Université de Montréal

On Choice Models in the Context of MDPs

par

Sobhan Mohammadpour

Département de mathématiques et de statistique
Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en informatique et recherche opérationnelle

14 août 2023

Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

On Choice Models in the Context of MDPs

présenté par

Sobhan Mohammadpour

a été évalué par un jury composé des personnes suivantes :

Simon Lacoste-Julien
(président-rapporteur)

Emma Frejinger
(directeur de recherche)

Pierre-Luc Bacon
(codirecteur)

Fabian Bastin
(membre du jury)

Résumé

Cette thèse se penche sur les modèles de choix, des distributions sur des ensembles d'alternatives. Les modèles de choix sur les processus décisionnels de Markov (MDP) peuvent décomposer de très grands espaces alternatifs en procédures étape par étape conçues pour non seulement combattre la malédiction de la dimensionnalité mais aussi pour mieux refléter la dynamique sous-jacente.

La première partie est consacrée à l'estimation du temps de trajet dans le cadre de la modélisation du choix de chemin. Les modèles de choix de chemin sont des modèles de choix sur l'ensemble des chemins utilisés pour modéliser le flux de circulation. Intuitivement, le temps de trajet est l'une des caractéristiques les plus importantes lors du choix des chemins, mais les temps de trajet ne sont pas toujours connus. En revanche, le cadre classique suppose que ces deux étapes sont séquentielles, car les temps de trajet des arcs font partie de l'entrée du processus d'estimation du choix de chemin. Pourtant, les interdépendances complexes signifient que ce modèle de choix de chemin peut complémenter toute observation lors de l'estimation des temps de trajet. Nous construisons un modèle statistique pour l'estimation du temps de trajet et proposons de marginaliser les caractéristiques non observées. En utilisant ces idées, nous montrons que nous sommes capables d'apprendre des modèles de choix de chemin sans observer de chemins réels et à différentes granularités.

La deuxième partie se concentre sur les échecs des MDP régularisés et comment la régularisation peut avoir des effets secondaires inattendus, tels que la divergence dans les chemins stochastiques les plus courts ou des fonctions de valeur déraisonnablement grandes. Les MDP régularisés ne sont rien d'autre qu'une application des modèles de choix aux MDP. Ils sont utilisés dans l'apprentissage par renforcement (RL) pour obtenir, entre autres choses, un modèle de choix sur les trajectoires possibles pour l'apprentissage par renforcement inverse, transférer des connaissances préalables au modèle, ou obtenir des politiques qui exploitent tous les objectifs dans l'environnement. Ces effets secondaires sont exacerbés dans les espaces d'action dépendants de l'état. Comme mesure d'atténuation, nous introduisons

deux transformations potentielles, et nous évaluons leur performance sur un problème de conception de médicaments.

Mots clés : Estimation du temps de trajet,Modèles de choix de chemin, Modélisation du choix d'itinéraire, Apprentissage par renforcement à entropie maximale, Processus de décision de Markov régularisé

Abstract

This thesis delves on choice models, distributions on sets of alternatives. Choice models on Markov decision processes (MDPs) can break down very large alternative spaces into step-by-step procedures designed to not only tackle the curse of dimensionality but also to reflect the underlying dynamics better.

The first part is devoted to travel time estimation as part of path choice modeling. Path choice models are choice models on the set of paths used to model traffic flow. Intuitively, travel time is one of the more important features when choosing paths, yet travel times are not always known. In contrast, the classical setting assumes that these two steps are sequential, as arc travel times are part of the input of the path choice estimation process. Yet the intricate interdependences mean that that path choice model can complement any observation when estimating travel times. We build a statistical model for travel time estimation and propose marginalizing the unobserved features. Using these ideas, we show that we are able to learn path choice models without observing actual paths and at different granularity.

The second part focuses on the failings of regularized MDPs and how regularization may have unexpected side effects, such as divergence in stochastic shortest paths or unreasonably large value functions. Regularized MDPs are nothing but an application of choice models to MDPs. They are used in reinforcement learning (RL) to get, among other things, a choice model on possible trajectories for inverse reinforcement learning, transfer prior knowledge to the model, or to get policies that exploit all goals in the environment. These side effects are exacerbated in state-dependent action spaces. As a mitigation, we introduce two potential transformations, and we benchmark their performance on a drug design problem.

Keywords: Travel time estimation, Route choice modeling, Path choice models, Maximum entropy reinforcement learning, Regularized Markov decision process

Contents

List of Tables

List of Figures

Liste des sigles et des abréviations

ARUM	Additive Random Utility Maximization Model
CM	Choice model
GFN	Graph flow network
IRL	Inverse reinforcement learning
MDP	Markov decision process
PCM	Path choice model
R-MDP	Regularized Markov decision process
R-SSP	Regularized Stochastic shortest path
RAM	Representative agent model
RL	Reinforcement learning
RUM	Random Utility Maximization Model
S-MDP	Stochastic Markov decision process
SQL	Soft Q-learning
SSP	Stochastic shortest path
TB	Trajectory balance

Introduction

Choice models, foundational to various disciplines such as economics, marketing, and transportation, serve as tools to understand and predict decision-making behavior. These models, based on the premise that individuals make decisions by choosing from a set of alternatives, allow researchers to forecast outcomes under varying conditions and analyze individual or group preferences. The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel was awarded to Daniel McFadden for his work on choice models in the year 2000.

This thesis focuses on two problems related to choice models in the context of sequential processes, that is, processes that are better described by a series of small choices instead of one single choice over a large number of alternatives (Rust, 1987; Aguirregabiria and Mira, 2010). We answer two questions here, can we learn the sequential model with data at different levels of granularity? And do choice models on sequential choice models have any questionable properties?

Choice models

For the purpose of this thesis, choice models are nothing more than distributions over a set of alternatives or choices, which may not necessarily be finite. Although, in theory, any distribution over the set of alternatives can be considered a choice model, our focus is on a specific subset of choice models. These models correspond to a stochastic process where an agent being modeled selects the alternative that maximizes the function

$$\max_i v_i + \epsilon_i,$$

where v_i denotes a deterministic utility, and ϵ_i is a stochastic utility. We refer to these models as additive random utility maximization (ARUM) (Feng et al., 2017).

ARUMs were first introduced by Thurstone (1927) as a tool to explain the presence of stochasticity in human behavior. His primary focus was on the scenario involving two choices, with ϵ_i following a normal distribution. Later, Luce (1959) introduced the axiom of choice. This axiom formalizes Arrow et al. (1951)'s independence of irrelevant alternatives' principle, stipulating that the ratio of probabilities for two alternatives should be independent of other

available choices. Although this axiom finds its roots in related psychological literature, Feng et al. (2018), among others, presents an instance where such an assumption may not be appropriate. For example, if consumers buy cameras solely based on their ranking on a website, and they have to choose between two Canon models and one Nikon model, should the score of one Canon camera affect the probability of choosing the other Canon camera? Adherence to the axiom of choice is achieved by assuming independent ϵ_i.

McFadden (1977) introduced the concept of parameter estimation to choice models. In his model, ϵ_i are independent and identically distributed (iid) Gumbel random variables. The probabilities in this context are given by the softmax of the deterministic utilities v. Rust (1987) extended McFadden (1977)'s results to Markov Decision Processes (MDPs) by employing a choice model at each state within an MDP. He simplified the impact of future states by considering the expected return, also known as the societal surplus or welfare, denoted by

$$w(v) = \mathbb{E}[\max_i v_i + \epsilon_i].$$

A critical complication with this generalization is that decisions are made concerning welfare and not the rewards, which could potentially lead to agents prioritizing welfare over rewards. This challenge is the main subject of Article 2. Since the choices are state-dependent, this extension of choice models to MDPs is referred to as the Markovian variant.

ARUMs are not the only type of choice model; representative agent models (RAM), are defined as

$$\arg\max_\pi \sum_i v_i \pi_i - w^*(\pi),$$

are an alternative to ARUMs. One popular choice for w^* is the negative entropy. The corresponding RAM is called entropy regularized RAM. The equivalence between entropy regularized RAM and the Gumbel ARUM was initially observed by Anderson et al. (1988). Subsequently, Hofbauer and Sandholm (2002) showed that ARUMs are, in fact, a subset of RAMs. This hierarchical relationship was further extended by Feng et al. (2017, 2018) and adapted to MDPs by Mai and Jaillet (2020).

In the following, first, we look at how we can estimate v using data generated from MDPs and then the failings of extending RUM to MDPs.

Choice models and traffic prediction

Choice models are widely used to predict people's behavior in transport systems. The most widely used path choice models—choice models operating over the set of paths, simple or otherwise—are defined based on a restricted set of paths. A choice model is then applied to this restricted see (Zimmermann and Frejinger, 2020). However, this process has its drawbacks. The parameters estimated can become exceedingly sensitive to the set of selected paths (Prato, 2009).

Yet, the introduction of more paths is not a straightforward solution. There are two problems; first, having an infinite (or large) number of choices does not mean an infinite (or large) welfare, yet welfare continues to grow in ARUM with an unbounded ϵ. Second, as the number of choices increases, the assumption of independence between choices becomes increasingly violated. This is particularly true as paths tend to intersect more with an increase in path number. Some Markovian choice models, notably the one proposed by Fosgerau et al. (2013), operate over the entire set of paths. Yet the probabilities are skewed towards regions with many paths. The solution to these challenges, along with potential mitigation strategies, are discussed in more detail in Article 2.

Attempts have also been made to extend RAMs to MDPs, one notable example being the work of Fosgerau et al. (2022). Their approach operates in the flow vector space rather than the more conventional observation space. The extraction of probabilities is left to the reader as there exists an infinite number of distributions over the set of paths. However, it is worth noting that the Markovian policy – which is independent of history – is unique.

The process of learning parameters of a path choice model—which is a choice model over the set of paths—requires information about trajectories that agents take in a given network. Intuitively, one of the most crucial features employed by the path choice model is travel time. To estimate travel time, we have several options: we can either observe precise trajectories along with the travel time between nodes, or we can leverage sensors, like radars, to directly measure the travel time (Bertsimas et al., 2019).

When we observe travel times for entire trajectories rather than on a turn-by-turn basis, we can still learn about arc travel time. However, related methods (Bertsimas et al., 2019) grapple with a significant issue: unobserved arcs are not considered in the problem, which means they could assume any value without altering the optimization objective. Therefore, regularization becomes crucial in these methods, a topic that contrasts with the method we present in Article 1.

Access to granular and accurate data is an important issue in practice. For example, detailed trajectory information may not be available due to the GPS sampling rate not being high enough(Mai et al., 2023). Given a path choice model, we can account for missing data, but we need arc travel time information to form the path choice model. Bertsimas et al. (2019) propose a solution to this problem by assuming that the path choice model follows the shortest path. They then integrate the shortest path process into a mathematical program used to estimate travel time, accounting for total trajectory travel time.

In this context, solving the resulting problem can be computationally expensive. To circumvent this, Bertsimas et al. (2019) propose separating the binary and continuous variables into two distinct programs. They then iterate on solving the programs sequentially while keeping variables of the other program fixed. This allows them to solve the shortest path part with the shortest path algorithm, skipping the need for costly big M constraints. While

the underlying mechanics of why this approach is effective might not be immediately clear, it appears there exists a synergy between the regularizer, the data, and the algorithm.

In Article 1, we propose a method that does not have the limitation of Bertsimas et al. (2019)'s approach. We pose the whole pipeline, travel time, and path choice model estimation as a single statistical inference problem. To deal with data at different levels of granularity, we integrate out the unobserved.

The proposed methodology is favorable to methods like optimizing the divergence between the observed travel time and the expected travel time for many reasons. First and foremost, while calculating the expected travel time for Markovian policies (choice models that are not history-dependent) is equivalent to solving a linear system of equations, not all choice models are Markovian, nor differentiating through the expected is cheap. Based on our experiments, even for small examples, it may take hours to calculate the gradient for a few thousand observations. Beyond the numerical difficulties, a more theoretical question remains, should we minimize the difference between the expected and observed travel times or the expected discrepancy? Jensen's inequality offers some guidance, proposing that for convex loss functions, the divergence of the expected is upper bounded by the expected divergence. However, in our context, this insight is not particularly helpful as our loss function is neither convex nor are we aiming to maximize the discrepancy of the expected.

Lastly, naively extending Bertsimas et al. (2019)'s method by replacing the shortest path with samples does not work as shown in the toy example in Article 1 since it ignores that the changes in path choice model induce a change in travel time estimation. Both of these alternatives are the original inspiration for the aforementioned toy example.

Choice models and regularized Markov decision processes

In Article 2, identifying an appropriate scaling for the input of the model proposed by Fosgerau et al. (2013) is not trivial. Although the model is convex and the optimum is confined within the feasible domain, it is necessary that the trajectory of the parameters, or the optimization path, be feasible too. However, barring methods similar to Oyama (2023), it is not hard for the value functions to diverge to infinity. In this context, appropriate scaling fixes the problem, as small steps will still stay inside the feasible and numerically stable region.

Mai and Frejinger (2022) provides the specific condition for convergence, but at the heart of the issue is an inherent inconsistency. Since a choice model is performed at each state, the reward for maintaining a loop can exhibit a positive mean, causing the expected reward to be infinite. This outcome contradicts the foundational assumptions of the stochastic shortest path (Bertsekas, 2012), which postulates that perpetual stay within the Markov Decision

Process (MDP) should return a $-\infty$ reward. This condition is a generalization of the negative loop condition in the conventional shortest-path scenario.

A popular workaround frequently adopted by reinforcement learning (RL) practitioners involves the introduction of a discount factor. However, this approach is not without its shortcomings, particularly in the context of transportation problems. Indeed, we are yet to encounter an RL practitioner who minimizes the discounted travel time during their commute. Such an individual might favor faster travel methods even if they do not necessarily bring them closer to their destination. Complicating matters further, discount factors aren't naturally inherent to the problem at hand, and the discretized nature of transportation MDPs, which are simplifications of continuous models, exacerbates the issue.

Choice models are linked to RL through regularized Markov decision processes Geist et al. (2019) (R-MDP). R-MDPs are the extension of RAMs to MDPs in a similar vain as how Rust (1987)'s method is an extension of ARUMs to MDPs. In Article 2, we exploit the structure of RUMs to reason about a large class of R-MDPs. Due to Anderson et al. (1988); Hofbauer and Sandholm (2002), we know that the Gumbel ARUM is equivalent to entropy regularized RL (Ziebart et al., 2008; Haarnoja et al., 2018). Hence any observation on RUM is directly applicable to entropy regularized or maximum entropy RL.

Contributions

In Article 1, we provide an end-to-end model to train path choice models while also learning the arc travel time matrix. We show that the traditional pipeline can lead to catastrophic results as the bias induced by different parts of the pipeline can be absorbed by the other parts, leaving the impression that the model works, but the parameter estimates can be far from the truth. We show how to use data at different levels of granularity.

In Article 2, we show how there is a survival bias embedded inside many regularizers, encouraging agents to stay alive rather than play the game. The survival bias, while in general useful in arcade environments (Bellemare et al., 2013), can lead to the agents preferring to spin in place or preferring structures with many paths like grid-like suburbs over efficient routes like highways. We provide two transformations for the regularizers to curb these behaviors. The two regularizers significantly improve the quality of solutions found by traditional soft Q-learning on the molecule design of problem of Bengio et al. (2021).

Thesis structure

The subsequent arrangement of this thesis unfolds as follows: The discussion on travel time and path choice model estimation is encapsulated in Article 1. Article 2 talks about the two possible fixes for R-MDPs. The conclusion follows thereafter, and two appendices are included, one for each article.

First Article.

Arc travel time and path choice model estimation subsumed

by

Sobhan Mohammadpour[1], and Emma Frejinger[1]

([1]) Département d'informatique et de recherche opérationnelle, Université de Montréal

This article will be submitted to Transportation Science.

My contributions: I was involved in all aspects of this work. We both worked on the literature review and writing. I proposed the example, the main ideas of the paper (the mixture and theorems), and the proofs. I coded and ran all of the experiments.

Résumé. Nous proposons une méthode pour l'estimation maximum de vraisemblance des paramètres du modèle de choix d'itinéraire *et* du temps de trajet sur chaque arc en utilisant des données de différents niveaux de granularité. Jusqu'à présent, ces deux tâches ont été abordées séparément. Pourtant, elles sont interdépendantes. Au moyen d'un petit exemple, nous illustrons que l'ignorance de cette interdépendance peut conduire à des estimations biaisées des paramètres du modèle de choix d'itinéraire. Les résultats sur les données simulées montrent que notre méthode, contrairement aux méthodes existantes, est capable de récupérer les valeurs des paramètres originale. De plus, les résultats sur les données réelles (ensemble de données des taxis jaunes de New York) montrent que notre méthode est rapide et améliore la qualité des estimations du temps de trajet par rapport à une méthode de référence.
Mots clés : Estimation du temps de trajet, modélisation du choix d'itinéraire, modèles de choix de chemin

Abstract. We propose a method for maximum likelihood estimation of route choice model parameters *and* arc travel time using data of different levels of granularity. Hitherto, these two tasks have been tackled separately. Yet, they are interdependent. By means of a small example, we illustrate that ignoring this interdependence can lead to biased route choice model parameter estimates. Results on simulated data show that our method, contrary to existing methods, is able to recover ground truth parameter values. Furthermore, results on real data (New York yellow cab dataset) show that our method is fast and improves the quality of travel time estimates compared to a benchmark method.
Keywords: Travel time estimation, route choice modeling, path choice models

1. Introduction

An extensive literature is devoted to traffic-related problems with the aim to improve performance (e.g., reduce congestion) through strategic and tactical planning as well as real-time traffic management. Central to many of these problems are, on the one hand, estimates of travel times in the network of interest and, on the other hand, models that can accurately predict traffic flow. In the transportation literature, the corresponding estimation/prediction tasks are referred to as *travel time estimation* and *route choice modeling*. As we discuss in the following, these two tasks are interdependent. However, existing methods address them separately, resulting in what we refer to as a *two-step estimation approach*. It consists of first estimating travel times. Then, treating travel time estimates as given, route choice model parameters are estimated in a second step. From a data point of view, these tasks do not need to be separated as they are both based on traffic data.

This work aims to estimate travel times and route choice model parameters simultaneously. Hence, we relax the strong assumptions required to separate the two tasks in the two-step approach.

In the following, we briefly introduce travel time estimation and route choice modeling (also referred to as path choice modeling) with pointers to relevant studies. We aim to

describe methods at a high level to highlight underlying assumptions as a motivation for our research.

1.1. Travel time estimation

Consider a network of nodes and arcs representing a given congested geographical area. Estimating the travel times in this network is difficult because it depends on numerous factors such as speed limits, traffic flows, road capacities, accidents, construction works, and states of traffic lights. Some factors are easy to observe (e.g., speed limit), whereas others are not (e.g., state of traffic lights and accidents). Even if the factors are observable, the sources of data are diverse. Travel time estimation methods are often tailored to the specificities of available data sources, such as GPS tracking (possibly low frequency) of vehicles (Jenelius and Koutsopoulos, 2013), and automatic vehicle identification equipment (Li and Rose, 2011). Moreover, the methods differ depending on network assumptions. For example, if travel times are assumed static (e.g., Bertsimas et al., 2019) or dynamic (e.g., Wang et al., 2019), or in equilibirum. The estimation can concern arc travel times, or travel times on longer segments, i.e., sequences of arcs (e.g., Jin and Sun, 2008; Jabari et al., 2020).

For strategic and tactical planning purposes (e.g., solving vehicle routing and network design problems), estimating arc travel times instead of travel times on longer road segments is important. Assuming that travelers minimize travel time (i.e., follow a *shortest path*), Bertsimas et al. (2019) propose a methodology to estimate arc travel times by only observing trip travel times between different OD pairs. Importantly, the requirements for data granularity are quite loose. Indeed, they do not need detailed path observations but can use such information if available. Bertsimas et al. (2019) is the closest to our work as we focus on estimating static arc travel times (the most widely used setting for route choice modeling) in large-scale networks under weak data assumptions.

1.2. Route choice modeling

Route choice models are typically parametric distributions defined over a set of paths (for a given OD pair). The distributions depend on the arc travel time matrix and other attributes such as arc lengths and red light locations. Route choice models are widely used to predict traffic flow. They are, therefore, central to various problems where the purpose is to predict flow in response to changes in network attributes, such as travel time or its structure (e.g., network design or facility location problems). Moreover, the interpretation of route choice model parameters, for example, value of time distributions, is used in cost-benefit analyses.

