12

A Contextual Decision-Making Framework

Eugene Santos Jr.,[1,*] *Hien Nguyen,*[2] *Keum Joo Kim,*[3] *Jacob A. Russell,*[4] *Gregory M. Hyde,*[5] *Luke J. Veenhuis,*[5] *Ramnjit S. Boparai,*[5] *Luke T. De Guelle*[5] and *Hung Vu Mac*[5]

12.1 Introduction

In Proactive Decision Support (PDS), the goal is to provide Commanders with the information that they need at the right time in order to make the right decision while dealing with a large expanse of contextual information about an uncertain and dynamic environment. Providing the Commander with too much information degrades their performance in time-critical situations; while not providing enough information can lead to uninformed and poor decisions. To compound the problem, Commanders do not all make decisions the same way, implying that identifying the right time and the right information for a Commander requires an understanding of how the individual Commander makes decisions. A Commander's decision-making process, itself, may also change over time as they learn from the consequences

[1] Professor of Engineering at the Thayer School of Engineering, Dartmouth College in Hanover, NH.
[2] Associate Professor of Computer Science, University of Wisconsin, Whitewater, WI.
[3] Research Associate, Dartmouth College in Hanover, NH.
[4] PhD graduate student, Dartmouth College in Hanover, NH.
[5] Undergraduate students, University of Wisconsin-Whitewater, Whitewater, WI.
* Corresponding author

of prior decisions. Therefore, to proactively assist a Commander, we need to understand his goals, anticipate his needs, and capture his decision-making process to correctly choose the context (relevant, critical pieces of information and actions) he needs. Unfortunately, there is a gap between the technologies involved in building such a framework to support Commanders automatically. Decision modeling techniques only focus on how decisions are made while user modeling focuses on the activities and information seeking behaviors. In addition, the use of techniques such as Cognitive Task Analysis (Heuer, 1999) do not scale well and are often useful only in well-defined and well-scoped decision-making tasks.

We develop a Double Transition Model (DTM) (Yu, 2013; Yu and Santos, 2016) to capture a Commander's decision-making process and assess the model through evaluations on synthetic and human Commanders in Naval warfare scenarios plus UAV operators in training simulations. A DTM can be used to derive a dynamic Markov Decision Process (*d*MDP) in which each state reflects a cognitive state of a decision maker that can be discovered/encountered over time and described by his context and each action represents the tasks and decisions that he makes (e.g., during a battle), while the rewards represent the effect of a decision's outcomes.

The novelty of our approach lies in the development and use of a computational model to (re-)construct, quantify, and recognize a Commander's behaviors as a decision-making process. We evaluate our approach through three assessments: The first evaluation focuses on how well our approach determines how a Commander's rewards work using synthetic Commanders. The second evaluation uses the Supervisory Control Operations User Testbed (Coyne and Sibley, 2015) to study how well DTMs can capture a real-world decision-making process. The third evaluation focuses on how well we recognize different decision-making styles by analyzing the graph structures of their DTMs, their reward functions, and the sequences of cognitive states and actions they go through with three human Commanders. Our findings show that the rewards and Double Transition Model's structure capture how a Commander makes a decision. Projecting trajectories of different decision-making styles into the same space, may help distinguish their different rewards.

In this chapter, we seek to address the following fundamental research question: How can we capture a Commander's sequential decision-making process in a computational model and evaluate it in real world situations? The heart of our approach is the idea that an individual faces a sequence of episodic tasks each with different subsequences of environmental and internal context changes caused by the actions taken by the individual. This is essential to capturing how decisions made by the Commander impact later decisions. In this way, understanding the entire process is personalized.

This chapter is organized as follows: The next section reviews related work on decision-making styles, Markov Decision Processes (MDP), and

Inverse Reinforcement Learning (IRL). The following section is a discussion of prior work on the Double Transition Model. The approach taken is discussed next, including the creation of testbeds, the proposed Inverse Reinforcement Learning algorithm, and an evaluation of the algorithm with the testbeds. The chapter ends with a discussion of what was done, what research is expected to be done next, and where the approach could potentially be useful in the future.

12.2 Background

Supporting a Commander in his decision-making process requires a sufficient understanding of how people make decisions in general. Once it is understood, a model can be built, and the learning of their decision-making style can then take place. This section describes the background knowledge that is needed to build our model: decision styles, Markov Decision Processes, and Inverse Reinforcement Learning.

12.2.1 Decision-Making Styles

A Commander's decision-making style represents how he processes data, makes sense of the received data and evaluates possible alternatives. Decision-making style has been defined as "the learned habitual response pattern exhibited by an individual when confronted with a decision situation" (Scott and Bruce, 1995). This area of research has extensively been studied in variety of domains including education management (Galotti et al., 2006), military (Thunholm, 2009), pharmaceutical training (McLaughlin et al., 2014), construction work (Esa et al., 2014), and work life balance (Michailidis and Banks, 2016). Researchers have shown that determining one's decision-making style is a crucial step to evaluating and distinguishing between good and poor decision-makers and helps improve the quality of decision-making process.

The five decision making styles proposed by Scott and Bruce (1995) have been investigated and applied in the cognitive community over the years. These are rational, intuitive, dependent, avoidant, and spontaneous decision styles. Rational decision makers consider all options and logically make a choice by reasoning over all options. Intuitive decision makers rely heavily on their own experience and choose heuristics that work best for them. Dependent decision makers rely on others' advice and decisions. Avoidant decision makers postpone the act of making decisions. Finally, spontaneous decision makers tend to make snap, quick decisions to respond to immediate needs. Similar to intuitive decision makers, in a battle setting, a naturalist decision-making style also refers to the use of past experience to make critical decisions under a lot of pressure and time constraints (Flin, 2001). In a study with 98 army captains, Thunholm (2009) has found that

military leaders tend to be more spontaneous and less rational, dependent, and avoidant than their staff members. Decision-making style is a very important factor to assess the quality of a decision and is also related to a broader concept that has been studied thoroughly, called cognitive style (Syagga, 2012). Syagga explored a specific relationship between cognitive styles and decision-making style such as "*People who are more extroverted in personality are more likely to have intuitive cognitive style, while those who are more introverted in personality are more likely to have analytic cognitive style.*" Syagga's study could not confirm the relationship, however, it could not reject the relationship and lays the groundwork for further study. In this work, we aim to identify the differences of decision-making styles by using a computational framework to capture Commanders' entire decision-making processes and analyze the Commanders within this framework.

12.2.1.1 Markov Decision Processes

The Markov Decision Process (MDP) was used as early as 1957 by Bellman (Bellman, 1957) and popularized by Howard (1960). Originally used in operations research, MDPs branched out and have been used in a variety of fields, including robotics, controls, and economics. They are typically used to model decision-making in well-defined scenarios where they can aid in making optimal decisions.

A MDP M is a 5-tuple $(S, A, P.(\cdot,\cdot), R.(\cdot,\cdot), \gamma)$, where S is a finite set of states. A is a finite set of actions. $P_a(s, s') = Pr(\{s_{t+1} = s' \mid s_t = s, a_t = a\})$ is the probability that by taking action a from state s at time t, we will transition to state s' at time $t + 1$. $R_a(s, s')$ is the immediate reward (often relaxed as the expected immediate reward) received after action a transitions from state s to state s'. Finally, $\gamma \in [0,1]$ is the discount factor, which controls for the ratio of importance between future rewards and present rewards. The formulation makes MDPs simple to design and use when in a well-defined space.

Conventionally, MDPs have a well-defined reward function and are looking for a policy π: $S \rightarrow A$ which determines which action should be taken at any given state. More specifically, they are looking for the optimal policy π^* which maximizes the expected reward of the system. The reward function is usually hand-crafted because it is unknown. An alternative approach to hand-crafting the reward is to approximate it. In the cases where the reward function is unknown, we could perform Inverse Reinforcement Learning to approximate it.

12.2.1.2 Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL), sometimes called Inverse Optimal Control, was originally proposed by Russell (1998) in a position paper, as a means for estimating how an individual or machine could assign preferences

for states when they were unknown. The first formulation was then defined by Ng and Russell (2000). IRL takes a MDP without a reward function $M \backslash R.(.,.)$, and approximates a reward function. In order to approximate the reward function, Ng and Russell's approach can take as a parameter the optimal policy π^*, but in practice the intent is to learn both the optimal policy and the reward function so they use a set Ξ_M of trajectories of the form $\vartheta = \{s_i, a_i, s_{i+1}, a_{i+1}, \ldots, s_{i+k}\}$. These trajectories are examples of the optimal behavior as demonstrated by an expert. This approach assumes that the reward function is linear and solves for the rewards via linear programming. Unfortunately, Ng and Russell's approach is under-constrained and has multiple optimal solutions, some of which are degenerate, and it is the luck of the draw as to whether you get the degenerate solutions.

Maximum Entropy (Maxent) IRL (Ziebart et al., 2008) maximizes the entropy of the system to have a single optimal solution solved via gradient descent. Unfortunately, in this work, Maxent did not converge and was fickle with respect to the hyperparameters. Maxent also happens to be the main approach upon which most other new approaches to IRL are based.

