

## Singapore Management University Institutional Knowledge at Singapore Management University

---

Research Collection School Of Information Systems

School of Information Systems

---

6-2013

# Understanding Sequential Decisions via Inverse Reinforcement Learning

Siyuan LIU  
*Carnegie Mellon University*

Miguel ARAUJO  
*Carnegie Mellon University*

Emma BRUNSKILL  
*Carnegie Mellon University*


Rosaldo ROSSETTI  
*Universidade do Porto*

Joao BARROS  
*Universidade do Porto*

*See next page for additional authors*

**DOI:** <https://doi.org/10.1109/MDM.2013.28>

Follow this and additional works at: [https://ink.library.smu.edu.sg/sis\\_research](https://ink.library.smu.edu.sg/sis_research)

 Part of the [Artificial Intelligence and Robotics Commons](#), and the [Theory and Algorithms Commons](#)

---

### Citation

LIU, Siyuan; ARAUJO, Miguel; BRUNSKILL, Emma; ROSSETTI, Rosaldo; BARROS, Joao; and KRISHNAN, Ramayya. Understanding Sequential Decisions via Inverse Reinforcement Learning. (2013). *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*. Research Collection School Of Information Systems.

**Available at:** [https://ink.library.smu.edu.sg/sis\\_research/3474](https://ink.library.smu.edu.sg/sis_research/3474)

This Conference Proceeding Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email [libIR@smu.edu.sg](mailto:libIR@smu.edu.sg).

---

**Author**

Siyuan LIU, Miguel ARAUJO, Emma BRUNSKILL, Rosaldo ROSSETTI, Joao BARROS, and Ramayya KRISHNAN

# Understanding Sequential Decisions via Inverse Reinforcement Learning

Siyuan Liu<sup>1</sup>, Miguel Araujo<sup>1</sup>, Emma Brunskill<sup>1</sup>, Rosaldo Rossetti<sup>2</sup>, Joao Barros<sup>3</sup>, Ramayya Krishnan<sup>1</sup>

<sup>1</sup>*Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, 15213. PA, USA.*

<sup>2</sup>*LIACC, Departamento de Engenharia Informatica, Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, 4200-465. Porto, Portugal*

<sup>3</sup>*Instituto de Telecomunicacoes, Faculdade de Engenharia, Universidade do Porto, Rua Dr Roberto Frias, 4200-465. Porto Portugal*

**Abstract**—The execution of an agent’s complex activities, comprising sequences of simpler actions, sometimes leads to the clash of conflicting functions that must be optimized. These functions represent satisfaction, short-term as well as long-term objectives, costs and individual preferences. The way that these functions are weighted is usually unknown even to the decision maker. But if we were able to understand the individual motivations and compare such motivations among individuals, then we would be able to actively change the environment so as to increase satisfaction and/or improve 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 series of complex activities (called sequential decisions in our work). A novel algorithm for the analysis of observational records is proposed. We also present a methodology that allows researchers to converge towards a summary description of an agent’s behaviors, through the minimization of an error measure between the current description and the observed behaviors.

This work was validated using not only a synthetic dataset representing the motivations of a passenger in a public transportation network, but also real taxi drivers’ behaviors from their trips in an urban network. Our results show that our method is not only useful, but also performs much better than the previous methods, in terms of accuracy, efficiency and scalability.

## I. INTRODUCTION

How people make sequential decisions? And how can we observe such decision behaviors and understand the undergoing process? Based on such a model, we can even retrieve people’s preference. 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 and effectively as possible. We are building systems that try to mimic human decisions in settings where we do not fully understand what is important and guides our own actions. Examples include tasks such as “driving well” [1], controlling helicopters [2], 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. Even though we do not have a clear understanding of our decision process whereas performing these

tasks, we are building systems that can improve the decision understanding by demonstrations. At the same time, some fields of Computer Science (e.g. Data Mining and branches of Artificial Intelligence) specialize on the knowledge discovery process, whereas psychology draws from the knowledge of other fields to help explain human behaviors. So the pertinent question is, given the amount of available behavior data regarding the fulfillment of activities, what if we could utilize these observations to accurately describe and reason about the decision-making process? Therefore, being able to describe a reward function in use plays a vital role in understanding what we observe. Discovering a 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 [3]. It would be much more interesting and challenging if we were able to characterize urban passengers or taxi drivers based on their mobility patterns, understanding how they weigh factors such as travel time, travel cost or travel time variance, etc. [4], [5], [6], [7], [8], [9], [10].

The challenges are as follows. First, human descriptions are often at different granularities, whereas computers struggle to ascertain the relevance of the relations that they identify. Deciding on the correct granularity and on what constitutes a piece of knowledge small enough to be understood by a domain expert, is hard to be incorporated in practice. Second, the fact is that 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 the decisions of animals and humans when performing complex tasks, especially when there exist sequential decisions.