Two data sources are required to estimate parameter values: First, a representation of the network in question, including attributes such as arc travel times. With relatively few

exceptions (e.g., Gao et al., 2010; Ding-Mastera et al., 2019; Mai et al., 2021, who study networks with stochastic travel times), the attributes are assumed to be fixed and known. Second, path choice observations. Most commonly, model parameters are estimated by maximum likelihood using path observations. Studies also investigate the case when path choices are only partially observed (e.g., Mai et al., 2023) or when only flows are observed (Fosgerau et al., 2022).

1.3. Strong assumptions

There is evidence in the literature that *travelers do not follow shortest paths* with respect to some known, deterministic generalized cost function (Jan et al., 2000; Lima et al., 2016). This is the very motivation for stochastic prediction models, such as random utility maximization (discrete choice) models, and there is ample empirical evidence in the related literature on route choice (see Frejinger and Zimmermann, 2021, for a survey), and stochastic traffic equilibrium models (Dial, 1971; Fisk, 1980; Baillon and Cominetti, 2008; Nikolova and Stier-Moses, 2014). The uncertainty stems from the fact that neither the analyst nor the travelers perfectly observe relevant network attributes, such as travel time (Di and Liu, 2016). In light of this literature, the shortest path assumption in Bertsimas et al. (2019) may be viewed as strong.

de Moraes Ramos et al. (2020) study the impact of inaccurate travel time estimates. They provide empirical evidence suggesting that errors in the first step can propagate to the second step hence impacting both the interpretation of parameters and the prediction accuracy.

Route choice modeling and travel time estimation are two interdependent tasks as they model the same process. Specifically, we can use information about the route choice model to improve the travel time estimation, and use travel time information to improve route choice estimation.

Exploiting this interdepency can lead to an improved estimator and enable estimation from otherwise unsuable data. Bertsimas et al. (2019) provide one such example where including the route choice model in the travel time estimation allows them to relax the need for observing the trajectories. In this work we go one step futhere and estimate the route choice model whitout observing the paths.

Contributions. We provide the following contributions:

— We propose a methodology to simultaneously estimate arc travel times and parameters of differentiable route choice models. We show that under weak assumptions, we can conveniently formulate a maximum likelihood estimator for the seemingly more complex joint estimation problem. Furthermore, we show that the solution to the model in Bertsimas et al. (2019) corresponds to a maximum likelihood estimate under

certain assumptions. An attractive property of our proposed joint likelihood is that it allows to mix observations of different levels of granularity naturally.

— We show through an illustrative example that the two-step approach, treating travel time as fixed and given in the route choice model estimation step, can lead to biased parameter estimates.

— Numerical results based on New York City taxi data show that our method is fast and performant. First, focusing on the travel time estimation task only, measured by mean-squared log error, we reach a performance and computing times comparable to those of Bertsimas et al. (2019) while solving the joint estimation problem. Focusing on both tasks, results on synthetic data illustrate that our travel time estimates better reproduce ground truth values compared to those obtained using the two-step approach.

1.4. Paper structure

The remainder of the paper is structured as follows. In Section 2, we describe the work of Bertsimas et al. (2019) and the arc-based route choice model of Fosgerau et al. (2013) that we use in our experiments. In Section 3 we provide an illustrative example, and in Section 4, we introduce our method. We report numerical experiments in Section 5, and finally, we provide concluding remarks in Section 6.

2. Related work

Our work is situated at the intersection between the literature on arc travel time estimation and route choice model estimation. A comprehensive literature review on these two topics is out of the scope of this paper. Instead, we focus this section on the most closely related works and refer to Mori et al. (2015), Oh et al. (2015), and Shaygan et al. (2022) for broader reviews on travel time estimation, and to Prato (2009), Frejinger and Zimmermann (2021), and Zimmermann and Frejinger (2020) for reviews of route choice models. Specifically, we outline the method of Bertsimas et al. (2019) in Section 2.1, and we discuss route choice models in Section 2.2.

2.1. A method for arc travel time estimation

Given a graph $G = (N, A)$ where N is a set of nodes and A a set of arcs, let O be a multi-set (i.e., a set with the possibility of duplicate elements) of observations $(o, d, t) \in N^2 \times \mathbb{R}_+$ from trajectories in G such that o is the origin, d is the destination, and t is the travel time. Bertsimas et al. (2019) focus on estimating the travel time of $a \in A$ using O.

Based on the assumption that the quality of travel time estimations is perceived on a multiplicative rather than an additive scale, Bertsimas et al. (2019) propose using the

mean-squared log error (MSLE),

$$\text{MSLE}(\hat{t}, t) = (\ln t - \ln \hat{t})^2 = \left(\ln(t/\hat{t})\right)^2, \tag{2.1}$$

as a loss function to compare estimate $\hat{t}$ and the observed t. Furthermore, they propose the convex surrogate,

$$\max\left\{\frac{\hat{t}}{t}, \frac{t}{\hat{t}}\right\}, \tag{2.2}$$

for $\text{MSLE}(\hat{t}, t)$, which they model using a second-order cone and linear constraints.

Bertsimas et al. (2019) formulate the estimation problem as a mixed-integer program whose objective is to minimize MSLE between travel times of shortest paths (i.e., predicted travel times) and observed travel times. We refer to this mixed-integer program as BDJM *model* (from the first letter of each contributing author's name). The objective also has a regularization term described in more detail below. The BDJM model has two sets of variables: Binary variables to select the shortest path between each OD pair and a set of continuous variables corresponding to arc and path travel times. Linear (including big-M) constraints ensure that, for each OD pair, the travel time of the shortest path is the sum of its arc travel times and is shorter than all other paths.

Since the BDJM model is hard to solve, Bertsimas et al. (2019) first relax some of the constraints related to the shortest paths and use a decomposition scheme where they iteratively solve the program on the real and binary variables separately until convergence. We call this the BDJM *method*.

Even after replacing the objective function with a convex surrogate, the BDJM model remains hard to solve due to a large number of binary variables. The authors, therefore, propose an algorithm based on iterative path generation, which results in solving a sequence of second-order cone problems.

The estimation problem is undetermined by nature, hence regularization may play an important role. Bertsimas et al. (2019) define the regularization on the sequence of arcs (i, j) and (j, k) as

$$\left|\frac{\boldsymbol{T}_{ij}}{\boldsymbol{L}_{ij}} - \frac{\boldsymbol{T}_{jk}}{\boldsymbol{L}_{jk}}\right| \frac{2}{\boldsymbol{L}_{ij} + \boldsymbol{L}_{jk}}, \tag{2.3}$$

where $\boldsymbol{T}_{ij}$ and $\boldsymbol{L}_{ij}$ are, respectively, the travel time and the length of the arc $(i, j) \in A$. The regularization for the whole problem is the sum of the regularization for consecutive arcs. The intuition is that the speed of travel time does not change dramatically between adjacent arcs. The full model and solution method are presented in Appendix A.1.

Bertsimas et al. (2019) evaluate their method on New York City's yellow cab dataset (New York City Taxi and Limousine Commission, 2016) limited to Manhattan and obtain high-quality results. Solving the problem is challenging because the network and the corresponding

data set are large. We use their method and the yellow cab dataset as our benchmark for the travel time estimation task.

2.2. Route choice models

Given a set of paths $R(o,d)$ connecting an origin-destination pair (o,d), we consider a distribution on $R(o,d)$ a route choice model $\mathbb{P}(r|o,d)$. There is extensive literature dealing with how to define such route choice models and how to estimate related parameters based on observed path choices. In this context, key challenges pertain to the large size (infinite if we consider paths with cycles) of $R(o,d)$, and the similarity of paths, for instance, due to physical overlap. Oyama (2017) reports roughly 10^{18} paths in a ten by ten grid which clearly shows that full path enumeration is not possible even for small networks.

We separate existing route choice models into three broad classes. First, models that are based on restricted sets of paths. The sets can be restricted in different ways through what is often referred to as choice set generation. The models in this class can be path-based or arc-based (Dial, 1971). Models in the second class consider path choice as a sequence of arc-choices following a Markov decision process (Fosgerau et al., 2013). These models are based on the full network and hence do not impose any restriction on the path set. Fosgerau et al. (2022) introduce a model and solution method that we consider belongs to a third class. They show that route choice model parameters can be estimated using flow vectors directly.

The classical way of estimating models of the first two classes is maximum likelihood estimation over path observations. The paths can be fully or partially observed (e.g., Mai et al., 2023).

In this paper, we limit our model to route choice models whose parameters are estimated by maximum likelihood. This is not a strong limitation. In Section 4.2 (solution approach) and onward, we restrict ourselves to differentiable route choice models. The models in the second class are differentiable, and so is the model in Fosgerau et al. (2022) in the third class. The latter is motivated by the fact that conic, convex, and quadratic problems are differentiable with respect to their decision variable. Hence the resulting Markovian policy is differentiable (Amos and Kolter, 2017; Agrawal et al., 2019). Models in the first class are differentiable if the path set restriction does not depend on travel time. This limitation is not very restrictive as many choice set generation algorithms can use distance instead of travel time.

We use the recursive logit model in Fosgerau et al. (2013) for the experimental section of this paper. There are two main reasons for this choice. First, the recursive logit model is differentiable and fast to evaluate. Second, this model, or its variants are used in many studies (e.g., Zhang et al., 2021; Cortés et al., 2023; Oyama, 2023; Knies et al., 2022; Gao, 2021;

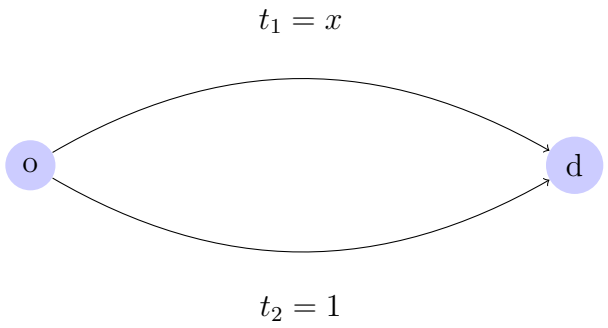


Figure 2.1 – A two-arc network. Arc travel times, t_1 and t_2, are given next to each arc.

Koch and Dugundji, 2020; Iizuka and Hato, 2020; Mai et al., 2015, 2018, 2023; de Freitas et al., 2019).

3. Two arcs, one hard network

The motivation behind our work is the interdependence between the problems of estimating arc travel times and route choice model parameters: Arc travel times are needed to estimate the route choice model, and a route choice model is needed to estimate the arc travel times, especially when full path observations are unavailable. With a small example (see Figure 2.1), we show that even if we start with a reasonable initial solution, the two-step approach using data without path observations (only travel time is observed) converges to a sub-optimal solution.

In the following, we assume that the probability of choosing the upper arc in Figure 2.1 is given by a logit model whose only feature is travel time,

$$p = \mathbb{P}(r = 1) = \frac{\exp(-x)}{\exp(-x) + \exp(-1)}. \tag{3.1}$$

This route choice model is fixed, and the parameter associated with travel time equals -1. The travel time of arc 2 is assumed to be one time unit. We have observed that a user has gone from node o to node d, and it took them two time units, but we do not know which arc they took. Our only unknown is x, whose starting value is one time unit.

We use the MSLE to compare our estimate of the travel time and the observation. We minimize the loss by changing the travel time of arc 1, that is, x in the graph shown in Figure 2.1.

Loss of expected travel time. In this example, minimizing the loss of the expected travel time,

$$\text{MSLE}(\mathbb{E}[t], 2) = \text{MSLE}(px + (1 - p), 2),$$

results in an odd estimate. Since the expected travel time needs to be 2, x has to be 3 for the expected (with the transition probabilities frozen) to be 2. However, if we do not freeze the transition probabilities, the optimum is close to 2.2.

If we repeat this process, using an updated p, x will increase again, leading to eventual divergence.

Expected loss. Minimizing the expected loss

$$\mathbb{E}[l(t,2)] = pl(x,2) + (1-p)l(1,2)$$

is more straightforward. The optimum of x, regardless of p (with frozen probabilities), will be 2. When x equals 2, the loss is approximately 0.35, but the loss at the optimum is closer to 0.32. The issue arises since the derivative of p with respect to x is not zero. The derivative of the loss $l(x,p)$ with respect to x is

$$\nabla_x l(x,p(x)) = \nabla_1 l(x,p(x)) + \nabla_2 l(x,p(x))\nabla_x p(x) \tag{3.2}$$

but the two step process assumes that $\nabla_x p(x)$ is zero; this can lead non-optimal solutions. Repeating this process, with updated p, will not change the value of x.

This illustrative example shows that even given a simple graph, it is possible to diverge or find sub-optimal solutions if arc travel times and route choice model estimations are done in a sequential fashion. Whereas iterative schemes are not used in practice, an important implication of this simple example is that bias in travel time estimates may be absorbed by route choice model parameter estimates. (See de Moraes Ramos et al., 2020, for an empirical investigation of this issue when they compare static and dynamic travel time data.) To address this issue, we introduce our methodology to estimate arc travel times and route choice model parameters simultaneously.

4. Methodology

In this section, we introduce a mixture that models trips in the network. We analyze the properties of the mixture and propose a solution approach. Whereas our methodology differs from both Bertsimas et al. (2019) and the route choice model estimation literature, we reuse the notation pertaining to the problem unless stated otherwise.

4.1. A mixture to model observations at different granularities

We represent every trip in G as a four-tuple

$$(o,d,t,r) \in N^2 \times \mathbb{R}_+ \times R(o,d), \tag{4.1}$$

where, like before, o is the origin, d is the destination, t is a travel time observation, and r is the path taken. We assume that they follow a mixture of distributions:

$$o \sim \mathcal{O}, \tag{4.2a}$$

$$d \sim \mathcal{D}(o), \tag{4.2b}$$

$$r \sim \mathcal{P}(o, d; \boldsymbol{T}, \boldsymbol{b}), \tag{4.2c}$$

$$\hat{t} = \sum_{(i,j)\in r} \boldsymbol{T}_{ij}, \tag{4.2d}$$

$$t \sim \mathcal{N}(\hat{t}). \tag{4.2e}$$

Origin o follows a distribution $\mathcal{O}$ over the set of all nodes, and d follows a distribution $\mathcal{D}(o)$ over the set of all nodes except o. Path r follows a distribution over the set of all paths $R(o, d)$ described by a route choice model $\mathcal{P}(o, d; \boldsymbol{T}, \boldsymbol{b})$ that is parameterized by $\boldsymbol{b}$. To simplify the notation we omit explicitly defining other features as they are assumed to be fixed and given. The travel time of path r is $\hat{t}$. The observed travel time t follows a distribution $\mathcal{N}$ parameterized by $\hat{t}$ with positive support, ideally $\mathbb{R}_+$ as some observations may end up having zero likelihood. It models the difference in travel time due to factors such as fluctuating arc travel time, non-optimal habits, and exceeding the speed limit. This mixture builds on the assumption that observed trajectories are independently and identically distributed (i.i.d.). Whereas this is a standard assumption, we note that it typically does not hold as the next trip often originates from a previous destination. The assumption could be relaxed, for example, if we have driver-specific information. In this case, we can make $\boldsymbol{b}$ a function of those features.

Given the mixture (4.2), the likelihood of an observation (o, d, t, r) is

$$f(o, d, r, t) = \mathbb{P}(o)\mathbb{P}(d|o)\mathbb{P}(r|o, d)f(t; \hat{t}), \tag{4.3}$$

where $f(t; \hat{t})$ is likelihood function of t parameterized by $\hat{t}$. If we do not observe any of the elements, we marginalize them. For instance, for an observation (o, d, t) where the path information is missing, the likelihood is

$$f(o, d, t) = \sum_{r\in R(o,d)} \mathbb{P}(o)\mathbb{P}(d|o)\mathbb{P}(r|o, d)f(t; \hat{t}) \tag{4.4}$$

$$= \mathbb{P}(o)\mathbb{P}(d|o)\mathbb{E}_{r\sim\mathcal{P}}\left[f(t; \hat{t})\right], \tag{4.5}$$

where $\mathcal{P}$ is shorthand for $\mathcal{P}(o, d; \boldsymbol{T}, \boldsymbol{b})$. The corresponding log-likelihood is

$$\ln f(o, d, t) = \ln \mathbb{P}(o) + \ln \mathbb{P}(d|o) + \ln \mathbb{E}_{r\sim\mathcal{P}}\left[f(t; \hat{t})\right]. \tag{4.6}$$

Similarly, if we observe the path but not its travel time, we obtain

$$f(o,d,r) = \int_0^\infty f(o,d,r,t)\mathrm{d}t \tag{4.7}$$

$$= \int_0^\infty \mathbb{P}(o)\mathbb{P}(d|o)\mathbb{P}(r|o,d)f(t;\hat{t})\mathrm{d}t \tag{4.8}$$

$$= \mathbb{P}(o)\mathbb{P}(d|o)\mathbb{P}(r|o,d)\int_0^\infty f(t;\hat{t})\mathrm{d}t \tag{4.9}$$

$$= \mathbb{P}(o)\mathbb{P}(d|o)\mathbb{P}(r|o,d). \tag{4.10}$$

The equality holds because probability density functions (pdf) have unit integral. Hence,

$$\ln f(o,d,r) = \ln \mathbb{P}(o) + \ln \mathbb{P}(d|o) + \ln \mathbb{P}(r|o,d). \tag{4.11}$$

If we are only estimating the route choice model and assume that travel times are fixed and given, then maximizing the log-likelihood (4.11) corresponds to maximizing the classical log-likelihood of the route choice models.

Furthermore, we note that it is straightforward to define a distribution over destinations even if we only observe the origin and the travel time. By definition, the probability that the destination of an observation (o,t) is d is

$$\mathbb{P}(d|o,t) = \mathbb{P}(o,d,t)/\mathbb{P}(o,t). \tag{4.12}$$

The term $\mathbb{P}(o,t)$ is a normalization term as it is constant with respect to the destination.

We can also use the mixture to model observations with partial path information. Let r' be the observed part of a partially observed path. The marginalized likelihood is then

$$f(o,d,r',t) = \sum_{r\in R(o,d)} \mathbb{P}(o)\mathbb{P}(d|o)\mathbb{P}(r,r')f(t;\hat{t}) \tag{4.13}$$

$$= \sum_{r\in R(o,d)} \mathbb{P}(o)\mathbb{P}(d|o)\mathbb{P}(r)\mathbb{P}(r'|r)f(t;\hat{t}) \tag{4.14}$$

$$= \mathbb{P}(o)\mathbb{P}(d|o)\sum_{r\in R(o,d)} \mathbb{P}(r)\mathbb{P}(r'|r)f(t;\hat{t}) \tag{4.15}$$

$$= \mathbb{P}(o)\mathbb{P}(d|o)\mathbb{E}_r[\mathbb{P}(r'|r)f(t;\hat{t})]. \tag{4.16}$$

Note that $\mathbb{P}(r'|r)$ is either one or zero, and if possible, samples should be generated in a way to guarantee that no sample gets rejected, i.e., $\mathbb{P}(r'|r)$ should not be zero. In case only the path choices are of interest, the objective is to maximize

$$\ln \mathbb{E}_r[\mathbb{P}(r'|r)]. \tag{4.17}$$

We further discuss partially observed paths in the following subsection. Specifically, we refer to Mai et al. (2023), which deals with partially observed trajectories.

It is possible to combine different log-likelihoods. For example, for an observation (o_1, d_1, t_1) and an observation (o_2, d_2, r_2, t_2), the total log-likelihood is

$$\ln f(o_1, d_1, t_1) + \ln f(o_2, d_2, r_2, t_2). \tag{4.18}$$

In other words, we have a formulation that can mix different types of data consistently. In contrast, using a standard weighting of losses (one loss for each data type) would introduce hyperparameters to control the ratio of the losses. Using other multi-objective optimization schemes yields similar complexities.

As discussed above, mixture (4.2) allows to model observations at various levels of granularity. Next, we discuss other properties and show that the approach in Bertsimas et al. (2019) is a special case.

Inspired by the literature on energy-based models (EBM) (LeCun et al., 2007), we aim to convert a class of loss functions to distributions whose log-likelihood is the original loss function. However, unlike the EBM literature, we do not require the loss function to be an energy function.

Proposition 4.1. *For any function $L : \mathbb{D} \times \Theta \to \mathbb{R}$ and a parameter vector $\theta \in \Theta$, let*

$$Z_L(\theta) = \int_{\mathbb{D}} \exp[-L(x, \theta)] \mathrm{d}x \tag{4.19}$$

be the partition function. Whenever $Z_L(\theta)$ converges,

$$f_L(x; \theta) = \begin{cases} \exp[-L(x, \theta)]/Z_L(\theta) & \text{if } x \in \mathbb{D} \\ 0 & \text{otherwise,} \end{cases} \tag{4.20}$$

is a valid probability density function. Furthermore, if Z_L is a constant independent of θ, the log-likelihood of the distribution corresponding to f_L is the original loss function.