Deep IRL (Wulfmeier et al., 2015) extends the approach from Maxent but trains a fully convolutional neural network (FCNN) to learn a nonlinear reward function capturing the relationship between features on the states instead of the reward function by Ng and Russell which was linear in the features. This approach is promising, but FCNNs are still difficult to explain because of their high dimensionality of parameters and nonlinear interaction between the parameters. This poses a significant barrier to better understanding the decisions made by a Commander.

12.3 Prior Work

In this section, we introduce the DTM which is a framework that uses MDPs to model an individual's decision-making style(s) and reconstructing the way that they view the world via IRL.

12.3.1 Double Transition Model

To capture the human decision-making process, we begin by assuming that the process can be determined by a sequential choice of actions and demonstrated figuratively through connections between cognitive states. The Double Transition Model (DTM) was originally proposed in Yu (2013) as a way of describing a human opinion formation process through computational simulation and applied to multiple domains (Yu and Santos, 2016; Santos et al., 2017). Nodes are used to represent human cognitive states and edges to represent their transitions during decision-making processes. A node (representing a cognitive state) is composed of two subgraphs, a Query Transition Graph (QTG) and a Memory Transition Graph (MTG).

The QTG is specialized for the instant interest of an individual and the MTG is for the memory transition associated with the QTG. Each node in the QTG represents a single query in the individual's mind and a set of random variables associated with it can be described as $U = [X, A, B, C, ...]$. A vector $[X, ?, b_1, c_2, ...]$ denotes a single instantiation of the set of random variables and represents an instant query of the individual, while X represents the target random variable of interest, and ?, b_1 and c_2 denote the unknown/known instantiation of variables $A, B, C \in U$, respectively. The MTG represents the underlying knowledge of the individual's mind represented by a Bayesian Knowledge Base (BKB) (Santos and Santos, 1999) stored on the state. This knowledge base can also be thought of as a representation of the relevant context and how different sets of context relate to each other.

Figure 12.1 shows a portion of a sample DTM, where an individual could undergo a memory transition from one subtask to another based on the change of his perception through MTGs. During the memory transition, he would ask himself a question: "Is there any action necessary for a current subtask?", and he takes an action to speed up one of his Unmanned Aerial Vehicles (UAVs) as he believes it is the right action to take at that time.

The random variables (such as 'Action' and 'event'), their relationships in the DTM, and the associativity between QTG and MTG cannot be known *a priori*. Cognitive states and relationships must be identified and extracted from the individual's information stream to identify the relevant context and incorporate it into the MTG and QTG. Transitions between cognitive states depend on the choice of action at the current instant. The ultimate goal of our research here is to reconstruct the sequence of decisions that the individual would make and assist his future decisions by investigating potential patterns of his decisions. For assisting Commander's decision-making processes, DTMs have been applied to previous work (Yu and Santos, 2016; Santos et al., 2017).

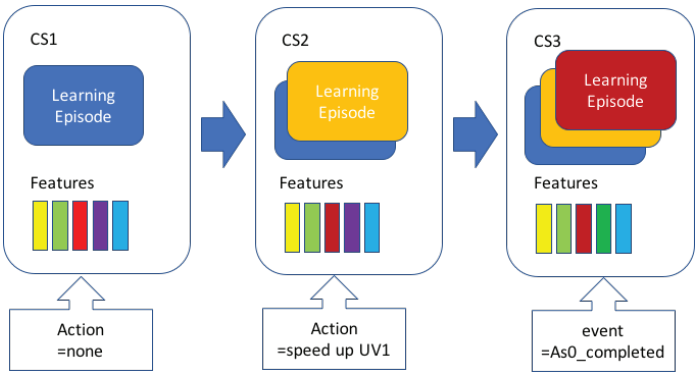

Figure 12.1. An example of DTM framework.

Here, the DTMs have been limited to just the MTG and the BKBs have been ignored in order to simplify the problem of identifying reward values for the states. A DTM at any given point in time has a mapping onto a MDP where the cognitive states and transitions in the DTM have a one-to-one mapping onto an equivalent MDP. Since the DTM is controlling and updating itself to correspond to the decision-maker being modeled the state and action spaces cannot be known *a priori*, and therefore must be dynamic. For this reason, we refer to the MDP derived from the DTM as a dynamic MPD (*d*MDP). Most of this chapter focuses on the *d*MDP derived from a snapshot of the DTM, however, it generalizes to instances where the state and action space could change.

12.4 Approach

In order to capture a Commander's decision-making process with a *d*MDP, a set of trajectories (which are representative of the decision-making process) need to be sampled from each Commander (thus forming a testbed). This section focuses on the capture and analysis of Commanders' decision-making styles and the development of a requisite, new Inverse Reinforcement Learning (IRL) algorithm. To start, it is necessary to understand limitations from previous iterations of IRL and testbeds created for evaluation. Conventionally, IRL algorithms and testbeds are used in domains such as robot motion planning which are relatively objective, but when dealing with people we do not want to regress towards the mean over all of them and lose their individuality. The new IRL algorithm tries not to make some of the assumptions common in domains such as robot motion planning. Therefore, this section begins by explaining how the testbeds were created and what types of information were included. We then discuss limitations of existing IRL algorithms and define the new IRL algorithm. We finish this section with an evaluation of the new IRL algorithm over our testbeds.

12.4.1 Testbed Construction

Due to the lack of publicly available traces of Commanders' decision-making processes or trajectories, we began our research by first synthesizing testbeds from virtual Commanders playing a simple toy game. Next, we extended to two more testbeds. The first one is the Supervisory Control Operations User Testbed (SCOUT) (Coyne and Sibley, 2015) which is under current development and analysis from the US Naval Research Laboratory. The second one is obtained from human Commanders played by students in a naval warfare simulation game.

12.4.1.1 Synthetic Commanders

Evaluation of IRL algorithms requires sufficient numbers of observed trajectories of Commanders' decisions in battlefield scenarios. Aside from the current lack of availability of such trajectories, evaluation is also not simple with human Commanders because the human Commanders' decision-making process is not fully observable and decisions about what context to include from a Commander's observations may be subjective. Furthermore, the data gathered from the human Commanders may not be repeated frequently enough to evaluate algorithm performance. In order to cope with these challenges, we designed synthetic Commanders associated with representative decision-making styles. We synthesized the Planner for a rational decision-maker and the Novice for a spontaneous decision-maker. The next few paragraphs introduce the world of actions that the Commander can take to contextualize the synthesized Commanders and then describe the design of the Commanders themselves.

The world is composed of two enemy ships, three friendly ships, and friendly scout planes. The goal is for the Commander to control the ships in a way that eliminates the enemy ships. In between actions that the Commander can take, snapshots of what the world looked like are visible and define the context of the space. These snapshots contain full instantiations of the variables in Table 12.1.

The Planner rationally chooses each action in a way that attempts to accomplish the goal of eliminating the enemy ships effectively and efficiently. When the enemy ships are concealed, they have a tactical advantage so it is advantageous to move multiple ships at a time to attack them. The design was meant to simulate an expert that knows how to traverse the domain.

The Novice makes some mistakes and can be thought of to be exploring the space at the same time as making the decisions. Generally, the novice will send one to two ships at a time to explore and will be at a tactical disadvantage (although the number of friendly ships is larger than the number of enemy ships so it is still winnable).

Table 12.1. Description of variables in synthetic Commander testbed.

Enemy ships	Variables (per ship) representing the location of the enemy ships
Fog 1-4	Variables representing explored or unexplored fog areas where the enemy ships may be hiding
Health	Variables for the health of each of the enemy ships (submarine and battleship) and each of the friendly ships (battleship 1, battleship 2, scout planes, aircraft carrier)
Friendly ships	Location of each of the friendly ships

The goal of the testbed is to evaluate the IRL algorithm under more appropriate conditions for our problem than the classic Grid World problem (Russell et al., 2003). Features in these other domains have no correlations with each other and can be treated independently, counter to what we expect to see. Since one of our goals is to address dynamism, having correlations between the features is important because we cannot make them orthogonal for factors that we have not yet seen. We use this synthetic toy problem as a means for testing before scaling up to real data such as the Supervisory Control Operations User Testbed.

12.4.1.2 Supervisory Control Operations User Testbed

This dataset was collected by the US Naval Research Laboratory from 20 volunteers executing pre-scripted Unmanned Aerial Vehicle (UAV) missions within the Supervisory Control Operations User Testbed (Coyne and Sibley, 2015). The testbed was developed to represent the tasks that operators would perform in dealing with multiple heterogeneous UAVs. Participants are encouraged to plan routes and navigate effectively by earning or losing points during twenty-minute long supervisory control missions. Each participant gets rewarded once the assigned vehicle is located within the range of the target sensor, and penalized when the vehicle enters a restricted airspace. In addition to this main task, participants are asked to respond to chat messages, which include requests to update UAV parameters (geographical locations, speed, altitude, etc.). The available context includes the location of the UAV, the score that Commanders accomplish, the type of tasks given, level of difficulty and biometric attributes of Commanders such as their pupil dilation during the mission. With the information provided, we apply our theoretical framework of DTMs into the task of building a single, unique DTM describing all Scout participants, and examine how the context embedded in the original dataset is represented in the DTM constructed. In particular, we focused on the context associated with individual characteristics such as risk-aversion, selection efficiency, and game performance.

