FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Understanding Behavior via Inverse Reinforcement Learning

Miguel Ramos de Araújo

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Rosaldo Rossetti (Ph.D.)

Co-Supervisor: João Barros (Ph.D.)

July 20, 2012

Understanding Behavior via Inverse Reinforcement Learning

Miguel Ramos de Araújo

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

July 20, 2012

Abstract

The execution of complex activities, comprising sequences of simpler actions, sometimes leads to the clash of conflicting functions that must be maximized. We simultaneously try to fulfill the objective of the task and to maximize our own satisfaction, thereby impacting energy, time and social interactions during the activity. Further differences regarding the way we believe something shall be done create deeper distinction between our actions.

Decisions are motivated by a desire to maximize a given function of satisfaction and objectives, unknown even to the decision maker himself. Were we able to understand individual motivations and to compare such motivations between individuals, and we would be able to actively change the environment so as to increase satisfaction and/or improving performance.

In this work, we approach the problem of providing high-level and intelligible descriptions of the motivations of an agent, based on observations of such an agent during the fulfillment of a complex activity. An algorithm for the analysis of observational records is then proposed. We also present a methodology that allows researchers to converge towards a summary description of the agent's behavior, through the minimization of an error measure between the current description and the observed behavior.

This work was validated using a synthetic dataset representing the motivations of a passenger in a public transportation network and through a real dataset representing the motivations of taxi drivers during their trips on an urban network.

Our results show that it is possible to recover linear combinations of reward functions when the original function is a linear combination itself. The applicability of the presented algorithm to large datasets is also demonstrated.

Resumo

A execução de actividades complexas, compostas por sequências de acções simples, leva ao conflicto entre funções que devem ser maximizadas. Procuramos simultaneamente cumprir os objectivos da tarefa e maximizar a nossa própria satisfação, com impacto na energia e tempo que estamos dispostos a utilizar para a tarefa e nas interacções sociais que estabelecemos durante a mesma. Uma maior distinção entre as nossas acções surge derivada de divergências de opinião em relação ao modo como acreditamos como algo deve ser feito.

As nossas decisões são motivadas pelo nosso desejo de maximizar uma determinada função de satisfação e objectivos, desconhecida até para nós mesmos. Se conseguíssemos compreender as motivações individuais, e eventualmente estabelecer comparações entre indivíduos, seríamos capazes de aplicar alterações no ambiente em que actuamos, aumentando a nossa satisfação e/ou melhorando o desempenho.

Neste trabalho abordamos o problema de providenciar descrições inteligíveis e de alto nível das motivações de um agente, baseando-nos em observações deste durante a execução de uma actividade complexa. Um algoritmo para a análise de registos observacionais é proposto. Também apresentamos uma metodologia que permite aos investigadores convergir para uma descrição sumária do comportamento do agente, através da minimização de uma medida de error entre a descrição actual e o comportamento observado.

Este trabalho foi validado num conjunto de dados sintéticos representando as motivações de um passageiro de transportes públicos e num conjunto de dados reais, representando as motivações de taxistas durante as suas deslocações numa cidade.

Os resultados mostram que é possível recuperar combinações lineares de funções de recompensa quando a função original é ela mesmo uma combinação linear. A aplicabilidade do algoritmo apresentado em grandes conjuntos de dados é também demonstrada.

Acknowledgements

When I finished High School and enrolled in my master's program, I thought I would make a simple transition to a consulting company. However, in my first years I came to realize that scientific research offered the ingredients I was looking for in a career: intellectual challenge, room for creativity and independence, but just enough pressure to force progress and to make good ideas come through. I'm deeply appreciated to Professor Rosaldo Rossetti for giving me the chance to get to know this world, despite barely knowing me in 2009, and, three years later, for accepting to be my advisor. Notwithstanding the distance, his constant guidance, input and in-depth reviews were impressive. By this time next year, I am going to be abroad in a PhD program; I am especially thankful to Professor João Barros for introducing me to the program and for making my 3 months stay in Pittsburgh possible, where I got to know Siyuan Liu, PhD. This dissertation would not be real without his ideas, help and feedback. Credit must be given to him. Finally, Professor Teresa Galvão's help regarding linear programming formulations was invaluable.

My family is always here to show me what is really important. They do not know how to say 'no' and are always supportive; we ought to stay together, no matter how rough things are. My father, my mother, my brother and, more recently, my wife. The long journey ahead is not going to be easy, but the time abroad will only make our love grow stronger.

Miguel Araújo

"Behavior is what a man does, not what he thinks, feels, or believes."

Emily Dickinson

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

CMU	Carnegie Mellon University
CRISP	Cross Industry Standard Process for Data Mining
DRIVE-IN	Distributed Routing and Infotainment through VEhicular Inter-Networking
FCT	Fundação para a Ciência e Tecnologia
GPS	Global Positioning System
ICTI	Information and Communication Technologies Institute
IRL	Inverse Reinforcement Learning
LP	Linear Programming
MDP	Markov Decision Process
RL	Reinforcement Learning

Chapter 1

Introduction

Researchers have been exploring general machine learning topics such as Learning from Demonstration and Supervised Learning with a clear goal in mind: we want to build systems that, based on real world data, accomplish tasks as efficiently as possible. We have been trying to mimic human behavior in settings where we do not fully understand what is important and guides our actions. Examples include tasks such as "driving well" [AN04], controlling helicopters [ACQN07], manipulation of objects by robots and other similar problems such as pole balancing and equilibrium. These complex activities have in common the need for constant awareness of the environment and the fact that they are composed of smaller and simpler actions taken consecutively.

However, the increased performance our systems are achieving does not always translate into a better comprehension of the domain at hand. Although we are able to mimic behavior and performance, we still lack understanding and are unable to explain animal and human[1] behavior when performing complex tasks. Psychology, the study of the mind and behavior, tends do be eclectic, drawing on knowledge from other fields to help explain and understand psychological phenomena. On the other hand, some fields of Computer Science (e.g. data mining and branches of Artificial Intelligence) specialize on the knowledge discovery process. Given the amount of data regarding the fulfillment of activities, what if we were able to accurately describe and reason about behavior, given these observations?

Evidence suggests that reinforcements condition animal and human learning. Therefore, being able to describe the reward function used plays a vital role in understanding what we observe. A high level description would allow us to explain many complex animal and human decisions. We would know how bees weigh different factors like nectar ingestion, flight distance, time and risk from wind and predators [NR00]. We would also be able to characterize urban passengers based on their mobility patterns, understanding how they weigh factors such as travel time, travel cost, travel time variance and comfort.

[1]Humans are animals, please mind the redundancy.

In this dissertation, the problem of extracting high level information from a record of complex activities performed by an agent is explored. As referenced, the majority of the algorithms try to mimic behavior; this issue is also further developed.

This chapter intends to explain the full scope of the project, its motivation and the structure of this thesis. The methodological approach and the expected contributions are also described.

1.1 Problem statement

Our actions can be portrayed as an optimization problem. We are constantly trying to maximize our own reward function which involves physiological concerns, satisfaction and short and long-term goals and objectives. As a consequence, our decisions define a policy and our focus is on the analysis of such actions of the decision maker (also called *agent*). This terminology is further detailed on the next chapter.

However, more than learning the reward function used, we want to be able to synthesize the information it carries. If the complex and often stochastic reward function is simplified to an understandable representation, then the Inverse Reinforcement Learning (IRL) model later proposed can be used to extract knowledge and justify behavior, increasing the existing knowledge of the domain at hand.

As an example, consider the bee foraging problem. We are not interested in replicating the bee decisions, rather interested in creating models which might help understand their behavior so that beekeepers can modify the environment in order to increase production.

We define the agent's movements as a set of transitions in a Markov Decision Process (MDP). We would like to find the reward function (R) that forces the resulting optimum policy to closely match the observed behavior (O), while being described as a linear combination of different vectors. Therefore, the problem is finding the set of αs that best fit the agent's process, where each ϕ_i is a simple reward function which the domain expert considers relevant for this particular problem, i.e.

$$R = \alpha_1\phi_1 + \alpha_2\phi_2 + ... + \alpha_d\phi_d \tag{1.1}$$

Let π^O be the observed policy and π^R the policy corresponding to the reward function R, then R shall be such that minimizes the number of states in which both policies differ, i.e.

$$\sum_{s\in S}[\pi^O(s) \neq \pi^R(s)]. \tag{1.2}$$

1.2 Aims and goals

In order to solve the proposed problem a set of high level aims were established:

- The development of an algorithm capable of providing users with feedback regarding the fitness of reward functions to observational data. This algorithm shall be able to compare two particular functions regarding their contribution to an explanation of the observed behavior.
- The proposal of an iterative methodology for the study of behavior understanding problems, using the aforementioned algorithm.
- The validation of such algorithm and methodology using both synthetic and real world data, contributing to the respective domains.

The development and implementation of the algorithm and case studies is tied to the following low level goals:

- Implementation of a Markov Decision Processes solver.
- Implementation of the proposed Inverse Reinforcement Learning algorithm.
- Regarding the first case study:
  - Building of an artificial multimodal road network where passenger routing decisions can be tested.
- Regarding the second case study:
  - Implementation of a map matching algorithm to match GPS positioning records to a graph description of the city network.
  - Translation of the set of taxi positions into taxi driver decisions on the road network.

Further details are provided in the appropriate chapters.

1.3 Expected contributions

The expected contributions of this research are both theoretical and practical. On the one hand, it contributes to the domain of Inverse Reinforcement Learning with a novel formulation and validations on real-world data. On the other hand, the case studies analyzed contribute to the state of the art in their domain.

The expected contributions are as follows:

- A linear programming formulation to the Inverse Reinforcement Learning problem.
- A methodology for tackling behavior analysis problems.
- Experiments with synthetic data regarding the recovery of the passenger's reward function in multimodal urban networks.

- Experiments with real-world data studying taxi drivers preferences from GPS positioning records.

Furthermore, to the best of our knowledge, this is the first work providing computing algorithms that focus on the testing of high level behavioral hypothesis from datasets of human activities.

1.4 Document organization

This dissertation can be divided in two parts. It starts by introducing the reader to theoretical topics in the domain and subsequently exposes the author's work on how inverse reinforcement learning techniques can be applied to behavior understanding problems.

In a second part, two case studies illustrate an application of the work presented. There is included an explanation on how these case studies improve the overall sturdiness of this dissertation and the reasoning behind some of the more technological decisions.

This document is organized in six chapters, with references in the end.

Chapter 2 introduces the reader to some concepts and considerations about the domain in study. Markov Decision Processes (MDP), Reinforcement Learning (RL), Inverse Reinforcement Learning (IRL) and Apprenticeship Learning are contrasted and explained in some detail.

Chapter 3 describes the theoretical work, most notably a novel IRL formulation and the proposed methodology.

In Chapter 4, one can find the first case study, based on the passenger traveling problem and built upon synthetic data.

Chapter 5 presents the second case study, extracting information from real taxi-positioning records.

Finally, chapter 6 concludes this work, providing an overview of the results obtained, main findings and future research directions.

Chapter 2

Literature Review

Humans and animals continuously perform complex activities [1] trying to maximize their long-term reward when selecting each action. At any given moment, the agent is able to recall a finite amount of information from his past experience; therefore we consider that all these activities possess the Markov Property [2].

This chapter describes some of the most common frameworks used to analyze such processes and compares some common artificial intelligence topics important for the understanding of this work, such as Markov Decision Processes, Reinforcement Learning and Inverse Reinforcement Learning. A different section is dedicated to Apprenticeship Learning, a problem with some similarities with the one we are considering.

2.1 Markov Decision Processes

Named after Andrey Markov, Markov Decision Processes (MDPs) have been studied at least since the 1950s [Bel57]. They provide a mathematical framework for modeling the decision making process in an environment known to the decision maker. The environment is represented as a state space S and, on each state, the decision maker may choose an action a from a set of actions available in the current state. This discretization of the environment is usually accompanied with a discretization of time; therefore, a MDP is considered a discrete-time stochastic control process.

Upon the selection of an action, the process responds with a transition into a new state s', according to a state transition function $P_a(s,s')$. Therefore, the next state s' depends on the current state s and on the chosen action a, possessing the Markov Property. The agent also gets a corresponding reward $R_a(s,s')$. This reward function might also be characterized simply as $R(s,s')$ or even $R(s)$, depending on exactly how the problem is modeled.

[1]For the purpose of this dissertation, these are activities composed of sequences of actions

[2]A stochastic process has the Markov property if the conditional probability distribution of future states of the process depends only on the present state, not on the sequence of events that preceded it [Mar54] [Fel71].