Proof. The density is necessarily positive as the exponent of any real number is positive. The integral of f over all reals equals the integral over $\mathbb{D}$; we can then take out the partition function because it is constant with respect to x and get the partition function divided by itself, which equals one. The log-likelihood is $\log f_L(x; \theta) = -L(x, \theta) - \log Z_L(\theta)$ hence maximizing the log-likelihood is equivalent to minimizing the loss function if Z_L is a constant. □

In our problem, $\hat{t}$ is the parameter of the distribution and x is the observed travel time.

Remark 1. *There are many examples of such functions. The mean squared error,* $\mathrm{MSE}(x, y) = (x - y)^2$*, corresponds to the normal distribution with $\sigma^2 = 0.5$ and the linear exponential function,* $\mathrm{LINEX}(x, y) = \exp(x - y) - (x - y) - 1$*, corresponds to the Gumbel distribution (Abdzaid Atiyah et al., 2020).*

Remark 2. *Whereas from an optimization perspective multiplying by a number does not change the minimum of the problem; it changes the corresponding distribution.*

Unfortunately, the partition function resulting from applying Proposition 4.1 to the MSLE is not constant. However, we create an equivalent asymmetric loss function with a constant partition function:

Proposition 4.2. *For any loss function $L : \mathbb{D} \times \Theta \to \mathbb{R}$, the loss function $\hat{L}(x, \theta) = L(x, \theta) + \log Z_L(\theta)$ has the same minimum as $\hat{L}$ but the partition function of $\hat{L}$, $Z_{\hat{L}}$ is a constant.*

Proof. By definition we have:

$$Z_{\hat{L}}(\theta) = \int \exp[-\hat{L}(x, \theta)]\mathrm{d}x \tag{4.21}$$

$$= \int \exp[-L(x, \theta) - Z_L(\theta)]\mathrm{d}x \tag{4.22}$$

$$= \frac{\int \exp[-L(x, \theta)]\mathrm{d}x}{Z_L(\theta)} \tag{4.23}$$

$$= \frac{\int \exp[-L(x, \theta)]\mathrm{d}x}{\int \exp[-L(x, \theta)]\mathrm{d}x} \tag{4.24}$$

$$= 1. \tag{4.25}$$

Since we optimize with respect to x, not θ, the gradient and extremum do not change. □

We use the transformation of Proposition 4.2 on the MSLE and obtain what we call the small-time biased MSLE (SMSLE),

$$\mathrm{SMSLE}_\gamma(t, \hat{t}) = \gamma \mathrm{MSLE}(t, \hat{t}) + \ln t, \tag{4.26}$$

that we use in our experimental results.

The next proposition links the work of Bertsimas et al. (2019) to maximum likelihood estimation.

Proposition 4.3. *The solution to the BDJM model is the maximum likelihood estimate of (4.2) where the route choice model is the shortest path algorithm and $\mathcal{O}$ and $\mathcal{D}(o)$ are not estimated.*

Proof. When $\mathcal{P}$ is the shortest path, the expectation in (4.6) is equal to $f(t; \hat{t})$ for the shortest path r. If we chose $\mathcal{N}$ to be the log-normal, $\ln f(t; \hat{t})$ is proportional to the MSLE. As such, the BDJM model results in a maximum likelihood estimate. □

Whereas it might seem possible to address the interdependence issue with observations having both path and travel time information, it can lead to a different estimator if the route choice model depends on the observed travel time.

Proposition 4.4. *Ignoring the link between travel time and route choice model when training a maximum likelihood estimator for the mixture can lead to a different estimator.*

PROOF. If the route choice model depends on the travel times, the maxima of (4.7) and (4.3) with regards to $\boldsymbol{T}$ can be different so the estimator can have a different limit point as the number of observations increases. □

Remark 3. *Proposition 4.4 implies that if the route choice model does not depend on travel time, it is possible to adress the interdependency issue by observing paths.*

4.2. Solution approach

To compute maximum likelihood estimates, we maximize the likelihood of the observed data. For this purpose, we assume that the function $\ln \mathbb{P}(r \sim \mathcal{P})$ is differentiable with respect to the travel time and the route choice model's parameters.

Assumption 4. *We assume that* $\nabla_{\boldsymbol{b},\boldsymbol{T}} \ln \mathbb{P}(r)$ *exists.*

We illustrate our method using a recursive logit model (Fosgerau et al., 2013). However, we are not limited to that class of route choice models. Note that, in this section, we focus on observations without full path information because having access to such information (or data with a similar level of granularity) makes the problem easier.

The gradient of the log-likelihood (4.6) is

$$\nabla \ln f(o,d,t) = \nabla \ln \mathbb{P}(o) + \nabla \ln \mathbb{P}(d|o) + \nabla \ln \mathbb{E}_{r\sim\mathcal{P}}\left[f(t;\hat{t})\right]. \tag{4.27}$$

The following proposition introduces estimators of the last term.

Proposition 4.5. *The offline estimator*

$$\mathbb{E}_{r\sim\mathcal{P}'}\left[\nabla f(t;\hat{t}) + f(t;\hat{t})\frac{\nabla \mathbb{P}_{\mathcal{P}}(r)}{\mathbb{P}'_{\mathcal{P}}(r)}\right] \tag{4.28}$$

and online estimator

$$\mathbb{E}_{r\sim\mathcal{P}}\left[\nabla f(t;\hat{t}) + f(t;\hat{t})\nabla \ln \mathbb{P}_{\mathcal{P}}(r)\right], \tag{4.29}$$

estimate $\nabla \ln \mathbb{E}_{r\sim\mathcal{P}}\left[f(t;\hat{t})\right]$ *in (4.27) for any distribution* $\mathcal{P}'$ *whose support is a superset of the support of* $\mathcal{P}$*, i.e. such that* $\mathcal{P} \ll \mathcal{P}'$*.*

PROOF. We rewrite the gradient of the expected likelihood as

$$\nabla \mathbb{E}_{r\sim\mathcal{P}}\left[f(t;\hat{t})\right] \tag{4.30}$$

$$= \nabla \mathbb{E}_{r\sim\mathcal{P}'}\left[f(t;\hat{t})\frac{\mathbb{P}_{\mathcal{P}}(r)}{\mathbb{P}_{\mathcal{P}'}(r)}\right] \tag{4.31}$$

$$= \mathbb{E}_{r\sim\mathcal{P}'}\left[\nabla f(t;\hat{t})\frac{\mathbb{P}_{\mathcal{P}}(r)}{\mathbb{P}_{\mathcal{P}'}(r)} + f(t;\hat{t})\frac{\nabla\mathbb{P}_{\mathcal{P}}(r)}{\mathbb{P}_{\mathcal{P}'}(r)}\right] \tag{4.32}$$

$$= \mathbb{E}_{r\sim\mathcal{P}}\left[\left(\nabla f(t;\hat{t})\frac{\mathbb{P}_{\mathcal{P}}(r)}{\mathbb{P}_{\mathcal{P}'}(r)} + f(t;\hat{t})\frac{\nabla\mathbb{P}_{\mathcal{P}}(r)}{\mathbb{P}_{\mathcal{P}'}(r)}\right)\frac{\mathbb{P}_{\mathcal{P}'}(r)}{\mathbb{P}_{\mathcal{P}}(r)}\right] \tag{4.33}$$

$$= \mathbb{E}_{r\sim\mathcal{P}}\left[\nabla f(t;\hat{t}) + f(t;\hat{t})\frac{\nabla\mathbb{P}_{\mathcal{P}}(r)}{\mathbb{P}_{\mathcal{P}}(r)}\right] \tag{4.34}$$

$$= \mathbb{E}_{r\sim\mathcal{P}}\left[\nabla f(t;\hat{t}) + f(t;\hat{t})\nabla \ln \mathbb{P}_{\mathcal{P}}(r)\right]. \tag{4.35}$$

We revert back to the original measure in (4.32). We call (4.32) the offline and (4.35) the online gradient estimator. □

Remark 5. *This type of gradient of expectation is called the REINFORCE (Williams, 1992) or score function estimator (L'Ecuyer, 1990; Mohamed et al., 2020). Furthermore, It is possible to use the same estimator for higher order gradient as shown by Mohamed et al. (2020).*

Next, we turn our attention to partially observed paths. If we apply the online estimator to (4.17), we obtain

$$\nabla \log \mathbb{E}_r[\mathbb{P}(r'|r)] = \frac{\mathbb{E}_r[\mathbb{P}(r'|r)\nabla \ln P(r)]}{\mathbb{E}_r[\mathbb{P}(r'|r)]}, \tag{4.36}$$

since $\mathbb{P}(r'|r)$ is independent of the choice model. Furthermore, $\mathbb{E}_r[\mathbb{P}(r'|r)]$ is equal to $\mathbb{P}(r')$ thus (4.36) is equal to

$$\frac{\mathbb{E}_r[\mathbb{P}(r'|r)\nabla \ln \mathbb{P}(r)]}{P(r')} = \frac{\sum_{r\in R(o,d)}\mathbb{P}(r)\mathbb{P}(r'|r)\nabla \ln \mathbb{P}(r)}{P(r')} = \sum_{r\in R(o,d|r')}\mathbb{P}(r|r')\nabla \ln \mathbb{P}(r), \tag{4.37}$$

due to the Bayes' rule with $R(o,d|r')$ being the set of all paths from o to d that contain r'. As a result, (4.36) is equal to

$$\mathbb{E}_{r|r'}[\nabla \ln \mathbb{P}(r)], \tag{4.38}$$

which is the gradient introduced in Mai et al. (2023).

In the remainder of this section, we describe several important aspects of our solution approach.

Inference. Inference depends on the performance metric. If the performance metric is likelihood, then we maximize the likelihood (i.e., find the mode). If the performance metric is the MSLE, we maximize the expected MSLE between the prediction and samples from our

mixture. Indeed, the empirical loss converges to the actual expected MSLE as the number of observations tends to infinity due to the uniform law of large numbers.

To simplify the inference process, we replace the expectation over paths with the empirical expectation over samples. To calculate the mode, we use an adaptive grid search between the ordered observation by bounding the size of the knots of the grid to be at least as far as a threshold and have an upper bound on the number of knots. Bertsimas et al. (2019) showed that for any distribution, the geometrical mean minimizes the MSLE. To minimize the MSLE on any path, we use the law of total expectation to calculate the geometrical average as

$$\exp\left(\mathbb{E}_{r\sim\mathcal{P},t\sim\mathcal{N}(\hat{t})}[\ln t]\right) = \exp\left(\mathbb{E}_{r\sim\mathcal{P}}\left[\mathbb{E}_{t\sim\mathcal{N}(\hat{t})}[\ln t|r]\right]\right). \tag{4.39}$$

Domain of $\boldsymbol{T}$. We restrict the travel time estimation to a reasonable domain. For instance, the restriction can be as simple as limiting the arc travel times by bounding the speed to be between the speed limit and a lower bound. Furthermore, the domain can encode restrictions from different types of observations, such as sensors that measure the average time to traverse a specific road segment.

We use projected gradient-based methods to maximize the log-likelihood. Projection on the rectangular (per arc) domain is inexpensive.

Mathematical model. Below, we define the mathematical program, ll is a log-likelihood obtained by marginalizing the unobserved in data. We assume that if the path choice model is infeasible, the log-likelihood is $-\infty$. While the path choice model may have constraints of its own, we solve the path choice model at each step, removing the need for including path choice model constraints.

$$\begin{aligned} \underset{\boldsymbol{T},\boldsymbol{b}}{\text{maximize}} \quad & ll(\boldsymbol{T},\boldsymbol{b};\text{data}) \\ \text{subject to} \quad & \boldsymbol{T} \text{ is valid.} \end{aligned} \tag{4.40}$$

Identifiability. The model may not be identifiable. For instance, when maximizing (4.7), i.e., travel time estimation without travel time observation, the model is undetermined if we use the MNL model. Indeed, we can multiply the utility parameter associated with travel time by a constant and divide all travel times by the same constant without changing the likelihood. Colinearities can generally create a large class of solutions with a very close likelihood. The bounds on $\boldsymbol{T}$ help to mitigate this issue.

Regularization. We propose the following differentiable regularization instead of the one proposed by Bertsimas et al. (2019) as theirs is not differentiable if added to the objective. Our regularization is the summation of

$$\text{MSLE}(\boldsymbol{T}_{ij}/\boldsymbol{L}_{ij}, \boldsymbol{T}_{jk}/\boldsymbol{L}_{jk})/(\boldsymbol{L}_{ij}+\boldsymbol{L}_{jk}) \tag{4.41}$$

over all 3-tuple (i,j,k) such that (i,j) and (j,k) are arcs.

Optimization. A source of complexity is the stochastic gradient when estimating (4.27). The stochastic objective makes line search and classical methods like L-BFGS-B (Zhu et al.,

1997) perform poorly. This is unsurprising as the stochastic nature of the problem defies the basic assumptions of smooth optimization. Instead, we use the Adam optimizer (Kingma and Ba, 2014) with a projection of gradients before and a projection of variables after each update. Mini-batches can be used to improve performance, speed up convergence, and for effective training over large datasets.

As the estimated gradient is noisy, we use a primal stopping criterion instead of using a first-order stopping criterion. We stop optimizing when we cannot improve the loss by a fixed threshold over the course of a fixed number of iterations.

Variance estimation and lack thereof. It is customary to analyze the estimator's variance. However, the variance estimators (DasGupta, 2008, p. 243) or "Hubber's sandwich estimator" (Freedman, 2006) do not have the standard interpretations for a few reasons. First, the objective is stochastic, the problem is not convex, and we cannot guarantee convergence to a global optimum. Second, not all variables are free in the constrained problem. Third, the gradient we estimate is, in many cases, not exact. Last but not least, we have a large number of parameters. As such, the standard methods do not apply, and estimating the variance is beyond the scope of this work.

Complexities. The methodology we describe in this section is based on relatively simple ideas. However, since it builds on top of route choice models, any complexity inherent to such models persists. The stochasticity of the evaluation and non-convexity are other sources of complexity. By choosing a differentiable route choice model, we avoid iterative schemes like the BDJM method (the solution to the BDJM model), and we do not need to resort to search-based methods as we optimize a differentiable problem.

Writing a tractable implementation requires some care. The BDJM method transfers most of the burden of a fast implementation to a solver, as calculating shortest paths is fast. We are faced with a few implementation challenges but with enough parallelism, efficient gradient calculation like Mai et al. (2018), and efficient operations by means of sparse matrices, we are able to have a fast implementation, as we show in the following section.

5. Experimental results

In this section, we present two sets of results. First, in Section 5.2, we test our method on Manhattan using the Yellow Cab dataset (NYC). The objectives are to assess the performance on the travel time estimation problem and compare it to the BDJM method. We also analyze the parameter estimates of the resulting route choice model. Since the Yellow Cab dataset does not contain path observations, we cannot use it to compare our method to the two step approach. Indeed, the second-step – route choice model estimation – is not possible using existing methods. In the second set of results reported in Section 5.3, we use simulated data so that we have access to ground-truth route choice model parameter values. We obsevations

with/without path observations and compare our method to the two-step approach. Next, we describe the experimental setup.

5.1. Experimental setup

We split the data into three datasets: training, validation, and testing. We report the square root of MSLE (RMSLE) and use NOMAD (Audet et al., 2021) to search the hyperparameter space by maximizing the validation loss as we found it works better than Bertsimas et al. (2019)'s recommended hyperparameter.

For the BDJM method, we use the MOSEK solver (MOSEK ApS, 2022) with their proposed termination criterion and a maximum of 30 iterations. We use MOSEK as we found it to be faster than Gurobi, CPLEX, and SCS.

Our method uses the primal stopping criterion to terminate after 50 iterations with less than 0.01 total improvement in objective. We employ the online (4.29) estimator as it constantly outperforms the offline estimator. We use 35 samples for NYC and 100 for the synthetic set to estimate $\mathbb{E}_r[f(t;\hat{t}]$.

There are five features in our route choice model (recursive logit proposed by Fosgerau et al., 2013). The first two relate to arc travel time in minutes: We partition the travel time matrix into two disjoint matrices such that the sum of those two matrices equals the original one, and the elementwise multiplication produces a zero matrix. The travel times for secondary streets are stored in the second matrix and the rest in the first. The third feature indicates red lights, stops, or intersections. Lastly, we have one indicator for left turns and one for U-turns. We can add left and u-turns by extending the current state with the knowledge of the previous state, i.e., arcs become nodes in a projected graph. We design the features such that discretization does not affect path probabilities. We fix the value of the $\beta_{\text{u-turn}}$ to -5, similar to Fosgerau et al. (2013). The arc utilities are linear in parameters.

The methods are coded in Julia (Bezanson et al., 2017). More details about data processing are given in Appendix A.2.

5.2. Yellow cab dataset

We report performance metrics – the number of iterations, computing time in minutes, and RMSLE – of our and the BDJM methods in Table 2.1 for different sizes of the training set. The first set of rows (2-6) is limited to the 6 am to 9 pm period on weekdays, whereas the second set of rows (7-11) is for the 9 am to 12 am period. The iteration budget (It.) is the maximum training time for one iteration of the hyperparameter optimization loop. We evaluate each model at most 50 times in the hyperparameter search loop. The training time columns (t.) reports the average training times in minutes for the best hyperparameters.

We run the experiments three times [1]. We note that the BDJM method and MOSEK are deterministic [2], but our method and NOMAD depend on the seed. Due to negligible variance (less than 5e-4 in most cases), we run the BDJM method only once. The BDJM method timeout in the larger set. However, the loss change in these cases is in the order of 1e-4 in the last two or three iterations, and training for longer periods did not improve the results.

Table 2.1 – Comparison of our and BDJM methods on the travel time estimation task on Manhattan on March 2016. It. and t. are the maximum time for each hyperparameter optimization step in minutes and the average optimization time for the best iteration. For each experiment, we do 50 hyperparameter optimization steps. The superscripts show how many of the experiments did not converge. Zeros are omitted.

size training set	It.	RMSLE ours	t. [min]	RMSLE ours with reg.	t. [min]	RMSLE BDJM	t. [min]
100	4	0.4343 ± 0.0039	$4^{2/3}$	0.4051 ± 0.0047	$3^{1/3}$	0.3705	1
1,000	8	0.3918 ± 0.0038	$9^{3/3}$	0.3891 ± 0.0075	$6^{1/3}$	0.3557	2
10,000	16	0.3472 ± 0.0017	7	0.3485 ± 0.0009	4	0.3571	$18^{1/1}$
100,000	32	0.3291 ± 0.0005	$30^{1/3}$	0.3289 ± 0.0004	$31^{1/3}$	0.3310	$41^{1/1}$
200,000	64	0.3264 ± 0.0001	$51^{1/3}$	0.3267 ± 0.0002	$52^{1/3}$	0.3285	$80^{1/1}$
100	4	0.3972 ± 0.0129	$5^{3/3}$	0.3900 ± 0.0114	$4^{2/3}$	0.3532	1
1,000	8	0.3570 ± 0.0015	$9^{3/3}$	0.3358 ± 0.0047	6	0.3306	2
10,000	16	0.3120 ± 0.0002	$16^{2/3}$	0.3113 ± 0.0011	$14^{1/3}$	0.3092	15
100,000	32	0.2936 ± 0.0001	30	0.2940 ± 0.0001	30^{1}	0.2983	$41^{1/1}$
200,000	64	0.2917 ± 0.0002	$57^{1/3}$	0.2916 ± 0.0001	54	0.2949	$83^{1/1}$

In terms of training time, our method scales well with the size of the training set, taking around 30 minutes to train with 100,000 data, roughly the same as the BDJM method. Our method has poorer performance on the smaller datasets (100 and 1,000 observations). We note that regularization plays an important role in data sparse settings and helps reduce the variance of the estimated $\boldsymbol{T}$ among experiments. Furthermore, regularization seems to help with overfitting in smaller datasets. For the larger sets (100,000 or 200,000), regularization, however, deteriorates performance (RMSLE), and NOMAD sets the regularization very low or to zero. We also note that even though our method is stochastic, the variance (in RMSLE) is relatively low.

