Journal of Machine Learning Research 13 (2012) 3253-3295 Submitted 10/11, Revised 9/12; Published 11/12

Linear Fitted-Q Iteration with Multiple Reward Functions

Daniel J. Lizotte DLIZOTTE@UWATERLOO.CA
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, ON N2L 3G1, Canada

Michael Bowling BOWLING@CS.UALBERTA.CA
Department of Computing Science
University of Alberta
Edmonton, AB T6G 2E8, Canada

Susan A. Murphy SAMURPHY@UMICH.EDU
Department of Statistics
University of Michigan
Ann Arbor, MI 48109-1107, USA

Editor: Sridhar Mahadevan

Abstract

We present a general and detailed development of an algorithm for finite-horizon fitted-Q iteration with an arbitrary number of reward signals and linear value function approximation using an arbitrary number of state features. This includes a detailed treatment of the 3-reward function case using triangulation primitives from computational geometry and a method for identifying globally dominated actions. We also present an example of how our methods can be used to construct a real-world decision aid by considering symptom reduction, weight gain, and quality of life in sequential treatments for schizophrenia. Finally, we discuss future directions in which to take this work that will further enable our methods to make a positive impact on the field of evidence-based clinical decision support.

Keywords: reinforcement learning, dynamic programming, decision making, linear regression, preference elicitation

1. Introduction

Within the field of personalized medicine, there is increasing interest in investigating the role of sequential decision making for managing chronic disease (Weisz et al., 2004; McKay, 2009; Kuk et al., 2010). Reinforcement learning methods (Szepesvári, 2010; Sutton and Barto, 1998) are already being used (Pineau et al., 2007; Murphy et al., 2007; Zhao et al., 2009) to analyze Sequential Multiple Assignment Randomized Trials (SMART) (Murphy, 2005). A patient's progression through a SMART is divided into stages, each of which consists of a (typically uniform) random assignment to a treatment, followed by monitoring and recording data on the patient's condition. The patient data collected during each stage are very rich and commonly include several continuous variables related to symptoms, side-effects, treatment adherence, quality of life, and so on. For the ith patient in the trial, we obtain a *trajectory* of *observations* and *actions* of the form

$$o_1^i, a_1^i, o_2^i, a_2^i, \ldots, o_T^i, a_T^i, o_{T+1}^i.$$

Here, a_t^i represents the action (treatment) at time t, and o_t^i represents measurements made of patient i *after* action a_{t-1}^i and *before* action a_t^i. The first observations o_1^i are baseline measurements made before any actions are taken.

To analyze these data using reinforcement learning methods, we must define two functions $s_t(o_1, a_1, ..., o_t)$ and $r_t(s_t, a_t, o_{t+1})$ which map the patient's current history to a state representation and a scalar reward signal, respectively. Applying these functions to the data from the ith patient gives a trajectory

$$s_1^i, a_1^i, r_1^i, s_2^i, a_2^i, r_2^i, ..., s_T^i, a_T^i, r_T^i.$$

These redefined data are treated as sample trajectories from a known policy which is typically uniformly random over possible actions. Once we have these, we will view ongoing patient care as a Markov decision process (MDP) (Bertsekas and Tsitsiklis, 1996), and apply batch off-policy reinforcement learning methods to learn an optimal policy that takes a patient state and indicates which action appears to be best in view of the data available. In an MDP, both the state transition dynamics and the reward distributions are assumed to have the *Markov property*. That is, given the value s_t of the current state, the distribution of next state S_{t+1} and current reward R_t is conditionally independent of s_j, a_j, r_j for all $j < t$. Clearly this can be achieved by including past history in s_t, but this may not be feasible. For this work, we will assume that practitioners can suggest state features (which may be summaries of history) that are "as good as a complete history" in terms of making predictions about future states and rewards: we want features that are rich enough to provide good predictions about action values, but that are simple enough to allow us to learn from a limited amount of data. In medical domains, we may additionally want the learned policy to be easily interpreted and implemented. The interplay between predictiveness, learnability, and interpretability makes the definition of s_t a challenging problem that requires a great deal of further investigation, particularly into the consequences of a non-Markov definition of state. However, the question of how s_t should be defined can be answered at least in part by the data themselves together with expert knowledge and feature/model selection techniques analogous to those used in supervised learning settings (Keller et al., 2006) *if we have an adequate definition of* r_t.

A major difficulty with using trial data in this way is that there is often no obviously correct way to define r_t. Indeed, any definition of r_t is an attempt to answer the question "*What is the right quantity to optimize?*"—a question that is driven by the objectives of individual decision makers and *cannot be answered by the data alone.* There are many reasonable reward functions one could define, since each patient record includes a multi-dimensional measurement of that patient's overall well-being. For example, data often include a measure of the severity of the symptoms the patient is experiencing, as well as a measure of the severity of the side-effects caused by the current treatment. These different dimensions are typically better addressed by some treatments than by others, and therefore the choice of which dimension to use as the reward will affect the resulting learned policy. For example, a policy that minimizes expected symptom level will tend to choose more aggressive drugs that are very effective but that have a more severe side-effect profile. On the other hand, a policy that minimizes expected side-effect measurements will choose drugs that are less effective but that have milder side-effects.

In clinical practice, doctors, patients, and families decide on a treatment by weighing different measures of well-being, like symptoms and side-effects, according to subjective preferences that are not known to us at the time of data analysis. Continuing our example, these preferences may lean more toward symptom reduction or side-effect reduction, depending on the individual decision makers involved, and an appropriate reward function definition should reflect these preferences. In

principle, one could elicit these preferences, use them to define the reward function of each individual decision maker, and then learn a policy for that reward function; however accurate preference elicitation can be difficult to achieve, and even when it is possible it can be a time-consuming process for the decision maker. Moreover, this approach is problematic because it does not give a complete picture of the quality of the available actions under different reward choices. Indeed, the decision maker will not know when very small changes in preferences might lead to different actions, or when one action is optimal for a broad range of preferences, or when another action is not optimal for any preference.

Rather than eliciting preference and producing a policy that recommends a single action per state, our "inverse preference elicitation" approach is to first consider *all* of the actions available at the current state. For each of the actions, we answer the question, *"What range of preferences makes this action a good choice?"* This provides much richer information about the possible actions at each stage. Furthermore, even if a preference is specified somehow, our methods allow the maker to immediately see if his or her preferences are near a "boundary"—that is, whether a small change in preference can lead to a different recommended action. In this case, according to the data analysis two or more actions perform comparably well, and therefore the final decision could be based on other less crucial considerations such as dosing schedule and difference in cost. We are interested in efficient algorithms that can exactly compute the optimal policy for a range of reward functions to investigate how our choice of reward function influences the optimal policy, and in turn how we can offer more flexible choices among good actions.

2. Related Applications, Existing Methods, and Our Contributions

Our approach can help explore trade-offs in different application domains besides sequential medical decision making as well. In e-commerce, one may wish to trade off short-term profits with increased brand-visibility. In robotics, one may wish to trade-off rapid task completion against wear-and-tear on equipment. Researchers are already considering trading off water reserves versus flood risk in water reservoir control (Castelletti et al., 2010); our approach could provide further insight here. Even within RL itself, our approach could provide a new perspective on trading off achieving high expected reward with avoiding risk, an issue explored by Mannor and Tsitsiklis (2011). Any problem for which it is difficult or undesirable to formulate a single scalar reward to drive decision making could benefit from our approach.

There is wide interest in making use of multiple reward signals for sequential decision making. Gábor et al. (1998) demonstrated that an MDP with multiple reward signals can be well-defined and solved so long as we are given a fixed partial ordering on reward vectors. Mannor and Shimkin (2004) offer a formalism where actions are chosen to ensure that the long-term average reward vector approaches a "target set". The target set induces an ordering (closeness) on reward vectors which drives the agent's actions. Natarajan and Tadepalli (2005) assume that a scalar reward function is constructed by taking a weighted sum of a reward vector, just as we will. They assume that the weights are given, and that the weights will change over time. Their strategy is to learn a dictionary of policies for different weight vectors that should eventually contain policies that work well for many different preferences. They note that "An interesting direction for future research is to investigate the number of different weight vectors needed to learn all the optimal policies within a desired degree of accuracy," which we will address as part of this work. Early work in this direction (Barrett and Narayanan, 2008) explored the problem of simultaneously computing optimal policies for a

class of reward functions over a small, finite state space in a framework where the model is known. Subsequent developments were made that focussed on the infinite-horizon discounted setting and black-box function approximation techniques (Castelletti et al., 2010; Vamplew et al., 2011). We extended the approach of Barrett and Narayanan (2008) to the setting with real-valued state features and *linear* function approximation, which is a more appropriate framework for analyzing trial data (Lizotte et al., 2010). We also introduced an algorithm that is asymptotically more time- and space-efficient than the Barrett & Narayanan approach, and described how it can be directly applied to batch data. We also gave an algorithm for finding the set of all non-dominated actions in the single-variable continuous state setting. This paper builds on our previous work by contributing:

- A general and detailed development of finite-horizon fitted-Q iteration with an arbitrary number of reward signals and linear approximation using an arbitrary number of state features
- A detailed treatment of 3-reward function case using triangulation algorithms from computational geometry that has the same asymptotic time complexity as the 2-reward function case
- A more concise solution for identifying globally dominated actions under linear function approximation, and method for solving this problem in higher dimensions
- A real-world decision aid example that considers symptom reduction, weight gain, and quality of life when choosing treatments for schizophrenia

3. Background

We begin by defining the mathematical framework for our problem and describing its relationship to the usual MDP formulation. We then discuss how two existing formalisms, *Inverse Reinforcement learning* and *POMDP Planning*, relate to our approach.

3.1 Problem Framework

For each patient, we assume we will choose a treatment action at each timepoint $t = 1, 2, ..., T$, after which they are no longer under our care. In this finite-horizon sequential decision making setting, the optimal policy in general depends on t (Bertsekas and Tsitsiklis, 1996), so we explicitly maintain separate r_t, Q_t, and V_t functions for each timepoint, and we define $Q_T \equiv r_T$. Furthermore, it is convenient for our purposes to allow the set of possible states $\mathcal{S}_t$ and the set of possible actions $\mathcal{A}_t$ to depend on time. We then designate the learned policy at a particular time point by $\pi_t : \mathcal{S}_t \to \mathcal{A}_t$.

We consider sets of MDPs that all have the same $\mathcal{S}_t$, $\mathcal{A}_t$, and state transition dynamics, but whose expected reward functions $r_t(s_t, a_t, \delta)$ have an additional parameter δ. One may think of δ as a special part of state that: i) does not evolve with time, and ii) does not influence transition dynamics. Each fixed δ identifies a single MDP by fixing a reward function, which has a corresponding optimal[1] state-action value function defined in the usual way via the Bellman equation:

$$Q_t(s_t, a_t, \delta) = r_t(s_t, a_t, \delta) + E_{S_{t+1}|s_t,a_t}[\max_{a \in \mathcal{A}_{t+1}} Q_{t+1}(S_{t+1}, a, \delta)].$$

1. In this work, most Q- and V-functions are either optimal or estimates of optimal. We omit the usual * superscript in most cases, and mark estimates with a hat $\hat{}$.

We will also refer to the optimal state value function $V_t(s_t,\delta) = \max_{a\in\mathcal{A}_t} Q_t(s_t,a,\delta)$, and the[2] optimal deterministic policy $\pi_t(s_t,\delta) \in \text{argmax}_{a\in\mathcal{A}_t} Q_t(s_t,a,\delta)$. The purpose of δ is to represent the *preferences* of a decision maker: we presume that a decision maker would like to follow an optimal policy $\pi_t(s_t,\delta)$ for the MDP indexed by the value of δ that represents their preferences, that is, the value of δ for which $r_t(s_t,a_t,\delta)$ is a reflection of their reward function.

In order to mathematize the relationship between preference and δ, we define the structure of $r_t(s_t,a_t,\delta)$ to be

$$\delta = (\delta_{[0]},\delta_{[1]},...,\delta_{[D-1]}), \tag{1}$$

$$r_t(s_t,a_t,\delta) = \delta_{[0]} r_{t[0]}(s_t,a_t) + \delta_{[1]} r_{t[1]}(s_t,a_t) + ... + (1 - \sum_{d=0}^{D-2} \delta_{[d]}) r_{t[D-1]}(s_t,a_t). \tag{2}$$

where $\sum_{d=0}^{D-1} \delta_{[d]} = 1$. Thus $r_t(s_t,a_t,\delta)$ is a convex combination of the *basis rewards* $r_{t[0]},...,r_{t[D-1]}$, identified by a vector δ of length D that identifies points on the $(D-1)$-simplex. The vector δ represents a preference that assigns weight to each of the basis rewards. For example, if $\delta_{[d]} = 1$ for some index d, then $r_t(s_t,a_t,\delta) = r_{t[d]}(s_t,a_t)$, and the optimal policy $\pi_t(s_t,\delta)$ will choose actions that optimize the expected sum of rewards as determined by $r_{t[d]}$. Intuitively, the magnitude of $\delta_{[d]}$ determines how much $\pi_t(s_t,\delta)$ "cares about" the dth basis reward. Preferences defined by non-linear combinations of reward have been considered in non-sequential settings (e.g., Thall 2008), but such approaches would be much more computationally challenging in the sequential decision making setting in addition to being much more challenging to interpret; we discuss this in Section 7.1. Throughout, we presume our data are trajectories where the ith one takes the form

$$s^i_1, a^i_1, r^i_{1[0]}, r^i_{1[1]}, ..., r^i_{1[D-1]}, s^i_2, a^i_2, r^i_{2[0]}, r^i_{2[1]}, ..., r^i_{2[D-1]}, ..., s^i_T, a^i_T, r^i_{T[0]}, r^i_{T[1]}, ..., r^i_{T[D-1]}.$$

3.2 Related Approaches

Inverse Reinforcement Learning (IRL) (e.g., see Ng and Russell, 2000) comprises a collection of methods that take as input trajectories acquired by observing an expert and then attempt to infer the reward function of that expert. While IRL methods also operate in a setting with unknown rewards, our goal is quite different since we explicitly assume that our data do *not* come from an expert—in fact actions are often chosen uniformly randomly. Furthermore, we do *not* attempt to recover the reward function of any particular agent; we will instead attempt to learn the optimal policy for a set of reward functions. IRL methods could be useful if one wanted to attempt to explicitly learn the preference (i.e., the δ) of a decision-maker under our proposed framework; we leave this as potential future work.

Partially Observable Markov Decision Process (POMDP) planning (Kaelbling et al., 1998) comprises a large collection of methods that are designed to learn policies in the face of *partial observability*, that is, when the current "nominal state"[3] of the system is not observed. In this framework, the agent maintains a *belief state* (i.e., distribution) over the current nominal state and defines a value function and policy over these belief states. In the simplest setting with k nominal states, the space of possible belief states is the set of vectors on the $(k-1)$-simplex, and value-based exact POMDP planners compute the optimal value function for all possible belief states. In effect,

2. The optimal policy may not be unique, but this does not concern us.
3. The term "nominal state" is used to denote the actual unobserved state of the system, so as to distinguish it from the "belief state," which comprises the agent's current beliefs about the system.

the POMDP planner defines and solves the *belief MDP* in which the belief states are considered to be the observed (continuous) state.

Our goal in this work is to learn the optimal value function and policy for all possible preferences δ, which also happen live on the simplex. The value functions we learn are similar in structure to those of the POMDP planning problem, but there are at least two important differences.

First and foremost, the value functions and policies we learn are functions of preference *and* of additional state (e.g., patient information) both of which are assumed to be observed. We will see that in our formulation, for any fixed state the optimal value function is piecewise linear in preference. The preference part of the value function has a structure similar to that of the value function of a belief MDP, which is piecewise linear in the belief state.

Second, in POMDP planning, value is always a *convex* function of belief state, and this property is crucial in the development of exact and approximate (e.g., Pineau et al. 2006, Wang et al. 2006) methods. However, we will show in Section 4.2.4 that because our approach estimates value functions using regression, the Q-functions in our problem are *not* convex in δ. Since we do not have convexity, we will develop alternative methods for representing value functions in Section 4.

Despite these two differences, it is possible to interpret our definition of preference as a "belief" that the agent/patient is in exactly one of D different hidden "preference states" each corresponding to a single basis reward. We will not approach the problem from this point of view since we prefer the interpretation that each agent (e.g., patient) has a true observable δ and a corresponding reward function given by (2), but there may be applications where the hidden "preference state" interpretation is preferable. In any case, the two differences mentioned above mean that even if we interpret δ as a belief over preference states, standard POMDP methods are not applicable.

4. Fitted-Q Iteration for Multiple Reward Functions

In order to illustrate the intuition behind our approach, we first describe an algorithm for learning policies for all possible δ in the simplest case: a finite state space with $D=2$ basis rewards. We then describe how to accomodate linear function approximation with an arbitrary number of features in the $D=2$ setting. We then give a generalization to arbitrary D, and we provide an explicit algorithm for the $D=3$ case based on methods from computational geometry.

4.1 Optimal Value Functions for All Tradeoffs: Finite State Space, $D=2$ Basis Rewards

To begin, we assume that the $\mathcal{S}_t$ are all finite, and that state transition probabilities $P(s_{t+1}|s_t,a_t)$ and expected rewards $r_{t[0]}(s_t,a_t),\ldots,r_{t[0]}(s_t,a_t)$ are estimated using empirical averages from the data set and "plugged in" where appropriate. From an algorithmic standpoint, in this setting there is no difference whether these quantities are known or estimated in this way. We therefore present our algorithm as though all expectations can be computed exactly.