12.4.1.3 Human Commanders

In addition to testing with synthetic users and UAV pilots, we use a team of three humans to play a Naval war game called Steel Ocean.[1] This game was selected because of its wide availability and its community of users which offer many videos with readily available examples of Naval Commanders with concrete traces. These episodic traces help us to create a platform to understand how Commanders differ in their decision-making process, why they differ, and what uniquely composes a Commander's

[1] Steel Ocean is available at http://store.steampowered.com/app/390670/Steel_Ocean/.

style over a set of tasks while still having control of how the tasks overlap. We created and recorded videos of 32 scenarios. Each scenario was played by three student volunteers who have some experience playing wargames online and involved a fleet battle between two teams: allies and enemies. One player acts as the Commander and the other two players act as fleet members. Depending on the role a person is filling, they have different ship types to choose from. Commanders use a battleship whereas the other two members use destroyers. Contextually speaking, we have three fundamental pieces in each scenario: A Commander's ship, a Support ship and an Attacker ship. A Commander's ship makes decisions predicated in part on context from the actions that the Attacker or the Support ship take. The Attacker ship is responsible for carrying out an action and the Support ship assists the Attacker ship in carrying out these actions. These pieces in each battle follow different formations in scouting the enemy and carrying out offensive and defensive actions. The formations are straight line, scattered line, triangle and D-formation as specified by United State Joint Force Command (United States Joint Forces Command, 2004). While we limit ourselves to these two specific ships, other ships do exist within the game such as submarines or aircraft carriers which enemy or ally players can use. The Commander in our simulation is the only player allowed to make any sort of decisions. The two other fleet members are allowed to make requests to the Commander which can ultimately influence the decisions a Commander makes. Along with requests, fleet members are expected to give reports to the Commander, so that the Commander has the context required to make informative decisions for their fleet. To communicate with each other in a battle, the three players use a VoIP server which allows the Commander to give verbal commands in real time to the fleet members. The fleet members are allowed to give reports and requests in real time to the Commander. We established a protocol to standardize our verbal communication. Narratives are compiled from recorded videos as we sample the movements and situations of each battle every 15 seconds and convert this data to a comma delimited file (csv). Each comma delimited file is a trace which, in turn, is used to create a portion of the DTM.

Each human Commander is also given a questionnaire (25 questions) which is used to measure their general decision-making styles (Scott and Bruce, 1995). There are five questions representative of each style and each question is rated on a scale from 1 to 5 with 1 as the lowest and 5 as the highest. We tallied the points for each group of questions for each style and determined the final style being the ones with highest scores. The results are shown in Table 12.2 below.

In summary, the testbed with Human Commanders brings several new dimensions of context to assess how the DTMs captures a Commander's decision-making process which are ground truth for Commanders' decision-

Table 12.2. Testbed of human Commanders.

Commanders	Decision Style	Number of Commanding Battles	Average Battle Length (in seconds)
Commander 1	Rational	16	643.5
Commander 2	Spontaneous and avoidant	8	542.87
Commander 3	Rational and dependent	8	518.87

making styles, and partially observable, dynamic environment with real-time actions.

12.4.2 A New Inverse Reinforcement Learning Algorithm

This work encountered limitations with existing IRL algorithms on the previously described testbeds. Here we describe the limitations briefly in order to motivate our algorithm but discuss them in more detail in the evaluation section.

The first prohibitive limitation that we encountered is that the definition of rewards in the original Ng and Russell (2000) approach yielded rewards on the states themselves. This is problematic because, when combined with the Markov Assumption, it implies that rewards are invariant to the order in which states are traversed. The assumption of the linear relationship between the features is also problematic and we will discuss this limitation in the Synthetic Commanders section.

In prior work (Santos et al., 2017), we used the Maximum Entropy Inverse Reinforcement Learning (Maxent) approach (Ziebart et al., 2008) because it yielded a, seemingly, better distribution of rewards than the Ng and Russell approach and because most of the other recent approaches build upon it. However, Maxent seems to be sensitive to the sampled trajectories being optimal, which is unlikely to be true in a dynamic environment like the ones that we are trying to work within. If the actions are not optimal, then there may not be a deterministic policy and Maxent may not converge. We also find that the linear function learned by Maxent does not hold in a dynamic environment which we take as evidence that we need a nonlinear reward function. Further analysis of Maxent appears in our Synthetic Commander testbed.

The implication of this analysis is that Commanders may have a nonlinear reward function or, in other words, the context is not defined by the full state space at every point in time. To remedy this, we could use an approach such as Deep Maximum Entropy Inverse Reinforcement Learning (Deep Maxent) (Wulfmeier et al., 2015), however, to recap, our end goal is explanation which Deep Maxent does not provide.

In order to design a new algorithm, it is necessary to avoid these common assumptions from existing IRL methods:

1. Maximum Entropy IRL does not account for the fact that the trajectory space itself is not complete, it is a sampling, and therefore it may not be representative, particularly in relatively unseen states.
2. We cannot make an assumption regarding linear combinations of states (or features) as any basis for determining rewards.
3. Trajectories may not be equally likely, there can be no assumptions on the distribution of the rewards for trajectories.
4. The frequency of state visitation, particularly with respect to features, may not correlate with reward and may correlate more with inevitability.
5. Trajectories do not have to be optimal nor do they need to be totally ordered.
6. There is an ordering (total or partial) of the importance of observed trajectories from the decision-makers.

We developed a new algorithm avoiding these assumptions while keeping any assumptions to a minimum. To recap IRL, the problem is when given a partially defined MDP $M\backslash R.(.,.)$, where only the reward function, $R.(.,.)$, is undefined, and a finite set Ξ_M of trajectories of the form $\vartheta = \{s_i, a_i, s_{i+1}, a_{i+1},\ldots, s_{i+k}\}$ that represent our observations of the decision-maker behavior, we wish to compute a reward function $R.(.,.)$ that reflects Ξ_M. To do so conceptually, our only assumption regarding the space of all trajectories in M are that trajectories in Ξ_M should be more important than other alternative trajectories not in Ξ_M but "close" to some trajectory in Ξ_M. We now provide our specific formulation of this concept for our IRL. First, given trajectories

$$\vartheta = \{s_i, a_i, s_{i+1}, a_{i+1},\ldots, s_{i+k}\} \tag{12.1}$$

and

$$\vartheta' = \{s_i, a'_i, s'_{i+1}, a'_{i+1}, s'_{i+2}\ldots, s'_{i+k}\} \tag{12.2}$$

from M is said to be a c-neighbor of ϑ if

$$|\{s'_j \mid s'_j \neq s_j, j = 1,.., k\} \cup \{a'_j \mid a'_j \neq a_j, j = 1,\ldots,k\}| = c \tag{12.3}$$

where c is a positive integer. Let $\Theta_c(\vartheta)$ be the set of all c-neighbors of trajectory ϑ in M. For some constant positive integer ξ, define $\overline{\Theta}_\xi(\vartheta) = \cup_{c=1}^{\xi} \Theta_c(\vartheta)$.

Given some trajectory $\vartheta = \{s_i, a_i, s_{i+1}, a_{i+1},\ldots, s_{i+k}\}$ from M, we define the expected value of ϑ as follows:

$$E(\vartheta) = \left(\prod_{j=i}^{i+k-1} P_{a_j}(s_j, s_{j+1})\right)\left(\sum_{j=i}^{i+k-1} R_{a_j}(s_j, s_{j+1})\right). \tag{12.4}$$

We also refer to Eq. 12.4 as the Trajectory Probability*Reward Sum (TP*RS). In other contexts we use just the Trajectory Probability ($\prod_{j=i}^{i+k-1} P_{a_j}(s_j, s_{j+1})$), Reward Sum ($\sum_{j=i}^{i+k-1} R_{a_j}(s_j, s_{j+1})$), or the expected linear reward sum:

$$E(\vartheta) = \sum_{j=i}^{i+k-1} P_{a_j}(s_j, s_{j+1}) R_{a_j}(s_j, s_{j+1}) \alpha^{j-i} \tag{12.5}$$

where $\alpha \in [0,1]$ a constant discount factor.

Returning to our single assumption on the trajectory space, we have the following: For each $\vartheta \in \Xi_M$, we introduce the following constraint:

$$\left|\overline{\Theta_{\xi}}(\vartheta) - \Xi_M\right| E(\vartheta) \geq \sum_{\vartheta' \in \overline{\Theta_{\xi}}(\vartheta) - \Xi_M} E(\vartheta'). \tag{12.6}$$

These constraints are linear in terms of $R.(.,.)$. We allow this constraint to be violated but wish to penalize such violations. We rewrite this as follows:

$$\left|\overline{\Theta_{\xi}}(\vartheta) - \Xi_M\right| E(\vartheta) = \delta_{\vartheta}^{+} - \delta_{\vartheta}^{-} + \sum_{\vartheta' \in \overline{\Theta_{\xi}}(\vartheta) - \Xi_M} E(\vartheta') \tag{12.7}$$

where δ_{ϑ}^{+} and δ_{ϑ}^{-} are unique variables to each ϑ that capture the slack and possible violation of the constraint. We also guarantee that they are non-negative with the constraints $\delta_{\vartheta}^{+} \geq 0$ and $\delta_{\vartheta}^{-} \geq 0$.