Next, we focus on route choice model parameter estimates resulting from our method. For the sake of illustration, we report those obtained based on 200,000 observations and the later morning data in Table 2.2. We note that there is some variation between the three

1. We acknowledge that running the experiments only three times is not enough to get a statistically significant variance and mean. However, we note that there is little variation between the runs.
2. MOSEK is only deterministic when the number of cores does not change.

different runs, but the parameter ratios remain relatively stable. Noteworthy is the difference between residential and non-residential roads, where the taxis prefer the latter.

Table 2.2 – $\boldsymbol{b}$ estimated on NYC using 200,000 observations from the late morning dataset.

Attribute	$\hat{\boldsymbol{b}}_{\text{non-residential}}$	$\hat{\boldsymbol{b}}_{\text{residential}}$	$\hat{\boldsymbol{b}}_{\text{intersection}}$	$\hat{\boldsymbol{b}}_{\text{left turn}}$	$\boldsymbol{b}_{\text{u-turn}}$ (fixed)
Runs 1	-3.27	-3.88	-0.36	-0.43	-5
Runs 2	-3.50	-4.50	-0.49	-0.54	-5
Runs 3	-3.39	-4.38	-0.32	-0.22	-5

Next we analyze why we do not observe a larger improvement in RMSLE for our method compared to BDJM. Figure 2.2 compares the travel time of paths sampled from the route choice model and the shortest path. In fact, 93.3% of sampled paths are less than 10% longer than the shortest path. In other words, the estimated route choice model confirms that the shortest path is a good proxy for taxi data. Still, the parameter estimates indicate that taxi drivers penalize travel time on secondary (residential) roads more severely than travel time on primary (non-residential) roads.

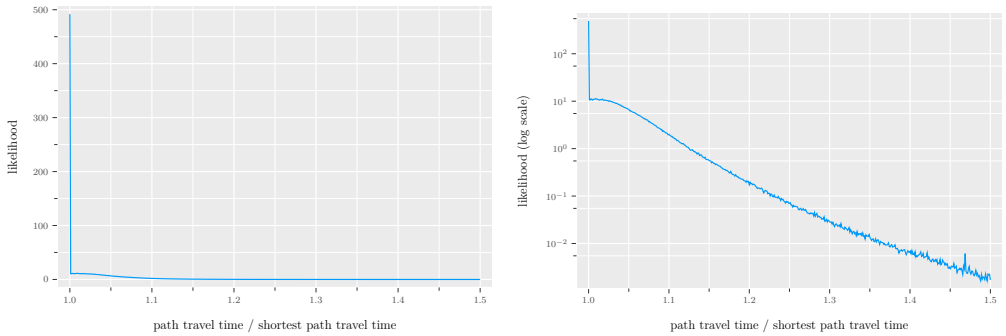
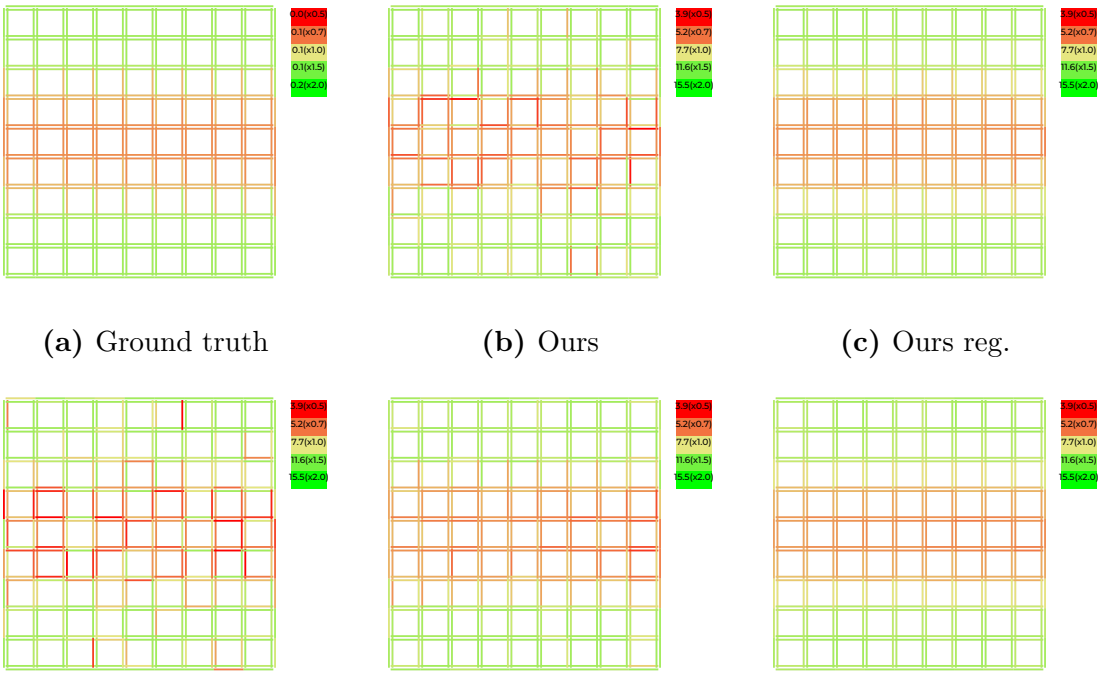


Figure 2.2 – Comparison of path travel time and shortest path, the x-axis shows the ratio while the y-axis shows the likelihood. The right figure is the left figure but with a log-scale axis.

We end this section with a few remarks. Our experiments showed little sensitivity to the initialization. The effects of regularization depend on the data-set size; the more data, the less regularization is required. As expected, low learning rates lead to more steps in the optimization loops (some results are shown in Appendix A.3). Note, however, that choosing a too large learning rate may lead to divergence. As initialization, we propose the parameters $\boldsymbol{b}_0 = -2$, $\boldsymbol{T}_0 = \boldsymbol{T}_{\min}/0.9$, $\log \eta = -2$ and $\lambda = 0$. Another viable alternative is to initialize using the solution of the two-step method and then use our method to improve the estimation results.



(a) Ground truth **(b)** Ours **(c)** Ours reg.

(d) BDJM's method **(e)** Ours path **(f)** Ours path and reg.

Figure 2.3 – Comparison of estimated arc travel time matrix, same color scale for all figure. reg. stands for regularized.

5.3. Synthetic data

In this section, we analyze results on data that has been simulated on a grid network with a route choice model we posit (see Appendix A.2 for details). Since we have access to observations at different levels of granularity, we can choose to use path observations or only observed OD times (as in the Yellow Cab data). This results in different versions of our method: with/without regularization (reg.) and with/without path observations.

We start with a qualitative analysis. Figure 2.3 displays a visualization of $\boldsymbol{T}$ for the methods; the ground truth is shown in Figure 2.3a. The RMSLE between the estimated $\hat{\boldsymbol{T}}$ and the ground truth is below 0.14 for our model, below 0.08 for our model when using path observations, and 0.22 for the BDJM method. However, based on visual inspection, our estimates of $\boldsymbol{T}$ are closer to the ground truth despite all methods having a similar RMSLE (ours is 0.31 while BDJM is 0.33). This highlights one of the weaknesses of measuring model performance with RMSLE. One alternative would be to compare predicted travel time for known paths, but it is not always possible.

Next, we turn our attention to an analysis of the parameter estimates, comparing them to the ground truth values. Here the main objective is to compare our method's estimates to ground truth values and those obtained using the two-step method, i.e., estimating a route

choice model assuming travel time is fixed and given. For the two-step method, we use travel times calculated assuming 90% of free-flow speed or the BDJM method in the first step.

The results are reported in Table 2.3. Note that we compare parameter ratios. The two-step methods cannot retrieve the ground truth parameter ratios. Given that recursive logit has been used to generate the data, the path choice assumption in the first step is misspecified for the free-flow method and BDJM's method. The bias in the travel times is absorbed by the travel time parameter estimate. On the contrary, our method achieves parameter estimate ratios close to the ground truth ratio. While $\boldsymbol{b}$ is well estimated in all cases, the methods using observed paths have a more accurate estimate of $\boldsymbol{T}$. Finally, we note that, unsurprisingly, having path observations makes the problem deterministic and easier to solve.

Table 2.3 – Comparison of log-likelihood and parameter estimates. Simulated data. $\boldsymbol{b}_{\text{u-turn}}$ is fixed to -5.

Model name	log-likelihood	$\hat{\boldsymbol{b}}_{\text{travel time}}/\hat{\boldsymbol{b}}_{\text{left turn}}$
Ground truth	-1.65	1.00
Ours	-1.83	1.01
Ours with reg.	-1.70	1.07
Ours with path	-1.67	0.96
Ours with path and reg.	-1.66	1.01
Two-step (1st step: 90% of free-flow speed)	-2.14	0.78
Two-step (1st step: BDJM method)	-1.88	0.72

6. Conclusion

Our proposed mixture effectively models traffic, ensuring compatibility with a broad array of route choice models and loss functions. It is designed for simultaneous estimation of arc travel time and route choice model parameters. Our approach underscores the advantage of marginalizing unobserved variables and utilizing stochastic gradient estimates, leading to a maximum likelihood estimation, even when observations occur at different levels of granularity.

Notably, we demonstrated that various data types can be amalgamated when computing the maximum likelihood estimate without necessitating a linear combination of losses as an objective. Speed is another advantage of optimizing this mixture. Moreover, our work illuminates the potential of estimating recursive models such as those suggested by Fosgerau et al. (2013) and Bertsimas et al. (2019), leading to maximum likelihood estimates of this mixture.

We illustrated with a small example as well as using results on simulated data that the conventional two-step approach for estimating route choice models may result in biased or suboptimal outcomes. This underscores the value of our methodology.

Despite these promising outcomes, our method requires further exploration across additional route choice models, datasets, and differen travel time distributions. As articulated in Section 4, the sole precondition is the differentiability of the route choice model. Yet, subsequent experiments are crucial to verify the practical efficacy of our methodology on this extensive range of route choice models.

Our choice of Manhattan as a test site was intended to align the experimental section with that of Bertsimas et al. (2019). Still, we acknowledge the importance of assessing the model's performance in diverse locations, particularly those deviating from Manhattan's grid-like structure. Moreover, testing the method on a mix of observations at different levels of granularity remains an area for future exploration. Lastly, we note that further investigation is needed for estimating time-dependent travel times.

Acknowledgments

We thank Adel Mohammadpour for constructive discussions about converting loss functions to distributions. We are grateful to Sebastien Martin for providing the code of Bertsimas et al. (2019) and to Eric Larsen for helping us with the exposition of the paper. Geographical data are copyrighted to OpenStreetMap contributors and available from OpenStreetMap contributors (2022). We ran most of the experiments on Calcul Quebec and the Digital Research Alliance of Canada's clusters.

Second Article.

Bias in regularized MPDs

by

Sobhan Mohammadpour[1], Emma Frejinger[2], and Pierre-Luc Bacon[3]

([1]) Université de Montréal
Mila, Institut québécois d'intelligence artificielle

([2]) Université de Montréal

([3]) Université de Montréal
Mila, Institut québécois d'intelligence artificielle

This article was submitted in part to ICLR 2024.

My contributions: I was involved in all aspects of this work. We all worked on the literature review and writing. I proposed the examples, the main ideas of the paper (the two fixes and theorems), and the proofs. I coded and ran all experiments. The molecule design experiments are based on code by Recursion Pharmaceuticals.

Résumé. L'apprentissage par renforcement régularisé par l'entropie est devenu un cadre proéminent dans l'apprentissage par renforcement inverse et les problèmes de contrôle optimal. Cependant, malgré sa popularité, une source importante de biais dans les estimations de valeur dans le cadre épisodique a été négligée. Ce biais entraîne une estimation de la valeur gonflée, ce qui contredit le principe de pessimisme jugé essentiel pour l'entraînement stable des agents d'apprentissage par renforcement profond. Pour aborder ce problème, nous proposons un facteur de correction simple mais efficace. De plus, nous démontrons que l'opérateur mellowmax incorpore intrinsèquement ce mécanisme de correction, évitant ainsi le problème de surestimation. Nous présentons également un modèle général pour créer des variantes douces dans le cadre plus large des processus de décision de Markov régularisés. Lorsqu'il est appliqué à la conception de séquences biologiques, notre approche est comparable à GFlowNets.
Mots clés : Apprentissage par renforcement à entropie maximale, processus de décision de Markov régularisé

Abstract. Entropy-regularized reinforcement learning has become a prominent framework in inverse reinforcement learning and optimal control problems. However, despite its popularity, an important source of bias in value estimates within the episodic setting has been overlooked. This bias results in inflated value estimation, which contradicts the principle of pessimism deemed essential for stable training of deep reinforcement learning agents. To tackle this issue, we propose a simple yet effective correction factor. Additionally, we demonstrate that the mellowmax operator inherently incorporates this correction mechanism, thus avoiding the overestimation problem. We also present a general template for creating mellow-variants within the broader framework of regularized Markov decision processes. When applied to biological sequence design, our approach is comparable to GFlowNets.
Keywords: Maximum entropy reinforcement learning, regularized Markov decision process

1. Introduction

Entropy-regularized reinforcement learning (RL) has gained prominence as a widely-used framework for inverse reinforcement learning (Ziebart et al., 2008) and control (Haarnoja et al., 2017, 2018). However, the added entropy has compounding effects. Environments with no reward can end up having large values and policies imitating a spinning top end up having a higher value than one that plays the game. The crux of the issue lies in the fact that the regularized reward is higher than the actual reward, biasing the agents towards maximizing the regularization instead of true reward that they are supposed to maximize.

In this paper, we explore this bias introduced by regularized RL (Geist et al., 2019), focusing on stochastic Markov decision processes (S-MDP) (Mai and Jaillet, 2020). We assert that the S-MDP perspective is not only equivalent to the prevalent framework but also provides a balance between expressiveness and structure for effective reasoning. Central to our argument is the notion that the maximum value of the regularizer should be zero at its maximum.

For every regularizer that has a non-zero maximum, we build new regularizers by subtracting its maximum. While it may seem that most environments possess a fixed number of actions, and thus a constant shift in the rewards would not yield significant alterations in policy, we empirically show that the fixed number of actions assumption does not hold in practice. We conducted a survey of 12 SAC implementations on GitHub, presented in Appendix B.4 and discovered that none of them adhered to the fixed action space assumption.

Building on the principle the regularizer should be zero at its maximum, we not only rediscover the mellowmax operator (Asadi and Littman, 2017) but also propose a versatile method for devising "mellow" variants of any operator. We demonstrate that such mellow operators exhibit desirable properties; specifically, for a certain class of undiscounted Markov Decision Processes (MDPs), they always yield a unique solution, in contrast to classical regularized operators, which can diverge.

To conclude the paper, we evaluate our enhanced version of soft Q-learning (SQL) (Haarnoja et al., 2017) on problem instances where the standard SQL has been reported to underperform. Although our primary focus remains on entropy-regularized RL, the approach we present exhibits a wide-ranging applicability to other regularized MDPs.

2. Preliminaries

A discounted Markov Decision Process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, \mathbb{A}, R, P, \gamma)$. Here, $\mathcal{S}$ represents the set of states, and $\mathcal{A}$ is the collection of all possible actions while $\mathbb{A}(s)$ represents the set of valid actions at stat s. The reward function, denoted by $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, or r_a if the state is obvious, maps state-action pairs to real numbers. The transition function, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, determines the probability of transitioning to the next state, where $\Delta(\mathcal{S})$ indicates the probability simplex over the set of states $\mathcal{S}$. Additionally, the discount factor, represented by $\gamma \in [0, 1]$, is included in our problem formulation.

When solving a Markov Decision problem under the infinite horizon discounted setting, the aim is to find a policy $\pi(s) : \mathcal{S} \to \Delta(\mathbb{A}(s))$ that maximizes the expected discounted return $V_\pi(s) \triangleq \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma R(S_t, A_t), |, S_0 = s\right]$ for all states. A fundamental result in dynamic programming states that the value function $V_{\pi^\star}$ for any optimal policy $\pi^\star$ must satisfy the Bellman equations (Bellman, 1954):

$$V^\star(s) = \max_{a \in \mathcal{A}(s)} Q^\star(s, a) \qquad \text{where} \qquad Q^\star(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) V^\star(s') \ .$$

In the infinite horizon setting, it is possible to demonstrate that considering only the space of stationary deterministic policies is sufficient when searching for optimal policies. This is not the case for the finite horizon setting, where the objective is to find a policy that maximizes the expected return within a fixed time horizon T: $V_\pi^t(s) \triangleq \mathbb{E}\left[\sum_{k=0}^{T-t} \gamma^k R(S_{t+k}, A_{t+k}) \Big|, S_t = s\right]$ for every state and time step t. In general, the finite horizon setting necessitates exploring the

broader space of nonstationary deterministic policies, or extending the state with t, making the learning problem more complex. Therefore, the majority of RL algorithms are formulated under the infinite horizon setting due to this important simplification.

However, adopting this perspective does not preclude addressing termination problem when modeling environments that have natural terminations. It is possible to tackle such problems by interpreting the termination event through the construction of an equivalent MDP with an appropriate absorbing state. We call a state s absorbing if s has only one action a such that $P(s|s,a) = 1$ and $R(s,a) = 0$. In this way, the infinite horizon setting can be used to study optimal policies in problems that have natural termination points, effectively bridging the gap between the infinite and finite horizon settings.

2.1. From Discrete Choice Models to Entropy-Regularized Reinforcement Learning

Representative Agent Models (RAM) are a class of models employed in various fields such as economics, marketing, and transportation, to describe and predict the choices made by individuals or agents in decision-making processes. The welfare or social surplus function (McFadden, 1977; Feng et al., 2018) is defined as

$$w^{\sigma}(r) = \max_{\pi \in \Delta} \sum_{a} r_a \pi(a) - \sigma w^*(\pi),$$

for some penalty function w^*, and a strictly positive temperature σ. The welfare function w^{σ} is the convex conjugate of σw^*. This can be seen as the value function of a regularized MDP with $\gamma = 0$. The choice probability

$$\pi^{\star} = \arg\max_{\pi \in \Delta} \sum_{a} r_a \pi(a) - \sigma w^* \pi,$$

is equal to the gradient of $w^{\sigma}(r)$ (Feng et al., 2018). Entropic regularization (when $\gamma \in (0,1)$) has been independently rediscovered by Dai et al. (2018); Geist et al. (2019); Nachum and Dai (2020). We explore this connection in the next subsection.

Random Utility Maximization (RUM) models provide an alternative point of view. In this setting, it is assumed that agents choose the action a^* that maximizes $\max_{a \in \mathcal{A}} X_a$. Here X_a are random vectors from our perspective, but each agent sees a sample. So while we have a stochastic problem, agents do not. We assume that X_a is iid up to a shift in location. Let $X_a = r_a + \sigma \epsilon_a$ for iid ϵ_a, a strictly positive temperature σ, and reward of action a, r_a. By changing the temperature, we can change the scale of $\sigma \epsilon_a$ and assume ϵ_a follows a standard distribution, one that does not have a location or scale parameter. RUM models in the form

$$\max_{a \in \mathcal{A}} r_a + \sigma \epsilon_a$$

are called additive RUM (ARUM). Their expected value,

$$w^\sigma(r) = \mathbb{E}_\epsilon[\max_a r_a + \sigma\epsilon_a],$$

is also referred to as the welfare function.

The Williams-Daly-Zachary (WDZ) theorem, a fundamental result in discrete choice analysis and econometrics (McFadden, 1977; Feng et al., 2018), states that the choice probability $P(\arg\max_{a'} r_{a'} + \sigma\epsilon_{a'} = a|r)$, is equal to the gradient of the welfare function, $\partial w^\sigma(r)/\partial r_a$. For instance, if ϵ_a are iid and follow a Gumbel distribution, the welfare of this Gumbel ARUM $w^\sigma(r)$ is the log-sum-exp $\text{lse}^\sigma(r) = \sigma \log \sum_a \exp(r_a/\sigma)$ and the choice probability is the softmax of the rewards $\pi(a) \propto \exp(r_a/\sigma)$.

It is important to note the equivalence of different choice models even if their base definitions are different. We call two choice models corresponding if they have the same welfare and choice probability. Hofbauer and Sandholm (2002) demonstrated that the convex conjugate of any ARUM's welfare function yields a penalty function for a corresponding RAM. Interestingly, they also highlighted that not every RAM has a corresponding RUM, indicating an asymmetry between the two types of models. Feng et al. (2018) showed that if the agents can choose the distribution of ϵ, the two classes become corresponding. The initial connection between RAMs and RUMs was established by Anderson et al. (1988), who demonstrated that the negative entropy RAM and the Gumbel ARUM correspond.