First, we consider two basis rewards $r_{t[0]}$ and $r_{t[1]}$ and corresponding preferences $\delta=(\delta_{[0]},\delta_{[1]})$. In this setting, the range of possible reward functions can be indexed by a single scalar $\delta=\delta_{[1]}$ by defining

$$r_t(s_t,a_t,\delta)=(1-\delta)\cdot r_{t[0]}(s_t,a_t)+\delta\cdot r_{t[1]}(s_t,a_t).$$

We will show that the optimal state-action value function $V_t(s_t,\delta)$ is piecewise-linear in the tradeoff parameter δ. Where appropriate, we will use the notation $V_t(s_t,\cdot)$ to represent the function of

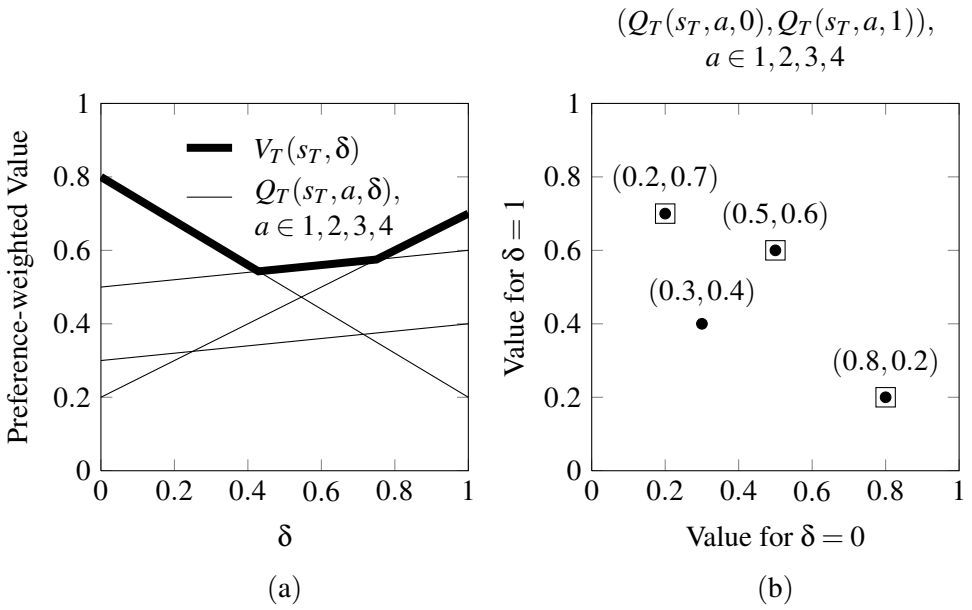

Figure 1: Computing V_T from Q_T for all δ by convex hull.

one argument (i.e., of δ) identified by fixing s_t. (We will use notation $Q_t(s_t,a_t,\cdot)$ and $r_t(s_t,a_t,\cdot)$ similarly.) We use an exact piecewise-linear representation of the functions $V_t(s_t,\cdot)$ for each state and timepoint, which allows us to exactly compute value backups for all δ more efficiently than the point-based representations of Barrett and Narayanan (2008). Our representation also allows identification of the set of dominated actions, that is, the actions that are not optimal for any (s_t,δ) pair. Value backups for finite state spaces require two operations: maximization over actions, and expectation over future states.

4.1.1 MAXIMIZATION

We begin at time $t = T$, the final time point,[4] and describe how to take a collection of functions $Q_T(s_T,a_T,\cdot)$ for all (s_T,a_T) and produce an explicit piecewise-linear representation of $V_T(s_T,\cdot)$ by maximizing over $a_T \in \mathcal{A}_T$. In Section 4.1.3, we show how this can be accomplished at earlier timepoints $t < T$ using a divide-and-conquer approach.

The Q-function for the last timepoint is equal to the terminal expected reward function r_T, which is linear in δ for each state-action pair as defined in (2), so we have $Q_T(s_T,a_T,\delta) = (1-\delta)\cdot r_{T[0]}(s_T,a_T) + \delta \cdot r_{T[1]}(s_T,a_T)$. To represent each $Q_T(s_T,a_T,\cdot)$, we maintain a list[5] of linear functions, one for each action, for each s_T. Figure 1(a) shows an example Q-function at a fixed state s_T at time T for four different actions, three of which are optimal for some δ and one which is not optimal for any δ. The linear function for each action can be represented by a list of tradeoff points (i.e., [0 1]) together with a list of their corresponding values (i.e., $[Q_T(s_T,a_T,0)\ Q_T(s_T,a_T,1)]$) at those tradeoff points. Each can also be represented by a point $(Q_T(s_T,a_T,0),Q_T(s_T,a_T,1))$ in the plane, as shown in Figure 1(b). These two equivalent representations offer an important concep-

4. We will write Q_T rather than $Q_{t=T}$ in this section, and similarly write $s_T, \mathcal{A}_T$, etc.
5. We denote an ordered list with objects a,b,c by $[a\ b\ c]$.

tual and computational insight that is well-established in the multi-criterion optimization literature (Ehrgott, 2005): the set of actions that are optimal for some $\delta \in [0,1]$ are exactly those actions whose line-representations lie on the upper convex envelope of the Q-functions, and equivalently, whose point-based representations lie on the upper-right convex hull of the set of points in the plane. In general, we can recover the actions that are optimal on any interval $[\delta, \delta']$ by finding the upper-right convex hull of the points $\{(Q_T(s_T,a,\delta), Q_T(s_T,a,\delta')) : a \in \{1...|A|\}\}$. This equivalence is important because the time complexity of the convex hull operation on n points in two or three dimensions is $O(n \log n)$ (de Berg et al., 2008)—as fast as sorting.

We make use of this equivalence to construct our piecewise-linear representation of $V_T(s,\cdot)$. Commonly-used convex hull routines produce output that is ordered, so it is easy to recover the list of actions that are optimal for some δ, along with the values of δ where the optimal action changes. These values are the "knots" in the piecewise-linear representation. We denote the list of knots of a piecewise-linear function $f(\cdot)$ by $\Delta(f(\cdot))$. The output of a convex hull algorithm is an ordered list of points, each of the form $(Q_T(s_T,a_T,0), Q_T(s_T,a_T,1))$. In this case, the list is $[(0.8,0.2)\ (0.5,0.6)\ (0.2,0.7)]$. We know from the order of this list that the second knot in $V_T(s,\cdot)$ (after $\delta = 0$) occurs where the lines represented by $(0.8,0.2)$ and $(0.5,0.6)$ intersect. Thus we can compute that the line represented by $(0.8,0.2)$ is maximal from $\delta = 0$ to $\delta = 0.43$, at which point it intersects the line represented by $(0.5,0.6)$. After finding the knots, we represent the piecewise-linear value function in Figure 1(a) by the ordered knot-list $\Delta(V_T(s_T,\cdot)) = [0.00\ 0.43\ 0.75\ 1.00]$ and value-list $[0.80\ 0.54\ 0.58\ 0.70]$, rather than by the list of points. To recover the policy at this point, we may also retain a list of lists containing the actions that are optimal at each knot: $[[1]\ [1\ 2]\ [2\ 4]\ [4]]$. This allows us to determine the action or actions that are optimal for any segment by taking the intersection of the action lists for the endpoints of the segment. Note that because $V_T(s_T,\cdot)$ is a point-wise maximum of convex[6] functions, it is convex.

Our representation allows us to evaluate $V_T(s_T,\delta) = \max_{a \in \mathcal{A}_T} Q_T(s_T,a_T,\delta)$ efficiently. Because our knot list and value list are ordered, we can use binary search to find the largest knot in $V_T(s_T,\cdot)$ that is less than δ. This tells us which linear piece is maximal for δ, so we only need to evaluate this single linear function. Thus computing $V_T(s_T,\delta)$ takes $O(\log|\Delta(V_T(s_T,\cdot))|)$ time, that is, the time for the cost of the search, rather than the $O(|\Delta(V_T(s_T,\cdot))|)$ time it would take to evaluate all of the linear functions at δ and then take the maximum.

4.1.2 EXPECTATION

We now show how we use our piecewise-linear representation of $V_T(s_T,\cdot)$ to efficiently compute a piecewise-linear representation of

$$Q_{T-1}(s_{T-1},a_{T-1},\cdot) = r_{T-1}(s_{T-1},a_{T-1},\cdot) + E_{S_T}[V_T(S_T,\cdot)|s_{T-1},a_{T-1}]$$

using the piecewise-linear representation of V_T. To do so, we must evaluate conditional expectations of V_T over possible future states.

Consider an example with two terminal states $s_T = 1$ and $s_T = 2$. Suppose that the probability of arriving in state j (conditioned on some (s_{T-1},a_{T-1})) is given by θ_j. Since each $V_T(j,\cdot)$ is linear over the intervals between $\Delta(V_T(j,\cdot))$, these two functions are *simultaneously* linear over the intervals between $\Delta(V_T(1,\cdot)) \cup \Delta(V_T(2,\cdot))$, and their weighted average is linear over the same

6. The $Q_T(s_T,a_T,\cdot)$ are each linear, and therefore convex.

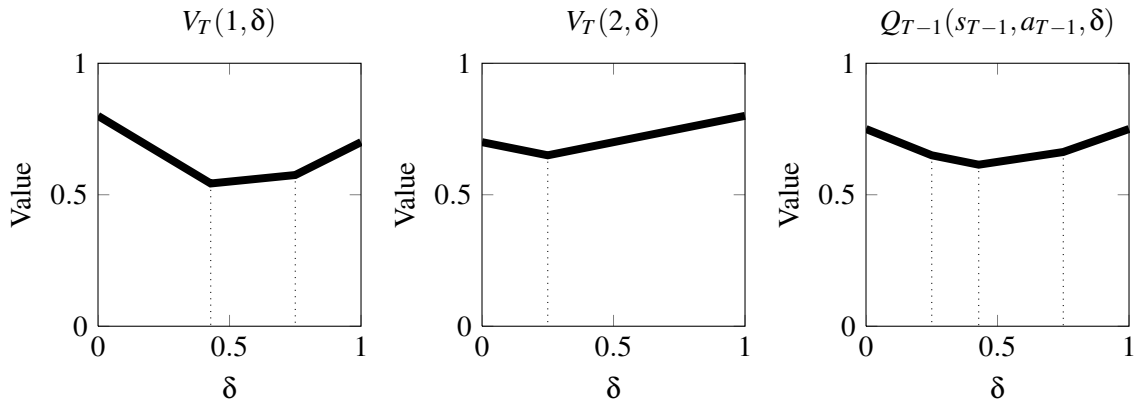

Figure 2: Computing expectations using unions of piecewise-linear representations. The graphs of $V_T(1,\delta)$ and $V_T(2,\delta)$ show the time T value function at two different states, $S_T = 1$ and $S_T = 2$. The graph of $Q_{T-1}(s_{T-1},a_{T-1},\delta) = 0.5 \cdot V_T(1,\delta) + 0.5V_T(2,\delta)$ shows the expected action-value function if the two states are each reached with probability 0.5 when starting from state $S_{T-1} = s_{T-1}$ and taking action $A_{T-1} = a_{T-1}$, and there is no immediate reward.

intervals. Therefore the expectation

$$E_{S_T}[V_T(S_T,\cdot)|s_{T-1},a_{T-1}] = \theta_1 \cdot V_T(1,\cdot) + \theta_2 \cdot V_T(2,\cdot)$$

is itself a piecewise-linear function of δ with knot-list $\Delta(V_T(1,\cdot)) \cup \Delta(V_T(2,\cdot))$. Since the function $r_{T-1}(s_{T-1},a_{T-1},\cdot)$ is linear, it does not contribute additional knots, so we can compute the piecewise-linear representation of $Q_{T-1}(s_{T-1},a_{T-1},\cdot)$ by computing its value-list at the aforementioned knots. The value list of $Q_{T-1}(s_{T-1},a_{T-1},\cdot)$ is

$$\left[\left(r_{T-1}(s_{T-1},a_{T-1},\delta) + \sum_j \theta_j V_T(j,\delta)\right) : \delta \in \Delta(V_T(1,\cdot)) \cup \Delta(V_T(2,\cdot))\right].$$

Let $k_j = |\Delta(V_T(s'_j,\cdot))|$. This construction uses $O(k_1+k_2)$ space and requires $O(k_1+k_2)$ evaluations of V_T. Note that because $Q_{T-1}(s_{T-1},a_{T-1},\cdot)$ is a positive weighted sum of convex functions, it is convex. Figure 2 illustrates this weighted sum operation.

We contrast the piecewise-linear representation approach with that of Barrett and Narayanan (2008). The expectation can also be computed using the point-based representation in Figure 1(b): let Ξ_i be the set of points in the point-based representation of $V_T(s_T,\cdot)$. One can compute the point-based representation of $Q_{T-1}(s_{T-1},a_{T-1},\cdot)$ by constructing a set of points

$$\begin{aligned}&\{(r_{(T-1)[0]}, r_{(T-1)[1]}) + \theta_1 \cdot (a_1,b_1) + \theta_2 \cdot (a_2,b_2)\},\\ &\text{where } r_{(T-1)[\delta]} = r_{T-1}(s_{T-1},a_{T-1},\delta)\\ &\text{for all } (a_1,b_1) \in \Xi_1,\ \ (a_2,b_2) \in \Xi_2.\end{aligned} \tag{3}$$

and then taking the upper-right portion of the convex hull of this set. Barrett and Narayanan (2008) advocate this procedure and prove its correctness; however, they note that the set given in (3) has

$|\Xi_1||\Xi_2|$ points that must be constructed and fed into the convex hull algorithm. Since $k_i = |\Xi_i| + 1$, computing the expectation in this way will take $O(k_1 k_2)$ space and $O(k_1 k_2 \log k_1 k_2)$ time, which is asymptotically much less efficient than our $O(k_1 + k_2)$ piecewise-linear representation based approach.

4.1.3 VALUE BACKUPS FOR $t < T-1$

The maximization procedure described in Section 4.1.1 relies on the linearity of $Q_T(s_T, a_T, \cdot)$. However, for $t < T$, we have shown that $Q_t(s_t, a_t, \cdot)$ is piecewise-linear. We now show how to compute V_t and Q_t from Q_{t+1} by first decomposing $Q_{t+1}(s_{t+1}, a_{t+1}, \cdot)$ into linear pieces and applying the expectation and maximization operations to each piece. Recall that

$$Q_t(s_t, a_t, \delta) = r_t(s_t, a_t, \delta) + E_{S_{t+1}}[V_{t+1}(S_{t+1}, \delta) | s_t, a_t].$$

We have shown by construction that $Q_{T-1}(s_{T-1}, a_{T-1}, \cdot)$ is convex and piecewise-linear. In general, $V_t(s_t, \cdot)$ is computed by taking a point-wise max over functions $Q_t(s_t, a_t, \cdot)$, and $Q_{t-1}(s_{t-1}, a_{t-1}, \cdot)$ is computed by taking a positive weighted sum of the convex functions $r_{t-1}(s_{t-1}, a_{t-1}, \cdot)$ and $V_t(s_t, \cdot)$. Since both of these operations preserve convexity and piecewise-linearity, it follows by induction that $Q_t(s_t, a_t, \cdot)$ is convex piecewise-linear for all $t \in 1, ..., T$. To compute $Q_t(s_t, a_t, \cdot)$, we first identify the knots in $E_{S_{t+1}}[V_{t+1}(S_{t+1}, \cdot) | s_t, a_t]$ and store them; this is done in the same way as for $t+1 = T$. We then compute the value-list as described above. To compute $V_t(s_t, \cdot) = \max_{a \in \mathcal{A}_t} Q_t(s_t, a, \cdot)$, we take the maximum over actions of these piecewise-linear Q-functions using Algorithm 2. First, we decompose the problem of finding $\max_{a \in \mathcal{A}_t} Q_t(s_t, a, \cdot)$ for $\delta \in [0, 1]$ into sub-problems of finding $\max_{a \in \mathcal{A}_t} Q_t(s_t, a, \cdot)$ over intervals of δ where we know the $Q_t(s_t, a, \cdot)$ are simultaneously linear. The ends of these intervals are given by $\bigcup_a \Delta(Q_t(s_t, a, \cdot))$. We then apply the convex hull algorithm to each of these intervals to recover any additional knots in $\max_{a \in \mathcal{A}_t} Q_t(s_t, a_t, \cdot)$.

The full backup procedure is described in Algorithm 1. In practice, we can avoid running the convex hull algorithm over every interval by checking each interval's end points: if for some action a_t^* we find that $Q_t(s_t, a_t^*, \cdot)$ is maximal at both ends of an interval in the current knot-list, then $\max_a Q_t(s_t, a, \cdot)$ has no knots inside the interval. Note that though we present our algorithms assuming the reward functions are linear, they will work for piecewise-linear reward functions as well.