An illustration of a sample MDP with three states and two actions per state can be seen in Figure 2.1. One can also find two rewards, +5 on the transition from state S_1 to state S_0 when action a_0 is applied and -1 on the transition from state s_2 to state s_0, when action a_1 is applied.

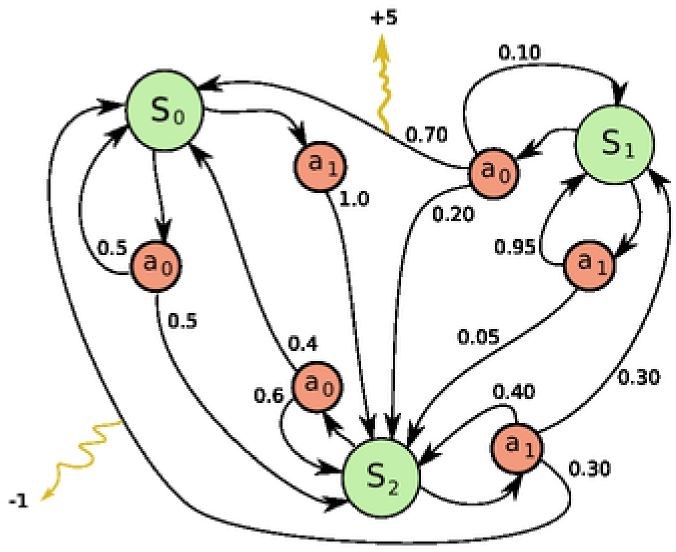


Figure 2.1: A sample Markov Decision Process [Mis12]

The main problem of MDPs is finding an optimum policy for the agent: a function $\pi : S \to A$ that maximizes a cumulative function of the rewards, typically the expected discounted sum over a potentially infinite horizon,

$$\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}), \tag{2.1}$$

where γ is the discount factor and usually satisfies $0 \leq \gamma < 1$.

Markov Decision Processes are a natural framework for modeling decision-making in situations where outcomes are partly random and partly under the control of the decision maker. They are used in a variety of areas including robotics, automated control, economics and manufacturing.

2.1.1 Notation, definitions and properties

As indicated in the previous section, MDPs are described by a tuple $< S, A, \{P_a\}, \gamma, R >$, where

- **S** is a set of states.
- **A** represents the set of actions applicable to the different states.
- $P_a(s,s')$ represents the transition probability from state s to state s', when action a is applied. As a consequence, $\sum_{s'} P_a(s,s') = 1, \forall s, s' \in S, a \in A$.

- $\gamma \in [0,1[$ is the discount factor.
- $R(s,s')$ represents the reward associated with the transition from s to s'.

A policy is defined as any map $\pi : S \rightarrow A$. The MDP problem is often characterized as finding the policy π that maximizes the reward obtained. The value function for a policy π at any state s represents the expected cumulative reward from state s and is given by

$$V^{\pi}(s_1) = E[R(s_1) + \gamma R(s_2) + \gamma^2 R(s_3) + ...|\pi], \tag{2.2}$$

or alternatively, $V^{\pi}(s_1) = \sum_{t=1}^{\infty} \gamma^{t-1} R(s_t)$,

where (s_1, s_2, s_3, ...) is the state sequence when we execute policy π starting from state s_1. The goal is thus finding a policy π so that $V^{\pi}(s)$ is maximized. It can be shown that there exists at least one optimal policy such that V is simultaneously maximized for all $s \in S$ [NR00] [BT96] [SB98].

$P_{\pi(s)}$ will be used to represent the transition probability matrix corresponding to the application of policy π. $P_{\pi(s)}(s,s')$ is the transition probability from state s to state s' when action $a = \pi(s)$ is applied to state s.

The action-value function for policy π, $Q^{\pi}(s,a)$, is defined as the value of taking action a in state s,

$$Q^{\pi}(s,a) = E[R(s,s') + \gamma V^{\pi}(s')|P_{\pi(s)}(s,s')], \tag{2.3}$$

where:

- $\pi(s)$ represents the action of policy π in state s.
- $V^{\pi}(s)$ is the value function for policy π in state s.
- $Q^{\pi}(s,a)$ is the action-value function.

For simplicity, we will often deal with *episodic* problems, in which *terminal states* are always reached, no matter what course of action the agent takes. Under these conditions, γ (the discount factor) might be 1 and the problem is called *undiscounted*.

A fundamental property of value functions used throughout reinforcement learning and dynamic programming is that they satisfy particular recursive relationships. For any policy π and any state s, the following consistency condition holds between the value of s and the value of its possible successor states. First identified by Bellman[Bel57], these are called the **Bellman Equations**:

$$V^{\pi}(s) = \sum_{s' \in S} P_{\pi(s)}(s,s')[R(s,s') + \gamma V^{\pi}(s')] \tag{2.4}$$

$$Q^{\pi}(s,a) = \sum_{s' \in S} P_a(s,s')(R(s,s') + \gamma V^{\pi}(s')) \tag{2.5}$$

The Bellman Equations average over all the possibilities, stating that the value of the start state must equal the (discounted) value of the expected next state, plus the reward expected along the way (see Sutton and Barto, *Reinforcement Learning: An Introduction*, 1998).

Similarly, the **Bellman Optimality** states that a policy $\pi : S \to A$ is optimal if and only if, for all $s \in S$, we have that

$$\pi(s) \in \arg\max_{a \in A} Q^{\pi}(s,a) \tag{2.6}$$

2.1.2 Algorithms

Given the Bellman equations and optimality, MDPs can be solved by linear programming or dynamic programming. MDPs are known to be solvable in polynomial time [PT87].

The linear programming formulation was first introduced in 1960 [Man60]. It uses S x A pairs of variables, $x_{s,a}$, representing the joint probabilities with which the state variable takes on the value of s and the decision value of a. More information can be found in the literature [Den70] [dGE67] [Put94].

On the other hand, dynamic programming variants work directly over two arrays indexed by state, obtained directly from the Bellman equations,

$$\pi(s) := \arg\max_{a} [\sum_{s' \in S} P_a(s,s')(R(s,s') + \gamma V(s'))], \tag{2.7}$$

$$V(s) := \sum_{s' \in S} P_{\pi(s)}(s,s')(R(s,s') + \gamma V(s')) \tag{2.8}$$

A dynamic programming implementation was necessary as part of this work; some of the most common variants will therefore be described.

2.1.2.1 Value Iteration

Value iteration may be considered the original version, proposed by Bellman himself in 1957 [Bel57].

Using this approach, the π array is not used, the value of $\pi(s)$ is calculated whenever it is needed. The two steps combined lead to the following expression

$$V(s) := \max_{a} [\sum_{s' \in S} P_a(s,s')(R(s,s') + \gamma V(s'))]. \tag{2.9}$$

This array is iterated until convergence.

2.1.2.2 Policy Iteration

Ronald Howard proposed an alternative variant which had the advantage of having a definite stopping condition [How60]. In this version, the π array is recalculated after finding a convergence of the values.

2.1.2.3 Prioritized Sweeping

Prioritized sweeping was later introduced [MA93] as a way to speed up policy iteration. The updating steps are preferentially applied to states which are considered important - either because there were changes around those states recently or because those states are of interest to the person using the algorithm.

2.1.3 Applications of MDPs

Being a mature modeling technique, Markov Decision Processes have been applied to many different areas. Among the many surveys of its applications, D. J. White published three articles analyzing some of the biggest application areas [Whi85] [Whi87] [Whi93].

Among the most common research areas, White identified *Inspection, maintenance and repair*, *Purchasing inventory and production* and *Water resources*. Other application areas include *Overbooking*, *Sports* as well as *Finances and investment*. Although this survey is 20 years old, this shows the wide range of applicability of the MDP framework.

More recently, other variants of MDPs have been explored, namely Partially Observable Markov Decision Processes with applications to dialog systems [WY07] and health sciences [HvBPM07]. This and others variants will not be described, as they do not provide any information necessary for understanding the work presented in this dissertation.

2.2 Reinforcement Learning

Reinforcement Learning (RL) can be described as a MDP whose reinforcement function is either unknown or highly stochastic. Furthermore, RL can be used in settings where the state space is considerably large, or even infinite, and thus exact methods are infeasible.

The scalar reward associated with the last transition is often directly obtained from the environment, which acts as an oracle, but might also be incorporated into the agent. In either case, it must be treated as a black-box. The problem changes into an online problem in which online performance is important and a proper balance between *exploration* and *exploitation* needs to be found.

In a reinforcement learning setting, the agent is concerned with maximizing the reward obtained given its imperfect knowledge of the reward function. As a consequence, the agent's focus is on performance and not on maximizing the knowledge it can obtain regarding the reward function.

2.2.1 Exploration

Exploration is the selection of unexplored or sub-optimal actions in order to improve the currently best policy. The exploratory method selected plays an important role in the learning process as randomly selected actions lead to poor performance.

Many researchers rely on *undirected* techniques [3] when performing exploration [McC97]. Among them, **ε-greedy** exploration is likely the simplest to understand. With this method, the agent selects what he believes to be the best answer with probability $(1-\varepsilon)$, and another action with probability ε. Typical values for ε vary widely.

A slightly more complicated **softmax** function is sometimes used instead. In this case, the function commonly used is [SB98]

$$\frac{e^{Q_t(a)/\tau}}{\sum\limits_{b=1}^{n} e^{Q_t(b)/\tau}}, \tag{2.10}$$

where $Q_t(n)$ represents the expected reward of action a at time t and and τ is a temperature parameter. For high temperatures ($\tau \to \infty$), actions have nearly the same probability and for low temperatures ($\tau \to 0$) the probability of the most-rewarding action tends to 1. Techniques to dynamically adapt the temperature are often employed.

On the other hand, it was proven that *undirected explorations* can cause the learning time to scale exponentially with the size of the state space [Wit91] and researchers have proposed *directed exploration* techniques in order to speed up the learning process. Different methods of directed exploration have been proposed, way beyond the limits of Reinforcement Learning, such as in the pathfinding domain [SKVS96].

Finally, issues such as the observability of the state need to be identified before selecting a proper exploration mechanism. In some problems, sensory limitations hide features of the environment from the agent, making different world states appear identical [LM93].

2.2.2 Exploitation

To obtain a lot of reward, a reinforcement learning agent must prefer actions that it has tried in the past and found to be effective in producing reward [SB98].

The problem of exploitation might be treated similarly to the identification of an optimum policy in a MDP. These *value function* approaches apply value iteration and policy iteration methods (or variants) over the whole state space. Computing such expectations is impractical for infinite or large state space MDPs.

Monte Carlo methods have also been applied with success in some areas. As an example, UCT [4][KS06] was responsible for a significant increase in the playing strength of computer Go programs over the last 5 years. Monte Carlo methods work by performing two steps in sequence: *policy evaluation* and *policy improvement*. In the first step, several runs of a given policy are performed in order to estimate the values of $Q^{\pi}(s,a)$. Subsequently, the current policy is improved based on the analysis of the Q function obtained.

[3] Approaches that do not use statistics from the learning experience to more efficiently guide the search.

[4] UCT stands for "Upper Confidence bounds applied to Trees" and is an extension to Monte Carlo search.

The problem with this methods is that they are often inefficient, as long trajectories are used to improve estimates only of the state-action pair that started the trajectory. Furthermore, the need for several runs implies that these methods only work in episodic, finite problems.

Temporal Differences algorithms have also had a great impact on Reinforcement Learning, combining Monte Carlo ideas and dynamic programming [SB98]. They learn directly from raw experience without a model of the environment dynamics and update estimates based in part on other learned estimates, without waiting for a final outcome.

Among them, **Q-Learning** [Wat89] is probably the most common. It acts similarly to *value iteration* updates, but the updates are done over a $Q(s,a)$ array using the following expression

$$Q(s_t,a_t) := Q(s_t,a_t) + \alpha_t(s_t,a_t)[R_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1},a_{t+1}) - Q(S_t,a_t)], \tag{2.11}$$

where action a_t is performed on state s_t and reward R_{t+1} is obtained. α_t is the learning rate $(0 < \alpha \leq 1)$.

2.2.3 Reinforcement Learning and behavior

When considering the goal of this dissertation and when compared to MDPs, Reinforcement Learning algorithms have the clear advantage of treating the reward function as an unknown that must be explored. They acknowledge that the policy to be followed and the reward function in use are inseparable and work towards a better knowledge of such function.

Nevertheless, reinforcement learning methods were developed under the perspective of the agent acting on the environment. When considering the behavior of a non-artificial agent (such as an animal), the reward function involves both the environment and the agent itself, whose actions we want to distinguish from those of its peers.