Next, we introduce a positive constant ω called the peak magnitude value where for all reward instances $R_{a_j}(s_j, s_{j+1})$, we introduce the bounds

$$-\omega \leq R_{a_j}(s_j, s_{j+1}) \leq \omega. \tag{12.8}$$

Lastly, to complete our linear programming formulation for our new IRL, we define our objective function as

$$\sum_{\vartheta \in \Xi_M} \delta_{\vartheta}^{+} - \delta_{\vartheta}^{-}. \tag{12.9}$$

Our goal is to maximize this objective function. Thus, our new IRL approach can be solved as linear programming problem which admits highly efficient algorithms such as Simplex and those for convex optimization. Again, unlike the prior IRL formulations, we make no assumptions about the type, or existence, of relationships between features and can avoid degeneracy as long as the trajectories can generate neighborhoods (sets of c-neighbors, or alternatively, branching neighbors as per below) with enough constraints.

The generation of the neighborhoods for the alternative trajectory set is critical because it is a characterization of the space of potential sequences of actions that the Commander could have taken. The selection of neighborhoods, therefore is giving a semblance of what options were possibly available to take as actions at any given state. These alternative trajectories then yield different constraints and different representations of

the expected possible trajectory space. In addition to the c-neighbors, we also try another method for neighborhood generation called branching.

Branching neighborhoods are formed as follows for each training trajectory $\vartheta = \{s_i, a_i, s_{i+1}, a_{i+1}, \ldots, s_{i+k}\}$: For each prefix trajectory of ϑ, say $\{s_i, a_i, s_{i+1}, a_{i+1}, \ldots, s_{i+l}\}$ where $0 \leq l < k$, we generate a random walk starting at s_{i+l} in the DTM for up to $k - l$ steps. Thus, each trajectory can have up to k neighbors through branching. Let $\Theta_b(\vartheta)$ be such a set of branching neighbors.

We can replace c-neighbors with branching neighbors, i.e., replace $\overline{\Theta}_\xi(\vartheta) = \cup_{c=1}^{\xi} \Theta_c(\vartheta)$ with $\Theta_b(\vartheta)$

$$\left|\overline{\Theta_\xi}(\mathcal{G}) - \Xi_{\mathrm{M}}\right| E(\mathcal{G}) = \delta_{\mathcal{G}}^{+} - \delta_{\mathcal{G}}^{-} + \sum_{\mathcal{G}' \in \overline{\Theta_\xi}(\mathcal{G}) - \Xi_{\mathrm{M}}} E(\mathcal{G}') \tag{12.7}$$

To

$$\left|\Theta_b(\mathcal{G}) - \Xi_{\mathrm{M}}\right| E(\mathcal{G}) = \delta_{\mathcal{G}}^{+} - \delta_{\mathcal{G}}^{-} + \sum_{\mathcal{G}' \in \Theta_b(\mathcal{G}) - \Xi_{\mathrm{M}}} E(\mathcal{G}') \tag{12.10}$$

Branching is expected to be useful in cases where there are few trajectories in order to expand the trajectory space.

Note—Given that our optimization problems are linear, the selection of the peak magnitude value for each instance of $R.(.,.)$ is theoretically irrelevant as different peak magnitudes correspond to a linear scaling of the feasible space and optimal objective value.

We still, however, have a possible problem that our state space is not separate from our feature space. While we ignore the features when learning the reward values, the cross product of the features defines an inequality of states with (possibly) irrelevant or noisy features. This means that there's an implicit dependence upon exact feature equivalency in the learning. As such, state merging is our initial attempt to abstract or generalize different states in the DTM. The idea is to merge states, resulting in drawing trajectories "closer" together and allowing for further analyses of a commander's decision-making style by interleaving trajectories.

Given a DTM, we simply define the distance between two trajectories to be the shortest path between any two states from one trajectory to the other. We further modify our distance definition as follows: Given two trajectories that share a common prefix, we remove the prefix from each trajectory. If the truncated trajectories begin with an action, delete the starting action from both truncated trajectories. If the two truncated trajectories are empty, then the distance is 0. Otherwise, the distance between the two trajectories is now defined as the distance between the two truncated trajectories. A distance of zero now indicates that there is a shared state between the two trajectories that does not occur in their common prefix. Our goal is to address situations where a common starting state for decision-making

results in a delay until a different decision is made—e.g., the situation is common among "standard opening moves".

Next, we define the distance between two states s_i and s_j, denoted $d(s_i, s_j)$, in the DTM as the sum over each feature γ in the two states as follows:

- If γ is not set in both s_i and s_j, add 1.
- If γ is non-numeric but has the same value in both s_i and s_j, add 0.
- If γ is non-numeric but has different values in s_i and s_j, add 1.
- Otherwise γ is numeric, add $(\gamma_i - \gamma_j)^2$ where these are the values in s_i and s_j, respectively.

The ordered pairs of trajectories are ranked in descending order of distance for non-zero distances. The top n% (for this paper, we chose n = 90) of this ranking. For each ordered pair ϑ and ϑ' in the top n%, let s_i and s_j' be two states in ϑ and ϑ', respectively, such that $s_i \neq s_j'$ and $d(s_i, s_j')$ is smaller than any other state distance pairs.

Once each s_i and s_j' are determined for each trajectory pair in the ordered ranking, we now merge the states in the DTM as follows: Let Q be a relation on the states where sQs' if and only s and s' is one of the state pairs for some ranked trajectory pair. Q induces a partition on the states of the DTM. For each partition q in Q where $|q| > 1$, for each state $s \in q$, construct $\widetilde{P}_a(s, \hat{s}) = \sum_{s' \in q} P_a(s', \hat{s})$. Next, normalize $\widetilde{P}_a(s, \hat{s})$ so that it is probabilistic. Introduce or replace $\widetilde{P}_a(s, \hat{s})$ into the DTM. Merger thus allows for transitions across trajectories which allows for generation of additional neighbors. Merging is done before IRL is conducted.

12.4.3 Evaluation

The overall goal of this evaluation is three-fold. First, we assess whether the new IRL algorithm could address the problems the previous IRL algorithms have on the Synthetic Commander testbed. Second, we evaluate how the DTM built from the Scout dataset could reflect the original context to describe individual differences. Third, we assess how our DTMs could help us identify different decision-making styles on the human Commander testbed.

These analyses need to control for different factors in the model and align them with the testbeds individually to perform targeted analysis. For the SCOUT and Human Commander testbeds, we would like to be able to directly compare Commanders. Unfortunately, the DTMs have no information about states that they have not seen, so they cannot be directly compared. To be able to perform these analyses we build a joint DTM using the union of the trajectories from each Commander. This projects each Commander's reward values into a space in which they can be directly compared. In this projected reward space, we compare the Commanders

decision-making styles and generalize over the unique characteristics of the testbeds.

12.4.3.1 Assessment with Synthetic Commanders

The Synthetic Commander testbed was designed to identify and test possible limitations of existing IRL algorithms and verify whether those limitations could be addressed by the new IRL algorithm(s). The main limitations addressed in this testbed are the linear assumption between features and the Markov Assumption in linking rewards to features on the states. The new IRL algorithm is intended to rectify these limitations by simply not using the features as a consideration, to avoid making the linear assumption, and learning the rewards on the edges instead of on the states (which we refer to as 3-tuples or triples (s, a, s')).

The most promising previous approach to us seemed to be Maxent (Ziebart et al., 2008). Maxent, like Ng and Russell, learns a linear function of rewards with coefficients $\boldsymbol{\theta} = [\theta_1, \ldots \theta_n]$, so it seems to make sense to evaluate the reward function's fit as we would evaluate an arbitrary linear regression model. Ideally, we would determine whether the fit itself is significant. However, we cannot compute an R^2 value because we have no ground truth and do not know which states the Commander would consider better or worse. The other significance test for a linear model is a test of whether the coefficients θ_i, themselves, have a significant impact on the overall model. This is performed by holding out θ_i and determining whether the reward value assigned is significantly different from the original reward. If the distribution of the rewards without the term is the same, then that term is fitting noise in the reward function and the model is unlikely to generalize if any coefficient is not significant. We held out the rewards over the Novice Commander and performed a t-test to compare the distributions, finding that 44 out of 74 coefficients were significant at a $p \leq 0.05$ level. This implies that 30 of the coefficients had no significant impact on the reward distribution and the model was likely not a very good fit (i.e., the model's fit was not significant). The t-test, however, is testing whether the value of the rewards changed by a significant amount but the magnitude of the reward values by themselves is not interpretable. Instead, we want to compare the ordering of the reward values in holding out θ_i versus the rewards with θ_i kept intact. We compared this using a Spearman Rank Order Correlation between the two distributions for each of the θ_i values held out and found that the distributions were correlated in 74/74 cases and the correlation was significant at a $p \leq 0.05$ level with a Spearman $r = 0.99$, implying that no coefficient ever had the ability to change the reward value by itself and therefore no coefficient was significant by itself. This is a strong indication that a linear model is not a good fit.

