment - Expected Sarsa:** A directed graph, showing the transitions between the 15 most common functions and industries. The data were generated by running 1000 episodes of 40 time steps (10 years) each.

Candidate	Uplift %	Factual Job	CF Job	Factual €	CF €	Months
5340607	28.87	(Material Recording and...(F432)-I46)	(Material Recording and...(F432)-(I46))	62531	62531	21
			(Material Recording and...(F432)-(I47))	17866	18464	6
		(Transport and Storage...(F933)-I53)	(Material Recording and...(F432)-(I47))	37200	46160	15
			(Manufacturing Laborers(F932)-(I24))	52080	91504	21
6564102	15.52	(Client Information Workers(F422)-I64)	(Client Information Workers(F422)-(I64))	163026	163026	54
			(Finance Professionals(F241)-(I64))	72456	109008	24
7824790	19.60	(Manufacturing Labourers(F932)-I70)	(Manufacturing Labourers(F932)-(I70))	8731	8731	3
		(Manufacturing Laborers(F932)-I82), (Manufacturing Laborers(F932)-I70)	(Manufacturing Laborers(F932)-(I70))	7778	8731	3
		(Manufacturing Laborers(F932)-I82)	(Manufacturing Laborers(F932)-(I70))	85704	104772	36
7941359	20.16	(Transport and Storage...(F933)-I49)	(Transport and Storage...(F933)-(I49))	20391	20391	9
			(Transport and Storage...(F933)-(I46))	6797	6921	3
			(Material Recording and...(F432)-(I46))	47579	62531	21
8034214	21.31	(Manufacturing Labourers(F932)-I52)	(Manufacturing Labourers(F932)-(I52))	25896	25896	12
			(Material Recording and...(F432)-(I52))	45318	60494	21

Table 4.3: **Naive Environment - Expected Sarsa:** Factual and Counterfactual paths for a sample of 10 employees whose income has **increased** in the CF world.

Candidate	Uplift %	Factual Job	CF Job	Factual €	CF €	Months
6084922	-18.30	(General Office Clerks(F411)-I84)	(General Office Clerks(F411)-(I84))	107184	107184	36
		(Government Regulatory...(F335)-I84)	(General Office Clerks(F411)-(I84))	231724	169708	57
7599315	-15.27	(Client Information Workers(F422)-I70)	(Client Information Workers(F422)-(I70))	16600	16600	6
		(Client Information Workers(F422)-I84), (Client Information Workers(F422)-I70)	(Client Information Workers(F422)-(I70))	9040	8300	3
		(Client Information Workers(F422)-I84)	(Client Information Workers(F422)-(I70))	131105	107900	39
7763946	-25.99	(General Office Clerks(F411)-I86)	(General Office Clerks(F411)-(I86))	16820	16820	6
		(Finance Professionals(F241)-I69)	(General Office Clerks(F411)-(I86))	25938	16820	6
			(General Office Clerks(F411)-(I84))	38907	26796	9
7867289	-38.98	(Transport and Storage...(F933)-I52)	(Transport and Storage...(F933)-(I52))	6827	6827	3
		(Social and Religious...(F263)-I84), (Transport and Storage...(F933)-I52)	(Transport and Storage...(F933)-(I52))	138624	81924	36
7888747	-24.58	(Client Information Workers(F422)-I61)	(Client Information Workers(F422)-(I61))	7265	7265	3
		(Building and Housekeeping...(F515)-I47)	(Client Information Workers(F422)-(I61))	15454	14530	6
		(Sales and Purchasing Agents...(F332)-I35)	(Client Information Workers(F422)-(I61))	38439	21795	9
			(Client Information Workers(F422)-(I84))	76878	60510	18

Table 4.4: **Naive Environment - Expected Sarsa:** Factual and Counterfactual paths for a sample of 10 employees whose income has **decreased** in the CF world.

4.1.3 Double Q-learning

Similarly to Sarsa, after training a model with the *Double Q-learning* algorithm and with the hyperparameters displayed in Table 4.2, I performed the aforementioned experiments. As can be

seen from Figure 4.9 and Figure 4.10 the model prefers certain functions and industries. Namely, the legal and finance occupations and the public administration industry. From Table 4.1 we can see that the mean counterfactual income has increased by 5.3% compared to the mean factual income.

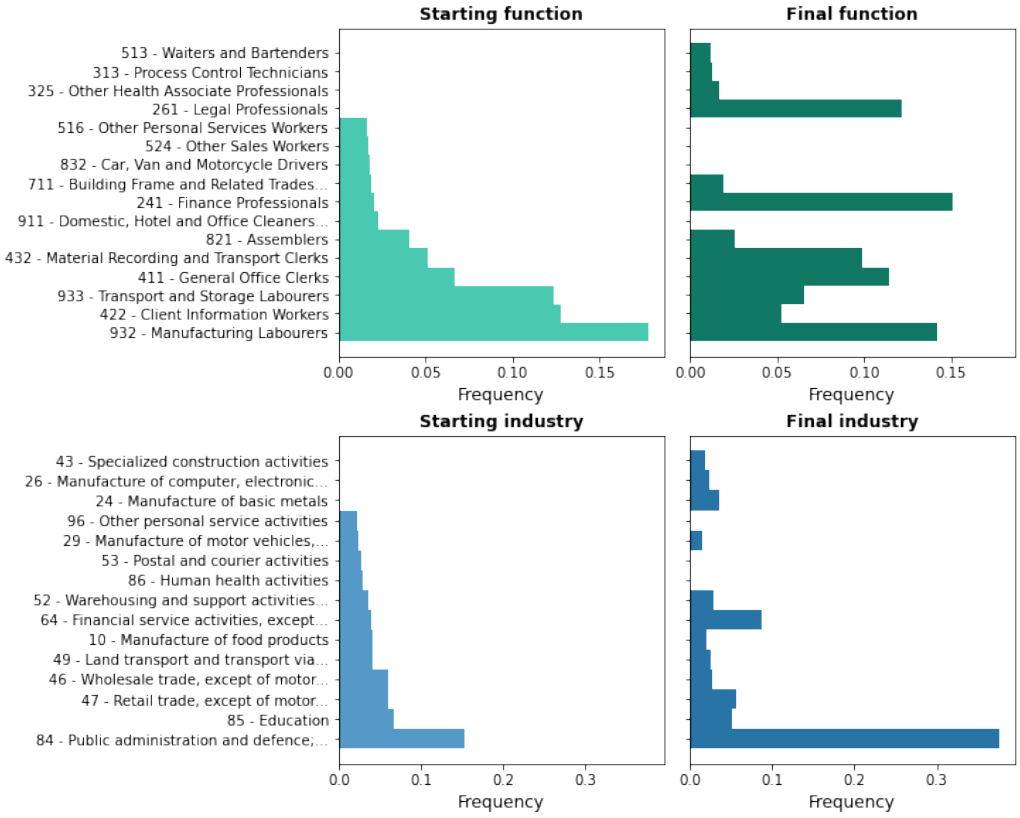


Figure 4.9: **Naive Environment - Double Q-Learning:** Starting and final distributions for the 12 most common functions and industries. The data were generated by running 1000 episodes of 40 time steps (10 years) each.