In the following, for brevity, we omit the temperature σ when using the welfare function w.

2.1.1. **Application to Reinforcement Learning.** In general, we can apply the RAM/RUM perspective to MDPs by fixing the discount factor to a non-zero value and introducing a choice model at every state on the actions. It follows that the value function is the welfare of the Q-function, which can be represented as $V(s) = w(Q(s,\cdot))$ where w is a welfare function. By adopting the Gumbel ARUM's welfare function as suggested by Rust (1987), we obtain the smooth Bellman equation, which forms the foundation of entropy-regularized RL:

$$V(s) = \mathbb{E}[\max_a Q(s,a) + \sigma\epsilon_a] \Leftrightarrow V(s) = \sigma \log \left(\sum_{a \in \mathbb{A}(s)} \exp(Q(s,a)/\sigma) \right). \tag{2.1}$$

The smooth Bellman equation, in the log-sum-exp form, has been rediscovered in RL through the SQL algorithm (Haarnoja et al., 2017). We call the Bellman equation with additive Gumbel noise the Gumbel MDP.

Using the RAM perspective, we can also recover the common entropy-regularized RL setting found in Maximum Entropy IRL (MaxEntIRL) (Ziebart et al., 2008) or Soft-Actor Critic (SAC) (Haarnoja et al., 2018), where the corresponding optimality equations are of

the form:

$$V^\pi(s) = \mathbb{E}_{a\sim\pi}[Q^\pi(s,a) - \sigma \log \pi(a|s)] \qquad \text{with} \qquad Q^\pi(s,a) = \mathbb{E}_{s'}[R(s,a) + V^\pi(s')] \ ,$$

and with the expectation over actions induced by the softmax policy. Here, the equivalence with the Rust (1987) model follows from the WDZ theorem. Interestingly, Fosgerau et al. (2013) and Garg et al. (2023) find the same model (not necessarily algorithm) as Ziebart et al. (2008) and Haarnoja et al. (2018), respectively. Geist et al. (2019) further explored the more general setting of regularized MDPs (R-MDP) in the form:

$$V(s) = \max_\pi \mathbb{E}_{a\sim\pi}[Q(s,a)] + \sigma w^*(\pi) = w(Q(s,\cdot)), \tag{2.2}$$

and Mai and Jaillet (2020) explores the hierarchies introduced by different choice models over the space of MDPs.

The entropy-regularized setting is subsumed by the R-MDP framework with w being the log-sum-exp function over actions when w^* is the negative entropy. Adopting this broader perspective, one can also inquire about the convex conjugate of the mellowmax operator (Asadi and Littman, 2017), which differs from the log-sum-exp function only through a state-dependent scaling factor $1/|\mathbb{A}(s)|$:

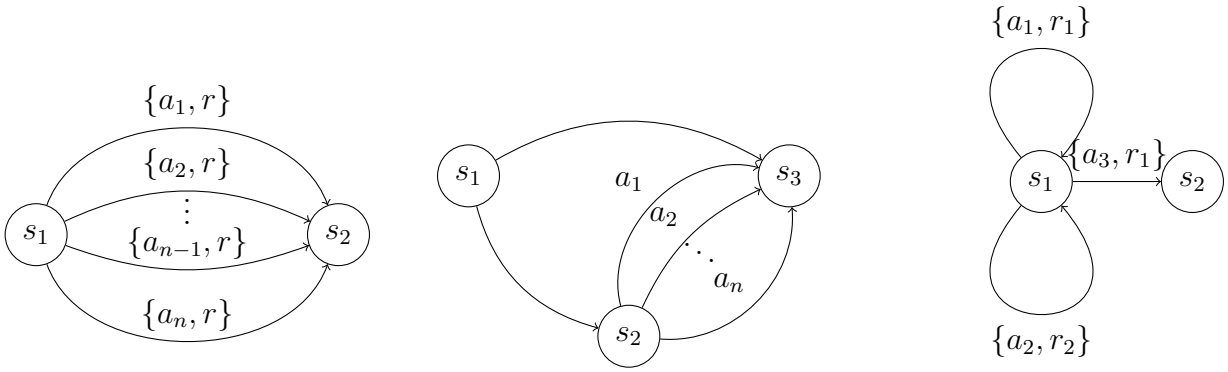
$$V(s) = \sigma \log \left(\frac{1}{|\mathbb{A}(s)|} \sum_{a\in\mathbb{A}(s)} \exp(Q(s,a)/\sigma) \right).$$

It follows from Geist et al. (2019) that the mellowmax corresponds to a RAM with a penalty term given by the KL divergence between the policy and the uniform distribution over actions. However, we find that interpreting the mellowmax operator as choosing w^* to be the negative entropy of the policy plus the maximum entropy $\log|\mathbb{A}(s)|$ to be more general and useful. In the Gumbel ARUM perspective, this corresponds to shifting the noise ϵ_a such that $\mathbb{E}[\max_a \epsilon_a]$ becomes zero. In the subsequent sections, we demonstrate that this shift plays a nontrivial role in the more general case of state-dependent action sets, while providing a built-in "de-biasing" mechanism in the episodic setting.

3. Welfare-Induced Bias

The RAM/RUM framework can introduce challenges when implemented within MDPs. The crux of the issue lies in the agents' prioritization of maximizing welfare over actual rewards. While this discrepancy is not inherently problematic in choice models, complications arise when the Bellman equation propagates welfare values instead of rewards. Indeed, decisions in the future will be made with respect to welfare values not actual rewards.

This propagation of welfare can lead to a compounding effect, ultimately causing agents to exhibit sub-optimal behavior. One notable manifestation of this issue is when agents engage in repetitive, non-productive actions, such as spinning in circles, without making



(a) An MDP with n identical actions

(b) An MDP with two steps

(c) Part of an MDP with two loops.

Figure 3.4 – Three example MDPs illustrating the welfare-induced bias. Edge annotations show the action and the corresponding reward if not zero. All transitions are deterministic.

meaningful progress toward the intended goal. To address this dilemma, it is crucial to reevaluate the balance between welfare and reward maximization in the context of sequential decision-making processes.

In the following, we present two stylized examples. The first illustrates that welfare increases with the number of actions, even when their rewards are equal:

Example 1. *(Bias due to $|\mathbb{A}(s)|$) In a one-step MDP with n actions with equal rewards $r_1 = r_2 = \ldots = r_n = r$ (illustrated in Figure 3.4a), the welfare of the Gumbel MDP is $r + \sigma \log n$. The result follows trivially from the definition of V (2.1): $\mathbb{E}[\max_{i \in \{1,\cdots,n\}} r + \sigma\epsilon_i] = \sigma \log \sum_{i=1}^{n} \exp(r/\sigma) = \sigma \log (n \exp(r/\sigma)) = r + \sigma \log n$.*

We illustrate the implication of Example 1 in Figure 3.4b. As the number of actions n increases, the welfare of taking one of the lower routes (through node s_2) increases. This can happen, for example, in city networks where agents prefer taking grid-like secondary roads instead of highways. A Gumbel MDP is equivalent to a Gumbel ARUM over all paths (Fosgerau et al., 2013), meaning that arcs having more paths passing through them (s_1 to s_2 in the example) have higher probability.

The second example illustrates that cycles can create infinite welfare glitches:

Example 2. *(Bias due to cycles) Consider an MDP that contains at least two deterministic cycles, i.e., there exists at least two distinct sequences of actions that, when taken, lead the agent to the starting state with probability one. We provide an illustration in Figure 3.4c where two cycles are depicted by actions a_1 and a_2. Let $r^m \triangleq \min_{i \in \{1,2\}} r_i$. If $r^m > -\sigma \log 2$, then the total reward for not leaving the cycle in the undiscounted Gumbel MDP becomes infinite, while the discounted total reward is finite and bounded from below by $(r^m + \log 2)/(1-\gamma) > 0$.*

Proof. Let $\mathbb{A}(s_1) \supseteq \{a_1, a_2, a_3\}$ denote the set of actions available to the agent in state s_1, where a_3 is the action that takes the agent out of state s_1. The welfare of choosing one of the

two loops, a_1 or a_2, is positive since

$$\mathbb{E}[\max_{i\in\{1,2\}} r_i + \sigma\epsilon_i] > r^{\mathrm{m}} + \mathbb{E}[\max_i \sigma\epsilon_i] = r^{\mathrm{m}} + \sigma\log 2 > 0,$$

where the last strict inequality holds by assumption. This means that a uniform policy over the two actions a_1 and a_2, will have a net welfare that is positive. The value function, due to the linearity of the expectation, is:

$$\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R(S_t,A_t)\,|\,S_0=s_0\right] = \sum_{t=0}^{\infty}\gamma^t\mathbb{E}\left[R(S_t,A_t)\,|\,S_0=s_0\right] \geq \sum_{t=0}^{\infty}\gamma^t(r^{\mathrm{m}}+\sigma\log 2) = \tfrac{(r^{\mathrm{m}}+\sigma\log 2)}{(1-\gamma)},$$

where $r^{\mathrm{m}} + \sigma\log 2$ is a lower-bound for the welfare of an agent choosing a_1 and a_2 greedily. If $\gamma < 1$, the value is $(r^{\mathrm{m}} + \sigma\log 2)/(1-\gamma)$, but if $\gamma = 1$ (i.e. undiscounted) the value diverges. □

Note that the same result holds if we replace the action with distinct sequences of actions and we have a similar result with n actions. Note that, by design, this example does not satisfy the condition provided by Mai and Frejinger (2022) for the convergence of undiscounted Gumbel MDPs.

A straightforward way to address these two problems is by using what we call mellow regularizers:

Definition 3. *$\hat{w}^*$ is the mellow version of w^* iff*

$$\hat{w}^*(\pi) = w^*(\pi) - w(\mathbb{0}).$$

While it may seem tempting to claim that shifts by a constant have no effect on probabilities in R-MDPs, consider the following:

Remark 4. *(Change in action space in terminations) If an MDP with termination has a constant action space, it needs to have $|\mathcal{A}|$ actions in the terminal state. Since by definition the terminal state has no reward, the reward of going to itself is $w(\mathbb{0})$, for an appropriately sized zero vector $\mathbb{0}$. And the value of the terminal state is $w(\mathbb{0})/(1-\gamma)$.*

As mentioned in Section 1, we have yet to see an implementation that uses this value for the terminal state. Instead all implementations that we have seen implement targets as $r+(1-t)V(s')$ for the reward r, next state s', and termination flag t, implicitly assuming that the value of the terminal state is zero. Thus, there can only be one action, as the following proposition shows.

Proposition 5. *In entropy-regularized RL, $w(\mathbb{0})$ can only be zero if there is one single action.*

Proof. Let N be the number of actions, $w(\mathbb{0})$ is equal $\log\sum_{i=1}^{N}\exp(0) = 0$ thus $\sum_i N\exp(0) = 1$ which implies N is one. □

Using Remark 4 and Proposition 5, we can see that the implementations mentioned in Section 1 model terminations in SQL as an action with one step thus breaking the constant action space assumption.

Remark 6. *By shifting the Gumbel noise in ARUM by $-\sigma \log n$, the welfare becomes*

$$\sigma \log \sum_a \exp\left[(r_a - \sigma \log n)/\sigma\right] = \sigma \log \frac{1}{n} \sum_a \exp(r_a/\sigma).$$

At this point, we have rediscovered mellowmax (Asadi and Littman, 2017).

Remark 7. *Using the procedure described by Hofbauer and Sandholm (2002), we observe that the mellowmax operator is equivalent to augmenting the rewards by $-\sum_a \pi(a|s)(\log n + \log \pi(a|s)) = H(\pi(\cdot|s)) - \log n$, i.e. subtracting the maximum entropy from the entropy function H.*

It is worth noting this also gives us a way of generalizing the mellowmax operator to continuous models by subtracting the maximum entropy possible in the action space. This perspective also adds context to the use of softmax policy by Asadi and Littman (2017).

A last reason to mellow the regularizers was given by Sørensen and Fosgerau (2022): If we define welfare as

$$w(r) = \mathbb{E}\left[\left(\max_a r_a + \epsilon_a\right) - \left(\max_a \epsilon_a\right)\right],$$

the mean of ϵ needs not to exist for w to exist. Knies et al. (2022) uses the $\log n$ factor as a feature instead in inverse reinforcement learning as a consequence of choice overload.

Finding the optimal policy for the new regularizer is trivial:

Proposition 8. *The constant shift does not change the choice probabilities.*

Proof. The gradient of a constant shift is zero, so by the WDZ theorem, the choice probabilities do not change.

$$\hat{\pi}(a) = \frac{\partial \hat{w}(r)}{\partial r_a} = \frac{\partial (w(r) - w(\mathbb{0}))}{\partial r_a} = \frac{\partial w(r)}{\partial r_a} = \pi(a)$$

□

Remark 9. *In deterministic MDPs, such as those presented in Fosgerau et al. (2013), which have no discount factor and exclusively negative rewards, the mellowmax operator consistently possesses a unique fixed point. This follows trivially from Mai and Frejinger (2022) who show that the soft Bellman operator is a contraction as long as $\sum_{a \in \mathbb{A}(s)} \exp(r_a/\sigma) < 1 \ \forall s \in \mathcal{S}$. The same result holds for mellowmax if $\sum_{a \in \mathbb{A}(s)} \exp(r_a/\sigma)/|\mathbb{A}(s)| < 1 \ \forall s \in \mathcal{S}$. Since r_a is assumed to be negative, this condition is always satisfied.*

Normalized entropy. One of the main drawbacks of entropy as a regularizer is that its maximum depends on the number of actions. Therefore, we propose using $H(\pi)/\max_{\pi'} H(\pi')$ as the regularizer, which we refer to as *normalized entropy*. Whereas this merely corresponds to a change of temperature in the constant action setting (even with termination), in the dynamic action setting, it can lead to large differences. The mellowmax operator can even be found in this form at an effective temperature $t/\log|\mathbb{A}(s)|$ at state s. This transformation is similar

to efficiency in the information theoretical sense, (Alencar, 2014, Chapter 4). In general, any regularization with a positive maximum can be scaled with this generic transformation:

Definition 10. *(Normalized regularizers) Let $w(\mathbb{0})$ be greater than zero, $\tilde{w}^*(x) = w^*(x)/w(x-x)$ is the normalized version of $w^*(x)$. Note that $\tilde{w}(\mathbb{0}) = 1$.*

This transformation also fixes the issue illustrated in Example 1.

It would be interesting to extend Mai et al. (2015)'s model similarly to how Knies et al. (2022) extended Fosgerau et al. (2013). Instead of adding the $\log n$ as a feature of the cost function, it will be a feature of the temperature function.

4. Regularized Stochastic Shortest Path

We turn our attention to undiscounted ($\gamma = 1$) infinite horizon MDPs with absorbing state, known as stochastic shortest path (SSP) problems (Bertsekas and Tsitsiklis, 1991; Bertsekas, 2012). In this context, a policy is considered *proper* if it guarantees reaching the absorbing state with probability one. We focus on a general non-contraction setting mentioned in Bertsekas (2012) and stochastic policies. Existing results do not apply here since the proofs in Bertsekas and Tsitsiklis (1991) are based on the assumption that all policies are proper, and Bertsekas (2012) considers deterministic policies.

We start by extending the definition of a proper policy to include the requirement that it must fall within the domain of w^*. It is important to note that a strictly convex function is ∞ outside its domain.

Assumption 11. *(Bertsekas, 2012, Assumption 2.1.1, extended) There exists at least one proper stochastic policy π in the domain of w^*.*

Moreover, we introduce an assumption ensuring that there exists a solution to the SSP problem. It is *always* possible to reach the terminal state.

Assumption 12. *(Bertsekas, 2012, Assumption 2.1.1) Policy evaluation diverges to $-\infty$ for any non-proper policy.*

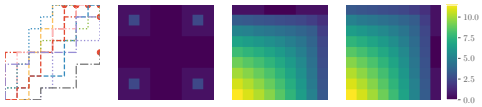
This assumption can be trivially satisfied by having strictly negative rewards and ensuring that it is possible to reach the absorbing state from any state. In entropy-regularized SSP, any action has a positive probability. Hence, if it is possible to reach the absorbing state, any policy in the domain of w^* is proper. We extend the results of Bertsekas (2012) with respect to stochastic shortest paths in Appendix B.1.

Next, we show the main result, namely that Assumption 12 holds for mellow regularized SSPs if it holds for the original problem.

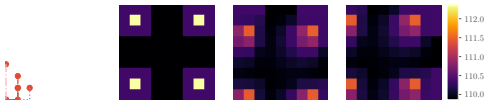
Proposition 13. *(Free mellow proposition) For any SSP problem where Assumption 12 holds, Assumption 12 also holds for the corresponding mellow regularized SSP.*

(a) Reward function.

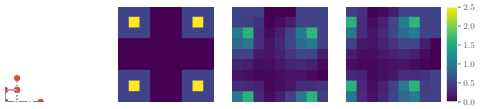
(b) Bellman Q-function



(c) Soft Q-function



(d) Soft Q-function, with three actions in the terminal state, $\gamma = 0.999$. Notice the change in the scale of the value functions.



(e) Mellow Q-function

Figure 3.5 – Results on the 2d hyper-grid of Bengio et al. (2021). The plots in the figure represent sample trajectories, values for stopping, moving up, and moving left respectively.

PROOF. The regularized reward $R_\pi(s,a) = R(s,a) + w^*(\pi) \leq R(s,a)$ due to the mellowness of w^*. If

$$\lim_{N\to\infty} \mathbb{E}\left[\sum_{t=0}^{N-1} R(S_t, A_t)\right], \tag{4.1}$$

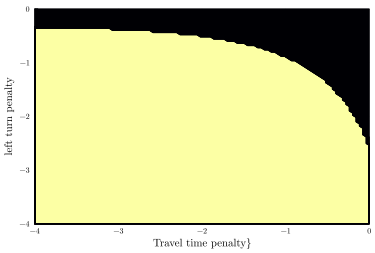
diverges to $-\infty$, then $\lim_{N\to\infty} \mathbb{E}[\sum_{t=0}^{N-1} R_\pi(S_t, A_t)]$ which is bounded from above by (4.1), diverges to $-\infty$. □

5. Experiments

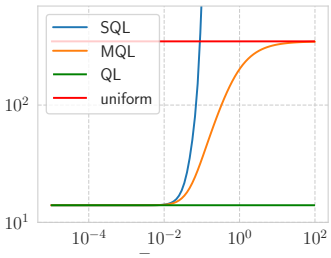
5.1. A Closer Look at Mellowmax

We start by looking at the choice probability of Figure 3.4b. We show the probability of taking the upper path, p, in Figure 3.6c. We can see that SQL (log-sum-exp) gives the same probability to all paths in the MDP, while the mellow version gives the same probability of taking the upper and lower paths. Note that this result holds regardless of the temperature.

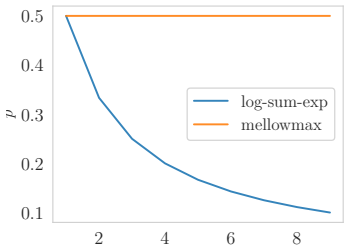
We illustrate the termination problem by using the 2d hyper-grid domain of Bengio et al. (2021). In this MDP, an agent is in a grid, and either ends the episode or increases one of its location components (i.e., move up or right, but not left). The agent starts at the origin $(0,0)$. Figure 3.5a shows the reward functions for the stop, up, and left actions. The episode ends only due to the stop action or moving out of bounds. We use $\gamma = 1$ and set the temperature to 1 unless specified. Figure 3.5b shows the usual hard Bellman Q-function, with the first column showing sample trajectories. Due to the accumulation of the welfare, as seen



(a) Domain for the model of Fosgerau et al. (2013), black points represent parameters valid for the mellow variant yet out of domain points for the original formulation.



(b) Expected trajectory length, as a function of temperature.



(c) Probability of taking the upper path p in Figure 3.4b as a function of n.

Figure 3.6 – Results for a real-world transport application (a), 2d hyper-grid (b), and Example 1 (c)

in Figure 3.5c, the entropy regularized version prefers the top right points as there are more paths to those points. Figure 3.5d illustrates what happens when we set γ to .999 and keep the number of actions to 3 in the terminal state. In Figure 3.5e, we show the mellow variant. As expected, the last two figures have the same policy, yet the large shift of Figure 3.5d can have an adverse effect when coupled with function approximators.