In this paper, we provide a methodology for describing the sequential decisions of individuals (illustrated in Figure 1). We present a new Linear Programming formulation for the Inverse Reinforcement Learning problem which enables domain experts to compare how different reward functions fit the observed behaviors. Based on both synthetic and real world datasets, our results show that it is possible to recover a linear combination of the reward functions which indicates an agent’s preference. The applicability of the presented algorithm to

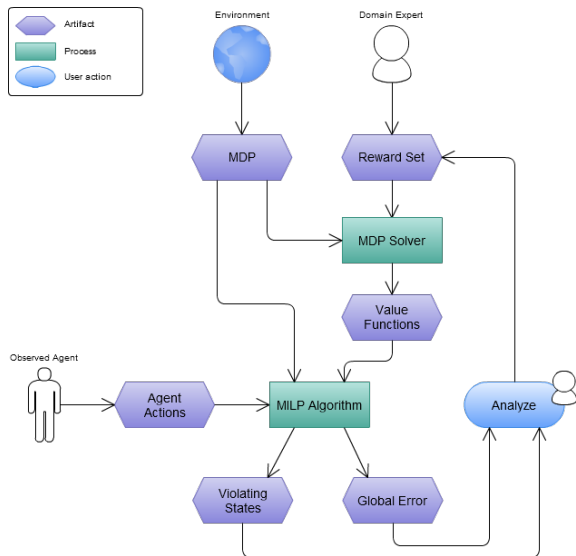


Fig. 1: Overview of Understanding Sequential Decisions

large datasets is also demonstrated. Furthermore, we compare our method with two latest related baseline methods via real-life taxi drivers' behavior logs. The results show that our method outperforms the previous methods in terms of accuracy, efficiency and scalability.

In summary, our contributions are:

- 1) We propose an interesting problem: identifying the preference from understanding sequential decisions and then we formalize it as an Inverse Reinforcement Learning problem.
- 2) We design a novel Linear Programming formulation to solve the problem which can interpret the reward functions describing the observed behaviors.
- 3) We illustrate applications of our method by well-designed case studies and evaluate it with baseline methods by large scale real-life taxi drivers' behavior logs. Interesting findings from our study are finally provided.

The remainder of this paper is organized as follows. Section II introduces related work in the area whereas Section III describes the notation used in this paper and provides some background in Markov Decision Processes and the Inverse Reinforcement Learning problem. Section IV formalizes our problem. Section V shows a new Linear Programming formulation and Section VI shows how the strengths of the previous formulation can be used to form a policy that practitioners should follow. Sections VII and VIII describe two experiments that illustrate how our proposals can be applied to the real world. Section IX discusses our results and points out some ideas for future work.

## II. RELATED WORK

Understanding user decisions is a problem gaining increased interest in today's research. A big part of human-computer interaction involves studying user satisfaction, human factors

and psychology and, more generally, the behavior of users when using a computer. Similarly, understanding user decisions browsing the Internet is essential to improve targeted advertising and PageRank algorithms. Mining behavior patterns has been used to discover users that share similar habits [11] and using sequential patterns to learn models of human behavior has been employed in lifestyle assistants to increase the life quality of elderly people [12].

Modeling decisions as Markovian processes<sup>1</sup> is a commonly used tool to simplify the representation of the knowledge of the decision maker (recent studies have statistically shown that Web users do not follow this property [13]). Section 3 provides more details on Markov Decision Processes (MDP) and their role in identifying the optimal decision in environments known to the decision maker that provide reinforcements.

On the other hand, understanding human behavior requires finding the reward function motivating the decisions. Inverse Reinforcement Learning (IRL), as first described by Russel [14], deals with the problem of identifying the reward function being optimized by an agent, given observations of its activity. The original algorithms of Ng and Russel [3] are detailed in the next section, but other algorithms have been proposed to tackle this task. Ratliff, Bagnell and Zinkevich approached this issue as a maximum margin structured prediction problem over a space of policies [15]. 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 [16]. They use the actions of the expert to update a prior on reward functions using a Markov Chain Monte Carlo [17] algorithm. In some recent publications, Brian Ziebart presented a maximum entropy approach [18] which has been successfully applied to different problems, such as route prediction given a driver's past information, destination prediction given incomplete journeys [19] and mouse movement prediction [20]. Rothkopf *et al.* stated the problem of inverse reinforcement learning in terms of preference elicitation, resulting in a principled (Bayesian) statistical formulation [21]. More details in this category can be found in reviews in the literature [22].