Inverse Reinforcement Learning, presented on the next section, attempts to model the reward function from the perspective of an observer of the agent's actions.

2.3 Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL), as first described by Russel [Rus98], deals with the problem of identifying the reward function being optimized by an agent, given observations of its activity. In his words, "It seems clear, however, that in examining animal and human behaviour we must consider the reward function as an unknown to be ascertained". He provided two "straightforward" reasons: firstly, the empirical hypothesis indicated for a given reward function may turn out to be wrong [5]. Secondly, multiattribute reward functions need to be calibrated and their parameters can not be pre-determined.

[5]Russel gave as an example the horses' gait selection for a given speed, which first was believed to be determined by energetic economy [HT81], which was later shown not to be true [FT91].

The **goal** of IRL is then to determine the reward function that an agent is optimizing, **given** observational records of an agent's decisions over time (behavior), a model of the physical environment (which might need to include the agent's body) and a measurement of the sensory inputs available to the agent.

In general, Inverse Reinforcement Learning can be seen as the dual problem of unsupervised reinforcement learning, whose task is to ascertain the optimum policy. However, this connection is certainly not bijective as a given policy can be optimal under several different reward functions (e.g. R = 0 is a trivial solution which makes *any* policy optimal).

More formally, we will consider a finite state space S, a set of actions A, transition probabilities P_{sa}, a discount factor γ and a policy π. We wish to find the set of possible reward functions R such that π is an optimal policy in the MDP defined by $(S, A, P_{sa}, \gamma, R)$. As the set contains many degenerate solutions, instead of finding the set of reward functions, we are interested in selecting the reward function that optimizes some other criteria.

2.3.1 Ng & Russel algorithms for Inverse Reinforcement Learning

Andrew Ng & Russel proposed a series of algorithms for the inverse reinforcement learning problem [NR00]. Using the notation previously described in 2.1.1 and representing functions as vectors indexed by state, their "main result" characterizing the set of solutions to the IRL problem is given by

$$(P_{\pi(s)} - P_a)(I - \gamma P_{\pi(s)})^{-1} R \succeq 0, \tag{2.12}$$

where R is a reward vector representing the function R(s) and $\succeq$ denotes non-strict vector inequality (i.e., $x \succeq y$ if and only if $\forall i : x_i \geq y_i$). Extensions to $R(s, s')$ are trivial.

Two problems can be identified: First, as previously mentioned, degenerate solutions such as $R = 0$ are part of the solution set (in fact, any constant vector or vectors arbitrarily close to 0 are solutions). Second, even after removing such degenerate solutions, it seems likely that more than one solution exists.

Values for R can be found as solutions to a linear programming problem, as the above inequality can clearly be expressed as constraints, but one can use the linear programming formulation to search for a more specific solution. Ng & Russel proposed that one should look for a solution that simultaneously makes π optimal and that makes single-step deviations from π as costly as possible.

Therefore, among all the functions R satisfying the inequality, we shall choose the one that maximizes

$$\sum_{s \in S} [Q^{\pi}(s, \pi(s)) - \max_{a \in A \setminus \{\pi(s)\}} Q^{\pi}(s, a)]. \tag{2.13}$$

This basic linear programming formulation allows one to select a single reward function defined by an array indexed by state, whose corresponding optimum policy matches the observed behavior.

2.3.2 Linearization of the reward function

While the above formulation allows one to find the exact reward array, in some situations it might be required that we linearize the reward function. On the one hand, the state space might be considerably large or even infinite (e.g. the state description contains a time component). On the other hand, our observations of the optimum policy might not take place over the whole state space (e.g. sampled trajectories or not all states are accessible from the initial state and thus are not observable) . In such cases, a linearization o the reward function is a possible solution [NR00], and R can be expressed as

$$R(s,s') = \alpha_1\phi_1(s,s') + \alpha_2\phi_2(s,s') + ... + \alpha_n\phi_n(s,s'), \tag{2.14}$$

where each function ϕ_i represents a possible reward function on the transition from s to s' and each α_i is a new linear programming variable whose values we want to figure out.

The new formulation draws from the main observation that in order for $\pi(s)$ to be optimum, on each state, the expected value of action $\pi(s)$ must be bigger than that of any other action.

If we define $E_{s'\sim P_a(s)}V^\pi(s')$ as the expected value of $V^\pi(s')$ over the distribution $P_a(s)$, then this relation can be described as

$$E_{s'\sim P_{\pi(s)}(s)}[V^\pi(s')] \geq E_{s'\sim P_a(s)}[V^\pi(s')] \tag{2.15}$$

and directly translated to

$$\sum_{s'} P_{\pi(s)}(s,s')V^\pi(s') \geq \sum_{s'} P_a(s,s')V^\pi(s'), \forall s,s' \in S, a \in A. \tag{2.16}$$

Due to the linearity of the reward function, V^π is also linear and given by

$$V^\pi = \alpha_1 V^{\phi_1} + \alpha_2 V^{\phi_2} + ... + \alpha_n V^{\phi_n}. \tag{2.17}$$

As each function ϕ_i is defined by the user, the values of V_i^π can be calculated using a common MDP solver. Therefore, the new linear programming formulation is defined by the constraints in 2.16 using αs as variables, as every other value is known.

Satisfiability and tie-breaking

Both the infinite nature of the state space and the linearization of the reward function might render this problem without satisfiable solutions, leading to its relaxation. Relaxation implies moving each inequality from the constraints to the maximization function, so that a solution is always possible.

Ng & Russel proposed that we should both respect as many constraints as we can and simultaneously look for the reward function which would penalize the agent the most in case he deviated from the observed policy, i.e.

$$maximize \sum_{s\in S_0} min_{a\in A\setminus\pi(s)}\{p(E_{s'\sim P_{\pi(s)}(s)}[V^{\pi}(s')] - E_{s'\sim P_a(s)}[V^{\pi}(s')])\} s.t. |\alpha_i| \leq 1.$$ [6]

The work presented in this dissertation builds upon this work of Ng & Russel.

A note on (Integer) Linear Programming Problems

Even though ILP is in general a NP-hard problem, several algorithms exist for solving integer linear programs, such as cutting-plane methods, branch and bound, branch and cut and branch and price.

Further information on this problem can be found in the literature, such as in *Advances in Linear and Integer Programming* [Bea96] or *Linear Optimization and Extensions* [Pad99].

2.3.3 Other algorithms for Inverse Reinforcement Learning

A similar problem to Inverse Reinforcement Learning has been studied in other fields under the name of Inverse Problem Theory, which is a theory on how to convert observed measurements into information about a physical object or system that we are interested in [Tar05]. Examples include estimating the Earth's gravitational field, which is determined by the density distribution of the Earth in the subsurface.

The closest work in economics is the task of *multiattribute utility assessment*, which studies how a person combines the various attributes of each available choice when making a decision [Rus98] [KR76].

Russel's extended paper on the subject introduced this topic to the computer science community and many ideas have been developed since. Ratliff, Bagnell and Zinkevich approached this issue as a maximum margin structured prediction problem over a space of policies [RBZ06]. They argue that the Maximum Margin Planning approach allows the demonstration of policies from more than a single MDP, demonstrating examples over multiple feature maps (their translation of MDP) and different start and goal states.

Deepak Ramachandran and Eyal Amir combine prior knowledge and evidence from the expert's actions to derive a probability distribution over the space of reward functions [RA07]. They use the actions of the expert to update a prior on reward functions using a Markov Chain Monte Carlo [Has70] algorithm.

In some recent publications, Brian Ziebart presented a maximum entropy approach [ZMBD08] which has been successfully applied to different problems, such as route prediction given a driver's past information, destination prediction given incomplete journeys [ZALMB08] and mouse movement prediction [ZDB12].

[6] p(x) is a penalizing function called whenever a restriction is broken. They selected p(x) = x if x $\geq$ 0, p(x) = 2x otherwise.

Both Ziebart's and Ramachandran's approaches try to deal with the problematic that demonstrated behavior is prone to noise and that the expert might not be infallible. A comparative evaluation of several methods (including maximum entropy and maximum margin) can be found in the literature [ZMBD08].

2.4 Apprenticeship Learning

Apprenticeship Learning is the task of teaching an agent (apprentice) to learn from an expert's demonstration of the "correct" performance on a given task. It is also known as *learning by imitation*[Bas99] and *teaching by showing*.

It differs from general unsupervised learning techniques, such as standard classification problems, because tasks are not single decision problems. In an apprenticeship environment, the activity to be learned is composed by a sequence of decisions towards an objective which might not be known to the learning agent.

In the context of human skill learning, it was applied to a complex manipulation task to be learned by an anthropomorphic robot arm [KGGW94]. Stefan Schaal investigated how data from demonstrations could benefit Reinforcement Learning methods (Q-learning, value-function learning and model-based learning) in both linear and nonlinear tasks (pole balancing). Researchers are wondering whether *imitation learning is the route to humanoid robots* [Sch99], given that, by definition, humanoid robots are not the most efficient way to perform a given task but rather an imitation problem.

Inverse Reinforcement Learning methods

The majority of the methods applied in apprenticeship learning settings try to directly mimic the demonstrator by applying a supervised learning algorithm to learn a direct mapping from states to actions [SHKM92] [AM02]. In 2004, Abbeel & Ng proposed Inverse Reinforcement Learning to tackle this task [AN04]. In fact, apprenticeship learning is one of the major source of interesting problems to IRL, given its clear-cut fit to these problems.

When applying an IRL algorithm to these problems, we are looking for a reinforcement function that more closely matches the observed behavior. This often involves finding each value of the $R(s,s')$ matrix. Abbeel & Ng approach involved approximating policies using a Monte Carlo method, but their formulation involved a quadratic programming problem instead of the linear programming problem explained in this chapter. They proved that their algorithm terminates in a small number of iterations.

Other approaches have been tried, such as Neu & Szepesvári junction of IRL and gradient methods [NS07]. They argue that their approach might work with less data, since it makes use of the knowledge of the model of the environment. A comparison of the two algorithms referenced can be found in Neu & Szepesvári 's work.

2.5 Summary

In this chapter, concepts and notation of Markov Decision Processes, Reinforcement Learning, Inverse Reinforcement Learning and Apprenticeship learning were described.

Markov Decision Processes algorithms look for optimum policies when reward functions are described and known in advance to the agent. Reinforcement Learning methods solve the same problem in settings where the reward function is unknown and exploration is necessary. Inverse Reinforcement Learning algorithms try to identify the reward function in use by an agent whose observations we analyze. Finally, apprenticeship learning problems try to imitate an expert decisions on a given environment; different algorithms may be applied to this task.

The most important algorithms for each task were also detailed, with especial attention paid to Ng & Russel 's algorithm to Inverse Reinforcement Learning which is used in this work.

Chapter 3

Understanding Behavior

This chapter is comprised of three main sections. In the first section, sample problems associated with behavior understanding are enunciated and its problematic is explored in greater detail. Subsequently, a new formulation for the linear programming problem is presented. Finally, the improved feedback from the new formulation allows for the proposal of a research methodology for tackling behavior understanding problems.

3.1 The problematic of Behavior Understanding

Understanding the behavior of an agent given records of its decisions is an issue strongly connected to the discovery of the reinforcement function that dictates the actions of such an agent. Consequently, and similarly to the problem of replicating the agent's actions, inverse reinforcement learning techniques are a promising tool in the analysis of behaviors. We might be interested in comparing two individuals performing a task (e.g. establishing the differences and/or similarities).

However, while the IRL approach plays an important part in solving the problem, its most obvious output (a $R_a(s,s')$ matrix) is not satisfactory because it is not intelligible. In order to overcome this difficulty, one would need to analyze such matrix so as to obtain higher level information, understandable by humans. Given the size of the reward matrix, we consider this to be a significant setback of the approach and a difficult research problem on its own.

Although not part of the original goal, the results obtained from the linearization of the reward function are a step towards a better comprehension of the underlying behavior. Through the analysis of the resulting αs one can determine whether a particularly simple reward function has either a positive or negative impact on the agent's decision process. Therefore, the proposed formulations (see Ng & Russel, 2000) allow us to have a rough idea on whether a given reward function might be part of the agent's decision function.

On the other hand, it would be interesting if we could quantify the relative importance that the agent attributes to each individual reward function. In practice, we would like the values of the αs to be meaningful when compared. For two reward functions i and j, our objective is to guarantee that when $\alpha_i > \alpha_j$, then the reward function i has a greater importance for the agent than reward function j. Unfortunately, importance is often volatile and heavily dependent on the circumstances. This requires reward functions to be ever more complex and the reward functions hypothesis space grows significantly. Further, importance is often difficult to measure analytically when using real data and the difficulty to validate the proposed models rises accordingly.