We illustrate the loop problem by using a similar 2d grid where an agent is once again initialized at the origin, but the episodes end when the agent reaches the top right corner, can move in all four directions, and gets a reward of one for terminating the episode. This time moving outside of the boundary does nothing. We set $\gamma = 0.9$ and measure the expected path length as a function of the temperature in Figure 3.6b. As expected, SQL diverges in higher temperatures, yet mellowmax (MQL) converges to the uniform policy.

Finally, we illustrate the convergence of Fosgerau et al. (2013) with mellowmax in Figure 3.6a. We use the original data of Fosgerau et al. (2013) with four attributes for travel time, left turn flags, node penalty (1 everywhere), and a u-turn penalty. The figure shows in black the parts of the domain that are only convergent for the mellow variant when we fix the last two attributes to their optimal value.

5.2. Application in Inference as Control

The entropy-regularized framework has led to new inference control methods for sampling large discrete spaces. These methods use an autoregressive model to address the curse of dimensionality step-by-step, resembling MCMC methods, but with a focus on learning policies to construct objects sequentially for sample generation based on a reward signal without rejection.

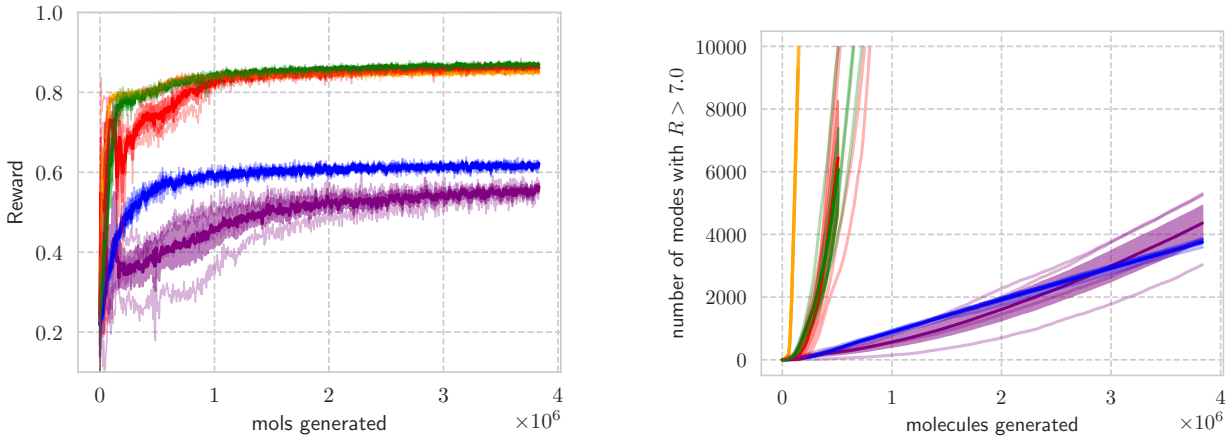


Figure 3.7 – Comparison of different methods. The left plot is the average reward of generated molecules. Right is the number of modes found at the level. Yellow is GFN, green is normalized mellowmax, blue is mellowmax, red is normalized SQL, and purple is standard SQL. The dark lines are average (with an exponential moving average on the right figure). The light lines are individual runs. The bars show the error.

Examining these problems within the MDP framework, the state space consists of a subset of graphs, and the action space involves adding elements to the graph. The agent decides when to stop generating a fragment with a random but finite horizon. Policies here represent conditional probabilities over the next graph change, given the current graph. Actions determine the next state deterministically from the current state. In these problems, the policy doesn't depend on history, and the graph construction order is generally irrelevant. For example, when building a molecule, the atom addition order does not matter, as the sequential structure is artificially designed to combat the curse of dimensionality.

Our experiment involves molecule fragment design, where an agent adds fragments to build a drug. The original GFlowNet (GFN) paper (Bengio et al., 2021) applies SQL to this MDP but struggles to learn meaningful representations in the molecule generation domain. GFN aims to learn a RUM on drug sets, propagating probabilities through the MDP.

We base our experiments on Malkin et al. (2022), and use the same neural network. However, in addition to the base model used by GFN, we use a large replay buffer, an exponential moving average target, and the tree backup (Sutton and Barto, 2018, Section 7.5). Hyper-parameters are provided in Appendix B.3.

We compare our variants of SQL with trajectory balance (TB) (Malkin et al., 2022). Figure 3.7 shows that our method finds high-quality molecules after the initial burn-in period. Empirically, TB is more stable to train, but our results contradict the claim SQL is not able to generate diverse candidates (Bengio et al., 2021). We hypothesize that larger molecules while being harder to manufacture, can have a higher reward. Thus having a bias towards larger molecules can help achieve better results. We analyze this claim in Appendix B.2

6. Conclusion

In conclusion, we have shown that bias in R-MDPs encourages agents to maximize their lifespan, which can be problematic in scenarios where the aim is to have shorter, more rewarding lifespans. While acknowledging the necessity of survival in most domains, as dead agents cannot accrue rewards, we proposed two solutions to mitigate this bias: a generalization of mellowmax and a state-dependent-action specific fix. We have delved into the relatively unexplored realm of the interplay of stochastic shortest and R-MDPs. By presenting theoretical models and real-world examples, we have offered a comprehensive study of this bias.

Our work has also yielded state-of-the-art results in the application of drug design MDP. This achievement is particularly remarkable, as we have demonstrated the feasibility of using Deep Q-Networks (DQN) in a domain where it was previously deemed impossible. The simplicity of DQN, along with its natural formulation in the presence of cycles and independence from the inverse dynamics compared to TB, makes it an attractive choice grounded on MDP formalism.

Beyond this work, we believe that the two fixes add two hyperparameters for practitioners. In particular, in inverse reinforcement learning, it might be interesting to include the number of actions both as a feature for the state-dependent temperature and the reward function. Furthermore, it is of interest to use the normalized entropy instead of entropy in benchmarks where the number of actions changes *between* environments.

Acknowledgments

We thank David Yu-Tung Hui and Emmanuel Bengio for many helpful discussions around Q-learning, GFlowNets, and molecule design. We acknowledge interesting discussions with Kolya Malkin, Moksh Jain, Matthieu Geist, and Olivier Pietquin. This research was enabled in part by support provided by Mila, Calcul Quebec, and the Digital Research Alliance of Canada.

Conclusion

This thesis has addressed two distinct challenges arising from the utilization of choice models on Markov Decision Processes (MDPs). The first of these challenges pertains to the necessity of learning from partial observations, and the second deals with bias in regularized MDPs.

In Article 1, we proposed a method to use data at different levels of granularity to estimate the arc travel time matrix and the parameters of a path choice model. Our method is compatible with a large class of path choice models, and we showed that we can leverage existing losses to choose the travel time variation distribution. We show that our mixture generalizes existing work (Bertsimas et al., 2019). In the experimental section, we took this to the extreme and trained a path choice model without observing paths. We show that classical methods can lead to wrong parameter estimates. We also show that our method is relatively fast even for large problems.

Beyond this work, the primary warning remains assumptions made about the path choice model during the travel time estimation phase; for instance, the shortest path assumption of Bertsimas et al. (2019) can lead to wrong parameter estimations for the path choice model. Yet the path choice model may be able to compensate for the wrong assumptions made in the travel time estimation phase so a large log-likelihood is not necessarily a sign that the parameters estimated for the path choice model are good.

We now turn to a few limitations of Article 1. Our study primarily concentrated on the choice model of Fosgerau et al. (2013), and more recent models proposed by Mai et al. (2015) and Fosgerau et al. (2022) were not explored. Furthermore, we did not consider gradient estimators like Fu and Hu (2012); Jang et al. (2016). The generation of efficient code for sparse automatic differentiation remains a challenging endeavor, requiring the manual computation of gradients in this case.

Access to larger-scale datasets beyond the one from New York would provide an interesting opportunity to test the model under varying conditions, especially in cases that incorporate a mix of missing or partial path observations. It could also be instructive to test the model's performance on larger geographical regions; for instance, the NYC yellow cab dataset, which

encompasses the entire city, not just Manhattan. We observed that our method scales much better than Bertsimas et al. (2019), offering an intriguing benchmark for comparison.

This foundational idea of Article 1, integrating out the unknown, bears implications that extend beyond transportation networks. When learning an IRL model, the same procedure can be applied to integrate the unobserved variables, given that the MDP structure is known but not the rewards. Future research could examine the feasibility of concurrently learning a world model alongside the IRL model, possibly in a latent space, similar to Schrittwieser et al. (2020).

We now turn our attention to Article 2 that identified two major problems with regularized MDPs, and provided two fixes, one for all regularizers and one for the state-dependent action setting. The first fix is an extension of the mellowmax operator (Asadi and Littman, 2017) to all regularizers while the second one is very similar to the notion of efficiency in the information-theoretic sense of the word (Alencar, 2014). These fixes are designed to help overcome preferences for staying alive or structures with more paths like grids and cliques.

A key contribution of Article 2 includes successfully demonstrating that, contrary to the claims of Bengio et al. (2021); Malkin et al. (2022), Soft Q-Learning (SQL) is capable of generating a substantial variety of molecules. This implies that the performance of GFlowNets may be more closely aligned with RL than shown by Bengio et al. (2021).

Lastly, We also introduced regularized stochastic shortest paths. The importance of regularized stochastic shortest path (R-SSP) settings cannot be overstated, as discounting is not always employed in practice. Even in Arcade learning environment (Bellemare et al., 2013), while most methods (Mnih et al., 2015; Hessel et al., 2018) use a discount factor, the final evaluation metric is the total score.

Rust (1987) discovered that Harold Zurcher, the agent he is trying to model, appeared to optimize for total return. Exploring the mellow version of Rust (1987)'s model would be interesting as we do not need to, as per Rust (1987)'s suggestion, use Abel's Theorem (Rust, 1987) to take the limit of the value function when the discount converges to one. One of Oyama (2023)'s main claims is that Fosgerau et al. (2013)'s value functions may not be finite, yet with the mellowmax fix, the value functions will be finite as shown, so it is interesting to compare the two approaches.

In Article 2, the focus was predominantly on entropy-regularized RL, although a multitude of other regularizers is potentially applicable. Consequently, future studies would benefit from evaluating the two proposed solutions on other regularizers. For example, Tsallis entropy (Tsallis, 1988) is not as severely impacted by the normalization problem as information entropy (Tsallis, 1988), yet the effect of the normalizing fix would be interesting. Standard choices in reinforcement learning seem to gravitate towards $k = 1/2$ and $q = 2$ (Martins and Astudillo, 2016; Lee et al., 2018; Chow et al., 2018; Geist et al., 2019), possibly due to the associated literature on fast l2 projections on the probability simplex (Chen and Ye, 2011;

Wang and Carreira-Perpinán, 2013). However, an efficient algorithm applicable to our use case remains elusive.

An intriguing observation is the prevalent focus on regularizers that are modular, i.e., have a diagonal hessian since they are both submodular and supermodular. For a definition of sub and supermodular functions on real numbers, refer to Murota (2003). However, as discussed in the Introduction, the exploration of non-modular regularizers, as in the three-camera problem, can be of significant interest.

In summary, while this thesis has reached its conclusion, it leaves open several intriguing avenues for further investigation. The problem of extending choice models to MDPs remains a compelling area of study. Our exploration, in its effort to address challenges, has inevitably unearthed new potential problems and questions that would undoubtedly drive future research in this field. The intersections of choice modeling, MDPs, and their complexities open a vast expanse of possibilities. The practical applications and theoretical richness promise a continued, vibrant future for this research domain.

Bibliography

Abdzaid Atiyah, I., Mohammadpour, A., Ahmadzadehgoli, N., and Taheri, S. (2020). Fuzzy C-means clustering using asymmetric loss function. *Journal of Statistical Theory and Applications*, 19(1):91–101.

Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., and Kolter, J. Z. (2019). Differentiable convex optimization layers. *Advances in neural information processing systems*, 32.

Aguirregabiria, V. and Mira, P. (2010). Dynamic discrete choice structural models: A survey. *Journal of Econometrics*, 156(1):38–67.

Alencar, M. S. (2014). *Information theory*. Momentum Press.

Amos, B. and Kolter, J. Z. (2017). Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR.

Anderson, S. P., de Palma, A., and Thisse, J. (1988). A representative consumer theory of the logit model. *International Economic Review*, 29(3):461–466.

Arrow, K. J. et al. (1951). *Social Choice and Individual Values*, volume 12. Yale University Press.

Asadi, K. and Littman, M. L. (2017). An alternative softmax operator for reinforcement learning. In *International Conference on Machine Learning*, pages 243–252. PMLR.

Audet, C., Digabel, S. L., Montplaisir, V. R., and Tribes, C. (2021). Nomad version 4: Nonlinear optimization with the MADS algorithm. *arXiv preprint arXiv:2104.11627*.

Baillon, J.-B. and Cominetti, R. (2008). Markovian traffic equilibrium. *Mathematical Programming Series B*, 111:33–56.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279.

Bellman, R. (1954). The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515.

Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. (2021). Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394.

Bertsekas, D. (2012). *Dynamic programming and optimal control: Volume II*. Athena scientific.

Bertsekas, D. P. and Tsitsiklis, J. N. (1991). An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595.

Bertsimas, D., Delarue, A., Jaillet, P., and Martin, S. (2019). Travel time estimation in the age of big data. *Operations Research*, 67(2):498–515.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98.

Chen, Y. and Ye, X. (2011). Projection onto a simplex. *arXiv preprint arXiv:1101.6081*.

Chow, Y., Nachum, O., and Ghavamzadeh, M. (2018). Path consistency learning in tsallis entropy regularized mdps. In *International conference on machine learning*, pages 979–988. PMLR.

Cortés, C. E., Donoso, P., Gutiérrez, L., Herl, D., and Muñoz, D. (2023). A recursive stochastic transit equilibrium model estimated using passive data from santiago, chile. *Transportation Research Part B: Methodological*, 174:102780.

Dai, B., Shaw, A., Li, L., Xiao, L., He, N., Liu, Z., Chen, J., and Song, L. (2018). Sbeed: Convergent reinforcement learning with nonlinear function approximation. In *International Conference on Machine Learning*, pages 1125–1134. PMLR.

DasGupta, A. (2008). *Asymptotic theory of statistics and probability*, volume 180. Springer.

de Freitas, L. M., Becker, H., Zimmermann, M., and Axhausen, K. W. (2019). Modelling intermodal travel in switzerland: A recursive logit approach. *Transportation Research Part A: Policy and Practice*, 119:200–213.

de Moraes Ramos, G., Mai, T., Daamen, W., Frejinger, E., and Hoogendoorn, S. (2020). Route choice behaviour and travel information in a congested network: Static and dynamic recursive models. *Transportation Research Part C: Emerging Technologies*, 114:681–693.

Di, X. and Liu, H. X. (2016). Boundedly rational route choice behavior: A review of models and methodologies. *Transportation Research Part B: Methodological*, 85:142–179.

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

Ding-Mastera, J., Gao, S., Jenelius, E., Rahmani, M., and Ben-Akiva, M. (2019). A latent-class adaptive routing choice model in stochastic time-dependent networks. *Transportation Research Part B: Methodological*, 124:1–17.

Feng, G., Li, X., and Wang, Z. (2017). Technical note—on the relation between several discrete choice models. *Operations Research*, 65(6):1516–1525.

Feng, G., Li, X., and Wang, Z. (2018). On substitutability and complementarity in discrete choice models. *Operations Research Letters*, 46(1):141–146.

Fisk, C. (1980). Some developments in equilibrium traffic assignment. *Transportation Research Part B: Methodological*, 14(3):243–255.

Fosgerau, M., Frejinger, E., and Karlstrom, A. (2013). A link based network route choice model with unrestricted choice set. *Transportation Research Part B: Methodological*, 56:70–80.

Fosgerau, M., Paulsen, M., and Rasmussen, T. K. (2022). A perturbed utility route choice model. *Transportation Research Part C: Emerging Technologies*, 136:103514.

Freedman, D. A. (2006). On the so-called "huber sandwich estimator" and "robust standard errors". *The American Statistician*, 60(4):299–302.

Frejinger, E. and Zimmermann, M. (2021). Route choice and network modeling. In *International Encyclopedia of Transportation*, volume 4, pages 496–503. Elsevier.

Fu, M. C. and Hu, J.-Q. (2012). *Conditional Monte Carlo: Gradient estimation and optimization applications*, volume 392. Springer Science & Business Media.

Gao, S., Frejinger, E., and Ben-Akiva, M. (2010). Adaptive route choices in risky traffic networks: A prospect theory approach. *Transportation Research Part C: Emerging Technologies*, 18(5):727–740.

Gao, Y. (2021). Estimation of tourist travel patterns with recursive logit models based on wi-fi data with kyoto city case study.

Garg, D., Hejna, J., Geist, M., and Ermon, S. (2023). Extreme Q-learning: MaxEnt RL without entropy. *arXiv preprint arXiv:2301.02328*.

Geist, M., Scherrer, B., and Pietquin, O. (2019). A theory of regularized markov decision processes. In *International Conference on Machine Learning*, pages 2160–2169. PMLR.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. (2017). Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pages 1352–1361. PMLR.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*.

Hinton, G., Srivastava, N., and Swersky, K. (2012). Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2.

Hofbauer, J. and Sandholm, W. H. (2002). On the global convergence of stochastic fictitious play. *Econometrica*, 70(6):2265–2294.

Iizuka, T. and Hato, E. (2020). Cost sensitive estimation methods of the rl-activity-scheduling models under disasters.

Jabari, S. E., Freris, N. M., and Dilip, D. M. (2020). Sparse travel time estimation from streaming data. *Transportation Science*, 54(1):1–20.

Jan, O., Horowitz, A. J., and Peng, Z.-R. (2000). Using global positioning system data to understand variations in path choice. *Transportation Research Record*, 1725(1):37–44.

Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Jenelius, E. and Koutsopoulos, H. N. (2013). Travel time estimation for urban road networks using low frequency probe vehicle data. *Transportation Research Part B: Methodological*, 53:64–81.

Jin, F. and Sun, S. (2008). Neural network multitask learning for traffic flow forecasting. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1897–1901. IEEE.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Knies, A., Lorca, J., and Melo, E. (2022). A recursive logit model with choice aversion and its application to transportation networks. *Transportation research part B: methodological*, 155:47–71.

Koch, T. and Dugundji, E. (2020). A review of methods to model route choice behavior of bicyclists: inverse reinforcement learning in spatial context and recursive logit. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on GeoSpatial Simulation*, pages 30–37.

Lafferriere, B., Lafferriere, G., and Nguyen, M. N. (2022). *Introduction to Mathematical Analysis I*. Portland State University Library.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. j. (2007). *Energy-Based Models*. The MIT Press.

L'Ecuyer, P. (1990). A unified view of the ipa, sf, and lr gradient estimation techniques. *Management Science*, 36(11):1364–1383.

Lee, K., Choi, S., and Oh, S. (2018). Sparse markov decision processes with causal sparse tsallis entropy regularization for reinforcement learning. *IEEE Robotics and Automation Letters*, 3(3):1466–1473.

Li, R. and Rose, G. (2011). Incorporating uncertainty into short-term travel time predictions. *Transportation Research Part C: Emerging Technologies*, 19(6):1006–1018.

Lima, A., Stanojevic, R., Papagiannaki, D., Rodriguez, P., and González, M. C. (2016). Understanding individual routing behaviour. *Journal of The Royal Society Interface*, 13(116):20160021.

Luce, R. D. (1959). *Individual choice behavior: A theoretical analysis*. Courier Corporation.

Mai, T., Bastin, F., and Frejinger, E. (2018). A decomposition method for estimating recursive logit based route choice models. *Euro Journal on Transportation and Logistics*, 7(3):253–275.

Mai, T., Bui, T. V., Nguyen, Q. P., and Le, T. V. (2023). Estimation of recursive route choice models with incomplete trip observations. *Transportation Research Part B: Methodological*, 173:313–331.

Mai, T., Fosgerau, M., and Frejinger, E. (2015). A nested recursive logit model for route choice analysis. *Transportation Research Part B: Methodological*, 75:100–112.

Mai, T. and Frejinger, E. (2022). Undiscounted recursive path choice models: Convergence properties and algorithms. *Transportation Science*, 56(6):1469–1482.

Mai, T. and Jaillet, P. (2020). A relation analysis of markov decision process frameworks. *arXiv preprint arXiv:2008.07820*.

Mai, T., Yu, X., Gao, S., and Frejinger, E. (2021). Routing policy choice prediction in a stochastic network: Recursive model and solution algorithm. *Transportation Research Part B: Methodological*, 151:42–58.

Malkin, N., Jain, M., Bengio, E., Sun, C., and Bengio, Y. (2022). Trajectory balance: Improved credit assignment in gflownets. *arXiv preprint arXiv:2201.13259*.