## III. PRELIMINARIES

### A. Markov Decision Processes

Markov Decision Processes (MDPs) [23] 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. Upon selection of an action, the process responds with a transition into a

<sup>1</sup>In a Markovian process decisions are assumed to be made based on the current state, not on the sequence of events that preceded it.

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

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

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

In this paper, we use the following notation:

- $S$  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.
- $\gamma \in [0, 1[$  is the discount factor.
- $R(s, s')$  represents the reward associated with the transition from  $s$  to  $s'$ .
- $\pi$  is defined as any map  $\pi : S \rightarrow A$ .

$P_{\pi(s)}$  will be used to represent the transition probability matrix corresponding to the application of policy  $\pi$ .

The value function for a policy  $\pi$  at any state  $s$  represents the expected cumulative reward from state  $s$  and is given by

$$V^\pi(s_1) = E[R(s_1) + \gamma R(s_2) + \gamma^2 R(s_3) + \dots | \pi], \quad (2)$$

or alternatively,  $V^\pi(s_1) = \sum_{t=1}^{\infty} \gamma^{t-1} R(s_t)$ , where  $\{s_1, s_2, s_3, \dots\}$  is the state sequence when we execute policy  $\pi$  starting from state  $s_1$ . The goal is thus finding a policy  $\pi$  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  $s \in S$  [3], [24], [25].

The optimum policy can be expressed in terms of the value function as follows:

$$\pi(s) \in \arg \max_{a \in A} \sum_{s' \in S} P_a(s, s')(R(s, s') + \gamma V^\pi(s')), \quad (3)$$

In each state, we apply the action with the biggest expected reward, taking into consideration the probability distribution of states to which it might lead us. Similarly, the action-value function for policy  $\pi$ ,  $Q^\pi(s, a)$ , is defined as the value of taking action  $a$  in state  $s$ .

A fundamental property of value functions used throughout reinforcement learning and dynamic programming is that they satisfy particular recursive relationships. For any policy  $\pi$  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 [23], these are called the **Bellman Equations**:

$$V^\pi(s) = \sum_{s' \in S} P_{\pi(s)}(s, s')[R(s, s') + \gamma V^\pi(s')] \quad (4)$$

$$Q^\pi(s, a) = \sum_{s' \in S} P_a(s, s')(R(s, s') + \gamma V^\pi(s')) \quad (5)$$

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 [25].

## B. Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL), as first described by Russel [14], deals with the problem of identifying the reward function being optimized by an agent, given observations of its activity. 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, we can see IRL 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, considering the notation used for describing MDPs, we wish to find the set of possible reward functions  $R$  such that  $\pi$  is an optimal policy in the MDP defined by  $(S, A, P_a, \gamma, R)$ .

*The original algorithms for IRL:* In 2000, Andrew Ng & Russel proposed a series of algorithms for the inverse reinforcement learning problem [3]. They translated the IRL problem to a Linear Programming problem with constraints leading to the optimal condition. Three cases were analyzed in the original paper:

- 1) IRL in Finite State Spaces;
- 2) Linear Function Approximation in Large State Spaces;
- 3) IRL from Sampled Trajectories.

Under a finite state space, the reward matrix  $\mathbf{R}$  can be obtained<sup>2</sup>. However, in the context of our problem of understanding behavior, we consider simply obtaining the reward matrix to be unsatisfactory, because it is not intelligible. In order to overcome this difficulty, one would need to analyze the reward matrix in order to obtain higher level information, understandable by humans. Given the possible size of the reward matrix, we consider this to be a significant setback of the approach as this problem is likely as difficult as the original one.

<sup>2</sup>They considered the reward to be a function of each state ( $R(s)$ ), and not part of the transition between two different states ( $R(s_1, s_2)$ ). The reward function must then satisfy  $(P_{\pi(s)} - P_a)(I - \gamma P_{\pi(s)})^{-1} R \succeq 0$ , where  $\succeq$  is defined as non-strict vector inequality.

Moreover, in practice, existing MDPs are often either infinite (e.g. the state description contains a time component), or our observations of the optimum policy do not fully encompass all the possible states (i.e. given this policy, not all states are accessible from the initial state and thus are not observable). In order to solve these two problems, Ng & Russel proposed a linear approximation of the reward function, expressing R as

$$R(s, s') = \alpha_1 \phi_1(s, s') + \alpha_2 \phi_2(s, s') + \dots + \alpha_n \phi_n(s, s') \quad (6)$$

Each function  $\phi_i$  would represent a possible reward function on the transition from  $s$  to  $s'$ . Due to the linearity of the reward function,  $V^\pi$  is also linear and given by

$$V^\pi = \alpha_1 V_1^{\phi_1} + \alpha_2 V_2^{\phi_1} + \dots + \alpha_n V_n^{\phi_1} \quad (7)$$

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. In Ng & Russel's algorithm, the idea is 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. Please note that  $V^\pi$  is now linearized and the  $\alpha$ s are the Linear Programming variables.