Finally, dependency between reward functions is difficult to ascertain. Two reward functions, although significantly different in nature and objective, might share similar optimum policies. This leads this and other models to believe the agent is valuing a specific reward function when, in fact, it is valuing a completely different one that shares the same actions. A clear and overly exaggerated example: on a MDP representing a city, the movements of a railfan[1] are certainly very similar to those of many passengers on their way to work. Their reasoning to take that path is very different but these differences can not be captured by our observation and their motivation might be wrongly identified. Further research on metrics to identify similarities between reward functions is necessary, maybe indirectly through the analysis of their equivalent optimum policies.

3.2 Sample Problems

3.2.1 Character motion behavior

The control of animated models has received great attention in the computer graphics community (character controllers). A realistic movement in an unknown environment is a challenging task when developers wish to provide an engaging and natural experience.

Seong Jae Lee and Zoran Popović [LP10] applied an Inverse Reinforcement Learning algorithm to a set of demonstrated obstacle avoidance behaviors. They demonstrated a normal style, a watchful style and a playful style to an IRL algorithm. They proceeded to replicate these styles in untrained environments using the obtained reinforcement functions.

What if we applied similar tools to players in interactive environments in order to characterize their behavior in different parts of the game? This could provide valuable feedback to game developers who would be able to understand how different game elements (such as the storyline, environment or color palette) impact the playful or cautious attitude of their players.

3.2.2 Poker players analysis and improvement tools

Poker is a card game of imperfect knowledge easily modeled as a Markov Decision Process, given the time discrete nature of the game and the finite number of actions on each state. Deception

[1]Also called rail buff, railway enthusiast or railway buff, is a person interested in the recreational capacity in rail transport

and bluffing are important characteristics of poker players and players regularly use information obtained in previous hands against the same opponent to guide their judgment in the current hand.

The most recent poker playing programs are capable of observing opponents and to construct models to dynamically adapt their actions to best exploit the opponents' play [BPSS98] [Dah01] [PRC$^+$08] [FR08].

What if we could use these knowledge to provide meaningful feedback to players so that their playing quality is improved? If exploitation of the player's weaknesses is possible (and the results presented in the previous articles support this conclusion), then there is knowledge that the player can use to improve his game.

Current analysis software statistically analyze and categorize decisions based on the current hand. An understandable player profile could be built using information from a whole playing session and analysis such as changes in behavior on losing streaks would now be possible. Furthermore, a more general psychological profile could potentially be created based on changes on the playing pattern over a poker session.

Even though this idea was presented instantiating poker, it may obviously be expanded to different games. As we are modeling the player, games of imperfect knowledge are the most suitable. Nevertheless, the performance of every playing program can be improved by exploring the weaknesses of the opponent and thus equivalent feedback can be provided in more traditional games such as chess.

3.3 A Mixed Integer Linear Programming Formulation

As described in the problem statement, we are trying to represent the reward function as

$$R = \alpha_1\phi_1 + \alpha_2\phi_2 + ... + \alpha_d\phi_d, \tag{3.1}$$

letting π be the observed policy and π^R the policy corresponding to the reward function R, then R shall be such that minimizes $\sum\limits_{s \in S} \pi(s) \neq \pi^R(s)$. We are also seeking algorithms that can provide experts with extra feedback regarding the fitness of the reward function to the observed behavior.

The biggest issue of the aforementioned linear programming problem is the satisfiability of each state's constraints, given by the relation of the optimum action and every other action on each state,

$$\sum_{s'} P_{\pi(s)}(s,s')V^{\pi}(s') \geq \sum_{s'} P_a(s,s')V^{\pi}(s'), \forall a \in A, s, s' \in S. \tag{3.2}$$

This is a problem which likely has no solution under the linearized formulation of the value function.

The proposal on this paper, in order to maximize the number of respected constraints, consists on the transformation of this problem into a mixed integer linear programming problem. A binary variable $C_{s,a}$, representing whether a given constraint is begin respected, is added and constraints are changed to:

$\sum_{s'} P_{\pi(s)}(s,s')V^{\pi}(s') - \sum_{s'} P_a(s,s')V^{\pi}(s') + M * C_{s,a} \geq 0, \forall s,s' \in S$, where M is an arbitrarily large constant.

$C_{s,a} = 1$ indicates that constraint *s,a* is being violated.

The problem is then changed to a minimization of the sum of the $C_{s,a}$'s, even though we decided to maintain the maximization of the distances proposed by Ng & Russel as a tie-breaker (this maximization is represented through the "minus" sign).

The formulation is then as follows:

$$minimize \sum_s \sum_a M * C_{s,a} - \sum_s \sum_a D(s,a), \tag{3.3}$$

such that

$$0 \leq \alpha_i \leq 1,$$
$$D(s,a) + M * C_{s,a} \geq 0, \forall s \in S, a \in A,$$

where

$$D(s,a) = \sum_{s'} P_{\pi(s)}(s,s')V^{\pi}(s') - \sum_{s'} P_a(s,s')V^{\pi}(s'). \tag{3.4}$$

Please remember that P_a is known, as it defines the structure of the MDP. Furthermore, V^{π} is a linear combination of the αs, which are the linear programming variables in this problem.

This formulation not only provides the user the optimal linear combination of the αs (as described), it also gives the user further feedback regarding which pairs *s,a* can not be explained with the current reward functions. This advantage is further explored in this chapter.

Modifications to the formulation

3.3.1 Limiting the values of the αs

Even though the proposed formulation is able to identify whether a specific reward function might play a part on the observed behavior, the ability to relate the proposed reward functions is missing. This can be achieved with the addition of a new constraint which also forces the existence of non-null rewards, namely

$$\sum \alpha_i = 1. \tag{3.5}$$

This new constraint, driving the solution space away from binary solutions (α_i either 0 or 1), might make the model disregard reward components whose contribution to the real reward function is very small.

3.3.2 Normalization to allow relative comparison

Given that we are trying to ascertain the relative importance attributed to each reward function, relative values of the αs are important. As such, normalization of the reward function's output is necessary in order to maintain correct equivalence.

A simple z-score normalization (also called standard score) over the ϕ functions was chosen. The output of each reward function is then transformed according to

$$\phi'(s) = \frac{\phi(s) - \mu}{\sigma} \tag{3.6}$$

The output of the reward function is transformed by subtracting the mean of the population value and dividing by its standard deviation.

3.3.3 Decreasing the number of binary variables

The presented number of binary variables is proportional to the number of states and actions. This formulation allows for a fine grained control over which state-action pairs produce an output different from the observed. However, this increase in the number of binary variables leads to a significant increase in the time complexity of the MILP solver.

One possible solution is to use a single binary variable per state, turning $C_{s,a}$ into C_s. This way, we are indicating that we are not interested in maximizing sub-optimal solutions on each state; whenever a single action outperforms the action of the generated policy, the corresponding C_s is immediately 1 and the MILP algorithm adapts accordingly.

3.4 A Methodology for Behavior Understanding

The advantage of the MILP formulation over the initial Linear Programming formulation is that more feedback is produced. Given a set of reward functions, not only is the number of violated restrictions asserted, the user is also able to find out exactly what state-action pairs are being violated in the MILP solution. He is then able to use his domain knowledge to add a new reward function to the reward functions set. We are able to extract a methodology for behavior understanding. Figure 3.1 and Algorithm 3.1 are two representations of the same procedure.

Understanding Behavior

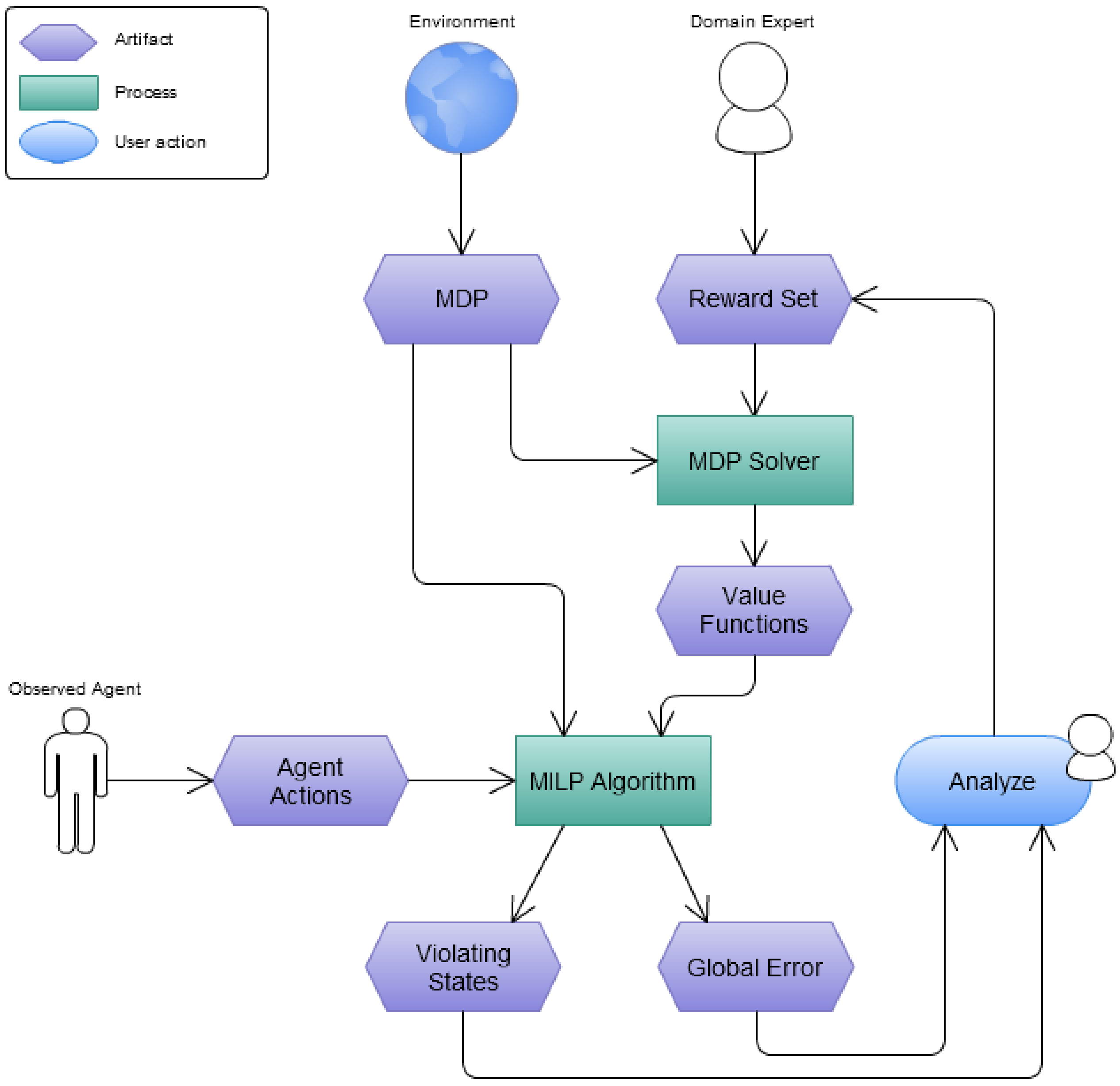


Figure 3.1: A methodology for Behavior Understanding

Algorithm 3.1 Behavior Understanding

```
Require: Environment e, ActivityLog al, $\mathbb{R}$ Threshold
 1: Let $mdp$ be $< S,A,\{P_a\} >$
 2: Let $RewardSet$ be a set of maps $\{S \rightarrow A\}$
 3: Let $ValueFunctions$ be a set of maps $\{S \rightarrow \mathbb{R}\}$
 4: Let $Constraints$ be a map $S \rightarrow A$
 5:
 6: $mdp \Leftarrow ParseEnvironment(e)$
 7: $Constraints \Leftarrow ParseDecisions(al)$
 8: $RewardSet \Leftarrow InitialRewardFunctions()$
 9: $ValueFunctions \Leftarrow \{\}$
10: for all $rf \in RewardSet$ do
11:     $ValueFunctions \Leftarrow ValueFunctions \cup MDPsolver(mdp, rf)$
12: end for
13:
14: repeat
15:     Let $ViolatingStates$ be a set $\{S\}$
16:     $\{\alpha\}, \varepsilon, ViolatingStates \Leftarrow MILP(mdp, Constraints, ValueFunctions)$
17:     if $\varepsilon > Threshold$ then
18:         $rf \Leftarrow GetNewRewardFunction(\varepsilon, ViolatingStates)$
19:         $RewardSet \Leftarrow RewardSet \cup rf$
20:         $ValueFunctions \Leftarrow ValueFunctions \cup MDPSolver(mdp, rf)$
21:     end if
22: until $\varepsilon < Threshold$
```

Firstly, the environment and the agent's decisions must be translated as a Markov Decision Process. This implies defining that state space, possible actions on each state and corresponding transition probabilities.