Conclusions from the analysis of Maxent are that it assumes that the features from states that are visited more frequently should have a higher reward. Therefore, Maxent increases the value of feature rewards proportionally to how often the containing state is visited. This is intuitive; however, it yields a bias towards more frequently occurring features. When the feature space is small and all the features are hand-picked to be important, this assumption may hold. Our goal is dynamic scenarios where feature relevance is not curated. In our case many features rarely change because they may not be relevant and it is ultimately a part of our task to identify which ones were relevant. As the number of features in the testbed grows, it becomes less likely that a linear model fit will do well. The number of features is expected to grow when modeling Commanders because they regularly encounter new situations, so this assumption is prohibitively expensive.

This informed the design of the IRL algorithm and biased it towards the need to accommodate nonlinear features. Explicitly making the features nonlinear would be likely to overfit so, instead, we ignore the features altogether, and nonlinear or linear features may be learned by the model. We identify some very simple structures that Maxent and the Russell and Ng approaches cannot handle as shown in Figure 12.2.

In the connections between nodes s_1 and s_2, the edges are directional because transitions between the states could have appeared either way within the trajectories. Having reward values assigned to the states implies that you can only have a symmetric relation and the weights of edges a_1 and a_2 must be equal because it is just a sum. In placing the rewards on the edges (s_1, a_1, s_2) and (s_2, a_2, s_1) we have no need to lose that information because the edges maintain the context of the prior state. The structure appearing in the left graph of Figure 12.2 appears 206 times where there are different rewards for edges a_1 and a_2 (~ 19% of all edges) in the Planner's DTM and 1420 times (~ 20% of all edges) in the Novice's DTM because the

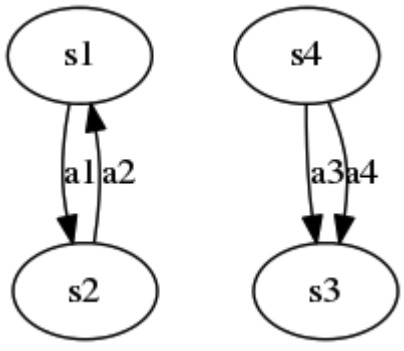

Figure 12.2. Problematic double transition model structures for conventional inverse reinforcement learning algorithms.

DTMs were constructed exhaustively. There seemed to be little difference between the c-neighbors and the branching neighbors at the node level in the graph. There may be variation amongst the relationships between the features, but it is nonlinear and analysis requires further interpretation.

In the right graph of Figure 12.2 we can see that there are 2 edges from s_4 to s_3 by taking 2 different actions a_3 and a_4. In the reward-on-state approach, the only way to differentiate these actions would be the probabilities of having taken the action.

The Synthetic Commander experiments demonstrated that we both need and have a means to learn nonlinear rewards in terms of the states and that we have potential to get a clear benefit from learning nonlinear rewards.

12.4.3.2 Supervisory Control Operations User Testbed

In this section, we gathered the context related to individual characteristics from the original Scout dataset and examined how well they were demonstrated in DTMs. To this end, we hypothesized the following based on the assumptions stated:

A1: A set of features collected from the original dataset is appropriate context to describe individuals.

A2: A trajectory in the DTM can be fully represented by the properties including trajectory probability, sum of rewards, expected linear sum and the product of probability and rewards.

H: If an appropriate DTM was built from a group of individuals' behaviors (trajectories), there should be some metric to reflect appropriate context to describe individuals in the DTM.

Each participant is involved in two consecutive sessions which correspond to two trajectories per participant. For individual characteristics, we addressed selection efficiency (Vogel et al., 2005), risk aversion (Thomas, 2016), and game performance. For selection efficiency, we collected the information regarding the number of actions, events and messages and computed the average repeated occurrence of unique instantiations in those fields during each session. For risk-aversion, we counted the number of actions violating restrictions. Therefore, the smaller the number of restricted operating zones (ROZ) violations the more risk-averse the individual was while playing the session. For game performance, we collected the maximum score an individual achieved during a session shown in Table 12.3. As stated by A1, the diverse range of attribute values in each column supports its fitness as the context to describe individuals.

For building the DTM describing Scout participants, we selected a set of 9 features in Table 12.4, which were identified as important out of 194

features through recursive partitioning and tree-based methods (Therneau et al., 2017). For the attributes having continuous values, we discretized them into multiple levels (i.e., 2, 5, 10 and 20). Due to the sparsity of the original data, we collected the observations only when the value of "ScoreChange" was meaningful (i.e., neither NA nor empty), and missing values were substituted by the observables identified at the closest time stamp.

In order to validate our hypotheses, we constructed a joint DTM from the 40 learning trajectories obtained from the 20 participants, and each trajectory was composed of difference numbers of cognitive states. We collected the derived trajectory properties from our IRL algorithm (i.e., trajectory probability, reward sum, expected linear reward with discount and trajectory probability x Reward sum) for all trajectories in Table 12.5 and confirmed the suitability of the DTM with respect to A2.

We summed up the absolute values of Pearson correlation associated with every pair and presented them in the last column in Table 12.6. By comparing the sum of correlation with respect to each trajectory property, we observed that value of RS and EXRW were higher than that of TP and TP*RW. We conducted additional correlation tests, and t-statistic (in the first row) and p-value (in the second row) were presented in Table 12.7. All p-values were larger than 0.05 and thus we could not reject the null hypothesis that each pair of attributes came from the same distribution under the correlation coefficients given in Table 12.6.

For assessing the p-values in Table 12.7, we examined the normality of each attribute with Shapiro-Wilk test (J.P. Royston, 1982) and obtained the p-values provided in Table 12.8. When the p-value is greater than 0.05, we can assert that the normality of the distribution of the attribute under consideration. Therefore, we could confirm the normality only for the five attributes: event, msgOut, score, RS and EXRW. Other attributes turned out to be significantly different from the normal distribution. In conclusion, it is hard to say whether the correlation test is significant or not from the analysis. However, this observation is pertinent to the fundamental challenge of our research in human decision making, since the distributions of some attributes describing human behaviors are significantly different from the normal distribution and is difficult to analyze under the assumption of normality.

The trajectory information in Table 12.5 was obtained from one of our implementation settings (i.e., merged with no branching) with respect to state management and search for neighbor trajectories, as explained earlier for our IRL algorithm.

We conducted further experiments to see how consistent our observations are and obtained the correlations between each individual characteristic and trajectory property under merged (M), merged-branched (MB), unmerged, and unmerged-branched (B) as shown in

Table 12.3. Individual characteristics of scout dataset.

PID	Action	event	msgIn	msgOut	ROZ	Score	PID	Action	event	msgIn	msgOut	ROZ	Score
1	2.60	7.11	1.17	3.82	0.00	9375.00	12	2.85	7.94	1.15	3.54	0.00	12875.00
1	2.52	9.63	1.13	3.12	1.00	5900.00	12	2.77	9.94	1.15	2.83	1.00	9750.00
2	2.98	7.78	1.15	4.45	0.00	9800.00	14	2.22	7.11	1.15	2.45	0.00	11075.00
2	2.34	10.06	1.13	3.00	1.00	7850.00	14	2.45	6.72	1.13	2.27	0.00	11050.00
3	2.64	8.53	1.15	3.47	0.00	12450.00	15	2.24	11.53	1.15	3.21	0.00	8075.00
3	2.81	8.39	1.15	2.78	0.00	11700.00	15	2.52	10.61	1.15	2.42	1.00	6650.00
4	2.53	9.16	1.15	4.00	0.00	11625.00	16	2.56	8.83	1.17	3.00	0.00	10525.00
4	2.58	7.79	1.13	3.79	0.00	9725.00	16	2.43	7.63	1.13	2.71	0.00	9350.00
5	2.48	9.95	1.17	4.17	0.00	10150.00	17	2.39	8.33	1.15	3.36	1.00	11850.00
5	2.72	9.79	1.13	3.19	1.00	9875.00	17	2.67	8.84	1.13	3.13	0.00	9275.00
6	2.64	10.11	1.15	4.00	1.00	12375.00	19	2.58	8.42	1.17	3.86	1.00	11675.00
6	2.76	7.53	1.13	4.00	1.00	8750.00	19	2.83	10.68	1.13	2.94	1.00	9175.00
7	3.55	8.06	1.17	4.62	0.00	11450.00	20	2.34	8.11	1.15	3.73	0.00	11250.00
7	2.90	7.65	1.13	3.69	1.00	7475.00	20	2.64	9.76	1.13	3.08	1.00	6575.00
8	1.69	8.58	1.15	2.08	1.00	8400.00	21	2.62	7.95	1.15	3.62	0.00	12850.00
8	2.25	7.79	1.13	2.53	1.00	6050.00	21	2.27	9.63	1.13	2.93	0.00	8900.00
9	2.57	8.94	1.17	3.00	0.00	11950.00	22	2.53	8.95	1.15	3.71	0.00	13600.00
9	2.75	7.84	1.13	3.06	0.00	9100.00	22	2.68	8.21	1.15	3.33	0.00	9225.00
10	2.53	7.95	1.15	4.27	0.00	12850.00	23	2.78	7.61	1.15	3.57	0.00	13525.00
10	2.35	9.17	1.13	2.40	0.00	10900.00	23	2.70	8.42	1.13	2.68	0.00	10300.00

Au.: Pl ensure that the in-text reference match up with the corrected number in the Table.

Table 12.4. Features selected as important through recursive partitioning and tree-based methods.