#### IV. UNDERSTANDING BEHAVIOR

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*).

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 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, but rather interested in creating models which might help us understand their behavior so that beekeepers can modify the environment in order to increase production.

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  $\alpha$ s 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 or not.

On the other hand, it would be interesting if we could quantify the relative importance that the agent ascribes to

each individual reward function. In practice, we would like the values of the  $\alpha$ s to be meaningful when compared. For two reward functions  $i$  and  $j$ , our objective is to guarantee that the reward function  $i$  has a greater importance for the agent than reward function  $j$ , whenever  $\alpha_i > \alpha_j$ . Unfortunately, importance is often volatile and heavily dependent on the circumstances. This requires reward functions to be even more complex and the reward functions hypothesis space to grow significantly. Furthermore, 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: in a MDP representing a city, the movements of a railfan<sup>3</sup> 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 cannot 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.

#### Problem Statement

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 more closely match the observed behavior (O), whereas being described as a linear combination of different vectors. Therefore, the problem is finding the set of  $\alpha$ s that best fit the agent's process, where each  $\phi_i$  is a simple reward function which the domain expert considers relevant for this particular problem, i.e.

$$R = \alpha_1 \phi_1 + \alpha_2 \phi_2 + \dots + \alpha_d \phi_d \quad (8)$$

Let  $\pi^O$  be the observed policy and  $\pi^R$  the optimal 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.

$$\text{minimize } \sum_{s \in S} [\pi^R(s) \neq \pi^O(s)]. \quad (9)$$

The formalization that follows is specified in [26], where the interested reader can find more details.

#### V. A NEW LINEAR PROGRAMMING FORMULATION

From our problem statement, we want to find R that is a linear combination of simpler functions and we want R to

<sup>3</sup>Also called rail buff, railway enthusiast or railway buff, is a person interested in the recreational capacity in rail transport.

be as close as possible to the observed behavior (minimizing the number of differences).  $\pi^O$  is the observed policy (e.g. decisions in each state) and  $\pi^R$  is the policy corresponding to the reward function we are describing.

Remembering the definition of  $\pi^R(s)$  in (3), a simple substitution in (9) leads to

$$\begin{aligned} & \text{minimize} \sum_{s \in S} [ \\ & \arg \max_{a \in A} \sum_{s' \in S} P_a(s, s')(R(s, s') + \gamma V^{\pi^R}(s')) \\ & \quad \neq \\ & \quad \quad \quad \pi^O(s) ] \end{aligned} \quad (10)$$

Given that the optimal action  $\pi^R(s)$  is a maximum (as given by (3)), for  $\pi^O$  and  $\pi^R$  to have the same value,  $\pi^O$  must be such maximum. Therefore,  $\forall s \in S, a \in A$ :

$$\begin{aligned} & \sum_{s' \in S} P_{\pi^O}(s, s')(R(s, s') + \gamma V^{\pi^R}(s')) \\ & \quad \geq \\ & \sum_{s' \in S} P_a(s, s')(R(s, s') + \gamma V^{\pi^R}(s')) \end{aligned} \quad (11)$$

We want this constraint to hold for the maximum number of states as possible; this is equivalent to solving the following mixed-integer Linear Programming problem:

$$\text{minimize} \sum_s C_s$$

such that:

$$\begin{aligned} & 0 \leq \alpha_i \leq 1 \\ & \sum_{s'} P_{\pi^O}(s, s')(R(s, s') + \gamma V^{\pi^R}(s')) - \\ & \quad - \sum_{s'} P_a(s, s')(R(s, s') + \gamma V^{\pi^R}(s')) + M * C_s \geq 0 \end{aligned}$$

where  $C_s \in \{0, 1\}$  is a binary variable and  $M$  is an arbitrarily large number.  $\sum_s C_s$  denotes the number of constraints which could not be respected. In our terminology, the number of states which cannot be explained with our solution.