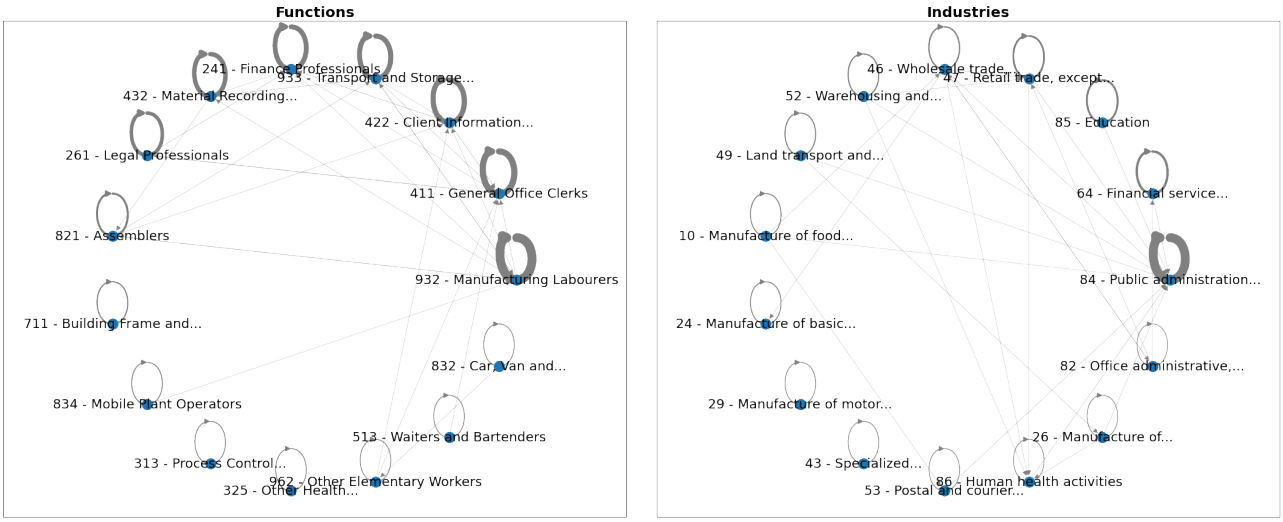


Figure 4.10: **Naive Environment - Double Q-Learning:** A directed graph, showing the transitions between the 15 most common functions and industries. The data were generated by running 1000 episodes of 40 time steps (10 years) each.

Candidate	Uplift %	Factual Job	CF Job	Factual €	CF €	Months
276256	12.93	(General Office Clerks(F411)-I84)	(General Office Clerks(F411)-(I84))	89320	89320	30
		(General Office Clerks(F411)-I85)	(General Office Clerks(F411)-(I84))	55986	62524	21
		(Waiters and Bartenders(F513)-I55)	(General Office Clerks(F411)-(I84))	14384	17864	6
		(Waiters and Bartenders(F513)-I55), (Other Sales Workers(F524)-I55), (Other Sales Workers(F524)-I56)	(General Office Clerks(F411)-(I84))	7521	8932	3
		(Other Sales Workers(F524)-I55), (Other Sales Workers(F524)-I56)	(General Office Clerks(F411)-(I84))	15047	17864	6
		(Waiters and Bartenders(F513)-I46)	(General Office Clerks(F411)-(I84))	292296	339416	114
2906652	11.29	(General Office Clerks(F411)-I78), (Client Information Workers(F422)-I82)	(General Office Clerks(F411)-(I78))	8196	8937	3
		(Client Information Workers(F422)-I82)	(General Office Clerks(F411)-(I78))	56119	62559	21
			(General Office Clerks(F411)-(I84))	64136	71456	24
7502099	17.65	(Client Information Workers(F422)-I61)	(Client Information Workers(F422)-(I61))	21795	21795	9
			(General Office Clerks(F411)-(I84))	72650	89320	30
8100517	21.66	(Other Elementary Workers(F962)-I53)	(Other Elementary Workers(F962)-(I53))	14124	14124	6
			(General Office Clerks(F411)-(I84))	63558	80388	27
8242420	11.47	(Client Information Workers(F422)-I61)	(Client Information Workers(F422)-(I61))	29060	29060	12
			(General Office Clerks(F411)-(I84))	29060	35728	12

Table 4.5: **Naive Environment - Q-Learning:** Factual and Counterfactual paths for a sample of 10 employees whose income has **increased** in the CF world.

Candidate	Uplift %	Factual Job	CF Job	Factual €	CF €	Months
5249751	-24.59	(Transport and Storage...(F933)-I84), (Client Information Workers(F422)-I84)	(Client Information Workers(F422)-(I84))	10049	10085	3
		(Client Information Workers(F422)-I65)	(Client Information Workers(F422)-(I84))	42008	40340	12
		(General Office Clerks(F411)-I84)	(Client Information Workers(F422)-(I84))	35728	40340	12
		(Legal, Social and Religious...(F341)-I84)	(Client Information Workers(F422)-(I84))	11276	10085	3
		(Legal, Social and Religious...(F341)-I84), (Administration Professionals(F242)-I84)	(Client Information Workers(F422)-(I84))	14424	10085	3
		(Administration Professionals(F242)-I84)	(Client Information Workers(F422)-(I84))	28928	20170	6
			(General Office Clerks(F411)-(I84))	173568	107184	36
5759008	-34.71	(Legal Professionals(F261)-I84)	(Legal Professionals(F261)-(I84))	28898	28898	6
			(General Office Clerks(F411)-(I84))	288980	178640	60
7669982	-10.71	(Client Information Workers(F422)-I84)	(Client Information Workers(F422)-(I84))	10085	10085	3
			(General Office Clerks(F411)-(I84))	151275	133980	45
8483505	-20.52	(Process Control Technicians(F313)-I10)	(Process Control Technicians(F313)-(I10))	9080	9080	3
			(Manufacturing Labourers(F932)-(I10))	45400	34220	15
8674767	-23.00	(Shop Salespersons(F522)-I47), (Sales and Purchasing Agents...(F332)-I14)	(Shop Salespersons(F522)-(I47))	8924	7716	3
		(Sales and Purchasing Agents...(F332)-I47), (Sales and Purchasing Agents...(F332)-I14)	(Shop Salespersons(F522)-(I47))	31158	23148	9

Table 4.6: **Naive Environment - Q-Learning:** Factual and Counterfactual paths for a sample of 10 employees whose income has **decreased** in the CF world.