Feature name	Content
MsgOut	Participant's written responses to messages in the Simulation's chat boxes
SE_EstDelay	Estimated delay across the network
RtPupilDiameter	Right pupil diameter size, not accounting for quality of the data
Asset2_NDdist	UAV's distance to the next destination
LtEyeLidOpeningQ	Quality of the left eyelid data from SmartEye
eventType	Marks when an event occurs in the simulation—both when a scripted event occurs and when the user performs an event
RtFILTPupilDiam	A quality weighted average filter for the right pupil size
ScoreChange	Shows when feedback was provided to the user about points that they have earned
Action	Shows more detailed information about what action the user took in response to an event

Table 12.9. For the case of branching, we computed the mean of 10 random runs and compared it against the results from non-branching method. The average values regarding EXRW (expected linear reward with discount) and RS (Reward sum) were 1.025 and 1.038 while those of TP (Trajectory probability) and TP*RW (trajectory probability x Reward Sum) were the same as 0.742. Therefore, we validated that our finding was consistent in different empirical settings with respect to state management and branching methods. Furthermore, we demonstrated the validity of the DTM to some degree since the sum of correlations with respect to EXRW and RW reflected the context to describe individuals found in the original data. However, we were still challenged by the low level of statistical confidence caused by the non-normality of attribute distributions observed in the Scout dataset.

12.4.3.3 Human Commanders

While assessment of the DTM model with the synthetic Commanders and SCOUT testbeds provides some insights on IRL algorithms and how features from the environment affect the decision-making process, the evaluation of human Commanders explores the relationship between decision making styles and rewards. The objectives of this evaluation are three-fold: First, we would like to assess whether we can distinguish different decision-making styles by comparing the graphical structures of different DTMs; second, we want to study how significant the reward values are by exploring the

Table 12.5. Information of Trajectories based on DTMs (TP=Trajectory probability, RS=Reward Sum, EXRW=Expected Linear Reward with Discount, TP*RW=Trajectory Probability * Reward Sum).

PID	TP	RS	EXRW	TP*RW	PID	TP	RS	EXRW	TP*RW
1	7.75E-31	-8200	-855794.5048	-6.35E-27	12	6.12E-59	-8300	-680113.0988	-5.08E-55
1	1.15E-27	-12800	-1758674.159	-1.48E-23	12	4.32E-76	-7400	-632474.3482	-3.19E-72
2	8.45E-30	-4700	-520933.5793	-3.97E-26	14	2.62E-21	-10000	-948829.9884	-2.62E-17
2	2.99E-31	-7600	-961511.1009	-2.27E-27	14	2.10E-36	-4000	-325949.996	-8.41E-33
3	1.08E-60	-7000	-900609.1025	-7.54E-57	15	7.39E-42	-5700	-458227.3474	-4.21E-38
3	1.13E-27	-4700	-513976.3881	-5.29E-24	15	4.38E-25	-11200	-1089398.327	-4.90E-21
4	1.02E-41	-7700	-889852.5266	-7.84E-38	16	1.23E-37	-10100	-1104708.871	-1.24E-33
4	3.16E-50	-4600	-419816.3757	-1.45E-46	16	3.01E-15	-12000	-1492579.663	-3.61E-11
5	4.84E-52	-7100	-884810.6541	-3.43E-48	17	1.13E-51	-10100	-1000422.76	-1.14E-47
5	1.12E-41	-10900	-1523688.639	-1.22E-37	17	1.09E-43	-2100	-198086.6633	-2.28E-40
6	2.07E-55	-9000	-1211881.894	-1.87E-51	19	3.21E-47	-11300	-1388526.463	-3.62E-43
6	7.70E-44	-6300	-801191.2841	-4.85E-40	19	1.89E-44	-8200	-1118135.543	-1.55E-40
7	1.66E-64	-5600	-508943.4366	-9.30E-61	20	6.83E-68	-5300	-448002.0003	-3.62E-64
7	1.51E-34	-6700	-763243.8041	-1.01E-30	20	3.40E-43	-5100	-565096.1046	-1.73E-39
8	4.82E-32	-7300	-861230.0344	-3.52E-28	21	1.14E-37	-5600	-745090.2646	-6.37E-34
8	5.76E-28	-8400	-830391.2891	-4.84E-24	21	4.37E-24	-9500	-1477729.987	-4.15E-20
9	5.49E-31	-8800	-1061067.041	-4.83E-27	22	6.52E-34	-9200	-1062696.482	-6.00E-30
9	6.67E-31	-9600	-1031492.496	-6.41E-27	22	3.05E-41	-6200	-822866.627	-1.89E-37
10	6.37E-69	-4000	-371733.4876	-2.55E-65	23	3.33E-37	-10800	-1325783.067	-3.60E-33
10	9.67E-41	-7200	-849345.5252	-6.96E-37	23	8.40E-32	-9900	-1290626.097	-8.31E-28

Table 12.6. Correlation between Individual characteristics and trajectory properties DTMs (TP=Trajectory probability, RS=Reward Sum, EXRW=Expected Linear Reward with Discount, TP*RW=Trajectory Probability * Reward Sum).

	Action	event	msgIn	msgOut	ROZ	Score	Sum
TP	-0.085	-0.154	-0.170	-0.153	-0.118	-0.062	0.742
RS	0.158	-0.155	-0.056	0.248	-0.292	0.072	0.982
EXRW	0.134	-0.221	0.066	0.188	-0.293	0.089	0.992
TP*RW	0.085	0.154	0.170	0.153	0.118	0.062	0.742

Table 12.7. Additional information of correlation Test (t-statistic, p-value, degree of freedom=38).

	action	event	msgIn	msgOut	ROZ	score
TP	-0.529	-0.959	-1.061	-0.955	-0.729	-0.384
	0.600	0.344	0.295	0.345	0.470	0.703
RS	0.986	-0.969	-0.349	1.579	-1.883	0.446
	0.331	0.339	0.729	0.123	0.067	0.658
EXRW	0.837	-1.397	0.406	1.181	-1.892	0.551
	0.408	0.171	0.687	0.245	0.066	0.585
TP*RS	0.529	0.959	1.061	0.955	0.729	0.384
	0.600	0.344	0.295	0.345	0.470	0.703

Table 12.8. Normality check with Shapiro test.

	action	event	msgIn	msgOut	ROZ
p	6.25E-03	1.77E-01	8.69E-06	8.62E-01	3.52E-09
	score	TP	RS	EXRW	TP*RS
p	4.07E-01	6.64E-14	9.37E-01	7.68E-01	6.64E-14

Table 12.9. Correlation analysis in different setting (M=merged, B=branched).

	action	event	msgIn	msgOut	ROZ	score	Sum
TP(M)	-0.085	-0.154	-0.170	-0.153	-0.118	-0.062	0.742
RS(M)	0.158	-0.155	-0.056	0.248	-0.292	0.072	0.982
EXRW(M)	0.134	-0.221	0.066	0.188	-0.293	0.089	0.992
TP*RS(M)	0.085	0.154	0.170	0.153	0.118	0.062	0.742
TP(MB)	0.085	0.154	0.170	0.153	0.118	0.062	0.742
RS(MB)	0.063	0.275	0.122	0.181	0.201	0.236	1.078
EXRW(MB)	0.053	0.354	0.149	0.131	0.234	0.193	1.115
TP*RS(MB)	0.085	0.154	0.170	0.153	0.118	0.062	0.742
TP	-0.085	-0.154	-0.170	-0.153	-0.118	-0.062	0.742
RS	-0.080	-0.080	0.260	0.101	-0.331	0.248	1.101
EXRW	-0.122	-0.217	0.243	0.032	-0.305	0.158	1.077
TP*RS	0.085	0.154	0.170	0.153	0.118	0.062	0.742
TP(B)	0.085	0.154	0.170	0.153	0.118	0.062	0.742
RS(B)	0.097	0.263	0.100	0.195	0.125	0.160	0.940
EXRW(B)	0.074	0.312	0.135	0.158	0.168	0.122	0.968
TP*RS(B)	0.085	0.154	0.170	0.153	0.118	0.062	0.742

reward distribution; lastly, we want to measure the effect of decision styles on rewards by combining the trajectories from Commanders with different decision-making styles.

We use the trajectories compiled from the 32 games played by the three student Commanders to create 32 DTMs (unmerged) to evaluate over the derived *d*MDP and use the number of states, the number of actions and density of the graph to measure the graph structure of each *d*MDP. The density of the graph is computed as follows: $d = \frac{|E|}{|V| * (|V| - 1)}$ with E being the number of actions and V being the number of states. The Table 12.10 below shows the statistics of these values broken down by Styles (we encoded Rational style to be 1, Spontaneous-Avoidant to be 2 and Rational-Dependent to be 3). As shown in Table 12.10, there are differences of means between states, actions and density among three Commanders. However, one-way ANOVA analysis confirmed that these differences are not statistically significant (as shown in Table 12.11).

To understand whether we could detect the differences of decision-making styles based on their rewards, we decided to use one training set which consists of all the trajectories from all three Commanders. For each Commander, we construct a test set by using only his own trajectories. We create a DTM for each test set and run IRL on each corresponding *d*MDP to determine his reward function and expected linear reward discount. The intuition behind this set-up is that different Commanders with different decision-making styles may have different ways of rewarding decisions. In order to be able to compare them, we should have the same reward space for all decision-makers.