In this Linear Programming formulation, the linear variables are the  $\alpha$ s because we can substitute:

$$\begin{aligned} & R(s, s') + \gamma V^{\pi^R}(s') \\ & = \end{aligned}$$

$$(\alpha_1(\phi_1(s, s') + \gamma V^{\phi_1}(s')) + \dots + \alpha_n(\phi_n(s, s') + \gamma V^{\phi_n}(s')))$$

#### A. Limiting the values of the $\alpha$ s

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

$$\sum \alpha_i = 1 \quad (12)$$

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

#### B. Normalization to allow relative comparison

Given that we are trying to ascertain the relative importance attributed to each reward function, relative values of the  $\alpha$ s are important. As such, normalization of the reward function's output is necessary in order to maintain correct equivalence. We opted for a simple z-score normalization over the  $\phi$  functions, each one being transformed according to:

$$\phi'(s) = \frac{\phi(s) - \overline{\phi(s)}}{\sigma_\phi} \quad (13)$$

## VI. UNDERSTANDING SEQUENTIAL DECISIONS

The advantage of the MILP formulation over the original Linear Programming formulation is that more feedback is produced. Given a set of reward functions, not only is the number of violated restrictions asserted, but the user is also able to find out exactly what state-action pairs are being violated in the MILP solution. Thus the user is then able to use the domain knowledge to add a new reward function to the reward functions set. Hence we are able to extract a methodology for behavior understanding.

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

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.

Second, using the observational records, the agent's decisions are obtained from the activity log, an observation of the agent's movements. These 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 whereas 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.

Third, 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 outputting the set of  $\alpha$ s, the global error  $\epsilon$  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 functions in the reward set not able to describe the behavior in 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.

#### A. 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 being the most common. Common value and policy iteration methods rely on the existence of a discount factor to converge to optimal solutions. On the other hand, if the activity is clearly limited and we do not wish to apply a discount factor to our MDP (i.e., we want to set  $\gamma = 1$  for the duration of the activity), we can still find optimum solutions by guaranteeing that the MDP is acyclic. In most scenarios, this can be easily done through the introduction of a time component and the MDP is easily solvable using dynamic programming. 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 [27]. However, integer programs with a constant number of variables may be solved in linear time as a LP-type problem<sup>4</sup> [28] [29]. 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, cannot be estimated.

#### B. Automatically finding better reward functions

The problem of understanding behavior is that we are not simply interested in replicating it, we rather need the reward functions to be meaningful and applicable to 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 observed behavior (i.e. reduce the error). Each reward function  $R$  in the reward set can be adapted through a function similar to the following:

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

<sup>4</sup>A.k.a. generalized linear program.

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

### VII. CASE STUDY I - SYNTHETIC DATA

In order to correctly assess the ability of the algorithm to determine the original reward function, a small test was conducted on a synthetic data setup. We considered the problem of understanding passenger choices in their individual travels. Several different means of transportation are available in today’s urban environment from buses, metros and taxis to individual transportation such as cars and walking. In this study, we restricted the study to buses, taxis and walking.

Passengers are driven by distinct desires: they might 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.

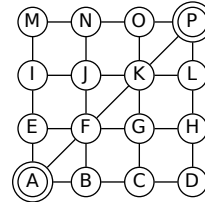


Fig. 2: Network layout used

#### A. Experimental setup

A small but densely connected synthetic network was modeled as a Markov Decision Process. The network layout allowed different possibilities between the chosen origin - destination pair, maximizing the number of decisions individual travelers may face. 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 2 shows the structure of the network used. Nodes A and P are the origin and destination, respectively.

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, we consider transitions to be deterministic and thus always leading to the same state. Nevertheless, more than one action corresponding to a given mode of transportation might be possible in a given state (e.g. two different walking directions). Figure 3 shows the possible



movements when using the Taxi, Bus or Walking. These connections were selected so that different reward functions generate different policies.

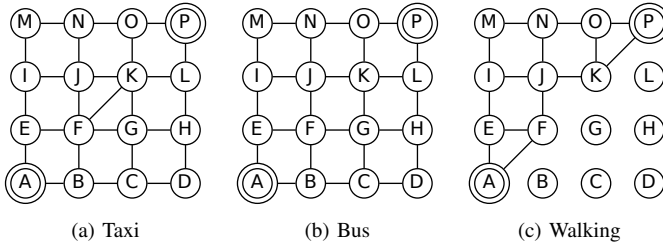


Fig. 3: Available connections per mode of transportation

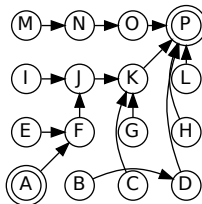


Fig. 4: Observed behavior

It was considered that every connection (edge) has a travel distance of 1. Walking takes 1 time units per transition, taxis take 1 time unit and buses take 2 time units. Additionally, taxis charge 3 monetary units per transition, whereas buses charge 2 monetary units regardless of the distance transversed.

### B. Observed behavior

The observed behavior synthetically created can be described by Figure 4. The observed behavior also includes the action selected in each state, but it was not included in the picture for simplification purposes.