4.2 Standard Environment

Next for the Naive Environment, I trained two models with Deep Q-Learning and Advantage Actor Critic methods. The implementations used are from the Stable Baselines 3 project ([Raffin et al., 2021]). The hyperparameters used for training where selected using grid search and can be found in Table 4.8. Then, I evaluated the learned policies against the baselines and reported the result below. In Table 4.7, I present the results from Counterfactual evaluation for each method on the Standard Environment. Subsequently, in the following subsections, I present more insights for each experiment described above.

Model	Mean Factual €	Mean CF €	Change %	p-value	Gainers %	Mean Gain %	Losers %	Mean Loss %
Baseline: Most Common	90386.16	95871.78	**6.18**	0.02	71.07	17.27	25.15	-13.39
Baseline: Highest Exp. Reward	90386.16	161774.45	**79.18**	0.00	96.01	80.37	0.04	-11.44
Deep Q-Learning	90283.42	94547.94	**4.7**	0.01	67.91	16.95	27.87	-13.71
A2C	90283.42	95616.29	**5.9**	0.00	70.82	17.22	25.35	-13.64

Table 4.7: **Standard Environment**: Factual vs Counterfactual career paths. The metrics reported are described in Section 3.4.

4.2.1 Baselines

In contrast to the Naive environment here the baselines show significant improvements for the counterfactual career paths. One important difference here is that the *Most Common* baseline is not choosing the most common transition, but the one with the highest probability. This probability is the one learned from the job applications dataset. The details are in Section 3.2.2. Therefore, the baseline does not reflect what most people are doing. Instead it applies for the job which the agent is more likely to be hired. The same probabilities are, of course, used for the *Highest Expected Reward* baseline.

Most Common baseline, as can been seen from Figures 4.11 and 4.12 it does not have a strong preference towards specific functions or industries. However, the *Highest Expected Reward* has a strong preference toward the Finance function (241) and the Public Administration industry (84). This is happening because, the estimated transition probabilities have a low variance and therefore the expected reward ($P(s'|s,a) * R(s')$) is mainly influenced by the rewards. In this specific environment, the Finance jobs in the Public administration industry are the highest in terms of salaries. Therefore, the model always aims for them.

Greedy most common

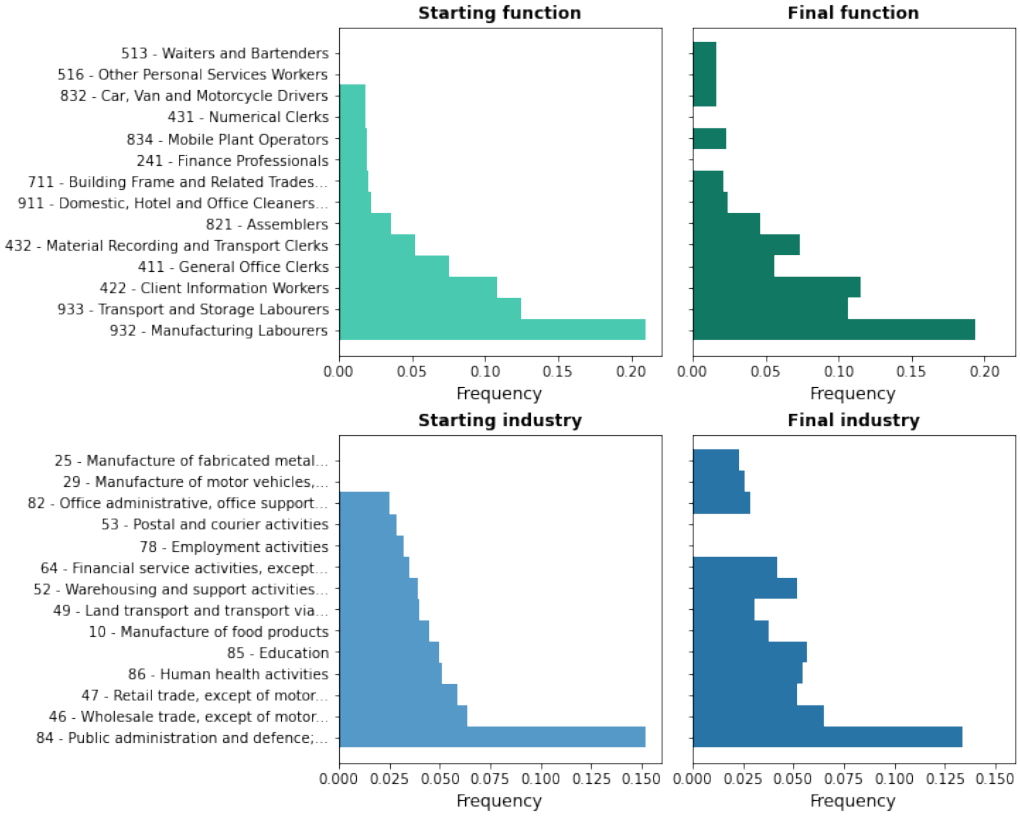


Figure 4.11: **Standard Environment - Greedy Most Common Baseline:** Starting and final distributions for the 12 most common functions and industries. The data were generated by running 1000 episodes of 40 time steps (10 years) each.