Martins, A. and Astudillo, R. (2016). From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International conference on machine learning*, pages 1614–1623. PMLR.

McFadden, D. (1974). The measurement of urban travel demand. *Journal of Public Economics*, 3(4):303–328.

McFadden, D. (1977). Modelling the choice of residential location. *Transportation Research Record*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Mohamed, S., Rosca, M., Figurnov, M., and Mnih, A. (2020). Monte Carlo Gradient Estimation in Machine Learning. *Journal of Machine Learning Research*, 21(132):1–62.

Mori, Mendiburu, A., Álvarez, M., and Lozano, J. A. (2015). A review of travel time estimation and forecasting for advanced traveller information systems. *Transportmetrica A: Transport Science*, 11(2):119–157.

MOSEK ApS (2022). *MOSEK optimization suite. Version 9.3*.

Murota, K. (2003). *Discrete Convex Analysis*. Society for Industrial and Applied Mathematics.

Nachum, O. and Dai, B. (2020). Reinforcement learning via fenchel-rockafellar duality. *arXiv preprint arXiv:2001.01866*.

New York City Taxi and Limousine Commission (2016). New york city taxi trip data. Data downloaded in June 2022 from `https://registry.opendata.aws/nyc-tlc-trip-records-pds`.

Nikolova, E. and Stier-Moses, N. E. (2014). A mean-risk model for the traffic assignment problem with stochastic travel times. *Operations Research*, 62(2):366–382.

Obando-Ceron, J. S. and Castro, P. S. (2020). Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. *arXiv preprint arXiv:2011.14826*.

Oh, S., Byon, Y.-J., Jang, K., and Yeo, H. (2015). Short-term travel-time prediction on highway: A review of the data-driven approach. *Transport Reviews*, 35(1):4–32.

OpenStreetMap contributors (2022). OpenStreetMap project database. Downloaded in June 2022 from `https://overpass-api.de`.

Oyama, Y. (2017). *A Markovian route choice analysis for trajectory-based urban planning*. PhD thesis, Doctoral Thesis, The University of Tokyo.

Oyama, Y. (2023). Capturing positive network attributes during the estimation of recursive logit models: A prism-based approach. *Transportation Research Part C: Emerging Technologies*, 147:104014.

Prato, C. G. (2009). Route choice modeling: past, present and future research directions. *Journal of Choice Modelling*, 2(1):65–100.

Rust, J. (1987). Optimal replacement of gmc bus engines: An empirical model of harold zurcher. *Econometrica: Journal of the Econometric Society*, pages 999–1033.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. (2020). Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609.

Shaygan, M., Meese, C., Li, W., Zhao, X. G., and Nejad, M. (2022). Traffic prediction using artificial intelligence: Review of recent advances and emerging opportunities. *Transportation research part C: emerging technologies*, 145:103921.

Sørensen, J. R.-V. and Fosgerau, M. (2022). How mcfadden met rockafellar and learned to do more with less. *Journal of Mathematical Economics*, 100:102629.

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

Thurstone, L. L. (1927). A law of comparative judgment. *Psychological review*, 34:273–286.

Tsallis, C. (1988). Possible generalization of boltzmann-gibbs statistics. *Journal of statistical physics*, 52:479–487.

Wang, H., Tang, X., Kuo, Y.-H., Kifer, D., and Li, Z. (2019). A simple baseline for travel time estimation using large-scale trip data. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–22.

Wang, W. and Carreira-Perpinán, M. A. (2013). Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *arXiv preprint arXiv:1309.1541*.

Williams, R. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256.

Zhang, W., Buehler, R., Broaddus, A., and Sweeney, T. (2021). What type of infrastructures do e-scooter riders prefer? a route choice model. *Transportation research part D: transport and environment*, 94:102761.

Zhu, C., Byrd, R., Lu, P., and Nocedal, J. (1997). Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound-Constrained Optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560.

Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, volume 8, pages 1433–1438.

Zimmermann, M. and Frejinger, E. (2020). A tutorial on recursive models for analyzing and predicting path choice behavior. *EURO Journal on Transportation and Logistics*, 9(2):100004.

Chapter A

Appendix for Paper 1

A.1. BDJM's model and method

Let $(o,d) \in W$ be the set of origin and destination in the training set, $\boldsymbol{t}_{od}$ be the geometrical average of the travel times observed between o and d, $\boldsymbol{n}_{od}$ be the number of observations between o and d, $\hat{\boldsymbol{t}}_{od}$ be the length of the shortest path from o to d, $K(o,d) \subseteq R(o,d)$ be a set of paths from o to d and z^r_{od} be a binary variable that is only one when the path r is the shortest path. The BDJM model is:

$$
\begin{aligned}
\underset{\boldsymbol{T},\hat{\boldsymbol{t}},\boldsymbol{z}\geq 0}{\text{minimize}} \quad & \sum_{(o,d)\in W} \boldsymbol{n}_{od}(\ln \hat{\boldsymbol{t}}_{od} - \ln \boldsymbol{t}_{od})^2 + \lambda \sum_{(i,j)\in E}\sum_{(j,k)\in E} \left|\frac{\boldsymbol{T}_{ij}}{\boldsymbol{L}_{ij}} - \frac{\boldsymbol{T}_{jk}}{\boldsymbol{L}_{jk}}\right| \frac{1}{\boldsymbol{L}_{ij}+\boldsymbol{L}_{jk}} \\
\text{subject to} \quad & \hat{\boldsymbol{t}}_{od} \leq \sum_{(i,j)\in r} \boldsymbol{T}_{ij} && \forall (o,d)\in W, r\in K(o,d), \\
& \hat{\boldsymbol{t}}_{od} \geq \sum_{(i,j)\in r} \boldsymbol{T}_{ij} - M(1-z^r_{od}) && \forall (o,d)\in W, r\in K(o,d), \\
& \sum_{r\in K(o,d)} z^r_{od} = 1 && \forall (o,d)\in W, \\
& z^r_{od} \in \{0,1\} && \forall (o,d)\in W, r\in K(o,d)
\end{aligned}
\tag{A.1.1}
$$

The BDJM method finds z using a shortest path algorithm and then solves the rest of the problem. The process is repeated until convergence. Let r^*_{od} be the shortest path from o

to d; the BDJM method solves the following program after each shortest path procedure:

$$\begin{aligned}
\underset{\boldsymbol{T},\hat{\boldsymbol{t}},\boldsymbol{x}\geq 0}{\text{minimize}} \quad & \sum_{(o,d)\in W} \boldsymbol{n}_{od}\boldsymbol{x}_{od} + \lambda \sum_{(i,j)\in E}\sum_{(j,k)\in E} \left|\frac{\boldsymbol{T}_{ij}}{\boldsymbol{L}_{ij}} - \frac{\boldsymbol{T}_{jk}}{\boldsymbol{L}_{jk}}\right| \frac{1}{\boldsymbol{L}_{ij}+\boldsymbol{L}_{jk}} \\
\text{subject to} \quad & \hat{\boldsymbol{t}}_{od} \leq \sum_{(i,j)\in r} \boldsymbol{T}_{ij} \quad \forall (o,d)\in W, r\in K(o,d), \\
& \hat{\boldsymbol{t}}_{od} = \sum_{(i,j)\in r^*_{od}} \boldsymbol{T}_{ij} \forall (o,d)\in W, \\
& \boldsymbol{x}_{od} \geq \hat{\boldsymbol{t}}_{od}/\boldsymbol{t}_{od} \qquad \forall (o,d)\in W, \\
& \hat{\boldsymbol{t}}_{od}\boldsymbol{x}_{od} \geq \boldsymbol{t}_{od} \qquad \forall (o,d)\in W.
\end{aligned} \tag{A.1.2}$$

The last constraint is modeled using a second-order cone as $ab \geq c$ is equivalent to $a+b \geq \|(a-b, 2\sqrt{c}\|$. The set K is populated using the previous shortest paths.

A.2. Data processing

For the real data, just like Bertsimas et al. (2019), we first download a recent map of the city from Open Street Map (OSM) (OpenStreetMap contributors, 2022) and cut out the parts outside a polygon that encircles the target (Manhattan). We fill the missing speed limit using official maximum speed limits based on the type of road. We remove all none intersection nodes as long as adding arcs to bypass that node will not create arcs larger than 100 meters. A non-intersection node is a node that has two neighbors such that they are all part of a one-way or two-way path. We keep stop signs and red light nodes unless the crossing is at most 50 meters or 100 nodes away from a red light or stop sign. We split arcs longer than 200 meters into two arcs connected to a new node in the middle of the two nodes. Note that we keep dead ends. We repeat this process until we cannot remove any more nodes. We then take the largest connected component of the graph. This transformation keeps the topology intact as we can easily interpolate the original graph's travel time. This process removes many nodes whose sole purpose is to convey the geometry of the streets, as OSM does not support curved arcs.

We match the origin and destination from the cab dataset [1] using the nearest node with an upper bound of 100 meters on the distance between the observation and the node. We remove trips that are shorter than 30 seconds or longer than 3 hours or have an average speed of less than 1 meter per second or greater than the city's maximum speed on the highway (50 miles per hour in the case of NYC) on the shortest path connecting the origin and destination. We then take the data observed in a time range of day groups (workdays, weekends, etc.). We show the results in Figure A.1.

1. The dataset is roughly 2 gigabytes, but the compressed version with the required columns only takes 150 megabytes, since we only consider a roughly 60-hour window, we even have fewer data

We take a training and validation set with 100, 1,000, 10,000, 100,000, and 200,000 observations. The test set contains 100,000 observations. We use the data from April 2016 to validate the code and show the results on May 2016.

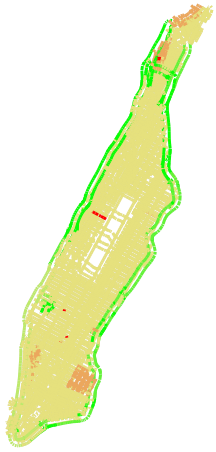


Figure A.1 – Map of Manhattan at free-flow speed.

For the synthetic tests, we simulated data on a ten-by-ten grid where each node is 600 meters away from its neighbors. The minimum and maximum speeds are 5.5 and 10 m/s, respectively. The travel time of the arc that ends in the ith column and jth row is $600/10 - 3(j - 3300)^2/3300^2$ clamped between the legal speed. We use $\boldsymbol{b} = (-2, -2, -5)^\top$ to generate 10,000 samples for the train, test, and validation sets. We generate five trajectories between each (o, d) uniformly. We multiply each observed travel time by a sample from the log-normal $(0.1, \sqrt{0.1})$. We show the grid in Figure 2.3a.

A.3. Hyperparameter and intialization figures

In this section, we analyze the effect of hyperparameters on our model. We take the best hyperparameter found on the Manhattan late-morning dataset with 10,000 observations and change one of the hyperparameters in each figure and train the model. We show both the training curve for different parameters and the final validation loss.

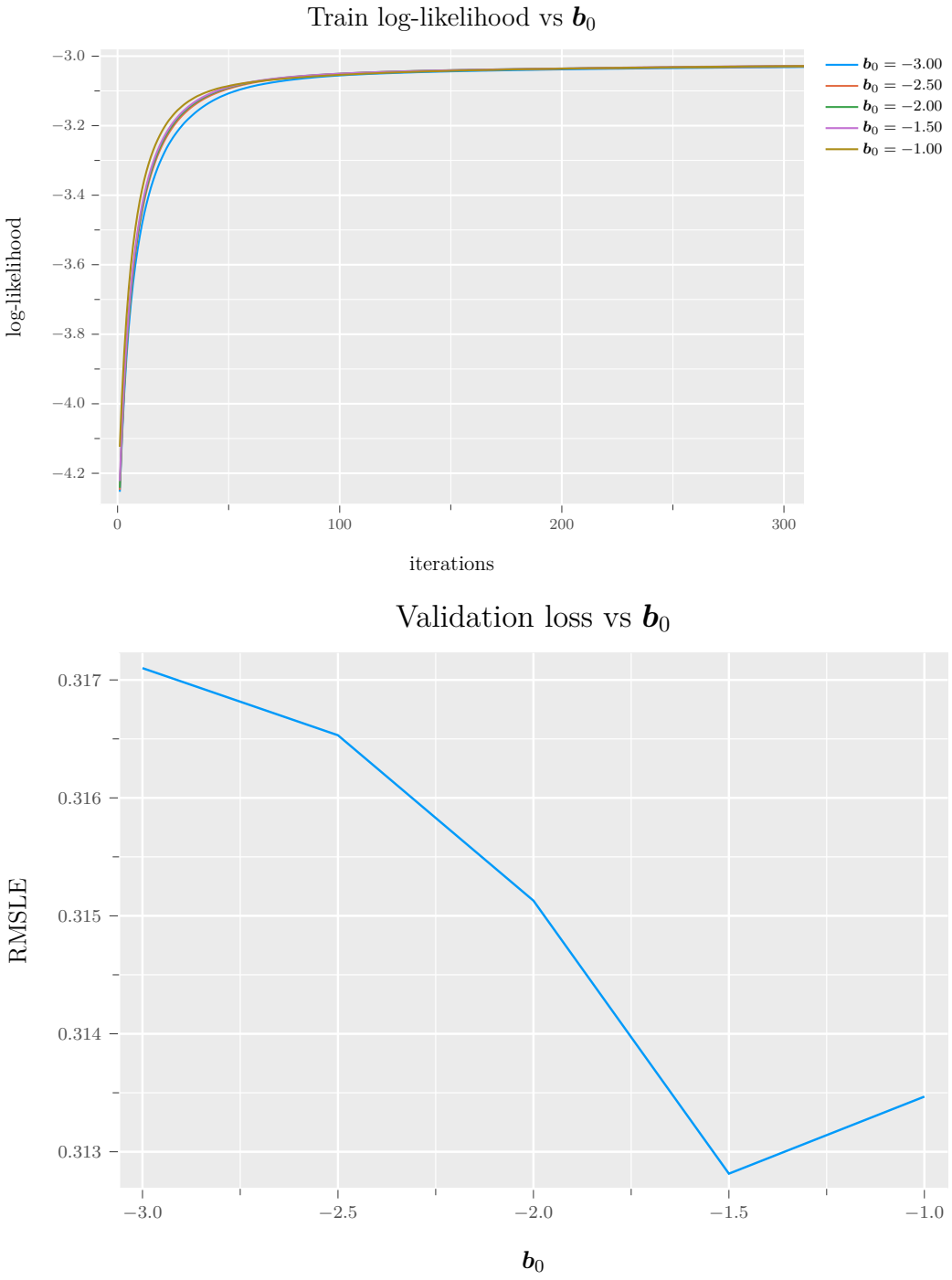


Figure A.2 – Sensitivity of our method to the choice of $\boldsymbol{b}_0$.

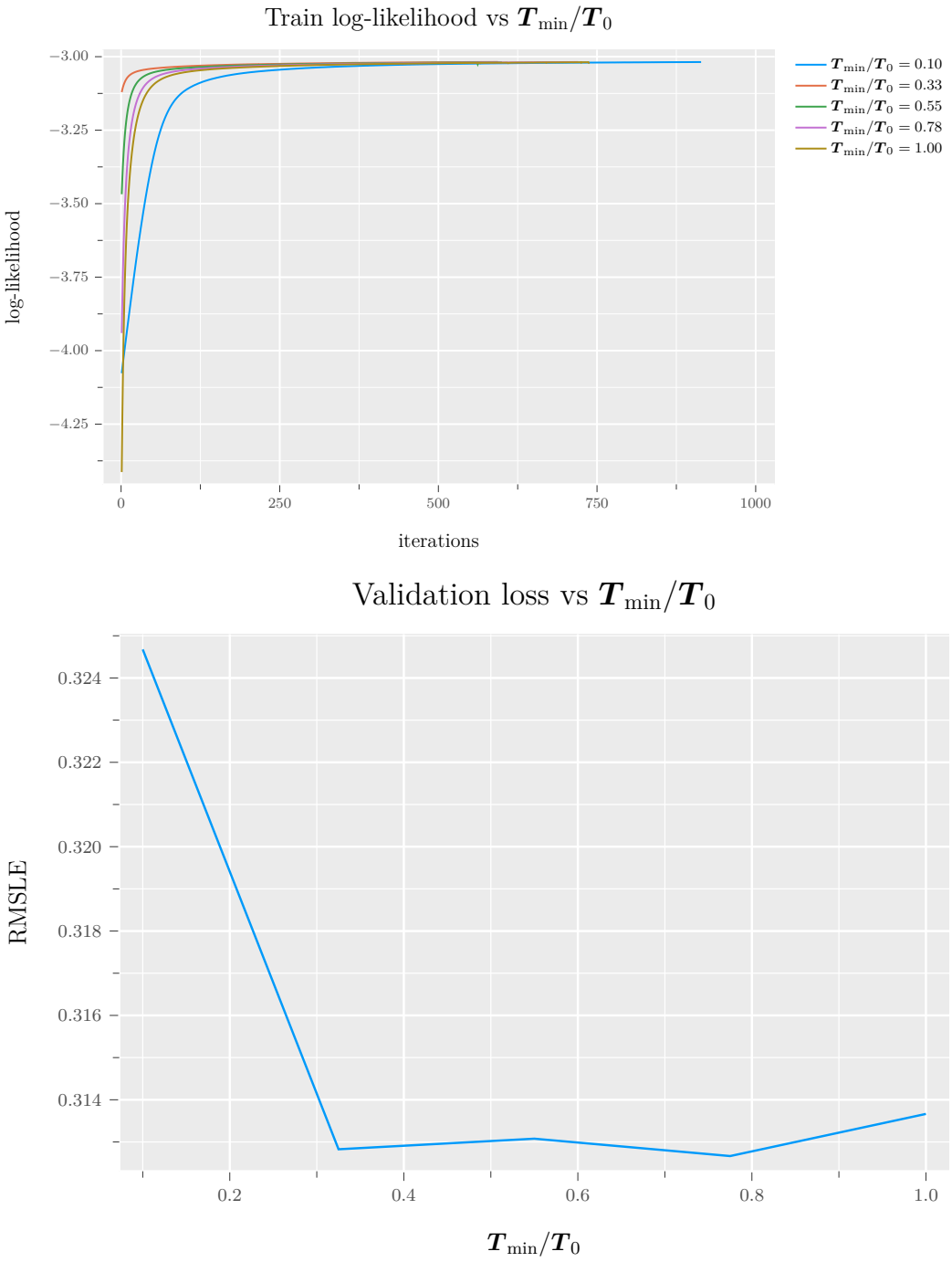


Figure A.3 – Sensitivity of our method to the choice of $\boldsymbol{T}_0$.