4.1.4 COMPLEXITY OF $Q_t(s_t, a_t, \cdot)$ AND $V_t(s_t, \cdot)$

Suppose there are $|\mathcal{S}|$ states and $|\mathcal{A}|$ actions at each stage. For any fixed s_T, each function $Q_T(s_T, i, \cdot)$, $i = 1..|\mathcal{A}|$, has 2 knots, $\delta = 0$ and $\delta = 1$. Applying Algorithm 2 to produce $V_T(s_T, \cdot)$ from these functions generates at most $|\mathcal{A}| - 1$ new internal knots, and therefore each $V_T(s_T, \cdot)$ has at most $(|\mathcal{A}| - 1) + 2$ knots. To compute $Q_{T-1}(s_{T-1}, a_{T-1}, \cdot)$, we take the expectation of $V_T(s_T, \cdot)$ over states s_T. Since $V_T(s_T, \cdot)$ might have different internal knots for every s_T, $Q_{T-1}(s_{T-1}, a_{T-1}, \cdot)$ may have as many as $|\mathcal{S}|(|\mathcal{A}| - 1) + 2$ knots. However, the knots in $Q_{T-1}(s_{T-1}, a_{T-1}, \cdot)$ will be the same for all s_{T-1} and a_{T-1}. Computing $V_{T-1}(s_{T-1}, \cdot)$ using Algorithm 2 adds at most $|\mathcal{A}| - 1$ new knots between each pair of existing knots, for a total of $(|\mathcal{A}| - 1|)(|\mathcal{S}|(|\mathcal{A}| - 1) + 1) + 2$. In general, $Q_t(s_t, a_t, \cdot)$ may have up to $O(|\mathcal{S}|^{T-t}|\mathcal{A}|^{T-t})$ knots, and $V_t(s_t, \cdot)$ may have up to $O(|\mathcal{S}|^{T-t}|\mathcal{A}|^{(T-t)+1})$ knots.

To compute $Q_t(s_t, a_t, \cdot)$ from r_t and V_{t+1}, our approach requires $O(|\mathcal{S}|^{T-t}|\mathcal{A}|^{(T-t)+1})$ time for each state, for a total of $O(|\mathcal{S}|^{(T-t)+1}|\mathcal{A}|^{(T-t)+1})$ time. In contrast, the approach of Barrett &

Algorithm 1 Value Backup - Finite State Space

/* $A \overset{\cup}{\leftarrow} B$ means $A \leftarrow A \cup B$ */
$\forall (s_{T+1}, \delta)$, $V_{T+1}(s_{T+1}, \delta) \triangleq 0$. $\forall s_{T+1}$, $\Delta(V_{T+1}(s_{T+1}, \cdot)) \triangleq \{0,1\}$.
for $t = T$ **downto** 1 **do**
  for all $s_t \in \mathcal{S}_t$ **do**
    for all $a_t \in \mathcal{A}_t$ **do**
      $\Delta(Q_t(s_t, a_t, \cdot)) \leftarrow \{\}$
      for all $s_{t+1} \in \mathcal{S}_{t+1}$ **do**
        $\Delta(Q_t(s_t, a_t, \cdot)) \overset{\cup}{\leftarrow} \Delta(V_{t+1}(s_{t+1}, \cdot))$
      end for
      for all $\delta \in \Delta(Q_t(s_t, a_t, \cdot))$ **do**
        $Q_t(s_t, a_t, \delta) \leftarrow r(s_t, a_t, \delta) +$
          $\sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) \cdot V_{t+1}(s_{t+1}, \delta)$
      end for
    end for
    Compute $\Delta(V_t(s_t, \cdot))$ by applying Algorithm 2
    to $Q_t(s_t, a, \cdot)$, $a \in \mathcal{A}_t$
  end for
end for

Algorithm 2 Max of Piecewise-Linear Functions

/* $A \overset{\cup}{\leftarrow} B$ means $A \leftarrow A \cup B$ */
input piecewise-linear functions $f_i(\cdot)$, $i = 1..k$ defined on $[\delta_0, \delta_1]$.
$\Delta^{\text{all}} = \bigcup_{i=1}^{k} \Delta(f_i(\cdot))$
$\Delta^{\text{out}} = \Delta^{\text{all}}$
for $i = 2$ **to** $|\Delta^{\text{all}}|$ **do**
  if $\text{argmax}_j f_j(\Delta^{\text{all}}_{i-1}) \neq \text{argmax}_j f_j(\Delta^{\text{all}}_i)$ **then**
    $\Delta^{\text{out}} \overset{\cup}{\leftarrow} \Delta(\max_j f_j(\delta), \delta \in (\Delta^{\text{all}}_{i-i}, \Delta^{\text{all}}_i))$
  end if
end for

Narayanan requires $O(|\mathcal{S}|^{2\cdot(T-t)+1}|\mathcal{A}|^{2\cdot(T-t)+1} \log |\mathcal{S}|^{2\cdot(T-t)+1}|\mathcal{A}|^{2\cdot(T-t)+1})$ time for each of $\log_2 |\mathcal{S}|$ pairs of piecewise-linear functions.

4.2 Optimal Value Functions for All Tradeoffs: Linear Function Approximation, $D = 2$ Basis Rewards

Here, we demonstrate how our previously developed algorithms for value backups over all tradeoffs can be extended to the case where we have arbitrary features of state variables and we use a linear approximation of the Q_t functions. Again, we first consider Q_T and V_T, which have the simplest form, and then describe how to compute Q_t and V_t at earlier time points. This treatment allows for linear function approximators based on an arbitrary number of state features rather than the single continuous state feature described in Lizotte et al. (2010).

Suppose the expected terminal rewards $Q_T(s_T, a_T, 0)$ and $Q_T(s_T, a_T, 1)$ are each linear functions of the form $Q_T(s_T, a_T, 0) = \phi_{s_T,a_T}^{\mathsf{T}} \beta_{T[0]}$ and $Q_T(s_T, a_T, 1) = \phi_{s_T,a_T}^{\mathsf{T}} \beta_{T[1]}$. Here, ϕ_{s_T,a_T} is a *feature vector*[7] that depends on state and action, and the weight vectors $\beta_{T[0]}$ and $\beta_{T[1]}$ define the linear relationships between the feature vector and the expectation of the two basis rewards at time T. From Equation (2), we have

$$\begin{aligned} Q_T(s_T, a_T, \delta) &= (1-\delta) \cdot \phi_{s_T,a_T}^{\mathsf{T}} \beta_{T[0]} + \delta \cdot \phi_{s_T,a_T}^{\mathsf{T}} \beta_{T[1]}, \\ &= \phi_{s_T,a_T}^{\mathsf{T}} \left[(1-\delta) \cdot \beta_{T[0]} + \delta \cdot \beta_{T[1]} \right]. \end{aligned} \tag{4}$$

A typical definition of ϕ_{s_T,a_T} might include a constant component for the intercept, the measurements contained in s_T, the discrete action a_T encoded as dummy variables, and the product of the measurements in s_T with the encoded a_T (Cook and Weisberg, 1999). One could also include other non-linear functions of s_T and a_T as features if desired. In particular, one could produce exactly the same algorithm described in Section 4.1 by using feature vectors of length $|\mathcal{S}| \times |\mathcal{A}|$ that consist of a separate indicator for each state-action pair. In this case the estimated parameters will be precisely the sample averages from Section 4.1. Note from (4) that regardless of the definition of ϕ_{a_T,s_T}, the function $Q_T(s_T, a_T, \delta)$ is linear in δ for any fixed s_T, a_T.

Recall that we have a set of trajectories of the form

$$s_1^i, a_1^i, r_{1[0]}^i, r_{1[1]}^i, s_2^i, a_2^i, r_{2[0]}^i, r_{2[1]}^i, ..., s_T^i, a_T^i, r_{T[0]}^i, r_{T[1]}^i,$$

for $i = 1...N$ with which to estimate the optimal Q functions. In order to estimate $Q_T(s_T, a_T, 0)$ and $Q_T(s_T, a_T, 1)$, we compute parameter estimates $\hat{\beta}_{T[0]}$ and $\hat{\beta}_{T[1]}$ using ordinary least-squares regression by first constructing a design matrix and regression targets

$$\Phi_T = \begin{bmatrix} \phi_{s_T^1,a_T^1}^{\mathsf{T}} \\ \phi_{s_T^2,a_T^2}^{\mathsf{T}} \\ \vdots \\ \phi_{s_T^N,a_T^N}^{\mathsf{T}} \end{bmatrix}, \quad \mathbf{r}_{T[0]} = \begin{bmatrix} r_{T[0]}^1 \\ r_{T[0]}^2 \\ \vdots \\ r_{T[0]}^N \end{bmatrix}, \quad \mathbf{r}_{T[1]} = \begin{bmatrix} r_{T[1]}^1 \\ r_{T[1]}^2 \\ \vdots \\ r_{T[1]}^N \end{bmatrix}.$$

We then compute parameter estimates

$$\begin{aligned} \hat{\beta}_{T[0]} &= (\Phi_T^{\mathsf{T}} \Phi_T)^{-1} \Phi_T^{\mathsf{T}} \mathbf{r}_{T[0]}, \\ \hat{\beta}_{T[1]} &= (\Phi_T^{\mathsf{T}} \Phi_T)^{-1} \Phi_T^{\mathsf{T}} \mathbf{r}_{T[1]}, \end{aligned}$$

using ordinary least squares.[8] These estimated parameters are then substituted into definition (4), giving $\hat{Q}_T(s_T, a_T, 0)$ and $\hat{Q}_T(s_T, a_T, 1)$. To construct an estimate $\hat{Q}_T(s_T, a_T, \delta)$ for arbitrary $\delta \in [0, 1]$, we could construct a scalar reward using $\mathbf{r}_{t[0]}$, $\mathbf{r}_{t[1]}$, and δ, and solve for the corresponding $\hat{\beta}_T(\delta)$, giving

$$\hat{\beta}_T(\delta) = (\Phi_T^{\mathsf{T}} \Phi_T)^{-1} \Phi_T^{\mathsf{T}} \left[(1-\delta) \mathbf{r}_{T[0]} + \delta \mathbf{r}_{T[1]} \right],$$

7. In statistical terms, $\phi_{s_T,a_T}^{\mathsf{T}}$ represents a row in the design matrix of the linear model.
8. We could also use ridge regression with a fixed ridge parameter. All of our techniques immediately apply in this case as well since the parameter estimates remain piecewise linear in δ.

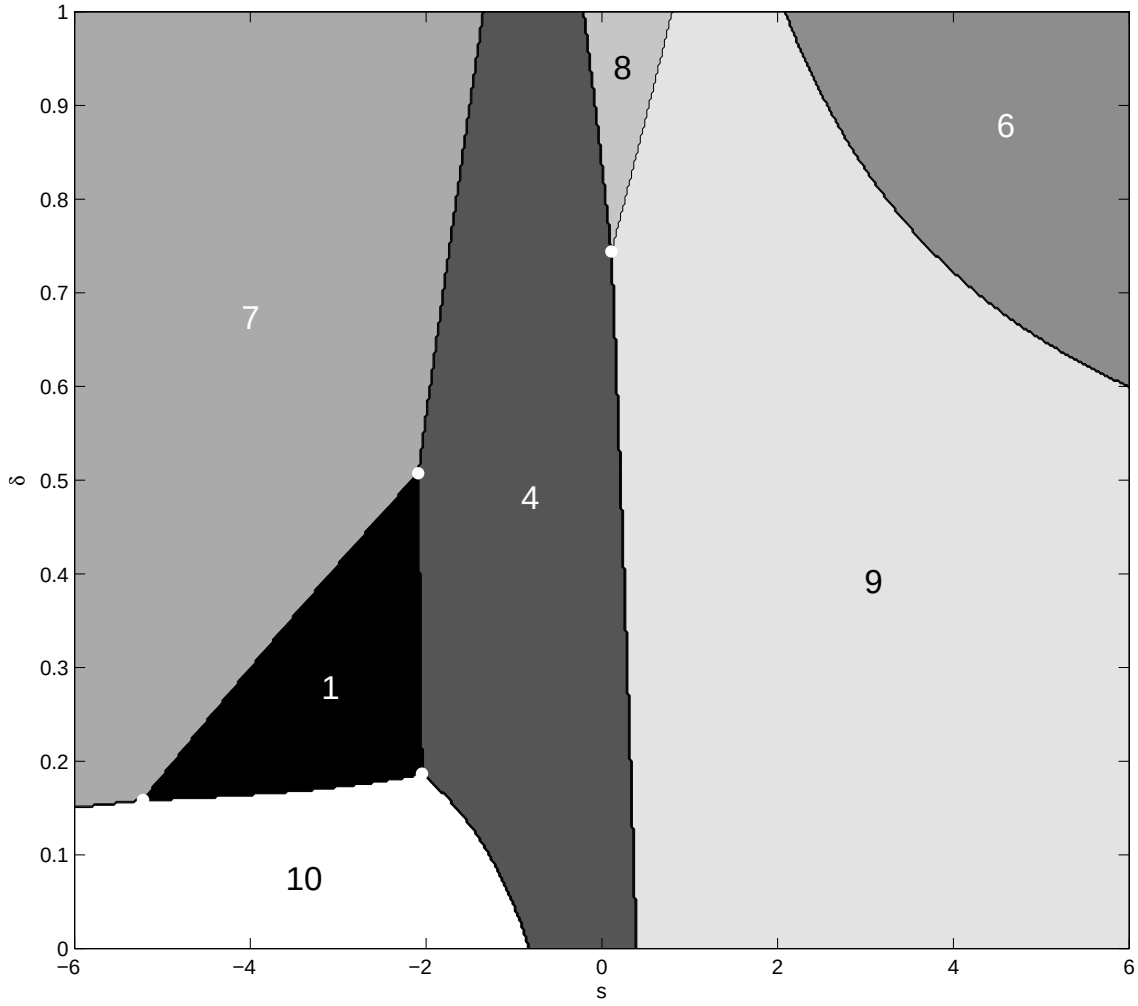

Figure 3: Diagram of the regions in (ψ_{s_T}, δ) space where different actions are optimal at time T. In this example, $\psi_{s_T} \in [-6, 6]$.

but by linearity,

$$\begin{aligned}\hat{\beta}_T(\delta) &= (1-\delta)\cdot(\Phi_T^{\mathsf{T}}\Phi_T)^{-1}\Phi_T^{\mathsf{T}}\mathbf{r}_{T[0]} + \delta\cdot(\Phi_T^{\mathsf{T}}\Phi_T)^{-1}\Phi_T^{\mathsf{T}}\mathbf{r}_{T[1]}, \\ &= (1-\delta)\cdot\hat{\beta}_{T[0]} + \delta\cdot\hat{\beta}_{T[1]}.\end{aligned}$$

Thus we only need to solve for $\hat{\beta}_{T[0]}$ and $\hat{\beta}_{T[1]}$, after which we compute $\hat{Q}_T(s_T, a_T, \delta) = \phi_{s_T,a_T}^{\mathsf{T}}\hat{\beta}_T(\delta)$ for any δ by taking convex combinations of these coefficient vectors. Therefore, for $t = T$, it is straightforward to exactly represent $\hat{Q}_T(s_T, a_T, \delta)$ for all δ.

4.2.1 MAXIMIZATION

For any fixed values of s_T and a_T, $\hat{Q}_T(s_T, a_T, \cdot)$ is a linear function. Therefore, we can use the convex hull method to identify the actions that maximize value at a given s_T, and use it to recover the knots in the piecewise-linear $\hat{V}_T(s_T, \cdot)$.

Figure 3 is an illustration of the pieces of a hypothetical $\hat{V}_T(s_T, \cdot)$ that is a maximization over 10 actions. In this example we define a scalar state feature ψ_{s_T} and we construct feature vectors

$$\phi_{s_T, a_T} = [[1 \ \psi_{s_T}] 1_{a_T=1} \ \ [1 \ \psi_{s_T}] 1_{a_T=2} \ \ \ldots \ \ [1 \ \psi_{s_T}] 1_{a_T=10}]^{\mathsf{T}}, \tag{5}$$

where $1_{a_T=k}$ is the indicator function that takes the value 1 if $a_T = k$ and 0 otherwise. Note that this choice of features is equivalent to defining $\hat{Q}$ using a separate linear regression for each action. Each number in Figure 3 marks the region where that action is optimal at time T. For example, a vertical slice at $\psi_{s_T} = -4$ of the value function has three linear pieces where actions 10, 1, and 7 are optimal.

In the finite state-space case, we explicitly represented $V_T(s_T, \cdot)$ separately for each nominal state s_T in the MDP in order to allow computation of expectations over terminal states. In contrast, in the linear regression setting, we represent $\hat{V}_T(s_T, \cdot)$ for each *observed* terminal state $s_T^1, \ldots, s_T^N$ in our data set. That is, we explicitly represent a one-dimensional slice of the value function for each of the s_T^i by applying Algorithm 2 to construct a piecewise-linear representation for $\hat{V}_T(s_T^i, \cdot)$.

4.2.2 REGRESSION ON STATE FEATURES

At stage $T-1$, the regression parameters of our estimate $\hat{Q}_{T-1}(s_{T-1}, a_{T-1}, \delta)$ are given by

$$\hat{\beta}_{T-1}(\delta) = (\Phi_{T-1}^{\mathsf{T}} \Phi_{T-1})^{-1} \Phi_{T-1}^{\mathsf{T}} \hat{\mathbf{y}}_{T-1}(\delta),$$

where, for $t \in \{1, \ldots, T-1\}$, we define

$$\hat{\mathbf{y}}_t(\delta) = ((1-\delta)\mathbf{r}_{t[0]} + \delta \mathbf{r}_{t[1]}) + \hat{\mathbf{v}}_{t+1}(\delta),$$

which are the one-step value estimates for time t, where

$$\hat{\mathbf{v}}_{t+1}(\delta) = \begin{bmatrix} \hat{V}_{t+1}(s_{t+1}^1, \delta) \\ \hat{V}_{t+1}(s_{t+1}^2, \delta) \\ \vdots \\ \hat{V}_{t+1}(s_{t+1}^N, \delta) \end{bmatrix}.$$

The components of the vector $\hat{\mathbf{y}}_{T-1}(\delta)$ are not linear in δ, so for $t < T$, solving the regression only for $\delta = 0$ and $\delta = 1$ does not completely determine $\hat{Q}_t(s_t, a_t, \delta)$. However, the components of $\hat{\mathbf{y}}_{T-1}(\delta)$ are each piecewise-linear in δ. We determine the intervals over which the components are simultaneously linear and then explicitly represent the state-value function at the knots $[\delta_1 \ \delta_2 \ \ldots \ \delta_K]$ between these intervals. The output accompanying this list of knots is a list of estimated parameter vectors $[\beta_{T-1}(\delta_1) \ \beta_{T-1}(\delta_2) \ \ldots \ \beta_{T-1}(\delta_K)]$, each given by $\hat{\beta}_{T-1}(\delta_k) = (\Phi_{T-1}^{\mathsf{T}} \Phi_{T-1})^{-1} \Phi_{T-1}^{\mathsf{T}} \hat{\mathbf{y}}_{T-1}(\delta_k)$. This collection of parameters is analogous to the *value list* in the finite state-space case, and completely defines $\hat{Q}_{T-1}(s_{T-1}, a_{T-1}, \cdot)$ for all s_{T-1} and a_{T-1}. As before, we can also retain a list of the optimal actions at each of the knots in order to later recover the policy.