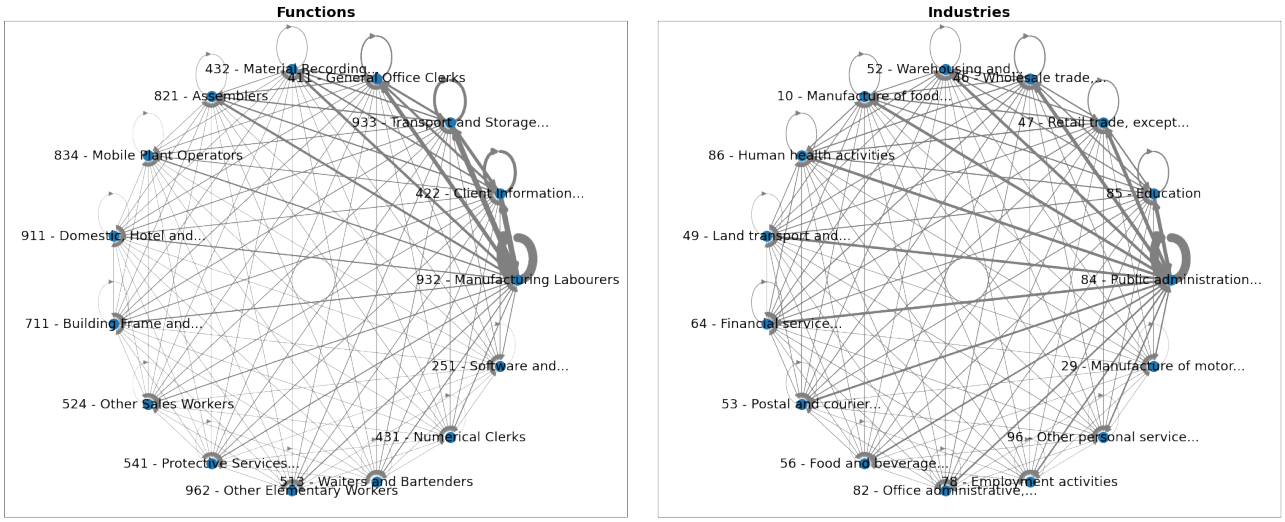


Figure 4.12: **Standard Environment - Greedy Most Common Baseline:** A directed graph, showing the transitions between the 15 most common functions and industries. The data were generated by running 1000 episodes of 40 time steps (10 years) each.

Greedy highest expected reward

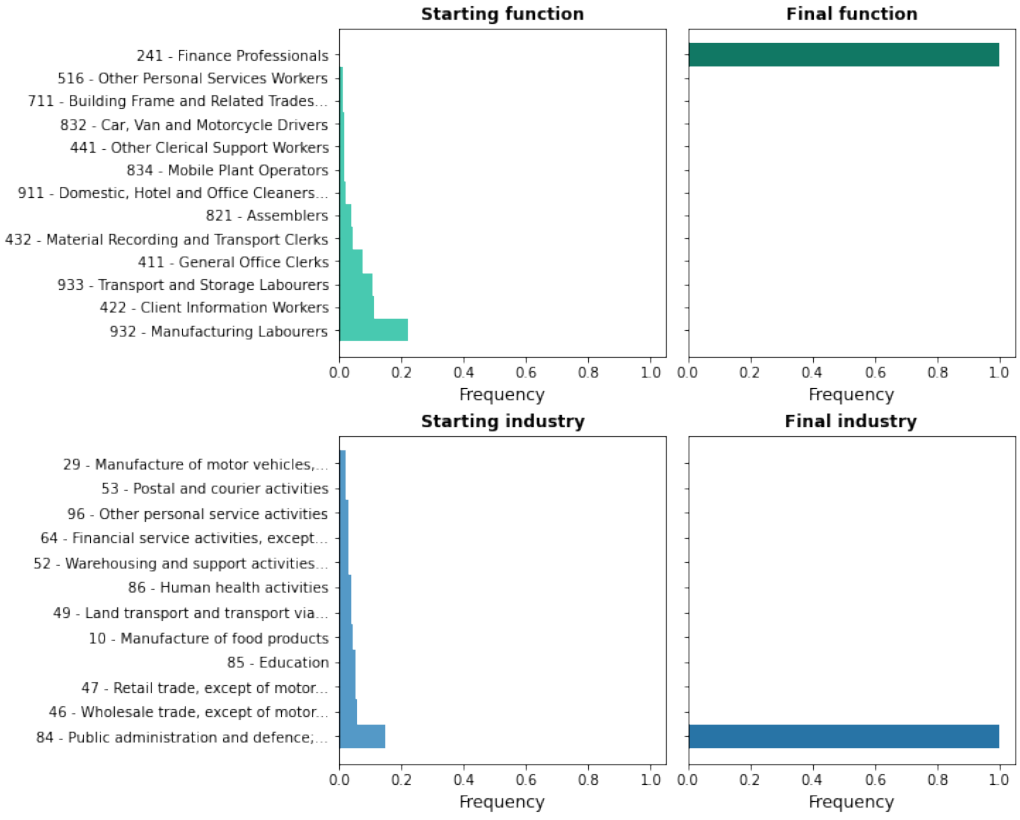


Figure 4.13: **Standard Environment - Greedy Highest Expected Reward Baseline:** Starting and final distributions for the 12 most common functions and industries. The data were generated by running 1000 episodes of 40 time steps (10 years) each.

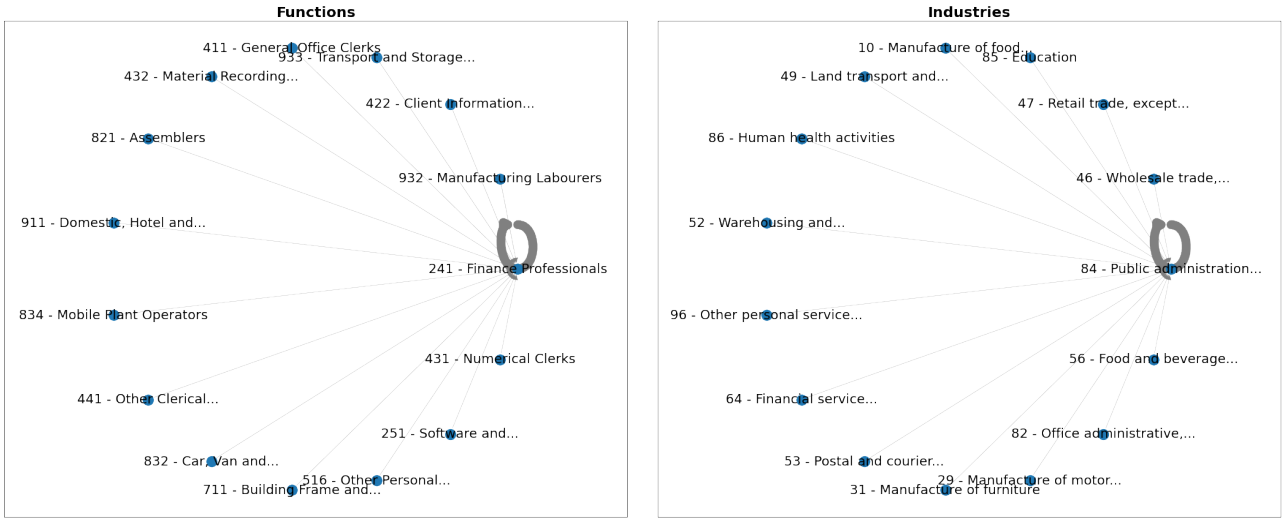


Figure 4.14: **Standard Environment - Greedy Highest Expected Reward Baseline:** A directed graph, showing the transitions between the 15 most common functions and industries. The data were generated by running 1000 episodes of 40 time steps (10 years) each.