As shown in the Table ~~12.13~~, there are differences between average reward sum and expected linear reward discount among three Commanders. Specifically, the first Commander with Rational decision-making style tends to achieve the highest expected linear reward discount compared to the other two Commanders with Spontaneous-Avoidant and Rational-Dependent styles. However, we performed one-way ANOVA analysis and found these differences are not statistically significant, as shown in Table 12.13.

Lastly, we are going to see how the DTMs constructed by using each Commander's own trajectories compared to the joint DTM created by combining his own trajectories with other Commander's trajectories would affect the rewards that he puts on each action from a specific state. For each Commander i, we generate a set of DTM which includes a DTM using his own trajectories, a DTM using his trajectories and the other Commander j($j \neq i$) unmerged and merged (as defined earlier in IRL section), respectively. For each DTM, we determined a set of attributes that is most significantly related to the reward distribution and used the k-Means clustering algorithm

Table 12.10. Statistics of graph structures of DTMs for three human Commanders.

Descriptives

		N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
						Lower Bound	Upper Bound		
States	1	16	20.56	8.422	2.105	16.07	25.05	11	38
	2	8	19.00	6.141	2.171	13.87	24.13	12	31
	3	8	15.25	4.743	1.677	11.28	19.22	9	23
	Total	32	18.84	7.265	1.284	16.22	21.46	9	38
Actions	1	16	15.75	7.197	1.799	11.91	19.59	7	29
	2	8	13.00	4.567	1.615	9.18	16.82	6	19
	3	8	11.00	4.342	1.535	7.37	14.63	4	18
	Total	32	13.88	6.179	1.092	11.65	16.10	4	29
Density	1	16	.045035	.0177797	.0044449	.035561	.054509	.0206	.0705
	2	8	.042585	.0219408	.0077573	.024242	.060928	.0204	.0934
	3	8	.054629	.0208191	.0073607	.037223	.072034	.0292	.0972
	Total	32	.046821	.0195332	.0034530	.039778	.053863	.0204	.0972

Table 12.11. One-way ANOVA on graph structures of DTMs for three Commanders.

ANOVA

		Sum of Squares	df	Mean Square	F	Sig.
States	Between Groups	150.781	2	75.391	1.472	.246
	Within Groups	1485.438	29	51.222		
	Total	1636.219	31			
Actions	Between Groups	128.500	2	64.250	1.766	.189
	Within Groups	1055.000	29	36.379		
	Total	1183.500	31			
Density	Between Groups	.001	2	.000	.888	.423
	Within Groups	.011	29	.000		
	Total	.012	31			

(MacQueen, 1967) to cluster its states using the reduced set of attributes. The main idea of clustering task is to (i) find out the values of attributes describing the centroid of each cluster corresponding with reward being zero and reward being negative; and (ii) evaluate how the states assigned to a single cluster.

The results are summarized in Table 12.14 below. As we can see, the DTM created through state merging between Commander 1 and Commander 2's trajectories provided the lowest errors within clusters and percentage of incorrectly clustered instances for Commander 1. The reward being zero is given to states of low damage for Commander 1 while reward being negative is given to no damage given. For Commander 2, the DTM created by using unmerged Commander 1's and Commander 2's trajectories generated the lowest errors and lowest percentage of incorrectly clustered instances. The reward being negative for movement to a specific location. For Commander 3, the DTM created by using unmerged Commander 2's and Commander 3's trajectories provided the lowest errors and lowest percentage of incorrectly clustered instances. Negative rewards are given to the actions of moving east of the Commander 3. Recall that Commander 2 has a spontaneous decision style while Commander 1 has a rational decision style and Commander 3 is a rational and dependent decision maker. The states in the DTMs created by using the combination and merged process of trajectories of a rational and spontaneous decision maker are better segmented than the states of the DTMs created by unmerged between Rational and Rational dependent decision styles or Rational alone. The rational decision makers often make decision based on thorough search and evaluation of alternatives (Scott and Bruce, 1995) while the spontaneous decision makers make react to the situations quickly. This gives some insights on how the combination of two highly contrasted decision-making styles would help a Commander to form his own style by distinguishing between actions with negative rewards from actions with zero rewards.

Table 12.12. Descriptive table of rewards broken down by decision making styles.

Descriptives

		N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
						Lower Bound	Upper Bound		
RewardSum	1	16	506.250000	766.3495721	191.5873930	97.891138	914.608862	.0000	2400.0000
	2	8	512.500000	372.0119046	131.5260702	201.490265	823.509735	.0000	1000.0000
	3	8	462.500000	726.9063606	257.0002084	-145.208926	1070.208926	.0000	1900.0000
	Total	32	496.875000	659.6599197	116.6125006	259.042237	734.707763	.0000	2400.0000
ExpectedLinearRewardWDiscount	1	16	6533.910894	9349.1443907	2337.2860977	1552.103503	11515.718285	.0000	28544.1248
	2	8	5035.241277	4076.4086944	1441.2281153	1627.278324	8443.204231	.0000	11076.5884
	3	8	3337.142067	5554.3166676	1963.7474903	-1306.382872	7980.667007	.0000	14183.0809
	Total	32	5360.051283	7403.1352361	1308.7017819	2690.936402	8029.166164	.0000	28544.1248

Table 12.13. One-way ANOVA analysis of rewards and decision-making styles.

ANOVA

		Sum of Squares	df	Mean Square	F	Sig.
RewardSum	Between Groups	12812.500	2	6406.250	.014	.986
	Within Groups	13476875.000	29	464719.828		
	Total	13489687.500	31			
ExpectedLinearRewardWDiscount	Between Groups	55628448.057	2	27814224.029	.491	.617
	Within Groups	1643370302.989	29	56667941.482		
	Total	1698998751.047	31			

Table 12.14. Effect of rewards on clustering of features using combinations of Commanders' trajectories.

Commanders	DTM trajectories	Description of centroid for cluster 0 (reward = 0)	Description of centroid for cluster 1 (reward = –100)	Sum of SQR error	Incorrectly clustered instances (%)
Commander 1	Commander 1	Reporting = Yes	Reporting = No	32	45.31%
	Commander 1 & 2 Unmerged	Damage_Given_high=Yes	Damage_Given=No	13	16%
	Commander 1& 3 unmerged	Movement_stationary=No	Movement_stationary=Yes	60	30.6548%
	Merged 1 & 2	Damage_Given_low=yes	Damage_Given_low=No	12	9.0395%
	Merged 1 & 3	Movement_stationary=No	Movement_stationary=Yes	24	28.2738%
Commander 2	Commander 1 & 2 Unmerged	Command__move_to_E4:_ship1_ship2 = No	Command__move_to_E4:_ship1_ship2 = Yes	4.0	1.9774%
	Commander 2	Movement__north=Yes	Movement__north=no	91.0	20.6897%
	Commander 3 & 2 Unmerged	Report__ship1:_Ship1_sank_ShipEnemy=No	Report__ship1:_Ship1_sank_ShipEnemy=Yes	35	4.1026%
	Merged 1 & 2	Command__move_to_E4:_shipYes,_ship2=No	Command__move_to_E4:_shipYes,_ship2=Yes	23	1.9774%
	Merged 2 & 3	Report__ship2_ShipEnemy_sank_Ship2=No	Report__ship2_ShipEnemy_sank_Ship2=yes	103.00	12.8205%
Commander 3	Commander 1 & 3 Unmerged	Movement__south =No	Movement__south =yes	25.00	24.4048%
	Commander 2 & 3 Unmerged	Movement__east =No	Movement__east =Yes	17.00	4.1026%
	Commander 3	Movement__east =Yes	Movement__east = No	91.0	48.9362%
	Merged 1 & 3	Command__engage_ShipEnemy_at_F1:_ship1_ship2 =No	Command__engage_ShipEnemy_at_F1:_ship1_ship2 =Yes	76.0	5.6548%
	Merged 2 & 3	Report_Is_Set =No Report__ship2:_ShipEnemy_sank_Ship2=No	Report_Is_Set =Yes Report__ship2:_ShipEnemy_sank_Ship2=Yes	171.0	12.8205%

For the spontaneous decision maker, using the trajectories from a rational decision maker and a spontaneous decision maker but not merging the states would help him learn separately how to evaluate the choices and segment the states more clearly. Similarly, a rational but dependent decision maker could learn from a spontaneous decision maker to help him to form his own unique style.

12.5 Summary of Evaluation

Our evaluation with three testbeds have assessed the DTM, the IRL algorithm used in this model, and the applications of this model in three different settings. As we could see from the Table 12.15, the complexity of our testbed is increasing from the Synthetic to Human Commanders testbed. The main results that we have achieved so far are:

- With the Synthetic Commander dataset, we identified limitations with existing IRL algorithms and demonstrated that learning rewards on (s, a, s') triples decreases the amount of information lost versus just having the rewards on the states.
- With the Scout dataset, we confirmed that the DTM constructed is valid, where EXRW and RW of trajectories could reflect the context to describe individuals, but the level of confidence is limited due to the non-normality of attribute distributions.
- With Human Commanders, there is a difference of graphical structures of the DTMs and expected linear reward discount for various decision-making styles. Unfortunately, due to the small sample size, the change is not statistically significant. Secondly, combining the trajectories from Commanders with different decision-making styles leads to a better segmentation of the actions of Commanders based on rewards.