Algorithm 3 Value Backup - Linear Function Approximation, $D=2$ Basis Rewards

$\forall(s,\delta)$, $\hat{V}_{T+1}(s,\delta) \triangleq 0$. $\forall s$, $\Delta(\hat{V}_{T+1}(s,\cdot)) \triangleq \{0,1\}$.
for $t = T$ **downto** 1 **do**
  $\Delta^{\hat{Q}_t} \leftarrow \{\}$
  for all $(s_t, a_t, s_{t+1}) \in \mathcal{D}$ **do**
    $\Delta^{\hat{Q}_t} \leftarrow \Delta^{\hat{Q}_t} \cup \Delta(\hat{V}_{t+1}(s_{t+1},\cdot))$
  end for
  for all $\delta \in \Delta^{\hat{Q}_t}$ **do**
    $\mathbf{y}_t^{(\delta)} = ((1-\delta)\mathbf{r}_{t[0]} + \delta\mathbf{r}_{t[1]}) + \hat{\mathbf{v}}_{t+1}(\delta)$
    $\hat{\beta}_t^{(\delta)} = (\Phi_t^\mathsf{T}\Phi_t)^{-1}\Phi_t^\mathsf{T}\mathbf{y}_t^{(\delta)}$
  end for
  for all $s_t \in \mathcal{D}$ **do**
    Compute $\Delta(\hat{V}_t(s_t,\cdot))$ by Algorithm 2
  end for
end for

4.2.3 VALUE BACKUPS FOR $t < T-1$

The procedure for computing the $\hat{V}_T(s_T^i,\cdot)$ relies on the linearity of $\hat{Q}_T(s_T^i,a_T,\cdot)$, but for $t<T$, $\hat{Q}_t(s_t,a_t,\cdot)$ is piecewise-linear in general. Thus to compute $\hat{V}_t(s_t^i,\cdot) = \max_a \hat{Q}_t(s_t^i,a,\cdot)$ for each s_t^i in our data set, we apply Algorithm 2 for each s_t^i, using regression to compute $Q_{t-1}(s_{t-1},a_{t-1},\cdot)$ from these functions then proceeds as we did for the $t=T-1$ case. The entire procedure is described in Algorithm 3.

4.2.4 NON-CONVEXITY OF $\hat{Q}_t(s_t,a_t,\cdot)$

For $t<T$, the resulting $\hat{Q}_t(s_t,a_t,\cdot)$ are not necessarily convex in the regression setting, as we alluded to in Section 3.2. To see this, recall that each element of $\hat{\beta}_{T-1}(\delta)$ is a weighted sum of the piecewise-linear $\hat{\mathbf{y}}_{T-1}(\cdot)$:

$$\begin{aligned}\hat{\beta}_{T-1}(\delta_k) &= (\Phi_{T-1}^\mathsf{T}\Phi_{T-1})^{-1}\Phi_{T-1}^\mathsf{T}\hat{\mathbf{y}}_{T-1}(\delta_k),\\ &= \mathbf{w}_{T-1}^\mathsf{T}\cdot\hat{\mathbf{y}}_{T-1}(\delta).\end{aligned}$$

Here, $\mathbf{w}_{T-1}$ is an $1\times N$ vector that depends on s and on the data, but does not depend on δ. Elements of $\mathbf{w}_{T-1}$ can be positive or negative, depending on the feature representation used and the particular data set on hand. Therefore, although each element of $\hat{\mathbf{y}}_{T-1}(\cdot)$ is a convex, piecewise-linear function, the $\hat{\beta}_t(\cdot)$, and therefore the $\hat{Q}_t(s_t,a_t,\cdot)$ may not be convex for $t<T$. One consequence of this non-convexity is that both the algorithm by Barrett and Narayanan (2008), as well as important algorithms from the POMDP literature (e.g., Pineau et al., 2003) that operate on convex piecewise-linear value functions, cannot represent the function $\hat{Q}_t(s_t,a_t,\cdot)$ for $t<T$.

4.2.5 COMPLEXITY OF $\hat{Q}_t(s_t,a_t,\cdot)$ AND $\hat{V}_t(s_t,\cdot)$

Suppose there are N trajectories and $|\mathcal{A}|$ actions at each time point. For any fixed s_T and a_T, the final learned Q-function $\hat{Q}_T(s_T,a_T,\cdot)$ has two knots, one at $\delta=0$ and one at $\delta=1$. The terminal value function $\hat{V}_T(s_T^i,\cdot)$ is constructed at each of N points in state space by applying Algorithm 2 to the

$\hat{Q}_T(s_T^i, a, \cdot)$ for each observed terminal state $s_T^1, s_T^2, ..., s_T^N$ in $\mathcal{D}$. Each resulting $\hat{V}_T(s_T^i, \cdot)$ has at most $|\mathcal{A}| - 1$ new internal knots, and therefore each has at most $(|\mathcal{A}| - 1) + 2$ knots in total. To compute $\hat{Q}_{T-1}(s_{T-1}, a, \cdot)$, we use regression with targets constructed from the N value function estimates $\hat{V}_T(s_T^i, \cdot)$. In general, the knots for each $\hat{V}_T(s_T^i, \cdot)$ will be unique. Thus each $\hat{Q}_{T-1}(s_{T-1}, a, \cdot)$, whose knots are the union of the knots of the $\hat{V}_T(s_T^i, \cdot)$, will have at most $N \cdot (|\mathcal{A}| - 1) + 2$ knots. Computing $\hat{V}_{T-1}(s_{T-1}^i, \cdot)$ using Algorithm 2 adds at most $|\mathcal{A}| - 1$ new knots between each pair of knots in the union, for a total of $(|\mathcal{A}| - 1|)(N \cdot (|\mathcal{A}| - 1) + 1) + 2$ knots. In general, $\hat{Q}_t(s, a, \cdot)$ may have up to $O(N^{T-t}|\mathcal{A}|^{T-t})$ knots, and $\hat{V}_t(s, \cdot)$ may have up to $O(N^{T-t}|\mathcal{A}|^{(T-t)+1})$ knots. To compute the expectation described in Section 4.2.2 at time t, our approach requires $O(N^{T-t}|\mathcal{A}|^{(T-t)+1})$ for each trajectory, for a total of $O(N^{(T-t)+1}|\mathcal{A}|^{(T-t)+1})$ time.

4.3 Optimal Value Functions for All Tradeoffs: Linear Function Approximation, $D > 2$ Basis Rewards

We have seen that, for $D = 2$ basis rewards, $\hat{Q}_t(s_t, a_t, \cdot) = \phi_{s_t, a_t}^{\mathsf{T}} \hat{\beta}_t(\cdot)$ is continuous and piecewise-linear, but not convex. This remains true for D reward functions and D tradeoffs $\delta = \delta_{[0]}, ..., \delta_{[D-1]}$, but as D increases, representing $\hat{Q}_t(s_t, a_t, \cdot)$ becomes more difficult. In the general case, $\hat{Q}_t(s_t, a_t, \cdot)$ and $\hat{\beta}_t(\cdot)$ are linear over pieces that are convex polytopes within the space of possible preferences. We prove this below and show how this insight can be used to develop representations of $\hat{Q}_t$ and $\hat{V}_t$. As in the $D = 2$ case, we can construct $\hat{Q}_t(s_t, a_t, \cdot)$ and $\hat{V}_t(s_t, \cdot)$ for all $t \leq T$ by taking pointwise maximums and pointwise weighted sums of piecewise-linear functions. All proofs are deferred to Appendix A.

Definition 1 (Linear functions over convex polytopes) *A function $f : \mathbb{R}^D \to \mathbb{R}$ is* linear *over a convex polytope $\mathcal{R} \subseteq \mathbb{R}^D$ if*

$$\exists \mathbf{w} \in \mathbb{R}^D : f(\delta) = \delta^{\mathsf{T}} \mathbf{w} \; \forall \delta \in \mathcal{R}.$$

Since $\mathcal{R}$ is a convex polytope, it can be decomposed into a finite collection of *simplices* $\mathcal{T}_i$, each with $\mathcal{D}$ vertices, such that $\mathcal{R} = \cup_i \mathcal{T}_i$ (Grünbaum, 1967). Each simplex is itself a convex polytope. For a simplex $\mathcal{T}$ with vertices $\delta^1, \delta^2, ..., \delta^D$, the weight vector $\mathbf{w}$ of a linear function $f(\delta) = \delta^{\mathsf{T}} \mathbf{w}$ defined over $\mathcal{T}$ can be computed from the values $y^1, y^2, ..., y^{D-1}$ that f takes on the vertices, together with the locations of the vertices themselves. This is accomplished by solving the system of linear equations for $\mathbf{w}$:

$$\begin{bmatrix} \delta_{[0]}^1 & \delta_{[1]}^1 & \cdots & \delta_{[D-1]}^1 \\ \delta_{[0]}^2 & \delta_{[1]}^2 & \cdots & \delta_{[D-1]}^2 \\ \vdots & \vdots & & \vdots \\ \delta_{[0]}^D & \delta_{[1]}^D & \cdots & \delta_{[D-1]}^D \end{bmatrix} \begin{bmatrix} w_{[0]} \\ w_{[1]} \\ \vdots \\ w_{[D-1]} \end{bmatrix} = \begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^D \end{bmatrix}. \tag{6}$$

Thus, a linear function over a convex polytope can be represented as a *piecewise-linear function* over simplices.[9]

Definition 2 (piecewise-linear functions over collections of convex polytopes)
A function $f : \mathbb{R}^D \to \mathbb{R}$ is piecewise-linear *over a collection $\mathcal{C}$ of convex polytopes if*

$$\forall \mathcal{R} \in \mathcal{C} \; \exists \mathbf{w}_{\mathcal{R}} \in \mathbb{R}^D : f(\delta) = \delta^{\mathsf{T}} \mathbf{w}_{\mathcal{R}} \; \forall \delta \in \mathcal{R}.$$

9. Equation (6) has a unique solution only if the determinant of the matrix in the equation is non-zero, that is, only if there are no collinearities in the vertices of $\mathcal{T}$.

Algorithm 4 Algorithm sketch for max

Identify the convex polytopes $\mathcal{R}_i$ where f_i is maximal. Each $\mathcal{R}_i$ is the intersection of $|\mathcal{A}_t|$ half-spaces.
Decompose each $\mathcal{R}_i$ into simplices
Evaluate $f_{\max}$ at each vertex in each resulting simplex
Recover the $\mathbf{w}$s as needed using Equation (6)

Algorithm 5 Algorithm sketch for sum

Identify the convex polytopes of the form $\mathcal{U} \cap \mathcal{V}$ over which the sum is linear
Decompose each of these polytopes into simplices
Evaluate f_{sum} at each vertex in each resulting simplex
Recover the $\mathbf{w}$s as needed using Equation (6).

Thus we can completely represent a piecewise-linear function as a collection of $(\mathcal{R}, \mathbf{w}_{\mathcal{R}})$ pairs.

Lemma 3 (Max of linear functions) *Given a set of functions $\{f_1, f_2, ..., f_N\}$ that are all linear over the same convex polytope $\mathcal{R}$, the function*

$$f_{\max}(\delta) = \max(f_1(\delta), f_2(\delta), ..., f_N(\delta))$$

is piecewise-linear over the collection of convex polytopes $\mathcal{C} = \{\mathcal{R}_1, \mathcal{R}_2, ..., \mathcal{R}_N\}$ given by

$$\mathcal{R}_i = \mathcal{R} \cap \bigcap_{j=1}^{N} \left\{ \delta \in \mathbb{R}^D : f_i(\delta) \geq f_j(\delta) \right\}, \; i = 1..N.$$

Note further that $\bigcup_{i=1}^{N} \mathcal{R}_i = \mathcal{R}$, and that the function $f_{\max}$ is convex (and therefore continuous) on $\mathcal{R}$, since each f_i is convex. These properties immediately suggest a strategy for computing a representation of $f_{\max}$ described in Algorithm 4. Figure 4 gives a pictorial representation of this strategy, which allows us to perform the max-over-actions portion of a value iteration backup for $D > 2$. We now address how to perform the weighted-sum-over-states portion of the backup.

Lemma 4 *[Sum of piecewise-linear functions] Given two functions, g_1 which is piecewise-linear over a collection $\mathcal{C}_1$ of polytopes, and g_2 which is piecewise-linear over a collection $\mathcal{C}_2$ of polytopes, their linear combination*

$$f_{\text{sum}}(\delta) = \alpha_1 g_1(\delta) + \alpha_2 g_2(\delta)$$

is piecewise-linear over the collection

$$\mathcal{C}_{1+2} = \{\mathcal{U} \cap \mathcal{V}\}, \text{ s.t. } \mathcal{U} \in \mathcal{C}_1, \; \mathcal{V} \in \mathcal{C}_2.$$

This property suggests a strategy for representing f_{sum} described in Algorithm 5. Figure 5 gives a pictorial representation of this strategy, which allows us to perform the weighted-sum-over-states portion of a value iteration backup for $D > 2$.

We can now use these strategies to construct a complete algorithm for the $D > 2$ case. At time T, we have

$$\begin{aligned} \hat{Q}_T(s_T, a_T, \delta) &= \phi_{s_T, a_T}^{\mathsf{T}} \left(\delta_{[0]} \cdot \hat{\beta}_{t[0]} + \delta_{[1]} \cdot \hat{\beta}_{t[1]} + ... + \delta_{[D-1]} \cdot \hat{\beta}_{T(D-1)} \right), \\ &= \delta^{\mathsf{T}} \mathbf{w}_T, \end{aligned}$$

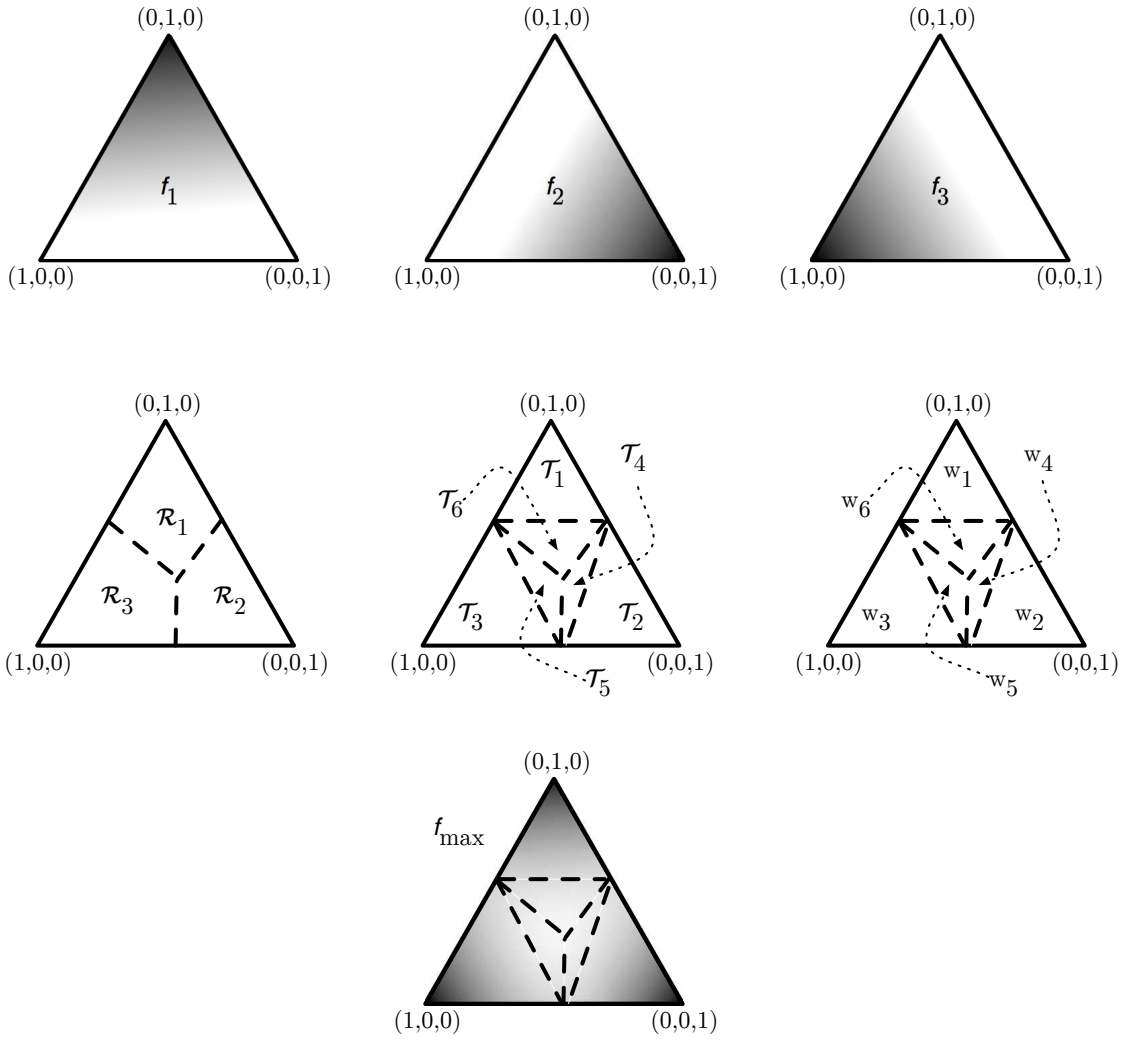

Figure 4: Pictorial representation of taking max of linear functions for $D=3$ basis rewards. The first row of triangles represents three linear functions f_1, f_2, and f_3; darker shading indicates higher function values. The second row shows the convex polytopes $\mathcal{R}_i$ over which f_i is maximal, the decomposition of each of these polytopes into simplices $\mathcal{T}_i$, and their corresponding weight vectors w_i. The continuous piecewise-linear function $f_{\max}$ is shown at the bottom.