4.2.2 Deep Q-learning (DQN)

As can be seen from Table 4.7, DQN managed to learn a policy which increased the mean cumulative income by 4.7%. However, it did not manage to outperform the two baseline methods. I assume this is because this method was not as able to learn, in the given training time, the dynamics of environment which would have enabled it to exploit the environment in the way the *Highest Expected Reward* baseline did.

Method	**Episodes**	**Learning rate**	γ	**Init e**	**e discount rate**	**min e**
DQN	40M	0.01	0.99	0.9	10e-8	0.1
A2C	40M	0.01	0.99	0.9	10e-8	0.1

Table 4.8: Hyperparameters for DQN and A2C training on the Standard environment.

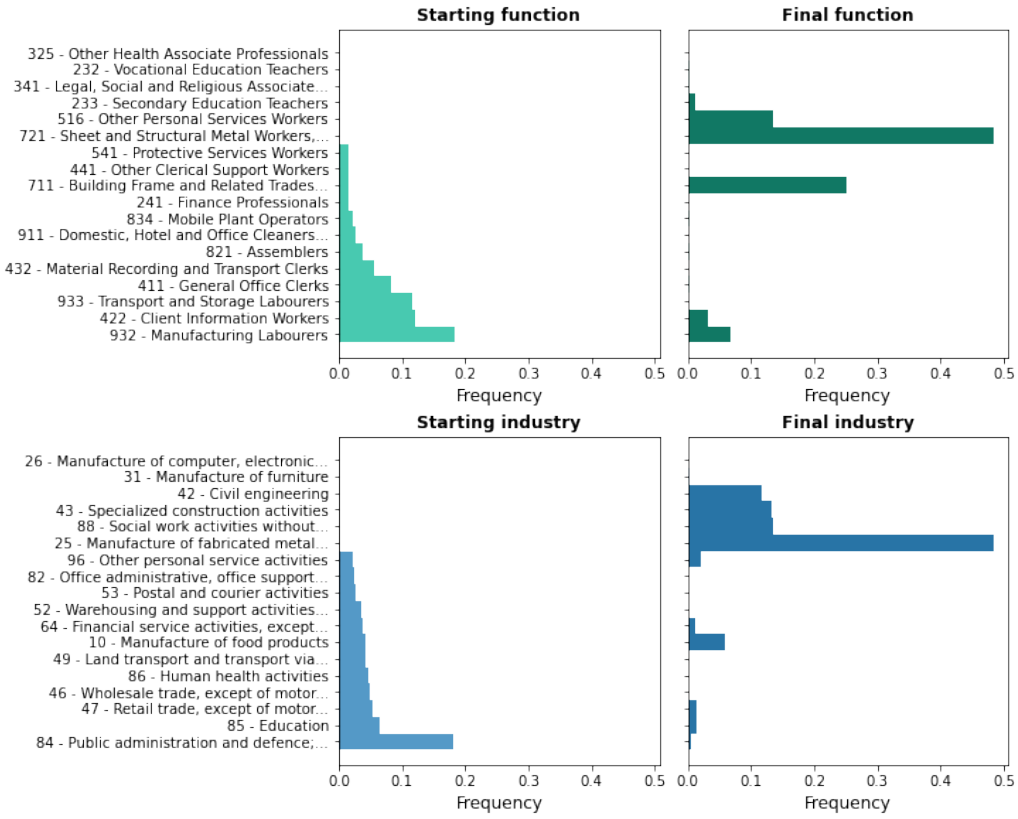


Figure 4.15: **Standard Environment - Deep Q-Learning:** Starting and final distributions for the 12 most common functions and industries. The data were generated by running 1000 episodes of 40 time steps (10 years) each.

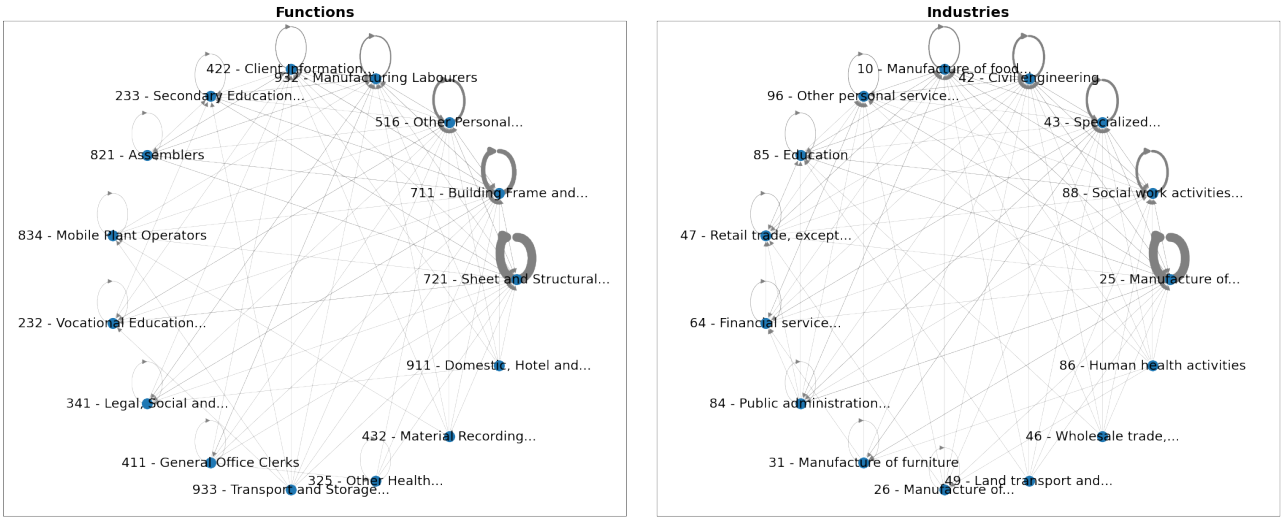


Figure 4.16: **Standard Environment - Deep Q-Learning:** A directed graph, showing the transitions between the 15 most common functions and industries. The data were generated by running 1000 episodes of 40 time steps (10 years) each.