### C. Reward functions

The proposed methodology was applied to this problem and the original reward function was identified after four simple reward functions were proposed. Due to space restrictions, the iterations and a complete description of the reward functions could not be included. Nevertheless, the four reward functions considered included minimizing travel time, minimizing travel distance, minimizing travel cost and a fourth reward function that applied a bias based on the distance to a node in the network. In this example, the reward function represents a safer region in the network.

### D. Results and Analysis

The reward function of the presented example was created based on a linear combination of two simple reward functions. Given that two of the proposed reward functions matched the two functions in the observed behavior, 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. We were able to observe that the feedback from the methodology iterations contributed to discovering the original function.

As a further test to this formulation, we wanted to test if the formulation was able to recover the original (single) reward function when presented with several possibilities. Table I shows the results of these tests. The numbers represent the output of the MILP formulation (i.e. the importance of the MILP formulation considering each function to be). In the first column we can see which function is in fact used, and also the the formulation is usually able to recover the original function when a good guess is provided.

Function	$\alpha_{time}$	$\alpha_{cost}$	$\alpha_{distance}$	$\alpha_{safety}$
Time	1	0	0	0
Cost	0	1	0	0
Distance	1/3	0	16/27	2/27
Safety	0	0	0	1
Cost + Safety	0	1/2	0	1/2

TABLE I: Inverse Reinforcement Learning on the simple reward functions

Note 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 also explain the observed behavior; it shall be noted that all the answers represent an error of 0 when compared with the original function. The second (least important) term of the minimization function tries to penalize deviations from the observed behavior, but that does not imply that we are closer to the real reward function.

## VIII. CASE STUDY II - REAL DATA

This case study was developed using GPS positioning records obtained from 500 taxicabs in a big city of 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 [4], [30], [31].

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 the 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 [32].

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 are always selecting what they believe is best.

The MDP states and transitions were created according to the data available and 12 taxi drivers (with plenty of data points) were chosen for this study. The movement of these taxi drivers were analyzed and their paths were considered as 12 different policies in our MDP.

Three different reward functions were implemented: minimize travel distance, minimize idle time and maximize number of passengers. These are vectors along which we intend to classify individual behavior when taxi drivers have no passengers to transport. Therefore, we wish to express the behavior  $R$  by Equation 14 below.

$$R = \alpha_1(\text{travel\_distance}) + \alpha_2(\text{idle\_time}) + \alpha_3(\text{number\_of\_passengers}) \quad (14)$$

#### A. Experimental setup

Different sources of information were merged to produce this work. A shapefile<sup>5</sup> of the whole city region was provided along with sequences of road segments tagged with taxi identifiers and timestamps representing taxi movements. The whole city's shapefile was composed of 264 425 road segments; this analysis was restricted to a cropped region containing 114 755 segments.

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

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 commonly used haversine formula.

<sup>5</sup>A popular geospatial vector data format. It describes the geometry of the network as points and polylines representing roads

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 shapefile's layout had to be minimized and intermediate nodes were removed; the resulting network can be seen in Figure 5. Figure 5a shows a specific region of the original network, whereas Figure 5b 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 could be preserved. The graph of the resulting network contained 24 752 nodes and 69 450 edges; the effect of turning sinuous roads into straight lines is evidenced in the Figure.

In order to build the MDP, travel time between adjacent nodes had to be obtained. Even though shapefiles provide a very accurate measure of distance, travel time is likely to be a more important feature for taxi drivers. The network was further restricted to those segments with travel time information, further reducing the number of edges to be incorporated in the MDP.

Finally, travel time extraction from GPS records is a research problem deserving considerable attention from the research community [33], [4]. In this work, a simple approach was chosen:

- 1) The shortest path between consecutive records was obtained using Dijkstra's algorithm [34].
- 2) Using the time stamp 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.

The MDP was built by considering a period between 9 am and 10 am and discretizing it in 30-second's intervals. Each state represented a (location, time interval) pair and



Fig. 5: Comparison between the original shapefiles and the minimized network. Two roads in which this difference is particularly notorious have been marked in red.

were connected along the roads obtained in the previous steps. Unlike the previous case study, this time the model was not deterministic. For each connection, there was a probability of reaching a special state (representing a passenger pick-up) that later randomly placed the taxi driver back in the network. Pick-up probabilities for each edge were part of the data available in this study.

### B. Reward functions

Three simple reward functions describing taxi drivers' desires were implemented.

Function  $\phi_1$  is a penalty function directly proportional to the distance traveled, obtained directly from the shapefile.

Function  $\phi_2$  rewarded the taxi driver for each passenger he transported. This was achieved by rewarding transitions to special drop-off states that sent the taxi driver back to a random node in the network; the probability to go to this special drop-off state from a given edge was equal to the probability of picking up a passenger in such an edge.

Finally, function  $\phi_3$  directly penalized the stoppage time of taxi drivers, directly proportional to the time spent without moving.