where

$$\hat{\beta}_{T[d]} = (\Phi_T^\mathsf{T}\Phi_T)^{-1}\Phi_T^\mathsf{T}\mathbf{r}_{T[d]},$$
$$w_{T[d]} = \phi_{s_T,a_T}^\mathsf{T}\hat{\beta}_{T[d]}.$$

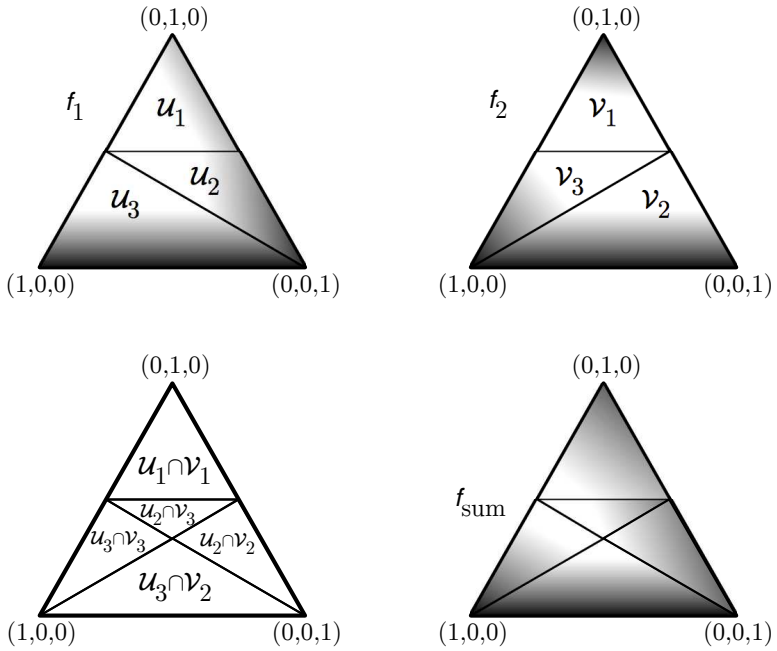

Figure 5: Pictorial representation of taking sum of piecewise-linear functions for $D = 3$ basis rewards. The top row shows the two functions to be added; darker shading indicates higher function values. In the second row, the left diagram shows the pieces over which the sum is linear. The right diagram of the second row shows the resulting continuous piecewise-linear function.

Thus for any s_T, a_T, the function $\hat{Q}_T(s_T, a_T, \cdot) = \phi_{s_T,a_T}^{\mathsf{T}} \hat{\beta}_T(\cdot)$ is linear over the piece $\mathcal{R} = \{\delta : \delta_{[d]} > 0\} \cap \{\delta : \sum_d \delta_{[d]} = 1\}$, which is the unit $(D-1)$-simplex. This is because each element of $\hat{\beta}_{T[d]}(\cdot)$ is linear in δ. It follows that $\hat{V}_T(s_T, \cdot) = \max_a \hat{Q}_T(s_T, a, \cdot)$ is piecewise-linear over the sets described in Lemma 3. To represent the stage T value functions $\hat{V}_T(s_T, \cdot)$, we apply Algorithm 4 to the Q-functions $\hat{Q}_T(s_T, a, \cdot)$ of each action a for each s_T in our data set. Given this value function at time T, we can compute $\hat{Q}_{T-1}(\cdot, \cdot, \hat{\beta}_{T-1}(\delta))$ by computing each element of $\hat{\beta}_{T-1}(\delta)$ as the weighted sum of $\hat{V}_T(s_T, \cdot)$ evaluated at the points s_T in our data set by repeated application of Algorithm 5. As in the $D = 2$ case, these weights are given by the columns of the matrix $(\Phi_T^{\mathsf{T}} \Phi_T)^{-1} \Phi_T^{\mathsf{T}}$. At this point, note that for any s_{T-1}, a_{T-1}, the function $\hat{Q}_{T-1}(s_{T-1}, a_{T-1}, \cdot)$ is piecewise-linear over the *same pieces*—they are the pieces identified in Lemma 4. Thus to compute $\hat{V}_{T-1}$ we can simply apply Algorithm 4 to each of these pieces. Backups to earlier timepoints proceed analogously.

4.3.1 COMPLEXITY

Note that the primitive operations required for fitted-Q iteration—pointwise max and pointwise weighted sum—are precisely the same as in the simpler settings discussed earlier, but the functions we are operating on are $(D-1)$-dimensional.

Suppose there are N trajectories and $|\mathcal{A}|$ actions at each time point. For any fixed s_T and a_T, the final learned Q-function $\hat{Q}_T(s_T, a_T, \cdot)$ has 1 piece $\mathcal{R}_1$ corresponding to the unit $(D-1)$-simplex. The terminal value function $\hat{V}_T(s_T^i, \cdot)$ is constructed at each of N points in state space by applying Algorithm 4 to the $\hat{Q}_T(s_T^i, a_T, \cdot)$ for each observed terminal state $s_T^1, s_T^2, ..., s_T^N$ in $\mathcal{D}$ and each action a_T. Each resulting $\hat{V}_T(s_T^i, \cdot)$ has at most $|\mathcal{A}|$ pieces $\mathcal{R}_1, ..., \mathcal{R}_{|\mathcal{A}|}$, supposing each action has a piece where it is optimal. To compute $\hat{Q}_{T-1}(s_{T-1}, a, \cdot)$, we use regression with targets constructed from the N value function estimates $\hat{V}_T(s_T^i, \cdot)$. In general, the pieces for each $\hat{V}_T(s_T^i, \cdot)$ may be unique. Thus each $\hat{Q}_{T-1}(s_{T-1}, a, \cdot)$ has pieces formed from all possible intersections between pieces of the N different $\hat{V}_T(s_T^i, \cdot)$, so there may be up to $|\mathcal{A}|^N$ such pieces. Applying Algorithm 4 again within each of these pieces means that each $\hat{V}_{T-1}(s_{T-1}^i, \cdot)$ may have $|\mathcal{A}|^{N+1}$ pieces. In general, $\hat{Q}_t(s, a, \cdot)$ may have up to $O(|\mathcal{A}|^{\sum_{i=1}^{T-t} N^i})$ pieces, and $\hat{V}_t(s, \cdot)$ may have up to $O(|\mathcal{A}|^{\sum_{i=0}^{T-t} N^i})$ pieces.

A more detailed complexity analysis would depend on how the pieces are represented, and on how Algorithms 4 and 5 are implemented using computational geometry primitives—we have already seen that for $D = 2$ basis rewards we can do much better than this worst-case bound. Intuitively this is because most of the intersections between pieces of the N different $\hat{V}_T(s_T^i, \cdot)$ are in fact empty. A general treatment of implementing Algorithms 4 and 5 is beyond the scope of this paper; however, we now present a detailed algorithm designed for the $D = 3$ case that is also much less computationally intensive than the above double-exponential bound suggests.

4.4 Optimal Value Functions for All Tradeoffs: Linear Function Approximation, $D = 3$ Basis Rewards

We now consider the $D = 3$ case specifically. The first "algorithmic trick" we will use is to represent functions of δ using two rather than three dimensions, that is,

$$r_t(s_t, a_t, \delta_{[0]}, \delta_{[1]}) = \delta_{[0]} r_{t[0]}(s_t, a_t) + \delta_{[1]} r_{t[1]}(s_t, a_t) + (1 - \delta_{[0]} - \delta_{[1]}) r_{t[2]}(s_t, a_t).$$

This follows from the constraint that $\sum_i \delta_{[i]} = 1$. Note that the set of points $(\delta_{[0]}, \delta_{[1]}) : \delta_{[0]} + \delta_{[1]} \leq 1,\ \delta_{[0]} \geq 0,\ \delta_{[1]} \geq 0$ is a convex polytope in $\mathbb{R}^2$. In fact it is a simplex, and therefore we can represent the linear function $Q_T(s_T, a_T, \cdot)$ by storing the corners of the simplex $\mathcal{T} = [(1,0)\ (0,1)\ (0,0)]$ together with the parameter vectors

$$\hat{\beta}_T(1,0) = (\Phi_T^\mathsf{T} \Phi_T)^{-1} \Phi_T^\mathsf{T} \mathbf{r}_{t[0]},$$
$$\hat{\beta}_T(0,1) = (\Phi_T^\mathsf{T} \Phi_T)^{-1} \Phi_T^\mathsf{T} \mathbf{r}_{t[1]},$$
$$\hat{\beta}_T(0,0) = (\Phi_T^\mathsf{T} \Phi_T)^{-1} \Phi_T^\mathsf{T} \mathbf{r}_{t[2]}.$$

We can compute a weight-vector representation of the function using Equation (6).

Consider two linear functions $\hat{Q}_T(s_T, 1, \cdot)$ and $\hat{Q}_T(s_T, 2, \cdot)$ over $\mathcal{T}$. To take their pointwise maximum, we must identify the pieces over which the maximum is linear, as described in Lemma 3. The boundary of these two pieces is a line in $\mathbb{R}^2$. If this line intersects $\mathcal{T}$, it will divide $\mathcal{T}$ into the two pieces. If it does not, then one function must be greater than the other over all of $\mathcal{T}$. Identifying the pieces can be accomplished by finding where (if anywhere) the dividing line given by $\hat{Q}_T(s_T, 1, \cdot) = \hat{Q}_T(s_T, 2, \cdot)$ intersects $\mathcal{T}$; this is illustrated in Figure 6. We represent $\hat{V}_T(s_T, \cdot)$ by recording the pieces $\mathcal{R}$ on either side of the dividing line. Each piece is identified by a set of vertices, along with the value of the max at each vertex. (Note that certain vertices will belong to both

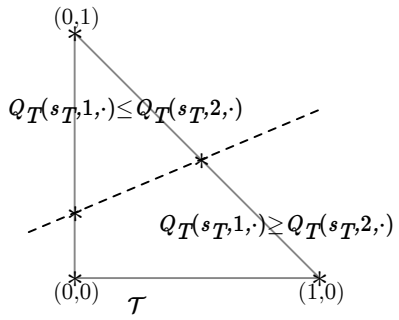

Figure 6: Identifying the pieces over which the max of two linear functions is linear.

pieces.) If there are more than 2 actions, we can take further maxes over each identified sub-piece, partitioning as necessary. This completes the max-over-actions step at time T.

To compute $\hat{Q}_{T-1}(s_{T-1}, a_{T-1}, \cdot)$, we compute each element of $\hat{\beta}_{T-1}(\cdot)$ at each vertex δ by taking a weighted sum over next states of $\hat{V}_T(s_T, \cdot)$, again with weights given by columns of $(\Phi_T^\intercal \Phi_T)^{-1}\Phi_T^\intercal$. From Lemma 4 we know that we need to identify all of the pieces formed by intersecting the linear pieces of the functions to be summed. Naïvely, one might compute the intersection of all pairs of pieces, but for $D=3$ basis rewards we can instead use a *constrained Delaunay triangulation* (CDT), which essentially gives us only the non-empty intersections and does so much more efficiently than enumerating all pairs. Figure 7 gives a schematic diagram of this procedure. The input to a standard Delaunay triangulation algorithm is a list of points in space. The output is a list of simplices (in this case triangles) that partition space and whose vertices come from the input points. The particular triangles chosen satisfy certain properties (de Berg et al., 2008), but the main appeal for our purposes is the algorithm's $O(n \log n)$ running time (Chew, 1987), where n is the number of points to be triangulated. A constrained version of the algorithm allows us to additionally specify edges between points that must be present in the output. The constrained version of the algorithm will add points as needed to satisfy this requirement; again Figure 7 illustrates this. The simplices (triangles) will form the pieces for the elements of $\hat{\beta}_{T-1}(\cdot)$, which will define our estimates $\hat{Q}_{T-1}$.

The output of the CDT algorithm is a set of pieces over which we know the sum of the piecewise-linear functions will be linear. There are in fact more pieces than are strictly necessary, because linear pieces that have more than three vertices (e.g., quadrilaterals) are divided up by the algorithm. Nonetheless, the output is convenient because we can determine the weight vector $\mathbf{w}$ for any simplex using Equation (6). Once we have determined these pieces and vertices, we evaluate $\hat{V}_T(s_T, \cdot)$ at each terminal state and each vertex. Each element of $\hat{\beta}_{T-1}(\cdot)$ is a piecewise-linear function whose pieces are given by the CDT algorithm, and whose values are given by the appropriate weighted sum of $\hat{V}_T(s_T, \cdot)$ evaluated at the vertices. This gives $\hat{Q}_{T-1}$. The max operation to obtain $\hat{V}_{T-1}$ can again be achieved by taking the max over each piece of $\hat{Q}_{T-1}$, and so on backward to $t = 1$. A complete description is given in Algorithm 6

The problem of finding intersections between lines and computing triangulations is well-studied in the field of computational geometry (de Berg et al., 2008). Though these problems may appear trivial, it is very important to avoid situations where there is "ill-conditioning." For example, if we were to use floating point arithmetic to define three lines that should intersect at the same point,

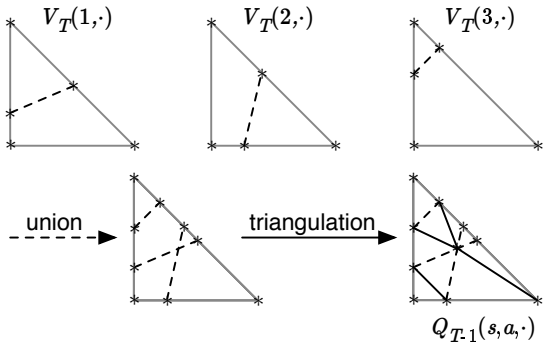

Figure 7: Computing the sum of three piecewise-linear functions. The three example value functions each have two linear pieces. The boundary between two pieces is shown by a dotted line. We take all of the vertices, plus the boundaries, and give them as input to a constrained Delaunay triangulation procedure. The output is shown.

we may find that the "intersection point" is different depending on which pair of lines is used to compute it. This can lead to many spurious points and edges being generated as we proceed with value iteration. We take advantage of CGAL, the Computational Geometry Algorithms Library (CGAL, 2011), which is designed specifically to avoid these problems.

Algorithm 6 Value Backup - Linear Function Approximation, $D=3$ Basis Rewards

$\forall(s,\delta)$, $\hat{V}_{T+1}(s,\delta) \triangleq 0$. $\forall s$, $\Delta(\hat{V}_{T+1}(s,\cdot)) \triangleq \{[(1,0),(0,1),(0,0)]\}$.
for $t = T$ **downto** 1 **do**
  $\Delta^{\hat{Q}_t} \leftarrow \{\}$
  for all $(s_t, a_t, s_{t+1}) \in \mathcal{D}$ **do**
    $\Delta^{\hat{Q}_t} \leftarrow \Delta^{\hat{Q}_t} \cup \Delta(\hat{V}_{t+1}(s_{t+1},\cdot))$
  end for
  $\Delta^{\hat{Q}_t} \leftarrow$ constrained_Delaunay_Triangulation$(\Delta^{\hat{Q}_t})$
  for all $\delta \in$ vertices$(\Delta^{\hat{Q}_t})$ **do**
    $\mathbf{y}_t(\delta) = \delta_{[0]}\mathbf{r}_{t[0]} + \delta_{[1]}\mathbf{r}_{t[1]} + (1-\delta_{[0]}-\delta_{[1]})\mathbf{r}_{t[2]} + \hat{\mathbf{v}}_{t+1}(\delta)$
    $\hat{\beta}_t(\delta) = (\Phi_t^{\mathsf{T}}\Phi_t)^{-1}\Phi_t^{\mathsf{T}}\mathbf{y}_t(\delta)$
  end for
  for all $s_t \in \mathcal{D}$ **do**
    Compute $\Delta(\hat{V}_t(s_t,\cdot))$ by Algorithm 4
  end for
end for

4.4.1 COMPLEXITY FOR $D = 3$

Any triangulation of n points in the plane contains $O(n)$ triangles (Brass, 2005), so the operation $\Delta^{\hat{Q}_t} \leftarrow$ constrained_Delaunay_Triangulation$(\Delta^{\hat{Q}_t})$ increases the size of $\Delta^{\hat{Q}_t}$ only linearly. It follows

that each $\hat{Q}_{T-1}(s_{T-1}, a, \cdot)$ has $O(N \cdot |\mathcal{A}|)$ pieces rather than the $|\mathcal{A}|^N$ given by the worst case analysis in Section 4.3.1. Therefore $\hat{Q}_t(s, a, \cdot)$ may have up to $O(N^{T-t}|\mathcal{A}|^{T-t})$ pieces, and $\hat{V}_t(s_t, \cdot)$ may have up to $O(N^{T-t}|\mathcal{A}|^{(T-t)+1})$ pieces. Note that these rates are the same as for the $D = 2$ special case discussed in Section 4.2.5. Intuitively this is because the triangulation of n points in d-dimensional space has $O(n^{\lceil d/2 \rceil})$, triangles (Brass, 2005), that is, the same asymptotic growth rate for $D = 2$ (one-dimensional preference space) and $D = 3$ (two-dimensional preference space).