4.2.3 Advantage Actor Critic (A2C)

Similarly to the DQN method, A2C managed to learn a policy which increased the mean cumulative income by 5.9%. Although this a little improvement compared to DQN, it did not

Candidate	Uplift %	Factual Job	CF Job	Factual €	CF €	Months
1164170	46.72	(Manufacturing Labourers(F932)-I78), (Assemblers(F821)-I25)	(Assemblers(F821)-(I25))	8106	8375	3
		(Assemblers(F821)-I25)	(Software and Applications...(F251)-(I86))	125625	203115	45
		(Domestic, Hotel and Office...(F911)-I81)	(Software and Applications...(F251)-(I86))	21486	40623	9
		(Business Services Agents(F333)-I25)	(Software and Applications...(F251)-(I86))	227940	270820	60
		(General Office Clerks(F411)-I87)	(Software and Applications...(F251)-(I86))	152424	243738	54
		(Client Information Workers(F422)-I46)	(Software and Applications...(F251)-(I86))	70002	121869	27
7826842	47.63	(General Office Clerks(F411)-I84)	(General Office Clerks(F411)-(I84))	8932	8932	3
			(Software and Applications...(F251)-(I86))	107184	162492	36
7873140	93.39	(Manufacturing Labourers(F932)-I96)	(Manufacturing Labourers(F932)-(I96))	7190	7190	3
			(Administration Professionals(F242)-(I84))	86280	173568	36
8046614	74.55	(Transport and Storage...(F933)-I53)	(Transport and Storage...(F933)-(I53))	7440	7440	3
			(Software and Applications...(F251)-(I86))	74400	135410	30
8509834	10.78	(Client Information Workers(F422)-I61)	(Client Information Workers(F422)-(I61))	7265	7265	3
			(General Office Clerks(F411)-(I47))	36325	41030	15

Table 4.9: **Standard Environment - Deep Q-Learning:** Factual and Counterfactual paths for a sample of 10 employees whose income has **increased** in the CF world.

Candidate	Uplift %	Factual Job	CF Job	Factual €	CF €	Months
7704548	-19.94	(General Office Clerks(F411)-I84)	(General Office Clerks(F411)-(I84))	8932	8932	3
			(Manufacturing Laborers(F932)-(I22))	125048	98336	42
8254475	-13.51	(General Office Clerks(F411)-I85)	(General Office Clerks(F411)-(I85))	7998	7998	3
		(Sales, Marketing and Public...(F243)-I85), (General Office Clerks(F411)-I85)	(Heavy Truck and Bus Drivers(F833)-(I49))	8401	7496	3
		(Sales, Marketing and Public...(F243)-I85)	(Heavy Truck and Bus Drivers(F833)-(I49))	12131	7496	3
		(General Office Clerks(F411)-I85)	(Heavy Truck and Bus Drivers(F833)-(I49))	31992	29984	12
		(Creative and Performing...(F265)-I65)	(Heavy Truck and Bus Drivers(F833)-(I49))	18058	14992	6
8498112	-36.68	(University and Higher...(F231)-I85)	(University and Higher...(F231)-(I85))	12547	12547	3
			(Manufacturing Laborers(F932)-(I22))	62735	35120	15
8554415	-10.21	(General Office Clerks(F411)-I49), (Transport and Storage...(F933)-I46)	(Transport and Storage...(F933)-(I46))	7685	6921	3
		(General Office Clerks(F411)-I49)	(Manufacturing Laborers(F932)-(I22))	31316	28096	12
8801913	-15.17	(Client Information Workers(F422)-I84)	(Client Information Workers(F422)-(I84))	10085	10085	3
			(Manufacturing Laborers(F932)-(I22))	10085	7024	3

Table 4.10: **Standard Environment - Deep Q-Learning:** Factual and Counterfactual paths for a sample of 10 employees whose income has **decreased** in the CF world.

manage to outperform the two baseline methods either. As mentioned before, I assume this is because this method was not as able to learn, in the given training time, the dynamics of the environment which would have enabled it to exploit the environment in the way the *Highest Expected Reward* baseline did.

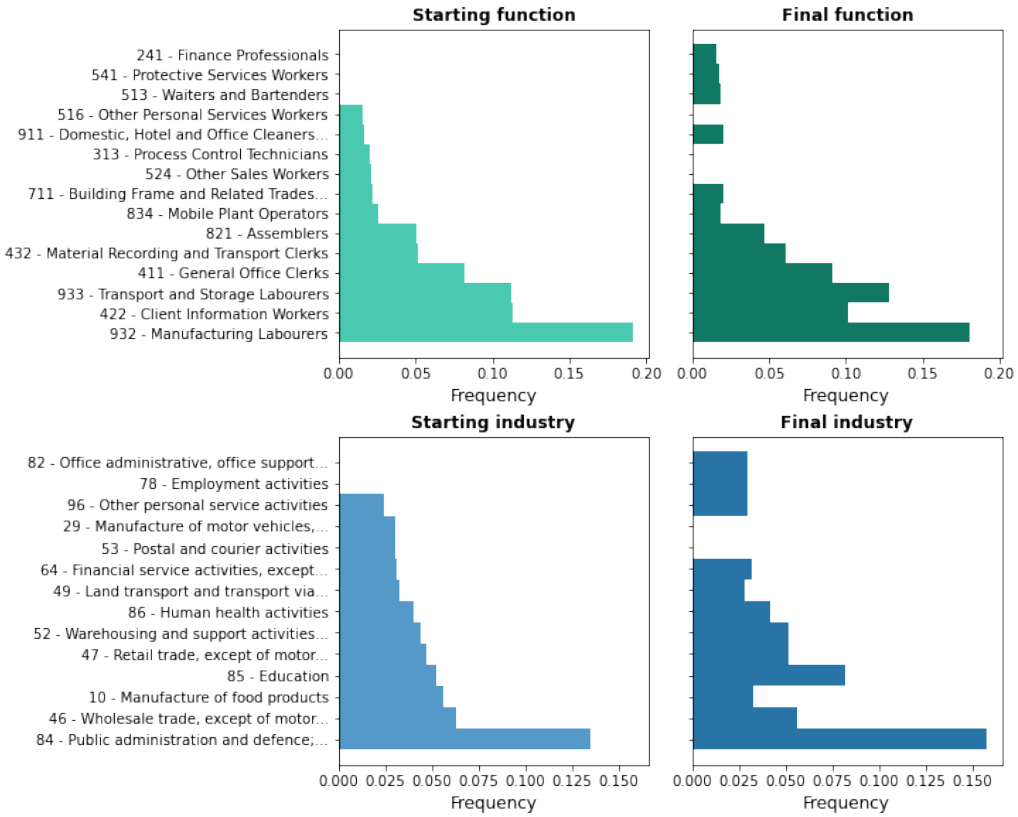


Figure 4.17: **Standard Environment - A2C:** Starting and final distributions for the 12 most common functions and industries. The data were generated by running 1000 episodes of 40 timesteps (10 years) each.