A correct definition of the MDP is independent of the particular agent to be analyzed and should be as complete as possible. Simplifying (reducing) the state space is essential to increase performance, but the state space chosen impacts the reward functions that can be implemented.

Secondly, using the observational records, the agent's decisions are obtained from the activity log, an observation of the agent's movements. There are translated as movements in the MDP and define a set (S_i, A_i), denoting that the agent applies action A_i when on state S_i. Ambiguities might arise while parsing the raw data from the logs. In that case, one must consider whether the state space needs to be redefined or if these can be attributed to observational error.

Thirdly, an initial reward set of simple reward functions is created. These are functions which the domain expert believes might be part of the agent's reward function. A MDP solver is then applied to each function in the reward set, obtaining the corresponding Value Functions.

The MILP algorithm is applied, using all the previously obtained information (the defined MDP, agent decisions and Value Functions) and outputing the set of αs, the global error ε and a set of violating states. The global error represents the percentage of states that can be explained with the currently defined reward functions. It is compared to the pre-defined threshold (representing a percentage of the states that we wish to cover) and either a new reward function is defined or computation may end.

Finally, in order to define a new reward function, the expert uses the feedback provided by the MILP algorithm. Is the global error acceptable? Can it be attributed to artifacts in the data? Why are the functions in the reward set not able to describe the behavior on some of the states? After it has been defined, the value function is obtained from the MDP solver and a new iteration of the process is performed.

3.4.1 Complexity analysis

We consider the definition of the Markov Decision Process and the definition of the agent's decisions to be a process external to the algorithm, as it involves parsing environment description files and activity logs, which may be defined in many different formats.

MDPs can be solved through Linear Programming or Dynamic Programming, with the later formulation the most common. While cycles in the MDP definition might render the process stochastic (thus it might not terminate), we are led to believe that they are avoidable in most realistic scenarios through the introduction of a time component in the state space. If each action leads to a state transition in which the time component is higher in the destination state than in the source state, then we are dealing with an acyclic graph and the dynamic programming approach runs in polynomial time. In this case, the time complexity is $O(|S| * |A|)$.

On the other hand, 0-1 integer programming is one of Karp's 21 NP-Complete problems [Kar72]. However, integer programs with a constant number of variables may be solved in linear time as a LP-type problem [2] [MSW96] [AE05]. This may be solved by a combination of randomized algorithms in an amount of time that is linear in the number of elements defining the problem, and subexponential in the dimension of the problem.

Finally, the number of iterations necessary to reach a global error below our pre-defined threshold depends on the reward functions proposed and, therefore, can not be estimated.

3.4.2 Automatically finding better reward functions

The problem of obtaining the original reward function is that we are not simply interested in replicating it, we rather need it to be meaningful and to carry high level information in the particular domain. This implies that having a computer program dictate an error-free reward function is not useful at all, even though it is possible.

On the other hand, computers are able to tune the reward functions given. More specifically, they are able to test whether slight variations of the reward functions might better explain the

[2] also called generalized linear program

observed behavior (i.e. reduce the error). Each reward function R on the reward set can be adapted through a function similar to

$$\alpha_i * R_i = \alpha_{i_0} * R + \alpha_{i_1} * R^2 + \alpha_{i_2} * ln(R) + \alpha_{i_3} * e^R. \tag{3.7}$$

This new reward function can be tested in place of the original and feedback regarding these αs can be given to the user whenever they are different from 0.

3.5 Technological decisions

The implementation of this methodology is important for the development of the case studies presented in the following chapters. The testing procedure can clearly be divided in nearly independent actions which communicate through a set of artifacts. A *Pipes and Filters* software architecture was then used, and the following major processes were implemented:

- **Markov Decision Process Solver**
  A MDP Solver is necessary to translate the reward functions (defined by the user) in a Value Function array (to be used by the MILP algorithm). The implementation in C++ followed a variant of the policy iteration method first described by Howard [How60].

- **Mixed Integer Linear Programming Modeling**
  Many implementations of MILP algorithms are freely available to be used. OpenOpt [Kro] is a numerical optimization framework for the Python [pyt] programming language; it provides solvers for both linear and non-linear problems. The package *lpSolve* was used by OpenOpt to solve the mixed integer programming problems.

The list of artifacts produced is quite extense (by-products of the whole process), including MDP definition files, observed policy files, value functions indexed by state files and violating states per iteration lists.

3.6 Summary

The problem of behavior understanding is illustrated with two sample problems which, as far as the author knows, have not been explored in the literature.

A new MILP formulation and a methodology for this problem are proposed, drawing from the state of the art presented in the previous chapter. Several variations and insights into the application of these proposals are provided.

In the following chapters, two case studies illustrating the application of these techniques are detailed. The following chapter, in particular, describes the methodological approach used for tackling the problems posed in the two case studies.

Chapter 4

Case Study I : Passenger motivations

In order to validate the algorithm presented in the previous chapter, a case study regarding public-transportation passengers' motivations was developed. A synthetic case study is essential because the reward function of the agent can be pre-defined and can be compared to the output of the MILP formulation. Therefore, the **primary objectives** of this case study are:

1. To assert whether the proposed algorithm is able to recover the original coefficients of a linear combination of reward functions.

2. To validate the implementation of the MDP solver and of the MILP algorithm.

4.1 Problem statement

Several different means of transportation are available in today's urban environment including buses, metros, taxis and individual transportation such as cars and walking. On a single journey, travelers often use more than one mode of transportation - we call these trips multimodal. In this study, we consider the problem of understanding passenger choices in their individual travels, restricting our discussion to buses, taxis and walking.

Passengers are driven by distinct desires: they may want to minimize monetary cost, minimize time, maximize comfort, minimize outdoor exposure, minimize travel time variance, minimize distance traveled, maximize the number of touristic landmarks they see, etc. By understanding how passengers value each individual function, city planners would be able to make more informed decisions and route navigation software would be able to provide travelers with better recommendations.

4.2 Contextualization - Multimodal routing and route advisory systems

While more and more car drivers are assisted by personalized and advanced guiding systems, traveling assistance is often limited or inexistent when travelers use other modes of transportation.

In urban networks, passengers who use public transportation systems may board buses, metros or even walk from one place to another. This multimodal transportation is often restricted by strict schedules and unpredictable delays. Travelers are constantly challenged by the changing environment, the increased complexity of the network and the lack of adequate information and guidance. Therefore, research is being conducted to adequately integrate existing information systems [BBM06], providing travelers with a single and ubiquitous source of information for their daily travels. As an example, information regarding delays from buses and taxis can be shown side by side with shortest walking path and estimated arrival time if one decides to take the metro.

As part of this effort, typical barriers multimodal travelers face have been identified, including the lack of public transport information in cars and other mobile platforms, the complexity of existing public transportation systems, the difficulty travelers face in transfers and the lack of end-trip and return-trip information [RBM07].

Consequently, route advisory systems plan to guide and help users through multimodal journeys. Among their objectives, they intend to integrate personalized multimodal trip management capacities, continuous on-trip access to multimodal trip data, personal orientation and guidance along trip sections and in transfer situations and interoperability of travel information systems [RBM07].

Due to the heterogeneity of the public transportation network, route advisory systems shall be aware of each individual passenger's preferences regarding travel time, price and comfort. The analysis of passenger decisions could prove a valuable source of information in order to improve the routing suggestions of this software.

4.3 Experimental setup

A small but densely connected synthetic network was created. The network layout allowed different possibilities between the chosen origin-destination pair, maximizing the number of decisions individual travelers may face. The road network was described as a Markov decision process. In order to reduce the state space, each state S is characterized by the passenger's current location. Other state descriptions are possible and a more comprehensive one would allow for different reward functions to be created (e.g. a state which included the total money spent would allow the creation of a reward function establishing the fastest route up to 2 monetary units). Both location and rewards were considered to be discrete. Figure 4.1 shows the structure of the network used. Nodes A and P are the origin and destination, respectively.

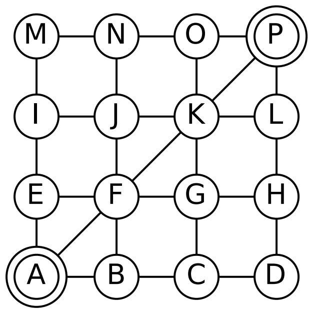
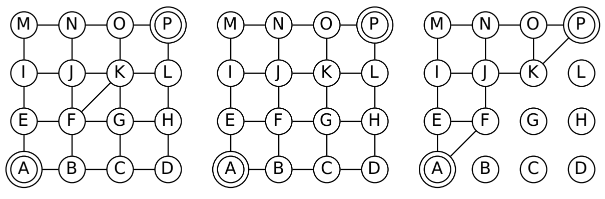


Figure 4.1: Network layout used

On each state, the passenger must select an action a, corresponding to a particular mode of transportation. The effect of the selection is determined by a function $P_a(s,s')$ which indicates the probability of transitioning from state s to state s'. In this specific setup, the transition is always made to the same state (i.e. values in the transition probability matrix are binary). Nevertheless, more than one action corresponding to a given mode of transportation might be possible on a given state (e.g. two different walking directions). Figure 4.2 shows the possible movements when using the Taxi, Bus or Walking. These connections were selected so that different reward functions generate different policies.



(a) Taxi (b) Bus (c) Walking

Figure 4.2: Available connections per mode of transportation

It was considered that every connection has a travel distance of 1. Walking takes 10 time units per transition, taxis take 1 time unit and buses take 2 time units. Further, taxis charge 3 monetary units per transition, while buses charge 2 monetary units regardless of the distance covered.

4.4 Observed behavior

The synthetically-created observed behavior is illustrated in Figure 4.3. It consists of a mapping $S \rightarrow A$, indicating the action selected on each state. Given the deterministic nature of the MDP (described in the previous section), Figure 4.3 shows the destination of the selected action. The mode of transportation was not included for simplification purposes.

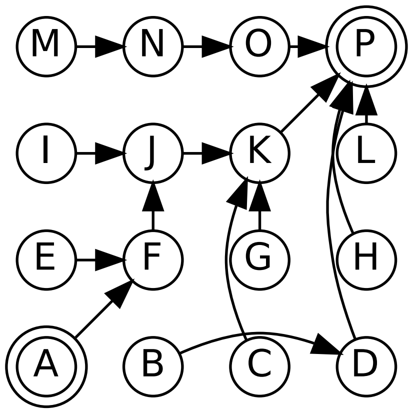


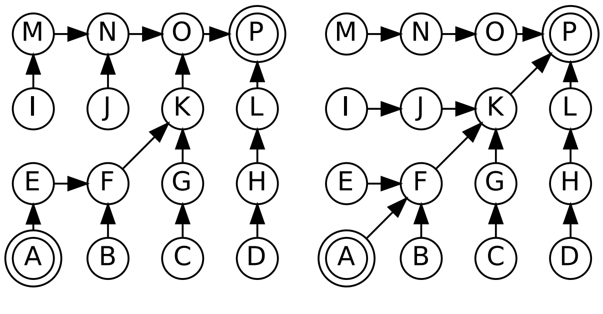
Figure 4.3: Observed behavior

4.5 Methodology application

This section will simulate the application of the devised methodology to the passenger motivations problem. Although this is constructed, plausible reasons for the proposal of each new reward function are provided.

Reward functions

Two of the most common reward functions are probably the time penalty reward function and the distance traveled penalty reward function. They penalize the agent for the time and distance of the journey, respectively. These two reward functions were considered to be the initial Reward Set; possible optimum policies for each reward function can be seen in Figure 5.1 [1]. Please note that the algorithm works with expected values per state, V_s, and not with optimum policies.



(a) Time (b) Distance

Figure 4.4: Optimum policies of the time and distance reward functions

[1]Given the definition of the MDP, other policies can equally be considered optimal under these reward functions. MDPs extracted from real data are less likely to have this property.

This reward set was fed to the MILP. The resulting optimal reward function coefficients are $(\alpha_{time}, \alpha_{dist}) = (0, 1)$ with $\sum_{c \in C} c = 8$, with states B and F not respecting the observed behavior in, respectively, 7 and 1 actions.

An analysis of the violating states would then be performed by a domain expert. In this case, we could take notice that, in the observed behavior, the transition from F is not by taxi to K but rather walking to J first. This might indicate a penalty based on the monetary cost. This reward function was therefore added; one of the optimum policies associated with cost minimization can be seen in Figure 4.5. Whenever possible, the agent walks; otherwise, he takes the bus.