5. Dominated Actions

The ability to compute $\hat{Q}$ and $\hat{V}$ for all preferences achieves our goal of informing the decision maker about the quality of available actions under different preferences, and of informing the decision maker about how the recommended policy changes with preference. In addition to achieving these primary goals, our representations of $\hat{Q}$ and $\hat{V}$ allow us to compute whether or not an action is *dominated* (not optimal for a given state no matter what the preference) and whether it is *globally dominated* (not optimal for *any* state-preference pair.) The notion of domination arises in POMDP planning as well, where certain actions may not be optimal for any belief state, but the notion of global domination has no direct analog in the POMDP setting since it is a property of additional observed state s that is not part of a typical POMDP.

The concept of domination is central to the field of multi-criterion optimization (Ehrgott, 2005), and is important in the medical decision making setting because it identifies treatment actions that are not appropriate no matter what the decision maker's preference is. One may also consider actions that are dominated for all patient states in a population of interest, that is, the actions that are globally dominated. Knowing this set of actions would be useful for developing a *formulary*—a list of treatments that should generally be made available to a patient population.

The general problem of analytically identifying the set of globally dominated actions is difficult, as we will illustrate, but we first provide solutions for low-dimensional problems and discuss the identification of globally dominated actions in higher dimensional settings. Our approach for determining if actions are dominated is to look for *certificates* of **non**-domination for each action. A point (s_t, a_t, δ) where $\hat{Q}_t(s_t, a_t, \delta) = \hat{V}_t(s_t, \delta)$ is a certificate that action a_t is not dominated at state s_t, and that action a_t is therefore not globally dominated.[10] All proofs are deferred to Appendix A.

5.1 Finite Case, $D = 2$ Basis Rewards

We showed in Section 4.1 how to exactly represent the $Q_t(s_t, a_t, \cdot)$ and $V_t(s_t, \cdot)$ functions for all s_t and a_t for $D = 2$ when the state space is finite by representing them as lists of knots (vertices) and knot-values (vertex-values). In addition to storing this information, we may also store the set of actions that are optimal at each knot, that is, we may store $\mathcal{A}^*(\delta) = \{a_t^* : Q_t(s_t, a_t^*, \delta) = V_t(s_t, \delta)\}$ for each δ in the knot list of $V_t(s_t, \cdot)$. Note that $\mathcal{A}^*(\delta)$ may contain more than one action. Suppose δ_k and δ_{k+1} are adjacent knots in $\Delta(V_t(s_t, \cdot))$. For all δ s.t. $\delta_k < \delta < \delta_{k+1}$, we have $\mathcal{A}^*(\delta) = \mathcal{A}^*(\delta_k) \cap \mathcal{A}^*(\delta_{k+1})$. Thus the set of actions that have a non-domination certificate at state s_t is given by

$$\bigcup_{k=1}^{|\Delta(V_t(s_t,\cdot))|} \mathcal{A}^*(\delta_k),$$

10. Note that in this work we determine which actions are *estimated* to be dominated, since we are using estimates $\hat{Q}_t$ and $\hat{V}_t$ to make this determination. Assessing our confidence that an action is truly non-dominated based on available data will require incorporation of appropriate statistical methods (Laber et al., 2009).

and any actions not in the above union are dominated at s_t. Note that recording this additional information does not increase the time complexity of the method. It also allows us to find every globally dominated action by computing the above union at each finite state, taking the union of those sets, and identifying actions not present in the union.

5.2 Regression Case, $D=2$ Basis Rewards, One State Feature

We now show how to identify all of the globally dominated actions in the linear function approximation setting. We first discuss the case with a single state feature ψ_{s_T}, $D=2$ basis rewards, and the last timepoint T. We also construct feature vectors ϕ_{s_T,a_T} so that the $\hat{Q}_t$ functions are built using separate regressions for each action; for example see (5). We can then define $\hat{\beta}_t^a(\delta)$ to be the 2×1 sub-vector of $\hat{\beta}_t(\delta)$ that aligns with the sub-vector of ϕ_{s_t,a_t} that is non-zero for $a_t = a$. We also define the matrix $B_T^a = \left[\begin{array}{cc} \hat{\beta}_T^a(0) & \hat{\beta}_T^a(1) \end{array}\right]$ for each action, so that

$$\hat{Q}_T(s_T, a_T, \delta) = \left[\begin{array}{cc} 1 & \psi_{s_T} \end{array}\right] B_T^a \left[\begin{array}{c} \delta \\ 1-\delta \end{array}\right]. \tag{7}$$

To find the globally dominated actions, we will search for certificates of non-domination in (ψ_{s_T}, δ) space and identify the actions that do not have a certificate. Figure 3 shows an example of this setting. In the example, actions 1, 4, 6, 7, 8, 9, and 10 have regions where they are optimal, and hence certificates of non-domination. Actions 2, 3, and 5 have no certificates, that is, they are not optimal for any combination of ψ_{s_T} and δ.

Each B_T^a is a constant given the data and the regression algorithm. The form of (7) clearly shows that $\hat{Q}_T(\cdot, a, \cdot)$ is a bilinear function of ψ_{s_T} and δ. To analytically identify the set of dominated actions, we analyze the boundaries between the regions where one action has higher value than another. These boundaries occur where $Q_T(\cdot, a_1, \cdot) = Q_T(\cdot, a_2, \cdot)$ for some actions a_1 and a_2, that is, where

$$\left[\begin{array}{cc} 1 & \psi_{s_T} \end{array}\right] B_T^{a_1} \left[\begin{array}{c} \delta \\ 1-\delta \end{array}\right] = \left[\begin{array}{cc} 1 & \psi_{s_T} \end{array}\right] B_T^{a_2} \left[\begin{array}{c} \delta \\ 1-\delta \end{array}\right],$$

which describes the hyperbola in δ and ψ_{s_T} given by

$$\left[\begin{array}{cc} 1 & \psi_{s_T} \end{array}\right] (B_T^{a_1} - B_T^{a_2}) \left[\begin{array}{c} \delta \\ 1-\delta \end{array}\right] = 0. \tag{8}$$

Along these boundaries, "triple-points" occur at (ψ_{s_T}, δ) points where three or more actions have exactly the same value. At these points, either all of the actions involved are optimal, or none of them are. We now show that if there exists a certificate of non-domination for action a, but there exists no certificate for a on the boundary of the domain of $V_T(s_T, \delta)$, then there exists a certificate for a at a triple-point.

Lemma 5 (Lizotte et al., 2010) *If action a is optimal at time T for some point (ψ_{s_T}, δ) but is not optimal for any (ψ_{s_T}, δ) on the boundary of the domain, then a is optimal for some (ψ_{s_T}, δ) that is a triple-point.*

From Lemma 5 we know that to find all actions that are optimal for some (ψ_{s_T}, δ) we need only check the boundaries and the triple points. The boundaries can be checked using Algorithm 2.

(Note that because $\hat{Q}_T(\cdot, a, \cdot)$ is bilinear in δ and in ψ_{s_T}, we can also use Algorithm 2 to identify for any fixed δ the actions that are optimal for some ψ_{s_T}.) We can then enumerate the $\binom{|\mathcal{A}|}{3}$ triple-points and check them to detect any regions that do not intersect the boundary of the domain, like for example the region where action 1 is optimal in Figure 3 where we have identified the triple-points with white dots. This procedure reveals all actions that are optimal for some (ψ_{s_T}, δ), and thereby identifies any actions that are not optimal for any (ψ_{s_T}, δ).

To compute the triple points, we must solve the following system of bilinear equations for ψ_{s_T} and δ:

$$\begin{bmatrix} 1 & \psi_{s_T} \end{bmatrix} (B_T^{a_1} - B_T^{a_2}) \begin{bmatrix} \delta \\ 1-\delta \end{bmatrix} = 0,$$
$$\begin{bmatrix} 1 & \psi_{s_T} \end{bmatrix} (B_T^{a_1} - B_T^{a_3}) \begin{bmatrix} \delta \\ 1-\delta \end{bmatrix} = 0.$$

There are many ways of interpreting this system of equations; it describes the intersection of two hyperbolas, as pointed out in our earlier work (Lizotte et al., 2010). We describe a more concise solution here. Note that any solution (ψ_{s_T}, δ) must have the property that the vector $[1\ \psi_{s_T}]$ is orthogonal to the two vectors given by $(B_T^{a_1} - B_T^{a_2}) \begin{bmatrix} \delta \\ 1-\delta \end{bmatrix}$ and $(B_T^{a_1} - B_T^{a_3}) \begin{bmatrix} \delta \\ 1-\delta \end{bmatrix}$. Since $[1\ \psi_{s_T}]$ is two-dimensional, this implies that these two vectors are collinear. Therefore the vector $\begin{bmatrix} \delta \\ 1-\delta \end{bmatrix}$ must satisfy

$$(B_T^{a_1} - B_T^{a_2}) \begin{bmatrix} \delta \\ 1-\delta \end{bmatrix} = \lambda (B_T^{a_1} - B_T^{a_3}) \begin{bmatrix} \delta \\ 1-\delta \end{bmatrix}. \tag{9}$$

Equation (9) describes the *generalized eigenvalue problem* (Golub and Van Loan, 1996). Common software packages can solve for λ and δ.[11] We have described the process of identifying globally dominated actions for bilinear functions; we can immediately extend this algorithm for piecewise bilinear functions by applying it between pairs of knots.

5.3 Regression Case, p State Features, Arbitrary D

Lizotte et al. (2010) conjectured that an analogue of Lemma 5 holds in higher dimensions, and that identifying all globally dominated actions for more state variables and/or reward functions would require computing intersections of surfaces in higher dimensions. We refine this conjecture, and we propose a solution method for finding globally dominated actions for

$$\hat{Q}_T(\psi_{s_T}, a, \delta) = \begin{bmatrix} 1 & \psi_{s_T}^{\mathsf{T}} \end{bmatrix} B_T^a \, \delta.$$

where ψ_{s_T} is p-dimensional, and δ is the D-dimensional vector of preferences defined as usual. We consider finding non-dominated actions over a "domain of interest," of the form $\mathcal{S} \times \mathcal{D}$, where $\mathcal{S}$ is a rectangle in $\mathbb{R}^p$, and $\mathcal{D}$ is a convex subset of valid preferences. To prove our method is correct, we require the following conjecture.

11. In practice one solves $(B_T^{a_1} - B_T^{a_2})x = \lambda(B_T^{a_1} - B_T^{a_3})x$ and then projects x onto the subspace $x_{[1]} = 1 - x_{[0]}$ by dividing it by $x_{[0]} + x_{[1]}$.

Conjecture 6 *If the system of polynomial equations of the form*

$$\begin{aligned}
\begin{bmatrix} 1 & \psi_{s_T}^{\mathsf{T}} \end{bmatrix} (B_T^{a_1} - B_T^{a_2})\,\delta = 0 \\
\begin{bmatrix} 1 & \psi_{s_T}^{\mathsf{T}} \end{bmatrix} (B_T^{a_1} - B_T^{a_3})\,\delta = 0 \\
\vdots \\
\begin{bmatrix} 1 & \psi_{s_T}^{\mathsf{T}} \end{bmatrix} (B_T^{a_1} - B_T^{a_k})\,\delta = 0
\end{aligned} \tag{10}$$

has a finite number of solution points, those points taken together are a continuous vector-valued function of the coefficients of the system.

Conjecture 6 is true for a single polynomial of one variable over the complex plain (Uherka and Sergott, 1977), and holds for the $D=2$ case. We believe the conjecture holds because it is known that systems of multivariate polynomials can be reduced to solving a collection of independent problems each involving a single polynomial of one variable. This reduction uses the methods of *elimination* and *extension* (Cox et al., 1997).

Proposition 7 *Assume Conjecture 6. If there exists a point* (ψ_{s_T}, δ) *in the interior of the domain of interest where action a is optimal, but a is not optimal at any point* (ψ_{s_T}, δ) *where* $p+D+1$ *actions are simultaneously optimal, then there exists a point* (ψ_{s_T}, δ) *on the boundary of the domain of interest where action a is optimal.*

We refer to a point where k actions are simultaneously optimal as a "k-tuple point." To find the set of globally non-dominated actions, we first solve (10) and check to see if any of the $(p+D+1)$-tuple points are optimal. If so, all of the point's associated actions not globally dominated. The system (10) of polynomial equations can be solved by computer algebra systems or using numerical approximation techniques (Cox et al., 1997; Sturmfels, 2002). By recursively applying the proposition to the boundaries of the original domain, we can ensure that we identify every action that is not globally dominated: first, we find $(D+p+1)$-tuple points inside the original $(D+p)$-dimensional domain of interest and check whether any of these are optimal. We then treat each of the $(D+p-1)$-dimensional boundaries as our new domains of interest, and look for $(D+p)$-tuple points in each of these, and so on until we check each of the 2^{D+p} zero-dimensional points at the corners of our original domain. Again, we have described the process of identifying globally dominated actions for functions linear in δ; we can immediately extend this algorithm to piecewise-linear functions by applying it within linear regions.

6. Application to Medical Decision Making

An important application of this work is the improvement of the use of sequential medical data for constructing clinical decision support systems. In this section, we briefly discuss how such systems are currently constructed, how preferences are currently addressed in the medical decision making community, and how the methods presented in this paper provide a novel and useful way of incorporating preferences in clinical decision support systems. We then present an example using real data that illustrates how our methods can be used to inform clinical decision making.

6.1 Clinical Decision Support, Evidence-Based Medicine, and Preferences

Currently, most clinical decision support systems are constructed using expert opinion (e.g., Working Group for the Canadian Psychiatric Association and the Canadian Alliance for Research on Schizophrenia, 1998; Miller et al., 1999). Although accumulated clinical experience is invaluable in making good treatment decisions, data-derived scientific evidence is playing an increasingly prominent role. Sackett (1996) state that "The practice of evidence based medicine means integrating individual clinical expertise with the best available external clinical evidence from systematic research."

In order to be effective, any evidence-based decision support system must leave room for individual clinical expertise to inform the final decision. The methods we have presented are able to do this by presenting treatment recommendations in a way that incorporates decision maker preferences. There is extensive literature on "preference elicitation," both within and outside the field of medical decision making. In the medical decision making field, however, preference elicitation is usually done at the population level and used to produce generic clinical guidelines, rather than to make recommendations tailored to individual patients (e.g., Bonnichsen, 2011; Davis et al., 2011). In other fields, preference elicitation is done before presenting any information about the available treatments (Boutilier, 2002; Thall, 2008). It is assumed that preference elicitation is able to reliably extract the preferences of the decision maker; in our setting, preference elicitation would attempt to find the δ that represents the preference of a decision maker, run fitted-Q iteration using $r_t(s_t, a_t, \delta)$, and recommend a single treatment. This approach leaves no room for individual clinical expertise.

Our methods provide a novel alternative to preference elicitation. Rather than trying to determine which of the uncountable number of possible preferences a user might have, we present, for each available action, the set of preferences for which that action is optimal. That is, we present the policy as a function of preference. We call this approach "inverse preference elicitation" because rather than eliciting a preference and recommending a treatment, we can easily and intuitively show for each treatment the set of preferences that are consistent with its recommendation. By using this approach, the time a user would have spent having his or her preference elicited is now spent directly considering the evidence for how preferences influence recommended treatment.

6.2 Example: CATIE Study

We illustrate inverse preference elicitation using data from the Clinical Antipsychotic Trials of Intervention Effectiveness (CATIE) study. The CATIE study was designed to compare sequences of antipsychotic drug treatments for the care of schizophrenia patients. The full study design is quite complex (Stroup et al., 2003; Swartz et al., 2003); we have therefore chosen a simplified subset of the CATIE data in order to more clearly illustrate the potential of the methods presented in this paper. CATIE was an 18-month study that was divided into two main phases of treatment. Upon entry into the study, most patients began "Phase 1," in which they were randomized to one of five possible treatments with equal probability: olanzapine, risperidone, quetiapine, ziprasidone, or perphenazine. As they progressed through the study, patients were given the opportunity at each monthly visit to discontinue their Phase 1 treatment and begin "Phase 2" on a new treatment. The set of possible Phase 2 treatments depended on the reason for discontinuing Phase 1 treatment. If the Phase 1 treatment was deemed to be ineffective at reducing symptoms, then their Phase 2 treatment was chosen randomly as follows: clozapine with probability $1/2$, or uniformly randomly from the set {olanzapine, risperidone, quetiapine} with probability $1/2$. If the Phase 1 treatment was deemed

to produce unacceptable side-effects, their Phase 2 treatment was chosen uniformly randomly from the set {olanzapine, risperidone, quetiapine, ziprasidone}.

In previous work, we used batch off-policy reinforcement learning to analyze data from this study using a single reward function (Shortreed et al., 2010). We now give two new analyses using the new methods we have presented to examine multiple rewards simultaneously. The basis rewards we consider are measures of symptoms, side-effects, and quality of life.