These three reward functions were chosen to represent 3 extremes in the function space. While functions  $\phi_1$  and  $\phi_3$  penalize drivers for moving and not moving (respectively), function  $\phi_2$  tries to replicate what we expect to be the taxi driver's objective. In order to guarantee that variations (e.g. using kilometers or meters in function  $\phi_1$ ) do not influence the results, the reward functions were normalized.

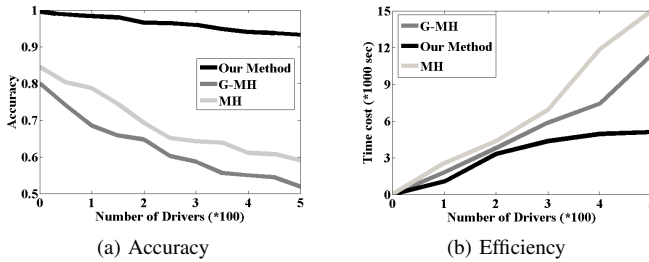


Fig. 6: Performance Evaluation

Taxi #	$\alpha_{idle}$	$\alpha_{passengers}$	$\alpha_{distance}$	Violated constraints	Accuracy (%)
1	0	1	0	15/856	98.3%
2	0	0.9994	0.0004	13/771	98.4%
3	0	1	0	28/706	96.1%
4	0	0.9651	0.0349	192/1113	82.8%
5	0	0.9879	0.0121	61/743	91.8%
6	0	0.9993	0.0007	11/818	98.7%
7	0.0039	0.9961	0	14/607	97.7%
8	0.9953	0.0047	0	15/940	98.4%
9	0	1	0	13/845	98.5%
10	0	0.9994	0.0006	12/1020	98.8%
11	0	0.9993	0.0007	14/609	97.7%
12	0	0.9920	0.0080	62/605	89.8%

TABLE II: Results of applying the MILP formulation to the different taxi drivers.

### C. Results and Analysis

Table II shows the result of 12 different sets of taxi records, each representing a different taxi driver. Violated constraints indicates how many constraints are violated in the final solutions (total number of constraints in the formulation). That is used to measure the accuracy (i.e. the percentage of states that the solution is able to explain). Time indicates how long the MILP formulation had to run in order to find a solution.

The results show that our formulation is able to predict with high accuracy what we empirically expected - taxi drivers do try to maximize the number of passengers. These results also show how sensitive the formulation is to the functions proposed; whenever the search space was big (the number of violated constraints was high), the algorithm was not able to terminate in an acceptable time. As a case study, our results show that this method can be used to validate conjectures about explanations to decisions in sequential environments. However, had we access to the relevant data, it is our belief that this case study could be expanded to discover new knowledge in this domain. As an example, one could test if the schedule of long-course trains or proximity to the local airport affect the taxis decisions. 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 paper would be important for both traffic planners and for the domain of behavior understanding. More details of our case studies can be found in [26].

### D. Comparison Study

Following last two studies, we compare our method with two baseline methods, MH and G-MH, proposed in [21] which are the latest and most related methods applying IRL in preference study (in our experiment, we follow the parameter training in [21]). We evaluate the efficiency and accuracy and illustrate the results in Figure 6.

In Figure 6a, the accuracy is defined as the percentage of states which can be explained by the resulting reward functions - the percentage of states in which what the reward function indicates is the optimal action, is in fact the action performed by the taxi driver.

In Figure 6b, the efficiency is evaluated by the time cost of running MILP formulation to find a solution. From the results, we can conclude that our method outperforms the previous methods in terms of accuracy, efficiency and scalability. The reason for such a performance is that 1) our reward function design can automatically find better reward functions and is able to allow relative comparison; 2) our MILP formulation is able to conduct fast computation.

## IX. CONCLUSIONS AND FUTURE WORK

This paper 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 and an alternative algorithm for Inverse Reinforcement Learning that provides more feedback than the alternatives. We utilize two case studies to illustrate our methods and evaluate our method compared with two baseline methods via real-life taxi drivers' behavior logs. The results show that our method outperforms the previous methods in terms of accuracy, efficiency and scalability. It has broad applications to human or animal's mobility and behavior analysis.

In the future, we will try to design a system which is able to automatically generate intelligible reward functions which are understandable and adaptive to the observed behaviors. On the other hand, we will try to retrieve the potential reasons behind the observed behaviors based on our discovered decision-making process.

**Acknowledgment** This research was supported by the T-SET University Transportation Center sponsored by the US Department of Transportation under Grant No. DTRT12-G-UTC11 and the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office. Part of this work was funded by the Fundacao para a Ciencia e Tecnologia under the Carnegie Mellon Portugal program. The authors also thank Teresa Galvao for valuable discussions regarding this work.