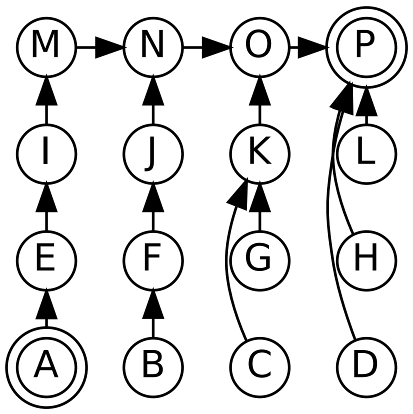


Figure 4.5: An optimum policy of the cost function

A new iteration of the MILP algorithm with this new reward functions set provides: $(\alpha_{time}, \alpha_{dist}, \alpha_{cost}) = (0, 0, 1)$ with $\sum_{c \in C} c = 7$. The cost function was able to better describe this behavior, with state B still to explain.

It might be noted that the agent prefers the right states whenever they seem interchangeable in the cost function. That right region could, e.g. represent a safer area or provide a more beautiful view by the river. A new reward function was added that penalizes agents the closer to the left they are. We assume that this is a reward function that provides meaningful feedback to the user analyzing the behavior. Once again, one optimum policy generated by this reward function can be seen in Figure 4.6.

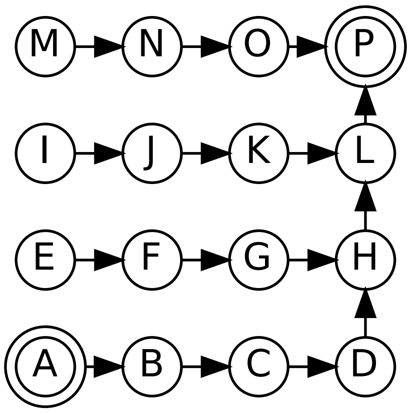


Figure 4.6: An optimum policy of the safety function

Solving the new MILP formulation provides new results: $(\alpha_{time}, \alpha_{dist}, \alpha_{cost}, \alpha_{safety}) = (0, 0, 0.5, 0.5)$ with $\sum_{c \in C} c = 0$. This implies that the resulting linear reward function is fully able to describe the observed behavior.

4.6 Results and Analysis

The reward function of the presented example was created based on a linear combination of 2 simple reward functions; the MILP formulation was able to completely obtain the original reward function. However, it might not always be the case, as a linear combination of the reward functions is not equivalent to a linear combination of the expected values, which is what the MILP formulation uses.

Please note that this is both a small network and that the reward functions presented are quite similar. Bigger networks and radically different reward functions will reduce the errors of the formulation.

As a further test to this case study, we wanted to test the ability to recover the original (single) reward function when presented with several possibilities. The results can be seen on Table 4.1.

Table 4.1: Inverse Reinforcement Learning on the simple reward functions

Original Function	α_{time}	α_{cost}	$\alpha_{distance}$	α_{safety}
Time	1	0	0	0
Cost	0	1	0	0
Distance	0.(333)	0	0.(592)	0.(074)
Safety	0	0	0	1

It can be seen that the IRL is often able to determine the original reward function. Possible variations are due to the fact that slightly different reward functions might explain the observed behavior. While the second (least important) term of the minimization function tries to penalize deviations from the observed behavior, it does not imply that we are closer to the real reward function.

4.7 Summary

This case study shows the ability of the MILP algorithm to obtain the correct coefficients of the reward function when it is a linear combination of simpler functions.

The application of the proposed methodology illustrates how information regarding the violating states can be used to guide the user towards a more precise reward function. The next chapter presents a real data case study which intends to analyze different characteristics, such as efficiency and error under realistic circumstances.

Case Study I : Passenger motivations

Chapter 5

Case Study II : Taxi Drivers Preferences

This case study was developed using GPS positioning records obtained from 476 taxi cars in Shenzhen, China, between 01-09-2009 and 30-09-2009. We wish to assert what reward functions better characterize the taxi driver's movements inside the city.

The **primary objectives** of this case study are:

1. To show that the devised algorithm can be applied to large scale data.

2. To have an empiric measurement of the efficiency of the algorithm.

5.1 Problem Statement

In a situation without passengers, taxi drivers need to decide between several alternatives regarding their next destination. In general, two actions are available - either waiting for passengers at its current location or driving along a nearby road.

In similar situations, individual driver's options are distinct. As an example, individual preference for a region of the city or knowledge of long-course-train schedules impact their choices. It shall also be noted that taxi drivers are not simply maximizing profit, as some of their actions indicate otherwise: they usually finish their shift earlier when they already earned above average, their average working time varies, some penalize late working hours higher than others, etc. The reward function they use is not simple.

We consider taxi drivers are in equilibrium, which means that their actions are already optimum considering the reward function they use. In MDP terms, they follow the optimum policy and are not exploring different possibilities but rather always selecting what they believe is best.

The MDP states and transitions were created according to the data available and one specific taxi driver [with plenty of data points] was chosen for this study. The movement of this taxi driver was analyzed and his path was coded as movements in the MDP.

Three different reward functions were considered: minimize travel distance, minimize stopped time and maximize number of passengers. These are vectors along which we intend to classify the taxi driver's behavior when he has no passengers to transport. Therefore, the behavior R can be expressed by Equation 5.1 below.

$$R = \alpha_1 travel\_distance + \alpha_2 stopped\_time + \alpha_3 number\_of\_passengers. \tag{5.1}$$

5.2 Contextualization - The collection of urban traffic data

Urban traffic, as a stochastic event, requires big quantities of data and domain knowledge in order to produce meaningful predictions. In fact, traffic modeling systems use different types of information, mostly aggregated and historical or raw and in real-time.

Previous efforts in traffic prediction relied in induction loop sensors placed in the road network, obtaining occupancies, vehicle counts and speed averages [SK03] [WCQZ06] [KCB00]. These inductors were used not only for travel time prediction but also for active traffic management and traffic lights control; research has been done in order to determine their best positioning [BCR01] [EFZM06]. Given the static position of the sensors, the system is more robust as data is more homogeneous. However, the coverage problem is hard to tackle and some sections of the network can not be covered without costly investments.

Recently, GPS information obtained from probe vehicles has changed research focus. Bus and taxi companies have equipped their vehicles with GPS devices in order to obtain real-time information regarding their speed, direction, occupancy and position, providing extensive test beds to the research community. Several studies using this data have been conducted, such as in Shanghai, China, [ODZ$^+$11] and in Ann Arbor, Michigan, USA [YNL07].

Nevertheless, identifying what kind of information can be obtained from these probe vehicles is not straightforward, as they might not be representative of the actual traffic state. As an example, taxi drivers looking for prospective customers might drive slower than usual. Furthermore, these systems are highly dependent on the distribution and coverage of the road network. Identifying proper sampling rates, both geographically and temporally, is a problem yet to be solved.

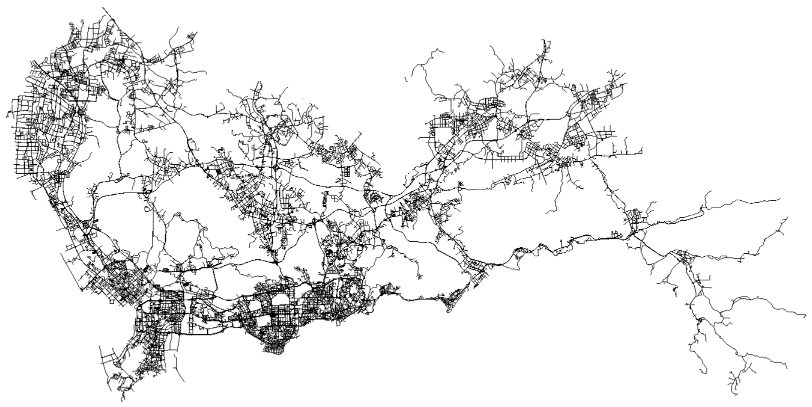
5.3 Experimental setup

5.3.1 Raw data

Different sources of information were merged to produce this work. A shapefile[1] of the whole Shenzhen region was provided and sequences of road segments tagged with taxi identifiers and timestamps represent taxi movements.

Finally, pre-processed data such as pick-up rate per road segment and income distribution per road segment are also part of the raw data available.

[1]A popular geospatial vector data format. They describe the geometry of the network as points and polylines representing roads


(a) Full network

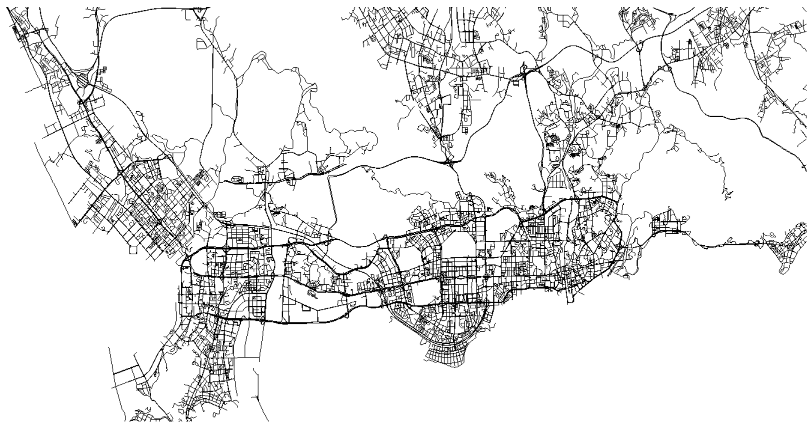

(b) Shenzhen's downtown region

Figure 5.1: The raw data representing the urban network

Figure 5.1 is a visualization of the Shenzhen's city map, with Figure 5.1a representing the whole map and Figure 5.1b being the downtown region studied. The whole Shenzhen's shapefile was composed of 264425 road segments; the cropped region was composed of 114755 road segments.

Taxi drivers' GPS positioning records had to be transformed into positions along edges of the underlying graph of the network. Distances were measured using an implementation of the haversine formula, usually represented as in Equation 5.2. The actual implementation followed a slightly different derivation that can be found in [Ric99].

$$d = 2r\arcsin\left(\sqrt{\sin^2\left(\frac{lat_2 - lat_1}{2}\right) + \cos(lat_1)\cos(lat_2)\sin^2\left(\frac{lng_2 - lng_1}{2}\right)}\right) \tag{5.2}$$

5.3.2 Map matching

Given the objective of this work, we are only concerned with decisions made by taxi drivers at intersections, which we consider decision points. The existing shapefiles layout had to be minimized and intermediate nodes were removed; the resulting network can be seen if Figure 5.2.

Figure 5.2a shows a specific region of the original network, while Figure 5.2b illustrates the same region without the removed nodes.

A node was removed whenever its indegree and outdegree is 2 and to the same nodes, so that one-way roads can be preserved. The graph of the resulting network contained 24752 nodes and 69450 edges.

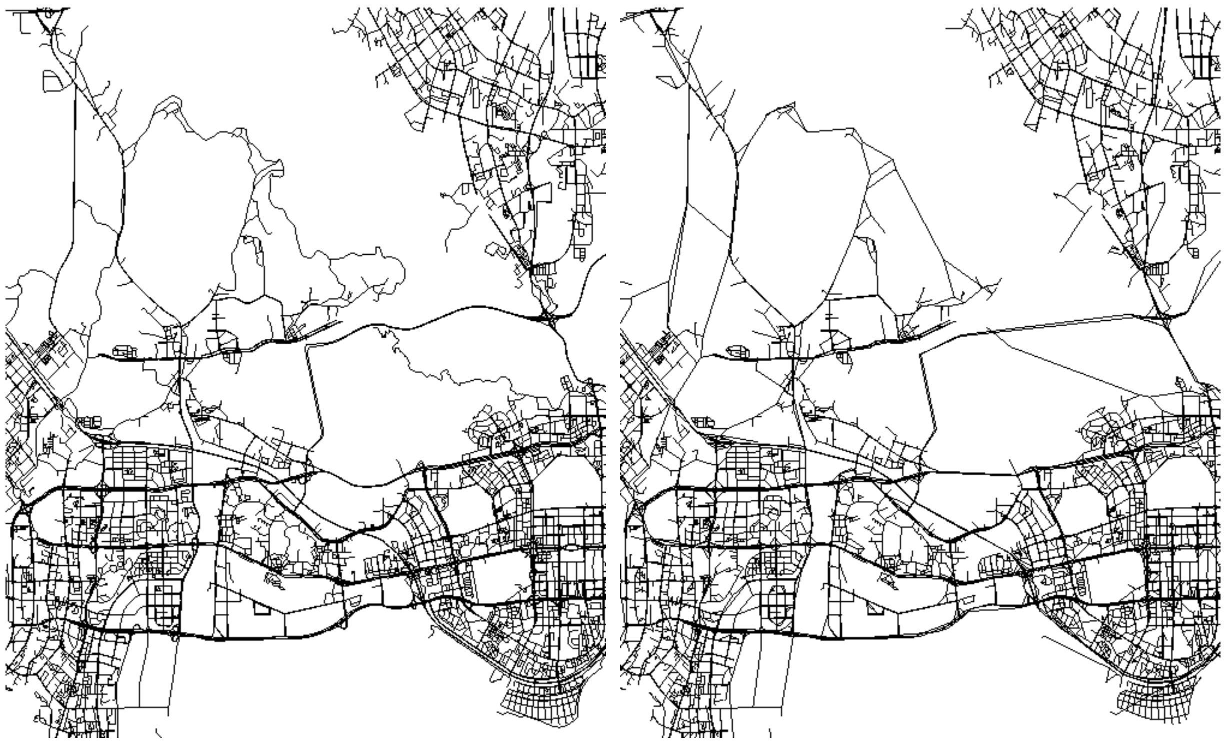

(a) Zoom from downtown region (b) Zoom from downtown region of the minimized network

Figure 5.2: Comparison between the original shapefiles and the minimized network.