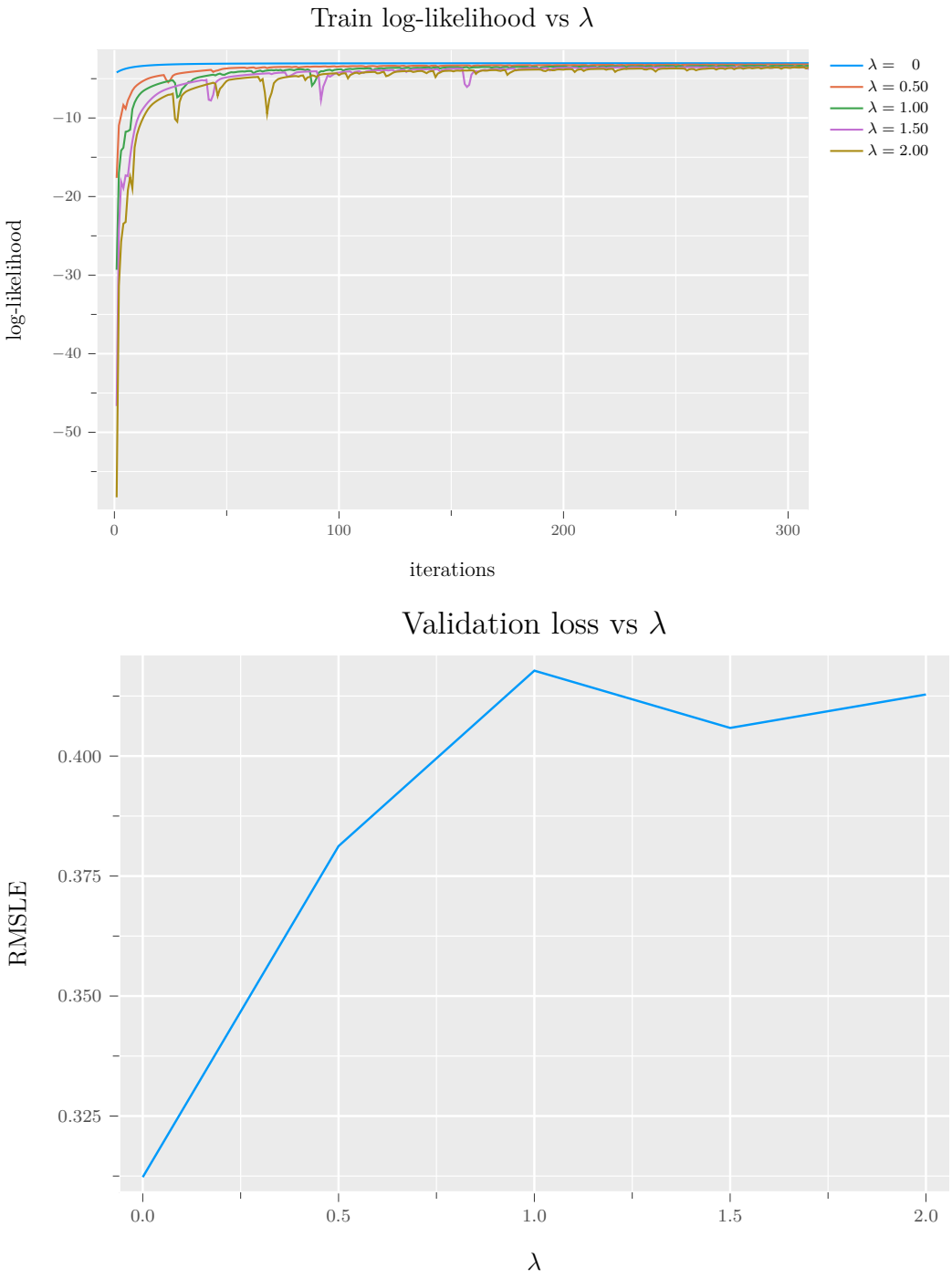


Figure A.4 – Sensitivity of our method to the choice of λ.

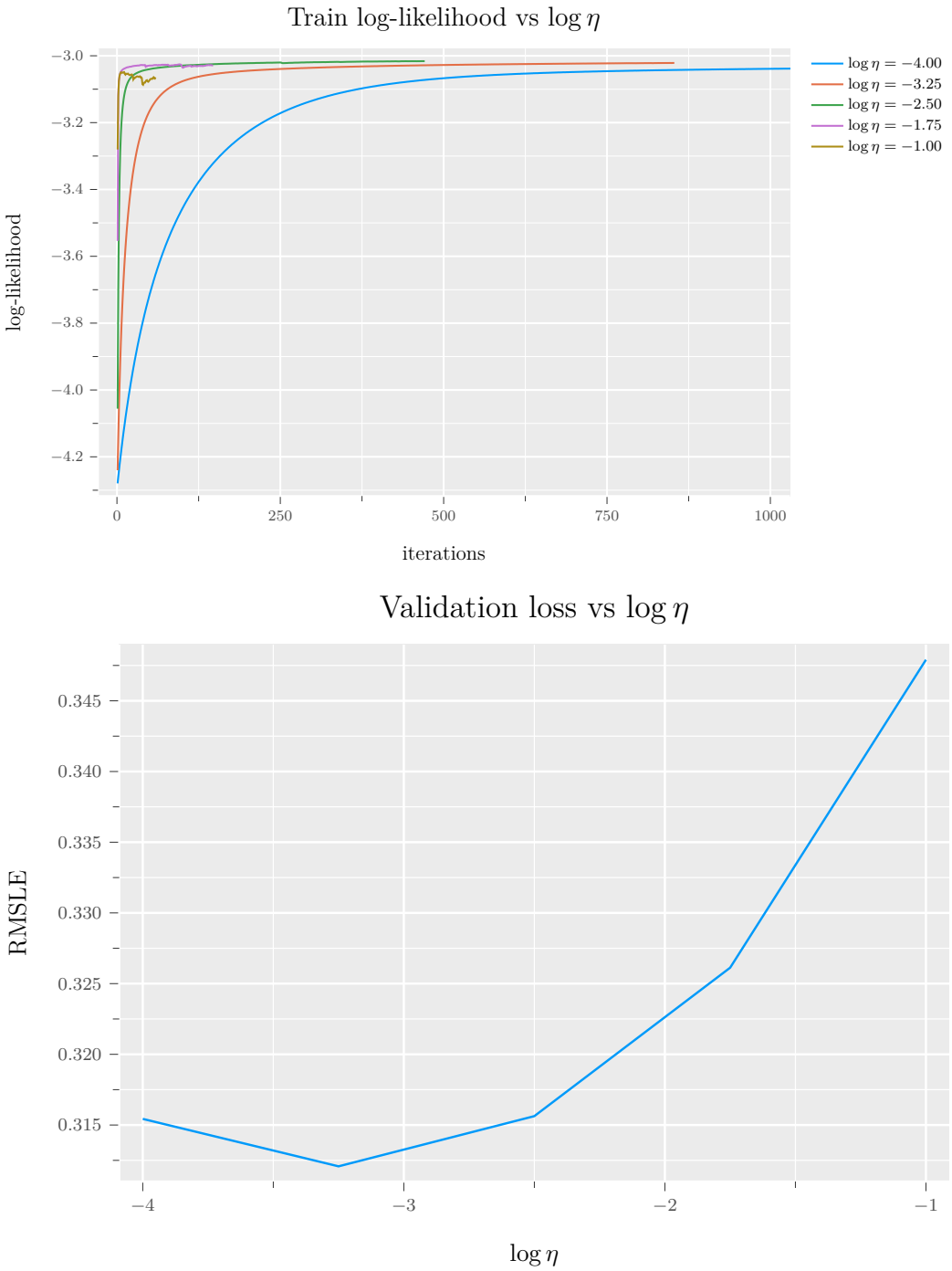


Figure A.5 – Sensitivity of our method to the choice of η. Logarithmic scale in the learning rate.

Chapter B

Appendix for Paper 2

B.1. Proofs – Regularized Stochastic Shortest Path

NIn this section we extend the proofs of (Bertsekas, 2012) to regularized stochastic shortest paths (R-SSP). While the original proof is for deterministic policies, we extend them to stochastic policies as deterministic policies may not even be in the domain of the regularizer (this is the case with entropy). As result of adding regularizer and working with stochastic policies, the original proofs do not hold. As a minor note, we consider a maximization problem whereas Bertsekas (2012) poses the problem as a minimization one.

Let T_π be the policy evaluation operator

$$T_\pi V = R + P_\pi V, \tag{B.1.1}$$

such that P_π is the transition matrix of policy π (i.e. $P_{\pi st} = \sum_{a \in \mathbb{A}(s)} P(t|s,a)\pi(a|s)$), and let T be the Bellman operator

$$TV(s) = \max_\pi \mathbb{E}_{a\sim\pi, s'\sim\mathbb{P}(s,a)}[R(s,a) + V(s')] - w^*(\pi). \tag{B.1.2}$$

Proposition 1. *(Bertsekas, 2012, Proposition 3.2.1, part a) For a proper policy π, the associated cost vector V_π satisfies*

$$\lim_{k\to\infty} T_\pi^k V = V_\pi,$$

Proof. The original proof holds with a modified reward function $R_\pi(s,a) = R(s,a) - \omega(\pi(s))$. Since the policy is proper, $\omega(\pi(s))$ is finite. □

Proposition 2. *(Bertsekas, 2012, Proposition 3.2.1, part b) A stationary policy π is proper iff*

$$V \leq T_\pi V$$

for some V.

PROOF. By Geist et al. (2019, Proposition i and ii, monotonocity of T_π) we have that

$$V \leq T_\pi V \leq T_\pi^2 V \leq T_\pi^3 V \leq \cdots \leq \lim_{k\to\infty} T_\pi^k V$$

holds. Since the value of π cannot diverge to $-\infty$, π has to be proper. Furthermore, the equation is trivially satisfied for any proper policy π as V_π, the value of π, exists and satisfies $V_\pi = T_\pi V_\pi$. So, $V \leq T_\pi V$ is satisfied for at least one V. □

Proposition 3. *(Bertsekas, 2012, Proposition 3.2.2, part a and b) The unique optimal cost vector satisfies the Bellman equation*

$$V^* = TV^*,$$

and we have

$$\lim_{k\to\infty} T^k V = V^*,$$

for any vector V.

PROOF. Let V and V' be two fixed points of T, let $\pi(a|s)$ be $\partial w(V(s)+R(s,\cdot))/\partial a$. Similarly we define $\pi'(a|s)$ as $\partial w(V'(s) + R(s,\cdot))/\partial a$. For all s, we have that

$$V(s) = \max_{q\in\Delta} \mathbb{E}_{a\sim q, s'\sim P(s'|s,a)}[R(s,a) + V(s')] - w^*(q) \tag{B.1.3}$$

$$= \mathbb{E}_{a\sim\pi, s'\sim P(s'|s,a)}[R(s,a) + V(s')] - w^*(\pi). \tag{B.1.4}$$

The last step holds from the definition of π and the WDZ theorem (McFadden, 1974; Feng et al., 2018; Geist et al., 2019). Since the (B.1.4) holds for all s, V has to be equal to the value of π or V_π and π is proper. We have that $V = T^k V \geq T_{\pi'}^k V$. By taking the limit when $k \to \infty$, we get $V \geq V_{\pi'} = V'$. By repeating the same process for π' we get that V and V' are equal.

To show that at least one fixed point exists, we start from V_{π_0} the value of a proper policy. We assume that such a policy exists in Assumption 11. Let π_1 be the policy computed from TV_{π_0} using the process used in the first part of the proof, we have that $V_{\pi_0} = T_{\pi_0} V_{\pi_0} \leq TV_{\pi_0} = T_{\pi_1} V_{\pi_0}$. By proposition 2, π_1 is proper. Using the monotonicity of T_{π_1} we have that $V_{\pi_0} \leq T_{\pi_1} V_{\pi_0} \leq T_{\pi_1}^k V_{\pi_0} = V_{\pi_1}$. We repeat this process to get a sequence of policies π_k and corresponding value functions V_{π_k}. The sequence of value function can either converge to a fixed point or diverge to infinity. We show that the value function has to converge.

Since π_ks are bounded vectors in $\mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$, by the Bolzano-Weierstrass theorem (Lafferriere et al., 2022, Section 2.4) there exists a policy π_l such that a subsequence of π_ks converges to π_l. Since the value of π_l, V_{π_l}, is bounded from below, it cannot be improper. Since π_l is proper, V_{π_l} exists by Proposition 1. Since the Bellman operator T is continuous, $TV_{\pi_l} = V_{\pi_l}$ holds and V_{π_l} is a fix point of T.

The remainder of this proof follows the one in Bertsekas (2012). □

B.2. Extended experiments

This section reports experiments with comparisons to GFlowNets (GFN) and SQL. Figure B.1 shows the average reward of the generated molecules, and Figure B.2 shows the number of modes found given a particular cutoff. Figure B.3 shows the size of the generated molecules (in terms of numbers of fragments). It seems that stopping the generation process early is a feature of mellowmax. Whilst this leads to smaller molecules, our performance measures do not reflect the synthesizability of these molecules. It is worth noting that the difference between normalized mellowmax and normalized SQL is equal to the temperature, while the gap between mellowmax and SQL is equal to the temperature times $\log |\mathbb{A}(s)|$. The small difference in target value between the two normalized models explains why they perform similarly.

As for the velocity of good candidate generation, after the initial burn-in period, our models manage to generate high-reward molecules. To our best knowledge, these are the first Q-learning-based model that manages to not diverge and continuously generate good and diverse candidates on the fragment design problem proposed by Bengio et al. (2021). Our results do not confirm that SQL diverges as claimed in Bengio et al. (2021); Malkin et al. (2022). However, SQL, especially the original version, comes with one caveat, it will diverge if not used in conjunction with any stabilizing method similar to Mnih et al. (2015).

We analyze the effect of the tree backup on the targets in Figure B.4. The results suggest that it plays a more important role early in the training process. It is worth noting that we train our model on the logarithm of the rewards. While early experiments showed that this is not important, it does change the hyper-parameters drastically. Lastly, the idea of using a large replay buffer comes from the idea that in order to generate diverse results, we need the Bellman equation to hold on to a very large space. The other details of the training scheme are the same as Malkin et al. (2022), save for learning rate, Huber loss (Obando-Ceron and Castro, 2020), exponential moving average targets that are reset to the Q-network every nth iteration, and gradient clipping. Unlike Obando-Ceron and Castro (2020), we found that the Huber combined with Adam (Kingma and Ba, 2014) performs better than both Huber combined with RMSProp (Hinton et al., 2012) and mean-squared-error combined with Adam. We clip the gradient by value and not by the norm.

B.3. Code

The code of the meolecule design experiment is hosted at `https://anonymous.4open.science/r/bias/RUNME.md`. The `RUNME.md` file contains the instructions to run the code after installing the package using `README.md`. We base our code on the repo XXXX. We ran the code on a cluster with RTX 8000 GPUs, 32GB of RAM, and eight cores; the runs took about 24 to 48 hours. The runs were also successful on V100-16GB GPUs. While we seeded

Figure B.1 – Average training reward. The right-hand-side is left-hand-side without error bars. Note the yellow line hidden behind the green and red lines.

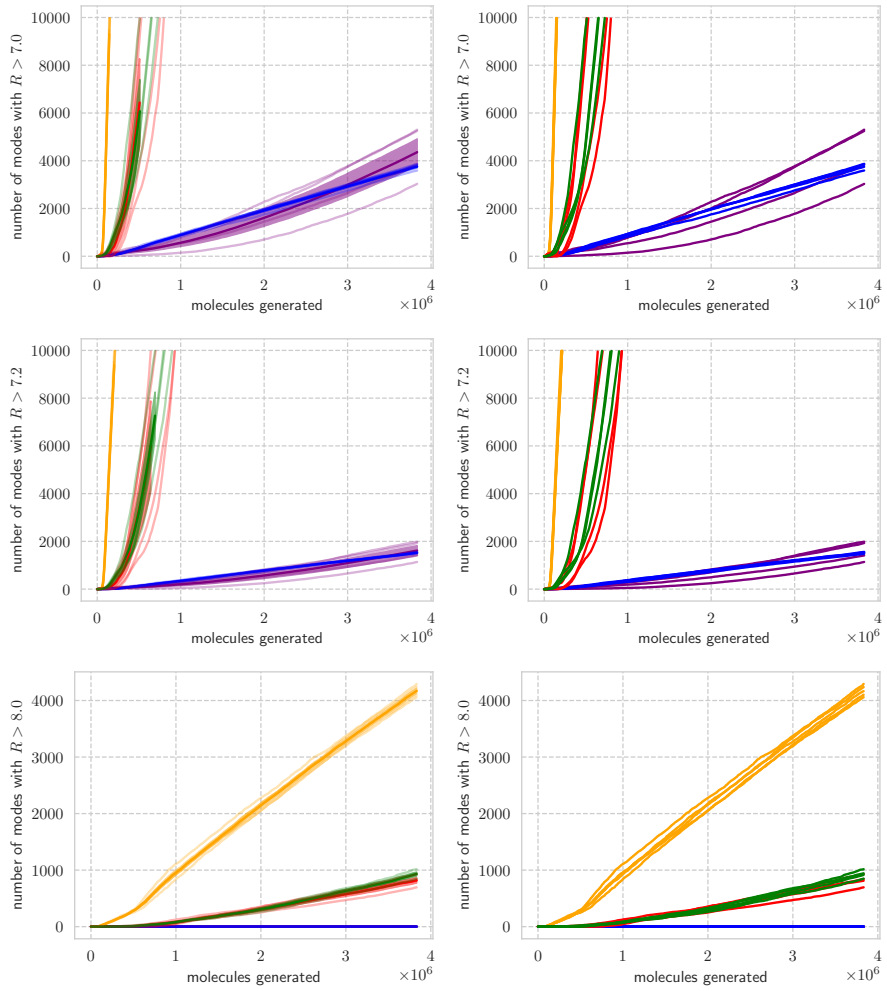
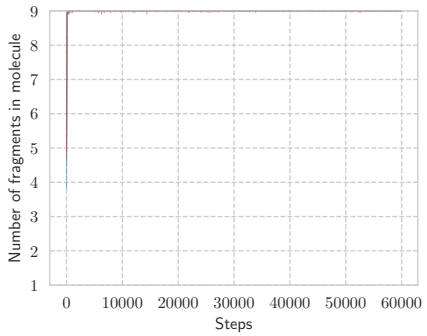
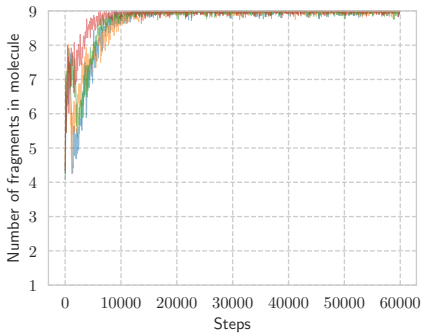


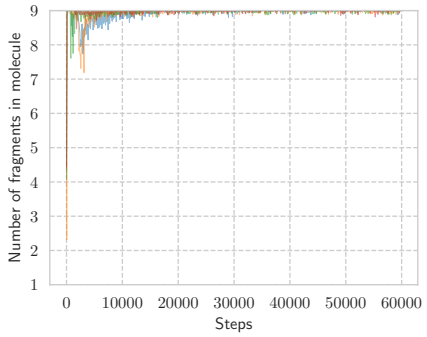
Figure B.2 – Number of modes generated at different levels. Yellow is GFN, green is normalized mellowmax, blue is mellowmax, red is normalized SQL, and purple is SQL. Right is left without error bars.



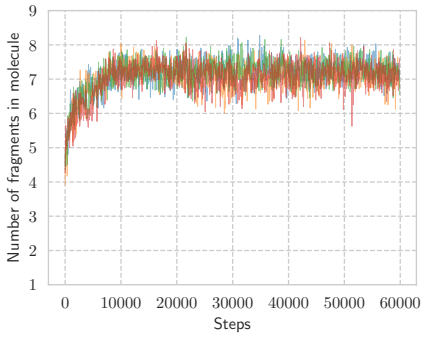
(a) Average trajectory length with trajectory balance.



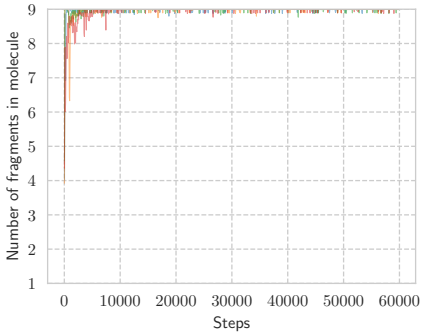
(b) Average trajectory length with normalized mellowmax.



(c) Average trajectory length with normalized SQL.



(d) Average trajectory length with mellowmax.



(e) Average trajectory length with SQL.

Figure B.3 – Length of trajectories. The largest molecule has nine fragments. Different lines are different seeds.

our experiments, the code suffers both from atomic operations, random numbers generated on GPU and multi-threading.

The code for the transportation example, including an implementation of Fosgerau et al. (2013) and its mellow version, is available at `https://anonymous.4open.science/r/jmdp-B30E/`. The code for the other figures is available at `https://anonymous.4open.`

(a) Effect of tree backup on normalized mellowmax.

(b) Effect of tree backup on normalized SQL.

(c) Effect of tree backup on mellowmax.

(d) Effect of tree backup on SQL.

Figure B.4 – Tree backup targets minus the original targets for different variants of SQL. Different lines are different seeds. Left-hand-side is the average difference, and the right-hand side is the variance of the difference.

`science/r/jmdp-41BE`. Both repos contain a `README.md` containing instructions needed to run the code.

B.4. Implementations

Table B.1 – 12 implementations of SAC that use state-dependent actions for terminations

URL
https://github.com/tinkoff-ai/CORL
https://github.com/vwxyzjn/cleanrl
https://github.com/denisyarats/pytorch_sac
https://github.com/haarnoja/sac
https://github.com/rail-berkeley/softlearning
https://github.com/sweetice/Deep-reinforcement-learning-with-pytorch
https://github.com/toshikwa/soft-actor-critic.pytorch
https://github.com/openai/spinningup
https://github.com/young-geng/CQL/blob/master/SimpleSAC
https://github.com/denisyarats/pytorch_sac_ae
https://github.com/thu-ml/tianshou/blob/master/tianshou
https://github.com/seungeunrho/minimalRL