Symptoms: PANSS For our symptom measurement, we use the Positive and Negative Syndrome Scale (PANSS) which is a numerical representation of the level of psychotic symptoms experienced by a patient (Kay et al., 1987). A higher value of PANSS reflects the presence of more severe symptoms. PANSS is a well-established measure that we have used in previous work on the CATIE study (Shortreed et al., 2010), and is measured for each CATIE patient both at the beginning of the study and at several times over the course of the study. Since having larger PANSS is worse, for our first basis reward $r_{[0]}$ we use 100 minus the percentile of a patient's PANSS at the end of their time in the study. We use the distribution of PANSS at the beginning of the study as the reference distribution for the percentile.

Body Weight: BMI Weight gain is an important and problematic side-effect of many antipsychotic drugs (Allison et al., 1999). Patients in the CATIE study had their Body Mass Index (BMI) (National Institutes of Health., 1998) measured at study intake and several times over the course of the study. Since in this population having a larger BMI is worse, for our second basis reward $r_{[1]}$ we use 100 minus the percentile of a patient's BMI at the end of their time in the study. We use the distribution of BMI at the beginning of the study as the reference distribution for the percentile.

Quality of Life: HQLS Measures of quality of life are intended to assess to what degree a patient's disease is impacting his or her daily life, in terms of a patient's relationships with others, ability to work, emotional state, and ability to carry out daily activities (Cramer et al., 2000). Patients in CATIE were administered the Heinrichs-Carpenter Quality of Life (HQLS) (Heinrichs et al., 1984) scale at intake and repeatedly as they progressed through the study. Since having a higher HQLS is better, for our third basis reward $r_{[2]}$ we use the percentile of a patient's HQLS at the end of their time in the study. We use the distribution of HQLS at the beginning of the study as the reference distribution for the percentile.

6.3 Symptoms versus Weight Gain

We begin by presenting the output of our algorithm for $D = 2$, using PANSS as described above for $r_{[0]}$, and BMI for $r_{[1]}$. In Figures 8, 9 and 10 we will present plots of the piecewise linear value function $\hat{V}_t(s_t, \cdot)$ for $t = 1, 2$ and for various representative values of s_t. When we plot $\hat{V}_t(s_t, \delta)$ as a function of δ, we simultaneously show the learned optimal action using the style and colour of the plotted line. Thus from our plots one can see both the learned value and the learned policy as a function of δ, which enables us to easily see for each action the range of preferences for which that action looks best.

6.3.1 PHASE 2 ANALYSES

Following the approach of our previous work (Shortreed et al., 2010), we use PANSS at entry to Phase 2 as a continuous state variable s_2 so that we can allow symptom severity to influence optimal action choice. We convert the PANSS scores at entry to Phase 2 into percentiles just as we did for the PANSS reward signal. Furthermore, we learn value functions for the Phase 2 Efficacy patients and

the Phase 2 Tolerability patients separately, since these two groups have different sets of possible actions.

We have relatively little data for Phase 2 Efficacy subgroup of patients. Therefore for this subgroup, we combine the actions of giving {olanzapine, risperidone, or quetiapine} into one "not-clozapine" action: $\mathcal{A}_2^{\text{EFF}} = \{\text{CLOZ}, \text{not-CLOZ}\}$. The other three drugs are much more similar to each other than they are to clozapine, which is much more toxic and is currently considered a "last resort" for use when symptoms are not effectively managed by other treatments (McDonagh et al., 2010). The feature vectors we use for Stage 2 Efficacy patients are given by

$$\phi_{s_2,a_2}^{\text{EFF}} = [1,\ 1_{a_2=\text{CLOZ}},\ s_2,\ s_2 \cdot 1_{a_2=\text{CLOZ}},\ 1_{\text{TD}},\ 1_{\text{EX}},\ 1_{\text{ST1}},\ 1_{\text{ST2}},\ 1_{\text{ST3}},\ 1_{\text{ST4}}]^{\mathsf{T}}.$$

Here, s_2 is the PANSS percentile at entry to Phase 2. Feature $1_{a_2=\text{OLAN}}$ is an indicator that the action at the second stage was clozapine, as opposed to one of the other treatments. We also have other features that do not influence the optimal action choice but that are chosen by experts to improve the value estimates.[12] 1_{TD} is an indicator variable of whether the patient has had tardive dyskinesia (a motor-control side-effect), 1_{EX} indicates whether the patient has been recently hospitalized, and 1_{ST1} through 1_{ST4} indicate the type of facility at which the patient is being treated (e.g., hospital, specialist clinic)

For Phase 2 Tolerability patients, the possible actions are $\mathcal{A}_2^{\text{TOL}} = \{\text{OLAN}, \text{QUET}, \text{RISP}, \text{ZIP}\}$, and the feature vectors we use are given by

$$\begin{aligned}\phi_{s_2,a_2}^{\text{TOL}} = [&1,\ 1_{a_2=\text{OLAN}},\ 1_{a_2=\text{QUET}},\ 1_{a_2=\text{RISP}},\ s_2,\ s_2 1_{a_2=\text{OLAN}},\ s_2 1_{a_2=\text{QUET}},\ s_2 1_{a_2=\text{RISP}}, \ldots \\ &1_{\text{TD}},\ 1_{\text{EX}},\ 1_{\text{ST1}},\ 1_{\text{ST2}},\ 1_{\text{ST3}},\ 1_{\text{ST4}}]^{\mathsf{T}}.\end{aligned}$$

Here we have three indicator features for different treatments at Phase 2, $1_{a_2=\text{OLAN}}$, $1_{a_2=\text{RISP}}$, $1_{a_2=\text{QUET}}$, with ziprasidone represented by turing all of these indicators off. Again we include the product of each of these indicators with the PANSS percentile s_2. The remainder of the features are the same as for the Phase 2 Efficacy patients.

Figure 8 shows a plot of the piecewise linear value function $\hat{V}_2(s_2, \cdot)$ for patients who are in Phase 2 of the study because of a lack of efficacy of the Phase 1 treatment. We plot $\hat{V}_2(s_2, \cdot)$ for three fixed values of s_2 corresponding to having low PANSS, moderate PANSS, or high PANSS at entry to Phase 2. (These correspond to setting $s_2 = 25$, $s_2 = 50$ and $s_2 = 75$, respectively.) For all three states shown in the plot, the learned policy indicates that clozapine is the best action for a reward based only on PANSS (i.e., for $\delta = 0$), but not-clozapine (olanzapine or risperidone or quetiapine) is best for a reward based only on BMI (i.e., for $\delta = 1$.) We have indicated the values of δ at which the decision changes from one action to the other by dropping down a dotted line. We see that, except for those with a strong preference for controlling BMI, clozapine appears to be the best choice among patients who found their Phase 1 treatment to be ineffective at controlling symptoms. It is clear from the plot that neither action is globally dominated since neither is dominated at any of our example states.

Figure 9 shows a plot of the piecewise linear value function $\hat{V}_2(s_2, \cdot)$ for patients who are in phase 2 of the study because they could not tolerate the side-effects of their Phase 1 treatment. Again we plot $\hat{V}_2(s_2, \cdot)$ for three different Phase 2 entry percentiles of PANSS: $s_2 = 25$, $s_2 = 50$ and

12. See Section 4.2 by Shortreed et al. (2010) for a more thorough discussion of these kinds of features. When we display value functions and learned policies in our examples, we set all of these indicators to 0 since they are not needed by the learned policy to select actions in the future.

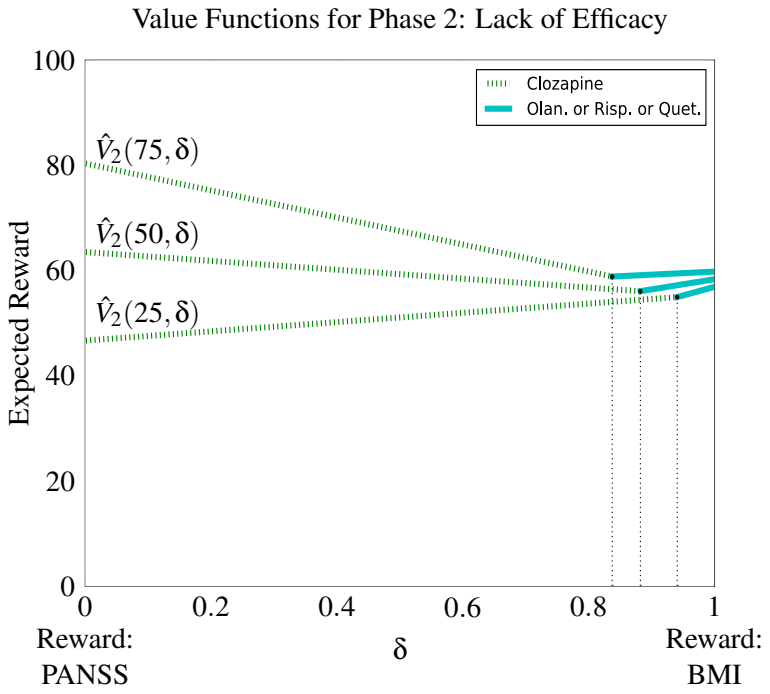

Figure 8: Multiple rewards analysis showing learned value function and associated learned policy for Phase 2 Efficacy patients. Three value functions are shown, with the associated action chosen by the learned policy, for $s_2 = 25$, $s_2 = 50$, and $s_2 = 75$.

$s_2 = 75$. Possible treatments are olanzapine, quetiapine, risperidone and ziprasidone. If we use a reward based only on PANSS (i.e., for $\delta = 0$), the learned policy indicates that olanzapine is the best action for those with high or moderate incoming PANSS, and that risperidone is best for those with lower incoming PANSS. Ziprasidone is best for a reward based only on BMI (i.e., for $\delta = 1$) independent of PANSS level. This result agrees with existing research on weight gain associated with these atypical antipsychotics (Allison et al., 1999). Again, we have indicated the values of δ at which the decision changes from one action to another by dropping down a dotted line. In this analysis, we found that quetiapine was globally dominated.

6.3.2 PHASE 1 ANALYSIS

For Phase 1 patients, the possible actions are $\mathcal{A}_1 = \{\text{OLAN}, \text{PERP}, \text{QUET}, \text{RISP}, \text{ZIP}\}$, and the feature vectors we use are given by

$$\begin{aligned}\phi^{\text{TOL}}_{s_1,a_1} = [&1,\ 1_{a_1=\text{OLAN}},\ 1_{a_1=\text{PERP}},\ 1_{a_1=\text{QUET}},\ 1_{a_1=\text{RISP}}, \cdots \\ &s_1,\ s_1 1_{a_1=\text{OLAN}},\ s_1 1_{a_1=\text{PERP}},\ s_1 1_{a_1=\text{QUET}},\ s_1 1_{a_1=\text{RISP}}, \cdots \\ &1_{\text{TD}},\ 1_{\text{EX}},\ 1_{\text{ST1}},\ 1_{\text{ST2}},\ 1_{\text{ST3}},\ 1_{\text{ST4}}]^{\mathsf{T}}.\end{aligned}$$

We have four indicator features for different treatments at Phase 2, $1_{a_1=\text{OLAN}}$, $1_{a_1=\text{PERP}}$, $1_{a_1=\text{QUET}}$, and $1_{a_1=\text{RISP}}$, with ziprasidone represented by turing all of these indicators off. We include the

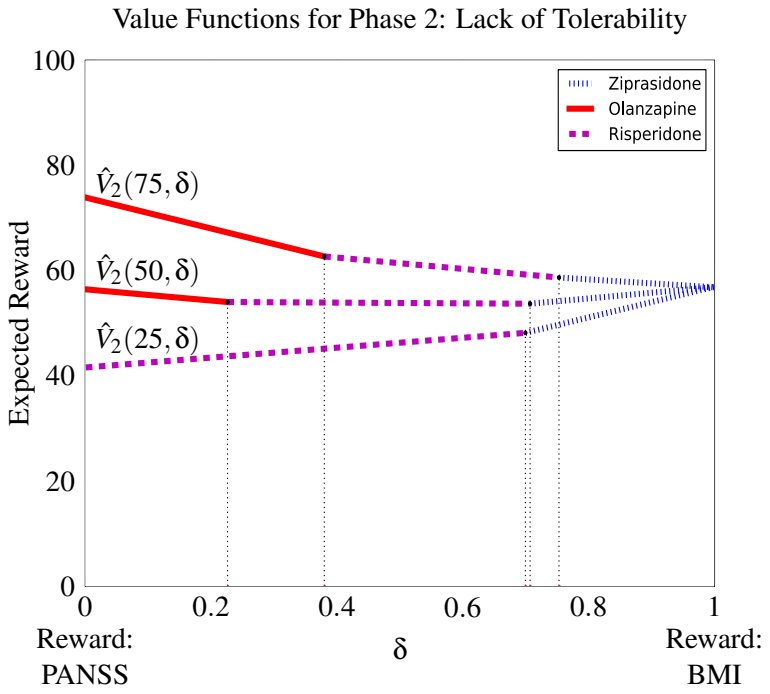

Figure 9: Multiple rewards analysis showing learned value function and associated learned policy for Phase 2 Tolerability patients. Three value functions are shown, with the associated action chosen by the learned policy, for $s_2 = 25$, $s_2 = 50$, and $s_2 = 75$.

product of each of these indicators with the PANSS percentile s_1 at entry to the study, and the remainder of the features are the same as for the Phase 2 feature vectors. (These are collected before the study begins and are therefore available at Phase 1 as well.)

Figure 10 shows a plot of the piecewise linear value function $\hat{V}_1(s_1,\cdot)$ for patients entering Phase 1 (the beginning) of the study. Again we plot $\hat{V}_1(s_1,\cdot)$ for three fixed values of $s_1 = 25$, $s_1 = 50$ and $s_1 = 75$. Possible treatments are perphenazine, olanzapine, quetiapine, risperidone and ziprasidone. For all three states shown in the plot, the learned policy indicates that olanzapine is the best action for a reward based only on PANSS (i.e., for $\delta = 0$). Ziprasidone is best for a reward based only on BMI (i.e., for $\delta = 1$), also independent of PANSS level. Again, the result agrees well with existing research (Allison et al., 1999). In this analysis, we found perphenazine and quetiapine to be globally dominated.

6.4 Symptoms vs. Weight Gain vs. Quality of Life

We now present the output of our algorithm for $D = 3$, using PANSS for $r_{[0]}$, BMI for $r_{[1]}$, and HQLS for $r_{[2]}$. We use the methods described in Section 4.4 to compute the value functions which map a state s_t and a three-element preference vector δ to an estimated value. Rather than display the shape of this value function using a surface or contour plot, we have elected to show only the regions of preference space where each action is optimal (i.e., the learned policy) mapped onto an equilateral triangle. This simplifies the presentation, but still allows us to easily see for each action

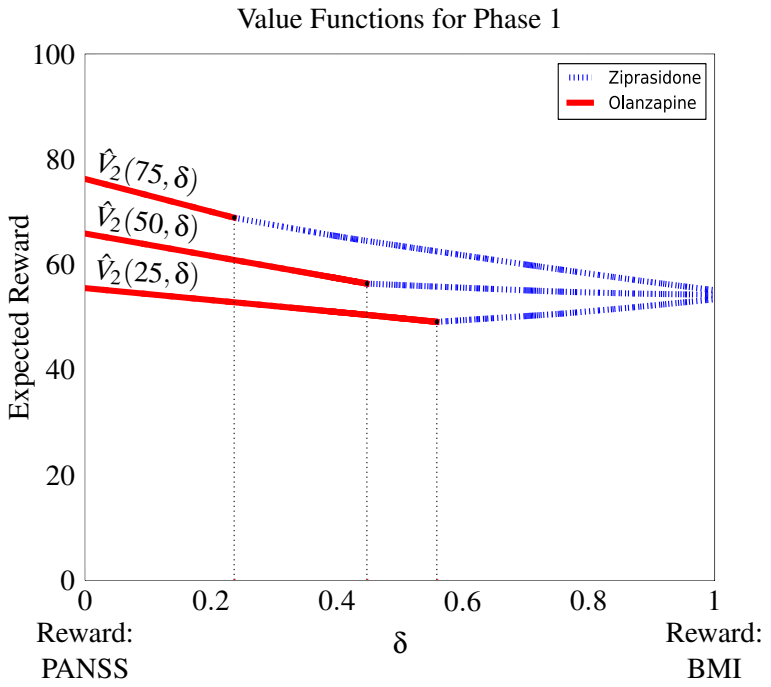

Figure 10: Multiple rewards analysis showing learned value function and associated learned policy for Phase 1 patients. Three value functions are shown, with the associated action chosen by the learned policy, for $s_2 = 25$, $s_2 = 50$, and $s_2 = 75$.

the set of preferences for which that action looks best.[13] In all examples, we show the policy for PANSS percentile $s_t = 50$.

6.4.1 PHASE 2 ANALYSES

We use the same state representation as for the $D = 2$ example. Because we are using the exact same $r_{[0]}$ and $r_{[1]}$ as we did for the $D = 2$ example as well, we can exactly recover the learned policy of our previous $D = 2$ analysis from our $D = 3$ analysis simply by considering all preferences of the form $\delta = (\delta, 1-\delta, 0)$, that is, the preferences along the upper-left edge of the triangle.

Figure 11 shows the learned policy for patients with $s_2 = 50$ whose Phase 1 treatment was not efficacious. As in the $D = 2$ case, we combine the actions of giving {olanzapine, risperidone, or quetiapine} into one "not-clozapine" action. We see that clozapine appears best if the reward is based only on PANSS or on HQLS, and "not-clozapine" appears best only if the preference assigns a relatively large weight to BMI. If we consider the upper-left edge of the triangle where the preferences assign zero weight to HQLS, we get precisely the same policy shown in Figure 8. We also see that clozapine appears best for all preferences that consider only PANSS and HQLS (bottom edge) and for most preferences that consider only HQLS and BMI (upper-right edge.) We hypothesize that this is because there is a strong association between control of schizophrenia

13. In addition to the policy, we have indicated the linear regions produced by the Delaunay triangulations using faint lines in order to give a sense of their complexity.