As the network was changed, the recorded GPS positions had to be modified. For each two adjacent points left in the modified network, their distance was calculated in the original graph and records were changed with a linear mapping.

5.3.3 Travel time extraction

In order to build the MDP, travel time between adjacent nodes had to be obtained. Even though shapefiles provides a very accurate measure of distance, travel time is likely to be a more important feature for taxi drivers.

Figure 5.3 shows an heat map of 476 taxis between 9am and 10am - the lighter the road, the more vehicles used it during that period. The network was further restricted to those segments with travel time information, Travel time extraction from GPS records is a research problem with considerable attention within the research community [QB98] [LM02] [HsYHmDc07]. In this work, a simple approach was chosen:

1. The shortest path between consecutive records was obtained using Dijkstra's algorithm [Dij59].
2. Using the timestamp label attached to records, velocity between two consecutive records was calculated. A threshold was specified and velocities below it were discarded.
3. The calculated velocity was aggregated per road segment and averaged.

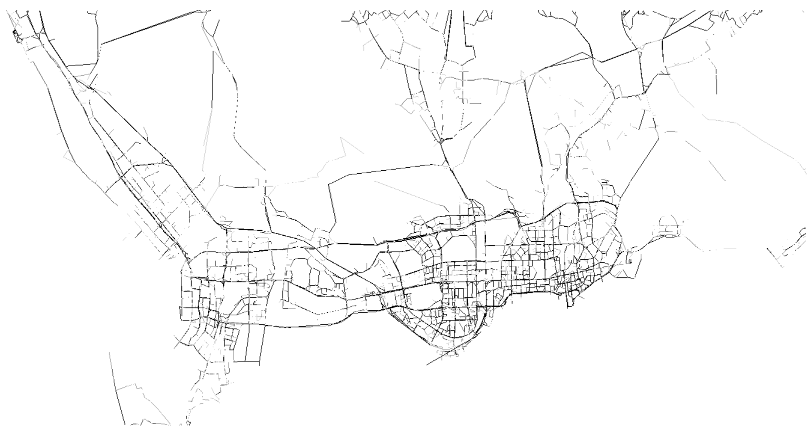


Figure 5.3: Heat map representing the movements of the 476 taxis from 9:00 to 10:00 am.

5.3.4 MDP description

The state space considered includes location and time of the day, (s,t). Given the travel time distribution between intersections (i.e. taking in consideration the average travel time between adjacent nodes in the network), the most suitable discretization of the time component is in intervals of 10 seconds. Therefore, in order to avoid an overwhelming number of states, a time period between 9am and 10am was selected to be analyzed. This resulted in a total of 4395896 MDP states. [2]

From statistical analysis, the probability of finding a passenger on a given location is known; let $pprob_{i,j}$ represent the probability of finding a passenger on the transition from location i to location j. Picking-up passengers was represented in the MDP by establishing that transitions between locations are made with probability $1 - pprob_{i,j}$ (i.e. not finding passengers).

On the other hand, with probability $pprob_{i,j}$, the system transitions to a state representing a general drop-off location. For each time interval t, there is such extra state and it is connected to each state whose time component is $t+1$. Given that there was no indication of probable drop-off locations, an uniform distribution was considered. Furthermore, the average journey length was considered for every transition from the drop-off states.

[2]Heatmaps in Figures 5.3 and 5.4 both illustrate this period.

This MDP could also incorporate other details often seen in reality, such as travel time variability between locations, but this study was restricted to average travel time due to the fact that these metrics were not easily available.

5.4 Observed behavior

Figure 5.4 shows a heat map of the movements of a particular taxi in this period:

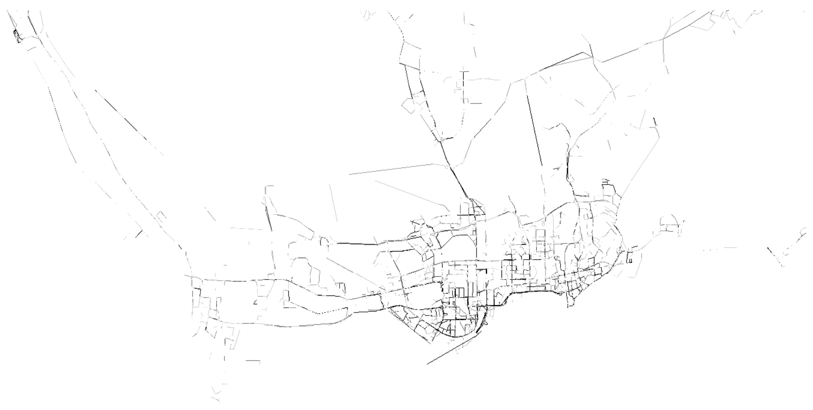


Figure 5.4: Heat map of the chosen vehicle.

Analysis of the GPS records of this individual taxi established his decisions on each road and intersection. For the purpose of this study, we considered the vehicle to be stopped whenever its velocity was below 1m/s, in order to absorb measurement errors.

A total of 202 observations were recorded; these observations will act as constraints in the Linear Programming formulation.

5.5 Reward functions

As indicated in section 5.1, three simple reward functions describing taxi drivers' desires were implemented.

ϕ_1 is a penalty function directly proportional to the distance traveled, obtained from an external file as described in a previous section.

Function ϕ_2 directly penalized the stoppage time of taxi drivers, directly proportional to the time spent without moving.

Finally, function ϕ_3 rewarded the taxi driver for each passenger he transported. This was achieved by rewarding transitions to the drop-off states described in section 5.3.4.

These three reward functions were normalized. This is especially important to guarantee that variations (such as using kilometers or meters in function ϕ_1) do not influence the results.

5.6 Results and Analysis

As described in section 3.3.3, *Decreasing the number of binary variables*, in order to decrease processing time, a single binary variable C_s was used to represent each taxi driver's decisions.

The resulting MILP model was composed of 202 binary variables (C_s) and 3 continuous variables in the range [0, 1] (α_i). A straightforward implementation on the referred solver took 29 seconds on a regular laptop. It is our opinion that 202 decision variables constitutes a significant number which shall allow the description of general tendencies in the observed behavior. Nevertheless, taking into consideration the required completion time, we are confident that more observations could be used.

Table 5.1 shows the number of violating states when each reward function was used alone, while Table 5.2 shows the values of α_i when all the functions were considered.

Table 5.1: Violating states for each reward function

Reward Function	Violating states	% Correct
Distance	143	29.3%
Stoppage time	37	81.7%
Passengers	99	51.0%

Table 5.2: Weight of each reward function used to describe the observed behavior

α_1	α_2	α_3	Violating states
0	1	0	37

These results demonstrate a preference for using function ϕ_2 alone over a linear combination of the proposed functions. In retrospective, the usefulness of function ϕ_2 is questionable and its definition lacks the properties required for a solid reward function. By simply penalizing stoppage time, many draws are created in its value function as different optimum policies can be easily constructed by forcing the vehicle to move; a good reward function should provide different rewards to different actions and multiple optimum policies should be avoided. As expected, further analysis shows that violating states are those where the taxi driver chose not to move.

Finally, expert knowledge and more and varied data would allow for a better analysis of this problem. Reward functions shall be defined so that whatever is the linear combination that best describes the observed behavior, meaningful information can be extracted. As an example, restricting the analysis to data regarding calls from the taxi central indicating passengers requesting taxis or data representing local knowledge (e.g. train schedules or major factories end-of-shift) would necessarily improve the quality of the results.

It is our opinion that this topic can be prolific in future work directions. Enquiring the city's taxi drivers could provide more information regarding their attitudes; cross-validating this information with results obtained from models such as the one proposed in this dissertation would be important for both traffic planers and for the domain of behavior understanding.

5.7 Summary

The second case study aimed to assess the applicability of the proposed algorithm when processing time is a concern, showing an application with real-world data. It is our opinion that the results are positive and that this is not a limitation in future implementations.

Finally, this case study illustrates the importance of iterations in the proposed methodology, which was not fully applied due to time constraints. Even though it only describes the addition of new reward functions, the removal of previously created functions shall certainly be considered when reward functions hinder the understanding of the agent's decisions.

Chapter 6

Conclusions

The extraction of high level knowledge from records is a widespread practice in datamining and in general knowledge extraction applications. Activity logs have the extra property of being highly structured and of having strong inter-record relationships and dependencies; algorithms in the field of Artificial Intelligence have been used to tackle problems with similar structure.

However, the extraction of human and animal motivations appears as a problem still to be solved. Most of the existing research is focused on problems of imitation learning and the synthesis of intelligible features and characteristics, which could help us understand why we behave and act the way we do, is yet to be studied in significant detail.

6.1 Final remarks

This dissertation was motivated by problems in the domain of urban traffic. Problems regarding human motivations arise on both the supply and demand side of the network, as passengers and drivers have a certain degree of freedom in their choices and, ultimately, are concerned with their own satisfaction.

Although they serve the higher purpose of testing the algorithm and methodology proposed, the two case studies presented intend to demonstrate problems in both sides of traffic modeling problems. Understanding passenger motivations, which represent passenger needs, is an important problem for traffic planers. As far as the author knows, automatic strategies to tackle this problem have not been proposed before. On the other hand, a better knowledge of taxi drivers profiles would allow taxi companies to assign their services in a way to increasing individual satisfaction. Many different problems of behavior understanding could be presented in this domain alone, but other unexplored possibilities in distinct domains are described in chapter 3.

The decision to approach the issue as an Inverse Reinforcement Learning problem and the creation of a link between motivations and reward functions used in reinforcements allowed for an objective definition of our primary goal, here revisited:

Let π^O be the observed policy and π^R the policy corresponding to the reward function R, can we find a reward function whose corresponding optimal policy matches the observed behavior? Then R shall be such that minimizes

$$\sum_{s \in S} [\pi^O(s) \neq \pi^R(s)]. \tag{6.1}$$

6.2 Contributions

This thesis effectively contributes to the domain of Inverse Reinforcement Learning and explores a problem overlooked by the research community - behavior analysis based on activity logs. It provides:

- A methodology for supporting behavior analysis problems.
- An alternative algorithm for Inverse Reinforcement Learning that provides more feedback than the alternatives.

Furthermore, the first case study presented indicates that it is possible to retrieve passenger preferences, given logs of their movements inside a city. The second case study provides a basis for the extraction of taxi driver profiles from GPS records, even though more in-depth work needs to be done in this area.

6.3 Future work directions

It is our belief that the inherently subjective nature of the problem (what is a useful, high-level and intelligible reward function?) renders fully automatic approaches impossible. Can this be proven? Would it be possible for a system to automatically extract intelligible reward functions, communicating them in a format understandable to us humans?

Considering the above limitation, is it possible for any methodology to guarantee that the error function is strictly decreasing? Simply finding a new reward function that explains previously violating states is not a guarantee that it will be included, given the limit on the sum of the αs.

Are there better alternatives to the linearization of the reward function? For finite MDPs, reward functions can be easily described in matrix form. In the current approach, high-level information is relegated to the simple reward functions that act as black-boxes. Can this high-level information be introduced in some other step of the whole process?

Finally, the algorithms shown work with value functions of the reward function. We then pretend that importance of the reward functions is the same as the importance of the value functions that we work with. Can this be proven, or better yet, can algorithms that work directly with reward functions be created?