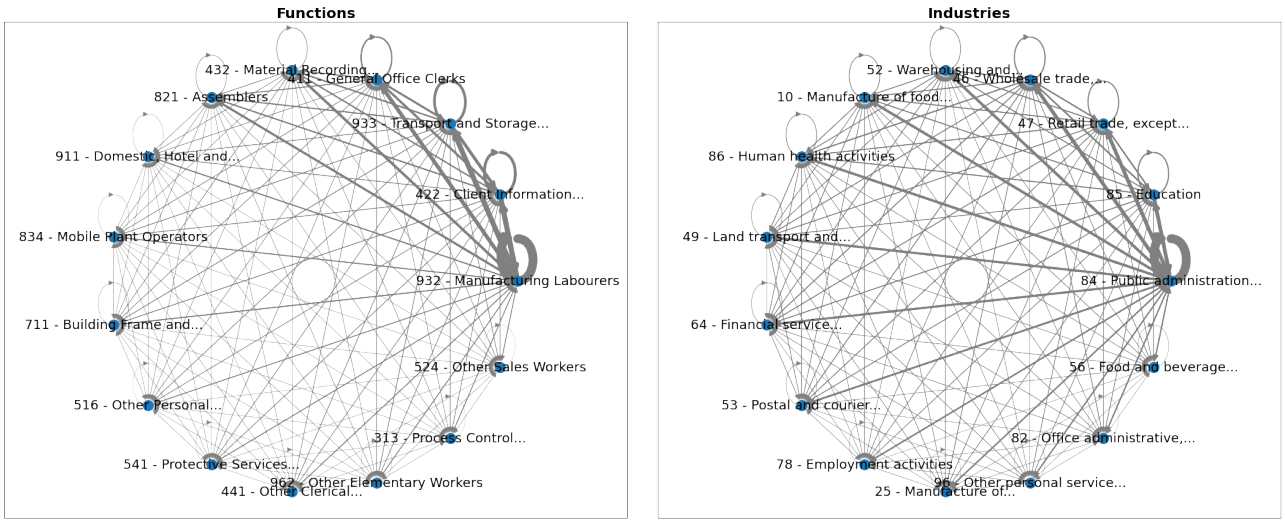


Figure 4.18: **Standard Environment - A2C:** A directed graph, showing the transitions between the 15 most common functions and industries. The data were generated by running 1000 episodes of 40 timesteps (10 years) each.

Chapter 5

Conclusions

Finally, I will discuss some conclusions I made through this research and the limitations I faced. Lastly, I will suggest some options which I think are interesting to be explored in the future.

5.1 Learned policies

As presented in the previous chapter, the methods I experimented with managed to learn policies which increase employees' incomes in the counterfactual world. Below I will discuss some conclusions I made by observing the learned policies.

5.1.1 Naive environment

In the Naive environment, the learned policies (Q-Leaning and Sarsa) manage to improve the mean accumulated by $5 - 6\%$. This is not a huge increase, but it is significant on big time scales, for example a lifetime. In addition, these methods managed to outperform the baselines.

5.1.2 Standard environment

On the Standard environment, the results are different from the Naive one. The two methods I used (DQN and A2C) managed to find policies which improve the counterfactual incomes by $4.5 - 6\%$. However, the baselines yield significantly bigger improvements. The *Highest Exp. Reward* baseline show improvements of 79% in the mean accumulated income. This observation, together with the observations from Figures 4.13 and 4.14 raise questions about the truthiness of the environment. As can be seen from the figures as mentioned earlier, the *Highest Exp. Reward* method achieves such high improvements by driving all employees towards Finance jobs in the Public Administration industry. Of course, the fact that the environment allows every employee, no matter their background, to achieve such a job means that are flaws in the learned environment dynamics.

After some investigation, I found this problem's root cause to be related to the data used to train the classification model which predicts the transitions probabilities (Section 3.2.2). The issue seems to be that many empty candidate profiles exist, which I can not simply discard. That is because I can not distinguish between people with no prior work experience and those with experience who did not list it on their profile.

5.1.3 Comparing the two environment

One important thing to mention is that the experiments' results from the two environments can not be compared directly. This is because each method is trying to learn a policy which

will exploit the dynamics of the environment it is trained on. Therefore, the ground truth is different for each environment. Consequently, the experiment results can only be compared in the scope of a specific environment.

5.2 Filtering the jobs

All the experiments I presented were performed by using only the 142 most common jobs in our datasets. This number was selected by keeping only jobs that appear at least 500 times in the data. Initially, I tried avoiding such filtering but faced two serious issues. The first one was the large state space. In the ISCO classification system there are 130 occupations on the ISCO3 level. In the SBI2 industry classification system there are 100 industries. That means there are 13000 ($130 * 100$) possible jobs. If we use all of them, we have at least 13000 states and 13000^2 transitions that our models should learn to navigate. In the Naive environment, I experimented with the two tabular methods (Q-Learning and Sarsa) and by training them with larger states, I could not get good policies within 24 hours of training. The second issues that arose, was in the Standard environment. There, the transitions prediction model I trained on the job applications dataset was unable to make sensible predictions for less common jobs. For example, it overestimated candidates' probability of being hired as medical doctors. That was probably because the dataset had very few and noise data about these positions. This resulted in policies where the agent every time was trying - and eventually was achieving - to become a doctor because it was a job with high rewards.

5.3 Formulating the environment

The reinforcement learning theory is strongly tied to an underlying Markov Decision Process (MDP) framework. However, to use reinforcement learning for career path recommendations, we have to tackle the many challenges that arise when formulating the job market as an MDP. I will present some that I face below.

5.3.1 The cost of an action

An assumption I made during this research is that there is no monetary cost for applying to a job. Obviously in reality, this is not true. Applying for a job takes time from other activities which can yield income for an employee. For example an employee might need to take a day off to attend an interview. Therefore, to have a formulation closer to reality, we have to estimate this cost and consider it when generating the rewards of the environment. Of course, in reality, there are other costs involved too. This could be, for example, lost family time or resting time. However, in the context of this thesis, I assumed that employees only focus on optimizing their incomes.

5.3.2 Not always working

Another assumption I made is that the employees are always working. In reality, however, a person could take, intentionally or not, breaks between jobs. That could be because of vacations, relocation, education, or other reasons. This is another design decision that someone should consider when formulating the job market as an MDP.

5.3.3 State space and the Markov property

The Markov property means that the evolution of the Markov process in the future depends only on the present state and not on past history. Therefore, when formulating the job market as an MDP we have to make sure that we respect the Markov Property. This could be done in two ways. The first option is to define a state to be a job and then assume that an employee's future depends only on their last job. This is what the *Naive environment* does. The second option is to avoid this over-simplifying assumption and take the employees' whole work experience, education and skills into the state. This is what the *Standard environment* partly does. Obviously, the second option is closer to reality. However the second option can easily yield states with too many dimensions and make policy learning hard and very slow. An option suggested in the literature for similar challenges is learning low-dimensional embeddings and reducing the state space size. This could be done for example, by using a recurrent neural network.