## REFERENCES

- [1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proc. of ICML*, 2004.
- [2] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In B. Scholkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19, Cambridge, MA, USA, 2007. MIT Press.
- [3] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proc. of ICML*. Morgan Kaufmann Publishers Inc., 2000.
- [4] Siyuan Liu, Yunhuai Liu, Lionel Ni, Jianping Fan, and Minglu Li. Towards mobility-based clustering. In *Proc. of ACM SIGKDD*, 2010.
- [5] Siyuan Liu, Ramayya Krishnan, and Emma Brunskill. How the urban taxi drivers collect decision knowledge. In *Proc. of WIN*, 2012.
- [6] Xueying Li, Huanhuan Cao, Enhong Chen, Hui Xiong, and Jilei Tian. Bp-growth: Searching strategies for efficient behavior pattern mining. 2012.
- [7] Po-Ruey Lei, Tsu-Jou Shen, Wen-Chih Peng, and Ing-Jiunn Su. Exploring spatial-temporal trajectory model for location prediction. In *Proc. of the 12<sup>th</sup> IEEE MDM*, 2011.
- [8] Md Reaz Uddin, Chinya Ravishankar, and Vassilis J. Tsotras. Finding regions of interest from trajectory data. In *Proc. of the 12<sup>th</sup> IEEE MDM*, 2011.
- [9] Yu Zheng and Xing Xie. Learning location correlation from gps trajectories. In *Proc. of the 11<sup>th</sup> IEEE MDM*, 2010.
- [10] Jing Yuan, Yu Zheng, Lihang Zhang, Xing Xie, and Guangzhong Sun. Where to find my next passenger. In *Proc. of the 13<sup>th</sup> UbiComp*, 2011.
- [11] Haiping Ma, Huanhuan Cao, Qiang Yang, Enhong Chen, and Jilei Tian. A habit mining approach for discovering similar mobile uses. In *Proc. of WWW*, 2012.
- [12] Valerie Guralnik and Karen Zita Haigh. Learning models of human behavior with sequential patterns. In *Proc. of the AAAIW*, 2002.
- [13] Flavio Chierichetti, Ravi Kumar, Prabhakar Raghavan, and Tams Sarls. Are web users really markovian? In *Proc. of WWW*, 2012.
- [14] Stuart Russell. Learning agents for uncertain environments (extended abstract). In *Proc. of COLT*, 1998.
- [15] Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. Maximum margin planning. In *Proc. of ICML*, 2006.
- [16] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *Proc. of IJCAI*, 2007.
- [17] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970.
- [18] Brian Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proc. of AAAI*, 2008.
- [19] 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 *Proc. of UbiComp*, 2008.
- [20] Brian D. Ziebart, Anind K. Dey, and J. Andrew Bagnell. Probabilistic pointing target prediction via inverse optimal control. In *Proc. of IUI*, 2012.
- [21] Constantin A. Rothkopf and Christos Dimitrakakis. Preference elicitation and inverse reinforcement learning. In *Proc. of the ECML PKDD*, 2011.
- [22] Shao Zhifei and Er Meng Joo. A review of inverse reinforcement learning theory and recent advances. *International Journal of Intelligent Computing and Cybernetics*, 5(3):293–311, June 2012.
- [23] R. Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(4):679–684, April 1957.
- [24] D. P. Bertsekas and J. Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, September 1996.
- [25] R. S. Sutton and A. G. Barto. *Reinforcement Learning: an Introduction*. MIT Press, 1998.
- [26] Miguel Araujo. Understanding behavior via inverse reinforcement learning. *M.Sc. Dissertation, Faculdade de Engenharia da Universidade do Porto*, 2012.
- [27] Richard M. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. 1972.
- [28] Jiri Matousek, Micha Sharir, and Emo Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4-5):498–516, 1996.
- [29] 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.
- [30] Siyuan Liu, Ce Liu, Qiong Luo, Lionel Ni, and Ramayya Krishnan. Calibrating large scale vehicle trajectory data. In *Proc. of the 13<sup>th</sup> IEEE MDM*, 2012.
- [31] Siyuan Liu, Yunhuai Liu, Lionel Ni, Minglu Li, and Jianping Fan. Detecting crowdedness spot in city transportation. *IEEE Transactions on Vehicular Technology*, (99):1–1.
- [32] Henry S. Farber. Reference-dependent preferences and labor supply: The case of new york city taxi drivers. *The American Economic Review*, 98(3), 2008.
- [33] Yanying Li and Mike McDonald. Link travel time estimation using single gps equipped probe vehicle. In *Proc. of IEEE ITS*, 2002.
- [34] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.