6.4 Lessons learned

In this section we would like to describe some of the biggest hurdles we had to overcome.

Firstly, artifacts in the real data and the transformation from the original format to the MDP structure must not be underestimated. It is well known that parsing the dataset is often the biggest barrier in datamining techniques, and some steps are easily overlooked if the focus is shifted to the engaging problem of understanding behavior too soon. Datamining methodologies such as CRISP (Cross Industry Standard Process for Data Mining) can and should be used when pre-processing the data, with proper adaptation.

Secondly, the analysis of the feedback provided by the algorithm is often difficult and visualization tools might need to be built to further improve the knowledge of the user analyzing the data. The second case study is an example of a situation in which visualization was not part of the process to reach the solution, yet it can be considered an essential helping tool to comprehend the data.

Finally, the proposal of a new reward function requires an in-depth knowledge of the domain at hand. Case Study II is an excellent example of the difficulties that arise in this process.

References

[ACQN07] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19, Cambridge, MA, USA, 2007. MIT Press.

[AE05] Karen Aardal and Friedrich Eisenbrand. Integer programming, lattices, and results in fixed dimension. In G.L. Nemhauser K. Aardal and R. Weismantel, editors, *Discrete Optimization*, volume 12 of *Handbooks in Operations Research and Management Science*, pages 171–243. Elsevier, Amsterdam, 2005.

[AM02] R. Amit and M. Mataric. Learning movement sequences from demonstration. In *Proceedings of the 2nd International Conference on Development and Learning*, pages 203–208, Cambridge, MA, USA, June 2002.

[AN04] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, New York, NY, USA, 2004. ACM.

[Bas99] Erdem Bascı. Learning by imitation. *Journal of Economic Dynamics and Control*, 23(9-10):1569–1585, September 1999.

[BBM06] Maurizio Bielli, Azedine Boulmakoul, and Hicham Mouncif. Object modeling and path computation for multimodal travel systems. *European Journal of Operational Research*, 175(3):1705–1730, December 2006.

[BCR01] Lucio Bianco, Giuseppe Confessore, and Pierfrancesco Reverberi. A network based model for traffic sensor location with implications on o/d matrix estimates. *Transportation Science*, 35(1), February 2001. Focused Issue on ITS-related Problems.

[Bea96] John E. Beasley. *Advances in Linear and Integer Programming*. Oxford Science Publications, Oxford, 1996.

[Bel57] R. Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(4):679–684, April 1957.

[BPSS98] Darse Billings, Denis Papp, Jonathan Schaeffer, and Duane Szafron. Opponent modeling in poker. In *AAAI 1998*, pages 493–499. AAAI Press, 1998.

[BT96] D. P. Bertsekas and J. Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, Belmont, MA, USA, September 1996.

[Dah01] Fredrik A. Dahl. A reinforcement learning algorithm applied to simplified two-player texas hold'em poker. In Luc De Raedt and Peter A. Flach, editors, *Proceedings of the 12th European Conference on Machine Learning*, pages 85–96, London, UK, September 2001. Springer-Verlag.

[Den70] Eric V. Denardo. On linear programming in a markov decision problem. *Management Science*, 16(5):281–288, January 1970.

[dGE67] Guy T. de Ghellinck and Gary D. Eppen. Linear programming solutions for separable markovian decision problems. *Management Science*, 13(5):371–394, January 1967.

[Dij59] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[EFZM06] Stacy M. Eisenman, Xiang Fei, Xuesong Zhou, and Hani S. Mahmassani. Number and location of sensors for real-time network traffic estimation and prediction: Sensitivity analysis. *Transportation Research Record: Journal of the Transportation Research Board*, 1964:253–259, 2006.

[Fel71] W. Feller. *Introduction to Probability Theory and Its Applications, vol II.* Wiley, New York, NY, USA, 1971.

[FR08] Dinis Félix and Luís Paulo Reis. Opponent modelling in texas hold'em poker as the key for success. In *Proceedings of the 2008 conference on ECAI 2008: 18th European Conference on Artificial Intelligence*, pages 893–894, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press.

[FT91] C. T. Farley and C. R. Taylor. A mechanical trigger for the trot-gallop transition in horses. *Science*, 253(5017):306–308, July 1991.

[Has70] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970.

[How60] Ronald A. Howard. *Dynamic Programming and Markov Processes.* The M.I.T. Press, Cambridge, MA, USA, June 1960.

[HsYHmDc07] Zhang He-sheng, Zhang Yi, Wen Hui-min, and Hu Dong-cheng. Estimation approaches of average link travel time using gps data. *Journal of Jilin University*, March 2007.

[HT81] Donald F. Hoyt and C. Richard Taylor. Gait and the energetics of locomotion in horses. *Nature*, 292:239–240, July 1981.

[HvBPM07] Jesse Hoey, Axel von Bertoldi, Pascal Poupart, and Alex Mihailidis. Assisting persons with dementia during handwashing using a partially observable markov decision process. In K.D.Baker and Monique Thonnat, editors, *Proceedings of the 5th International Conference on Computer Vision Systems*, March 2007.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Yorktown Heights, NY, USA, 1972.

[KCB00] Jaimyoung Kwon, Benjamin Coifman, and Peter Bickel. Day-to-day travel time trends and travel time prediction from loop detector data. *Transportation Research Record: Journal of the Transportation Research Board*, 1717(1):120–129, 2000.

[KGGW94] M. Kawato, F. Gandolfo, H. Gomi, and Y. Wada. Teaching by showing in kendama based on optimization principle. *Proceedings of the International Conference on Artificial Networks*, 1994.

[KR76] R. L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. Wiley, New York, NY, USA, 1976.

[Kro] Dmitrey Kroshko. OpenOpt: Free scientific-engineering software for mathematical modeling and optimization, 2007–.

[KS06] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *ECML'06 Proceedings of the 17th European Conference on Machine Learning*, pages 282–293, 2006.

[LM93] Long-Ji Lin and Tom M. Mitchell. Reinforcement learning with hidden states. In Jean-Arcady Meyer, Herbert L. Roitblat, and Stewart W. Wilson, editors, *Proceedings of the second international conference on From animals to animats 2 : simulation of adaptive behavior*, pages 271–280, Cambridge, MA, USA, 1993. MIT Press.

[LM02] Yanying Li and Mike McDonald. Link travel time estimation using single gps equipped probe vehicle. In *Proceedings of the iEEE 5th Conference on Intelligent Transportation Systems*, pages 932–937, September 2002.

[LP10] Seong Jae Lee and Zoran Popovic. Learning behavior styles with inverse reinforcement learning. *ACM Transactions on Graphics*, 29(4):122:1–122:7, July 2010.

[MA93] Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13:103–130, 1993.

[Man60] Alan S. Manne. Linear programming and sequential decision models. *Management Science*, 6(3):259–267, April 1960.

[Mar54] Andrey A. Markov. *The Theory of Algorithms*, volume 42. Academy of Sciences of the USSR, 1954.

[McC97] Andrew Kachites McCallum. Efficient exploration in reinforcement learning with hidden state. In *AAAI Fall Symposium on Model Directed Autonomous Systems*. AAAI, 1997.

[Mis12] MistWiz. Example of markov decision process (mdp) transition automaton, June 2012.

[MSW96] Jiří Matoušek, Micha Sharir, and Emo Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4-5):498–516, 1996.

[NR00] Andrew Y. Ng and Stuart J. Russel. Algorithms for inverse reinforcement learning. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[NS07] Gergely Neu and Csaba Szepesvári. Apprenticeship learning using inverse reinforcement learning and gradient methods. In Ronald Parr and Linda C. van der Gaag, editors, *Uncertainty in Artificial Intelligence*, pages 295–302. AUAI Press, 2007.

[ODZ$^+$11] Kaoru Ota, Mianxiong Dong, Hongzi Zhu, Shan Chang, and Xuemin Shen. Traffic information prediction in urban vehicular networks: A correlation based approach. *Wireless Communications and Networking Conference*, pages 1021–1025, March 2011.

[Pad99] M. Padberg. *Linear Optimization and Extensions (Algorithms and Combinatorics)*. Springer-Verlag, Berlin, Germany, July 1999.

[PRC$^+$08] Marc Ponsen, Jan Ramon, Tom Croonenborghs, Kurt Driessens, and Karl Tuyls. Bayes-relational learning of opponent models from incomplete information in no-limit poker. In *Proceedings of the 23rd national conference on Artificial intelligence*, volume 3, pages 1485–1486, Menlo Park, CA, USA, 2008. AAAI Press.

[PT87] Christos H. Papadimitriou and John K. Tsitsiklis. The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, August 1987.

[Put94] Martin L. Puterman. *Markov Decision Processes: Discrete stochastic dynamic programming*. Wiley, New York, NY, USA, April 1994.

[pyt] Python programming language. www.python.org.

[QB98] Cesar A. Quiroga and Darcy Bullock. Travel time studies with global positioning and geographic information systems: an integrated methodology. *Transportation Research Part C: Emerging Technologies*, 6(1-2):101–127, February 1998.

[RA07] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, pages 2586–2591, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[RBM07] Karl Rehrl, Stefan Bruntsch, and Hans-Joachim Mentz. Assisting multimodal travelers: Design and prototypical implementation of a personal travel companion. *iEEE Transactions on Intelligent Transportation Systems*, 8(1):31–42, 2007.

[RBZ06] Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. Maximum margin planning. In William Cohen and Andrew Moore, editors, *ICML '06 Proceedings of the 23rd International Conference on Machine Learning*, pages 729–736, New York, NY, USA, 2006. ACM.

[Ric99] Doctor Rick. Deriving the haversine formula - http://mathforum.org/library/drmath/view/51879.html, 1999.

[Rus98] Stuart Russell. Learning agents for uncertain environments (extended abstract). In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103, New York, NY, USA, 1998. ACM.

[SB98] R. S. Sutton and A. G. Barto. *Reinforcement Learning: an Introduction*. MIT Press, Cambridge, MA, USA, 1998.

[Sch99] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233–242, December 1999.

[SHKM92] Claude Sammut, Scott Hurst, Dana Kedzier, and Donald Michie. Learning to fly. In Derek H. Sleeman and Peter Edwards, editors, *Proceedings of the Ninth International Conference on Machine Learning*, pages 385–393, San Francisco, CA, USA, 1992. Morgan Kaufmann.

[SK03] Anthony Stathopoulos and Matthew G. Karlaftis. A multivariate state space approach for urban traffic flow modeling and prediction. *Transportation Research Part C*, 11(2):121–135, April 2003.

[SKVS96] Yury Smirnov, Sven Koenig, Manuela M. Veloso, and Reid G. Simmons. Efficient goal-directed exploration. In *Proceedings of the thirteenth national conference on Artificial intelligence*, volume 1, pages 292–297. AAAI Press, 1996.

[Tar05] Albert Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2005.

[Wat89] C.J.C.H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, 1989.

[WCQZ06] Yaqin Wang, Yue Chen, Minggui Qin, and Yangyong Zhu. Dynamic traffic prediction based on traffic flow mining. In *Proceedings of the 6th World Congress on Intelligent Control and Automation*, volume 2, pages 6078–6081, 2006.

[Whi85] Douglas J. White. Real applications of markov decision processes. *Interfaces*, 15(6):73–83, 1985.

[Whi87] Douglas J. White. Further real applications of markov decision processes. *Interfaces*, 18(5):55–61, 1987.

[Whi93] Douglas J. White. A survey of applications of markov decision processes. *The journal of the Operational Research Society*, 44(11):1073–1096, 1993.

[Wit91] Steven D. Witehead. Complexity and cooperation in q-learning. In Lawrence A. Birnbaum and Gregg C. Collins, editors, *Proceedings of the Eighth International Workshop on Machine Learning*, pages 363–367, 1991.

[WY07] Jason D. Williams and Steve Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, April 2007.

[YNL07] Jungkeun Yoon, Brian Noble, and Mingyan Liu. Surface street traffic estimation. pages 220–232, New York, NY, USA, 2007. ACM.

[ZALMB08] Brian D. Ziebart, Anind K. Dey Andrew L. Maas, and J. Andrew Bagnell. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In *Proceedings of International Conference on Ubiquitous Computing (Ubicomp 2008)*, pages 322–331, September 2008.

[ZDB12] Brian D. Ziebart, Anind K. Dey, and J. Andrew Bagnell. Probabilistic pointing target prediction via inverse optimal control. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, pages 1–10, New York, NY, USA, February 2012. ACM.

[ZMBD08] Brian Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Twenty-Third AAAI Conference on Artificial Intelligence*, 2008.