5.4 Future work

By closing this thesis, I want to give some ideas for future work. I believe that the main challenge when trying to build such a recommendation system is the unavailability of online interaction with the environment. As mentioned in the earlier chapter, the system can not start interacting with the job market (environment) to learn its dynamics. That is because applying for jobs and waiting for outcomes is a process which requires human interactions (interviews etc) and has delayed feedback. Therefore, such a system has to learn how to navigate the job market through offline data. In addition, due to risk of deploying such a system in production, we have to able to evaluate its performance from offline data too. This raises several challenges which need be addressed in future work. More specifically, I believe that more effort should be spent on creating better simulations of the job market. This includes incorporating features like seniority and education in the environment – something I did not have the necessary data to do. In addition, better models for predicting the hiring (transition) probabilities should be developed and used.

Bibliography

[Al-Dossari et al., 2020] Al-Dossari, H., Nughaymish, F. A., Al-Qahtani, Z., Alkahlifah, M., and Alqahtani, A. (2020). CareerRec: A Machine Learning Approach to Career Path Choice for Information Technology Graduates. *Engineering, Technology & Applied Science Research*, 10(6):6589–6596.

[Dawson et al., 2021] Dawson, N., Williams, M. A., and Rizoiu, M. A. (2021). Skill-driven recommendations for job transition pathways. *PLoS ONE*, 16(8 August):1–20.

[Fuller, 2008] Fuller, S. (2008). Job Mobility and Wage Trajectories for Men and Women in the United States. Technical Report 1.

[Gugnani et al., 2019] Gugnani, A., Reddy Kasireddy, V. K., and Ponnalagu, K. (2019). Generating unified candidate skill graph for career path recommendation. *IEEE International Conference on Data Mining Workshops, ICDMW*, 2018-Novem:328–333.

[Guo et al., 2022] Guo, P., Xiao, K., Ye, Z., Zhu, H., and Zhu, W. (2022). Intelligent career planning via stochastic subsampling reinforcement learning. *Scientific Reports 2022 12:1*, 12(1):1–16.

[Joseph et al., 2012] Joseph, D., Fong Boh, W., Ang, S., and Slaughter, S. A. (2012). The Career Paths Less (or More) Traveled: A Sequence Analysis of IT Career Histories, Mobility Patterns, and Career Success. Technical Report 2.

[Kokkodis and Ipeirotis, 2021] Kokkodis, M. and Ipeirotis, P. G. (2021). Demand-aware career path recommendations: A reinforcement learning approach. *Management Science*, 67(7):4362–4383.

[Levine et al.,] Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems.

[Li et al., 2017] Li, L., Jing, H., Tong, H., Yang, J., He, Q., and Chen, B.-C. (2017). NEMO: Next Career Move Prediction with Contextual Embedding. In *Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion*, pages 505–513, New York, New York, USA. ACM Press.

[Liu and Tan, 2020] Liu, R. and Tan, A. (2020). Towards interpretable automated machine learning for STEM career prediction. *Journal of Educational Data Mining*, 12(2):19–32.

[Liu et al., 2016] Liu, Y., Zhang, L., Nie, L., Yan, Y., and Rosenblum, D. S. (2016). Fortune teller: Predicting your career path. *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, (1):201–207.

[Long and Ferrie, 2006] Long, J. and Ferrie, J. (2006). "Labour Mobility" Oxford Encyclopedia of Economic History.

[Lou et al., 2010] Lou, Y., Ren, R., and Zhao, Y. (2010). A Machine Learning Approach for Future Career Planning. pages 1 – 4.

[Meng et al., 2019] Meng, Q., Zhu, H., Xiao, K., Zhang, L., and Xiong, H. (2019). A hierarchical career-path-aware neural network for job mobility prediction. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (November):14–24.

[Mnih et al., 2016] Mnih, V., Badia, A. P., Mirza, L., Graves, A., Harley, T., Lillicrap, T. P., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. *33rd International Conference on Machine Learning, ICML 2016*, 4:2850–2869.

[Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning.

[Moscarini and Thomsson, 2007] Moscarini, G. and Thomsson, K. (2007). Occupational and Job Mobility in the US. *Source: The Scandinavian Journal of Economics*, 109(4):807–836.

[Oentaryo et al., 2018a] Oentaryo, R. J., Lim, E.-P., Ashok, X. J. S., Prasetyo, P. K., Ong, K. H., and Lau, Z. Q. (2018a). Talent Flow Analytics in Online Professional Network. *Data Science and Engineering*, 3(3):199–220.

[Oentaryo et al., 2018b] Oentaryo, R. J., Oentaryo, R. J., Jayaraj, X., Ashok, S., Lim, E.-p., and Kokoh, P. (2018b). JobComposer : Career Path Optimization via Multicriteria Utility Learning.

[Paparrizos et al., 2011] Paparrizos, I., Cambazoglu, B. B., and Gionis, A. (2011). Machine learned job recommendation. *RecSys'11 - Proceedings of the 5th ACM Conference on Recommender Systems*, pages 325–328.

[Prudencio et al., 2022] Prudencio, R. F., Maximo, M. R. O. A., and Colombini, E. L. (2022). A Survey on Offline Reinforcement Learning: Taxonomy, Review, and Open Problems.

[Raffin et al., 2021] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021). Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268):1–8.

[Shahbazi et al., 2019] Shahbazi, B., Akbarnezhad, A., Rey, D., Ahmadian Fard Fini, A., and Loosemore, M. (2019). Optimization of Job Allocation in Construction Organizations to Maximize Workers' Career Development Opportunities. *Journal of Construction Engineering and Management*, 145(6):04019036.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. 2 edition.

[Topel and Ward, 1992] Topel, R. H. and Ward, M. P. (1992). Job Mobility and the Careers of Young Men. *The Quarterly Journal of Economics*, 107(2):439–479.

[Van Hasselt,] Van Hasselt, H. Double Q-learning.

[Wang et al., 2013] Wang, J., Zhang, Y., Posse, C., and Bhasin, A. (2013). Is it time for a career switch? In *Proceedings of the 22nd international conference on World Wide Web - WWW '13*, pages 1377–1388, Rio de Janeiro, Brazil. ACM Press.

[Watkins and Dayan, 1992] Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning 1992 8:3*, 8(3):279–292.

[Xu et al., 2019] Xu, H., Yu, Z., Yang, J., Xiong, H., and Zhu, H. (2019). Dynamic Talent Flow Analysis with Deep Sequence Prediction Modeling. *IEEE Transactions on Knowledge and Data Engineering*, 31(10):1926–1939.

[Yamashita et al., 2022] Yamashita, M., Li, Y., Tran, T., Zhang, Y., and Lee, D. (2022). Looking Further into the Future: Career Pathway Prediction. Technical report.