Table 12.15. Attributes of three testbeds.

	Partially observable	**Dynamic**	**Complete control of environment**	**Real time**	**Ground truth availability**
Synthetic	Semi	No	Yes	No	Complete
SCOUT	No	No	Yes	Yes	No
Human Commander	Yes	Yes	Yes	Yes	Partially

12.6 Discussion and Future Work

This chapter has introduced a framework for modeling Commanders decision making styles, described Inverse Reinforcement Learning as a means for describing a Commander's decision-making process, and evaluated it over three testbeds: synthetic Commanders, Supervisory

Control Operations User Testbed (SCOUT), and human Commanders. Here, we discuss the difficulties we encountered in this work followed by future directions.

The Synthetic Commander testbed showed that it is useful to have rewards on the (s, a, s') triples. The exhaustiveness of the generation of neighborhoods for the Synthetic Commanders meant that many of the possible c-neighbors were already existing within the trajectories themselves. This is not expected to happen frequently with real-world cases but if it does occur in more real-world cases then an alternative method for generating neighbors might be required.

While investigating the records of the 20 participants in the SCOUT testbeds, we realized that the cognitive processes of individuals are difficult to classify into any "typical" cognitive style, since they are composed of multiple layers and facets that would be observed differently according to the situations given. We focused on specific tendencies in human behavior such as the risk-aversion, selection efficiency and game performance, which are known to be critical in affecting behaviors. From our empirical study, we observed some limited associations between the individual characteristics and trajectory properties observed in DTMs. There are likely to be other individual characteristics that would impact a Commander's decision and we will continue to work on further understanding them.

Lastly, there are differences in values from the following three categories: graph structure, expected linear rewards with discount and trajectories for our three human Commanders. However, these differences are not statistically significant (with p-value < 0.05). Differences are caused in part due to the small sample size of battles ($n = 32$) and the average length of the battles (around 10 minutes). Interestingly, we found that Commander 1 has achieved highest expected linear reward with discount and also achieved the highest number of wins among the three Commanders (6 wins versus 2 wins for Commanders 2 and 3). The wins and losses are automatically defined by the Steel Ocean game. His games are also longer in length (642.5 seconds versus 542.9 seconds for Commander 2 and 518.9 seconds for Commander 3). Interestingly, in our data, games that took longer more often resulted in wins. For instance, games that lasted longer than 10 minutes had a 66.67% chance of winning and games that were shorter than 10 minutes had a 6.25% chance of winning. This is also backed up by Commander 1's personality test coming up strongly rational. One could infer that a rational commander would be more willing to focus on future events rather than focusing on the instant gratification of the reward at hand.

There are a number of directions to pursue for our future work. First, we would like to study how the DTMs change over time, especially when we have a junior Commander working to get experience and get better at Commanding. How a junior Commander's rewards and sequences of actions

change gives us a potential to understand how people learn and when they use different decision styles. This means that, in a battle, a Commander may have a sequence of actions where he is acting spontaneously but also has a sequence of actions where he is acting more rationally. A DTM could capture these changes and that could be used to distinguish dynamic decision styles over time. Second, we also want to strengthen our evaluations with hypothetical and human Commanders by adding more data points and finding how using a more experience Commander's DTM could help train less experience Commander. Lastly, we would like to verify whether the reward distribution plays a significant role in reinforcing good decisions, enabling us to learn from bad decisions.

Our new IRL approach with its basis as a linear programming formulation enables us to extend and scale with updated computations as new trajectories with yet unseen states and/or actions are introduced. A large, well-understood literature of tools and techniques from linear and convex optimization are readily applicable to our new IRL algorithm including standard sensitivity analysis, variable impact analysis, and gap-analysis to explain changes in reward values reflecting our dMDPs.

Acknowledgements

This work supported in part by ONR Grant No. N00014-1-2154, ONR/NPS Grant No. N00024-15-1-0046, AFOSR Grant No. FA9550-15-1-0383, DURIP Grant No. N00014-15-1-2514, and University of Wisconsin - Whitewater undergraduate research grants summer 2017.

References

Bellman, Richard. 1957. A Markovian decision process. Journal of Mathematics and Mechanics: 679–684. http://www.iumj.indiana.edu/IUMJ/FULLTEXT/1957/6/56038.

Coyne, Joseph T. and Ciara, M. Sibley. 2015. Impact of task load and gaze on situation awareness in unmanned aerial vehicle control. Naval Research Laboratory Washington United States. https://www.nrl.navy.mil/itd/imda/sites/www.nrl.navy.mil.itd.imda/files/pdfs/079%20Coyne,%20Sibley.pdf.

Esa, Muneera, Anuar Alias and Zulkiflee Abdul Samad. 2014. Project managers' cognitive style in decision making: a perspective from construction industry. International Journal of Psychological Studies 6(2): 65.

Flin, Rhona. 2001. Decision making in crises: the Piper Alpha disaster. pp. 103–116. In MANAGING CRISES: Threats, Dilemmas, Opportunities.

Galotti, Kathleen M., Elizabeth Ciner, Hope E. Altenbaumer, Heather J. Geerts, Allison Rupp and Julie Woulfe. 2006. Decision-making styles in a real-life decision: Choosing a college major. *In*: Personality and Individual Differences 41(4): 629–639.

Heuer Jr., Richards J. 1999. Psychology of Intelligence Analysis. Central Intelligence Agency Washington DC Center for Study of Intelligence.

Howard, Ronald A. 1960. Dynamic Programming and Markov Processes.

McLaughlin, Jacqueline E., Wendy C. Cox, Charlene R. Williams and Greene Shepherd. 2014. Rational and experiential decision-making preferences of third-year student pharmacists. American Journal of Pharmaceutical Education 78(6): 120.

Michailidis, Evie and Adrian P. Banks. 2016. The relationship between burnout and risk-taking in workplace decision-making and decision-making style. Work & Stress 30(3): 278–292.

MacQueen, James B. 1967. Some Methods for classification and analysis of multivariate observations. Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability. Berkeley, University of California Press 1: 281–297.

Ng, Andrew Y. and Stuart J. Russell. 2000. Algorithms for inverse reinforcement learning. In Icml, pp. 663–670.

Royston, J.P. 1982. An extension of Shapiro and Wilk's W test for normality to large samples. Applied Statistics 31(2): 115–124.

Russell, Stuart. 1998. Learning agents for uncertain environments. pp. 101–103. *In*: Proceedings of the Eleventh annual conference on Computational learning theory, ACM.

Russell, Stuart J., Peter Norvig, John F. Canny, Jitendra M. Malik and Douglas D. Edwards. 2003. Artificial Intelligence: A Modern Approach. Vol. 2, no. 9. Upper Saddle River: Prentice Hall.

Santos Jr., Eugene and Eugene S. Santos. 1999. A framework for building knowledge-bases under uncertainty. Journal of Experimental & Theoretical Artificial Intelligence 11(2): 265–286.

Santos Jr., Eugene, Hien Nguyen, Jacob Russell, Keumjoo Kim, Luke Veenhuis, Ramnjit Boparai and Thomas Kristoffer Stautland. 2017. Capturing a Commander's decision making style. *In* Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security, Defense, and Law Enforcement Applications XVI, vol. 10184, p. 1018412. International Society for Optics and Photonics.

Scott, Susanne G and Bruce, Reginald A. 1995. Decision-making style: The development and assessment of a new measure. Educational and Psychological Measurement 55(5): 818–883.

Syagga, Laura. 2012. Intuitive Cognitive Style and Biases in Decision Making. PhD diss., Eastern Mediterranean University (EMU).

Therneau, Terry, Beth Atkinson, and Brian Ripley. "R Rpart Manual." Accessed April 12, 2017. https://cran.r-project.org/web/packages/rpart/rpart.pdf.

Thomas, P.J. 2016. Measuring risk-aversion: The challenge. Measurement 79: 285–301.

Thunholm, Peter. 2009. Military leaders and followers—do they have different decision styles? Scandinavian Journal of Psychology 50(4): 317–324.

United States Joint Forces Command. 2004. Commander's Handbook for Joint Battle Damage Assessment. Retrieved from http://www.dtic.mil/doctrine/doctrine/jwfc/hbk_jbda.pdf.

Vogel, Edward K., Andrew W. McCollough and Maro G. Machizawa. 2005. Neural measures reveal individual differences in controlling access to working memory. Nature 438(7067): 500–503.

Wulfmeier, Markus, Peter Ondruska and Ingmar Posner. 2015. Deep inverse reinforcement learning. CoRR, abs/1507.04888.

Yu, Fei. 2013. A Framework of Computational Opinions. Dartmouth College.

Yu, Fei and Eugene Santos Jr. 2016. On modeling the interplay between opinion change and formation. pp. 140–145. In FLAIRS Conference. Proceedings of the 29th International FLAIRS Conference, Key Largo, FL 140–145.

Ziebart, Brian D., Andrew L. Maas, J. Andrew Bagnell and Anind K. Dey. 2008. Maximum entropy inverse reinforcement learning. In AAAI 8: 1433–1438.