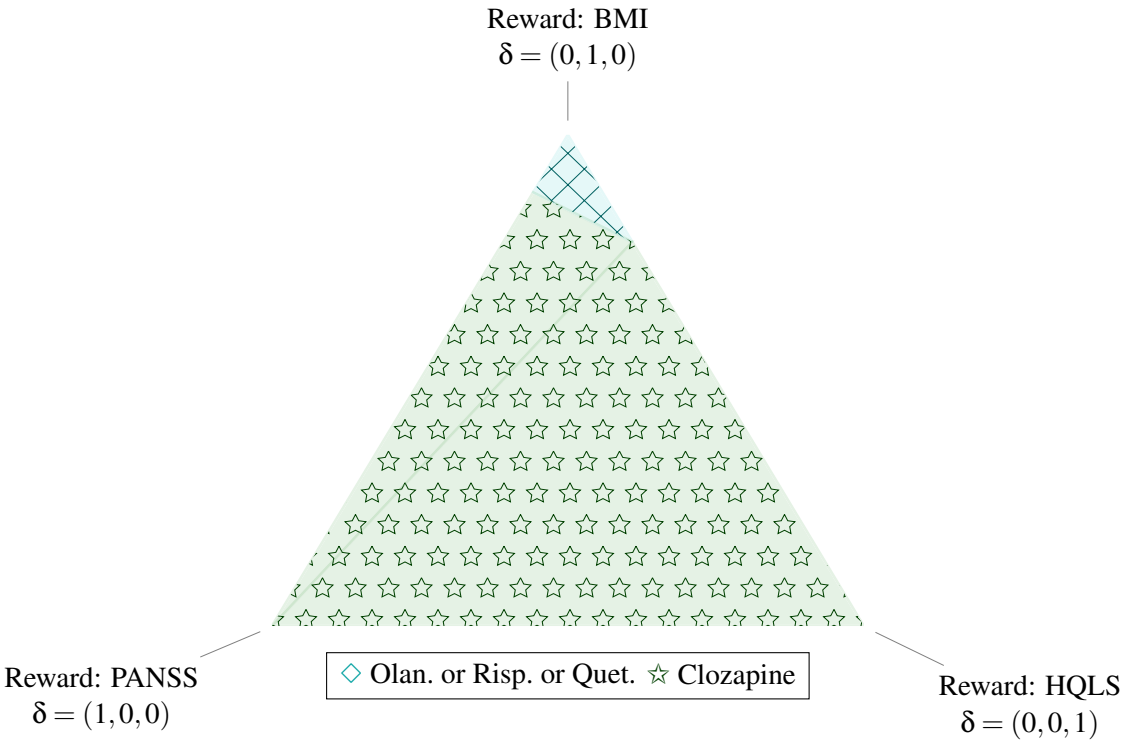

Figure 11: Multiple rewards analysis using PANSS, BMI, and HQLS, showing learned policy for Phase 2 Efficacy patients with $s_2 = 50$.

symptoms and quality of life; thus treatments that work well for PANSS should also work somewhat well for HQLS. We note however that clozapine occupies a narrower range on the upper-right edge than it does on the upper-left edge. In this analysis it is clear that neither action is globally dominated because neither is dominated at state $s_2 = 50$.

Figure 12 shows the learned policy for patients with $s_2 = 50$ whose Phase 1 treatment was not tolerable due to side-effects. Here, we see that olanzapine appears best if the reward is based only on PANSS or on HQLS, and ziprasidone appears best if the preference assigns a relatively large weight to BMI. For intermediate preferences, risperidone appears best. Again if we consider the upper-left edge of the triangle where the preferences assign zero weight to HQLS, we get precisely the same policy shown in Figure 9. We also see that olanzapine appears best for all preferences that consider only PANSS and HQLS (bottom edge.) Note that horizontal lines in the triangle represent sets of preferences where the weight on BMI is held constant. Over much of preference space, these horizontal lines are completely contained within one treatment's optimal region, meaning that given a weight for BMI, the policy usually does not depend on the relative preference for PANSS versus HQLI. We hypothesize again that this is because there is a strong association between symptom control and quality of life. In this analysis, we found that quetiapine was globally dominated.

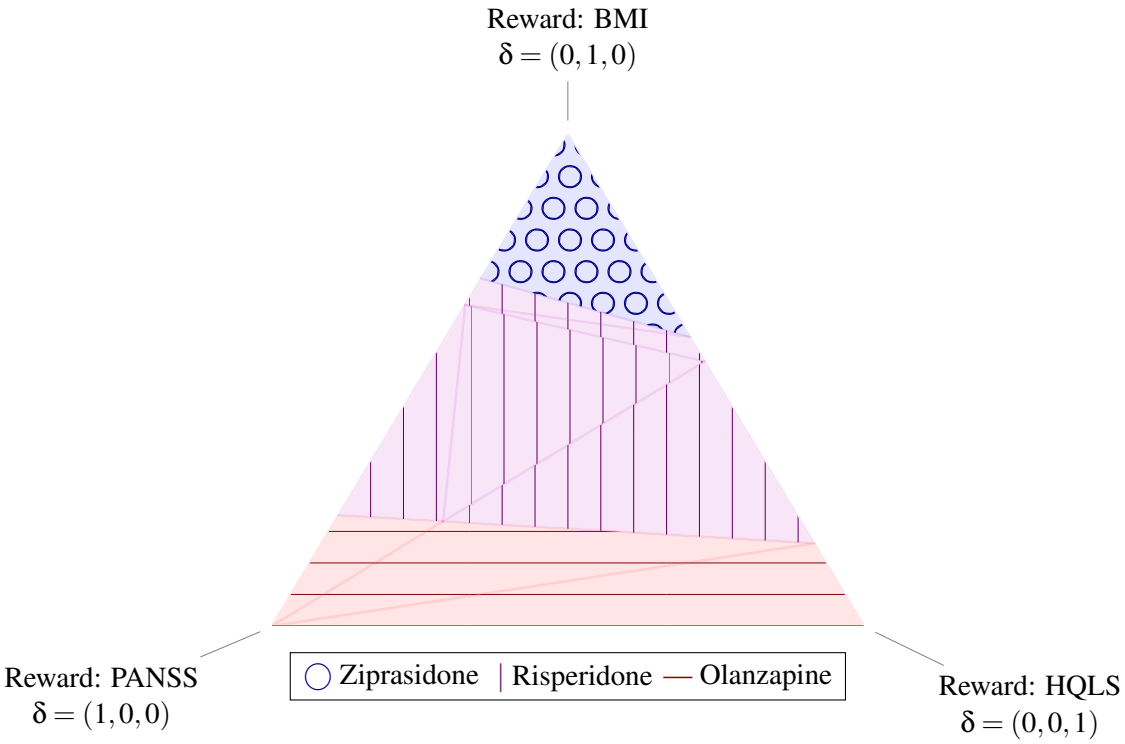

Figure 12: Multiple rewards analysis using PANSS, BMI, and HQLS, showing learned policy for Phase 2 Tolerability patients with $s_2 = 50$.

6.4.2 PHASE 1 ANALYSIS

Figure 13 shows the learned policy for patients with $s_1 = 50$. Again we see that ziprasidone appears best for preferences that assign a large importance to BMI, and olanzapine appears best for other preferences. Again if we consider the upper-left edge of the triangle where the preferences assign zero weight to HQLS, we get precisely the same policy shown in Figure 10. Interestingly, the region where ziprasidone appears best increases as we decrease the importance of PANSS, indicating it may be preferable for patients who are more concerned with weight control and quality of life than with very tight control of symptoms. In this analysis, we found that quetiapine, risperidone, and perphenazine were dominated at our example state $s_1 = 50$, but we found no action to be globally dominated.

6.5 Limitations

We note that unlike our previous work using this data, this analysis does not attempt to remove bias induced by missing data, nor does it provide measures of uncertainty for the learned policy (Shortreed et al., 2010). Both of these limitations indicate important directions for future work, as we discuss below.

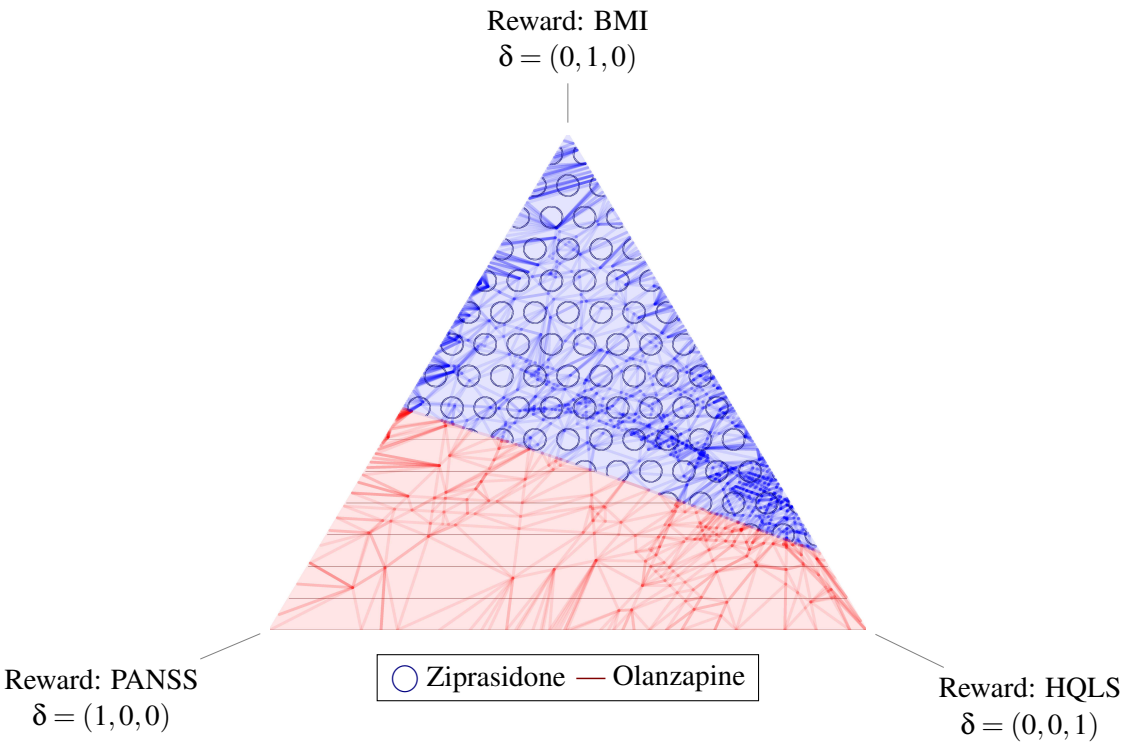

Figure 13: Multiple rewards analysis using PANSS, BMI, and HQLS, showing learned policy for Phase 1 patients with $s_1 = 50$.

7. Discussion and Future Work

The methods we have presented comprise a crucial first step towards a data analysis method that can be deployed in clinical decision support systems. However, there are challenges that remain to be addressed.

7.1 The Meaning of Rewards and the Effect of Scaling

Consider an analysis with $D = 2$ basis rewards at its final time point $t = T$. For a preference of $\delta = 0.5$, two actions a_1 and a_2 for which $0.5r_{[0]}(s_T, a_1) + 0.5r_{[1]}(s_T, a_1) = 0.5r_{[0]}(s_T, a_2) + 0.5r_{[1]}(s_T, a_2)$ are indistinguishable. One can think of the preference as setting an "exchange rate" for $r_{[0]}$ and $r_{[1]}$: in this case, the basis rewards can be exchanged at a one-to-one rate and our happiness with the overall result of an action remains the same. For $\delta = 0.75$, the two actions would be indistinguishable if $0.25r_{[0]}(s_T, a_1) + 0.75r_{[1]}(s_T, a_1) = 0.25r_{[0]}(s_T, a_2) + 0.75r_{[1]}(s_T, a_2)$, meaning that the loss of one unit of $r_{[1]}$ would have to be compensated by a gain of three units of $r_{[0]}$ in order for the actions to be considered equivalent. The stronger the preference for $r_{[1]}$, the more units of $r_{[0]}$ we need in order to "make up" for the loss a unit of $r_{[1]}$. Note that this interpretation would not be possible had we chosen to define reward as a non-linear function of preference.

In our example analysis, we chose to convert all of the rewards to percentiles before using them; thus we interpret a preference of $\delta = 0.5$ to mean that the "exchange rate" is one-to-one

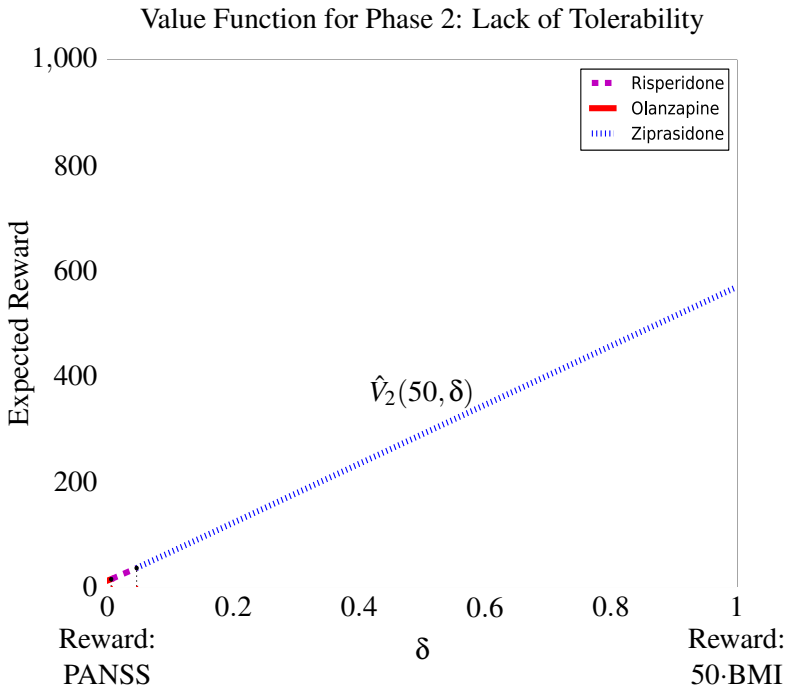

Figure 14: Multiple rewards analysis showing learned value function and associated learned policy for Phase 2 Tolerability patients, **using** 50·**BMI as one basis reward**. Note how scaling the BMI reward affects the regions where different actions are optimal. (Compare with Figure 9.)

for percentiles of PANSS and percentiles of BMI. The exchange rate at $\delta = 0.5$ could be shifted however by first multiplying one or both rewards by a constant factor before using them in our algorithm. If we fed $2 \cdot \text{BMI}$ into our algorithm as $r_{[1]}$, the exchange rate at $\delta = 0.5$ would be two percentiles of PANSS equals one percentile of BMI, and the preference at which the policy changes from one action to the other in Figure 9, for example, would shift to the left. The ordering of recommended treatments (olanzapine for lowest δ, risperidone for moderate δ, ziprasidone for high δ) would remain the same, however.

A more extreme version is illustrated in Figure 14, where we use $50 \cdot \text{BMI}$ as one basis reward. In this analysis, the "exchange rate" for $\delta = 0.5$ is one unit of BMI equals 50 units of PANSS. Note that there are still three non-dominated actions, but the regions where two of them are optimal are now very small and "compressed" into an area very near $\delta = 0$. This illustrates a potential pitfall: if the exchange rate represented by $\delta = 0.5$ is not "moderate," resulting decision aids will be at best unhelpful and at worst misleading. However, it also supports the use of the exact algorithms we have presented: even if the rewards are poorly scaled, the set of non-dominated actions remains the same, and they retain their ordering according to delta.

Note that if the region where an action is optimal is very small, a naïve grid-search over δ may not detect it. For example, if we ask try to determine in the Figure 14 example which treatments are optimal near a preference of $\delta = 0$ by checking nearby δ, we may miss risperidone. On the

other hand, the exact methods we have presented will correctly recognize that the risperidone is non-dominated. A practitioner using our methods might then wish to change the analysis by rescaling one or more of the basis rewards. The nature of this rescaling will of course depend on the application at hand; we intend to formalize this problem in future work.

7.2 Value Function and Policy Approximations

We have shown that the complexity of constructing the exact value function is potentially exponential in the time horizon of the problem. However, we have also shown in our example that although the value function may be very complex, the learned policy may still be very simple. Figure 10 illustrates this: each faint triangle in the figure represents a linear piece of the value function. Though there are many pieces, by and large adjacent pieces recommend the same action. This reflects a large-scale smoothness in the Q-functions, and suggests that a simple, smooth function might approximate the piecewise-linear Q-functions very well while reducing computational cost. Some existing algorithms for POMDPs that approximate the value function (e.g., Pineau et al. 2006, Wang et al. 2006) may be useful, but novel modifications will be needed to use these approximations in our setting. Another class of approximations introduced by Poupart and Boutilier (2002) focuses on compressing the state space of the POMDP. As we discussed in Section 3.2, the number of states in a POMDP roughly corresponds to the number of basis rewards considered by our method. Thus, these methods may lead to a way of computing a simplified or "compressed" view of preferences when the number of basis rewards is large, which could be used both to reduce computational cost and to help users better understand their preferences.

7.3 Measures of Uncertainty and Similar Q-values

In our example, almost all preferences are associated with exactly one optimal action. In practice, it may make more sense to recommend more than one action for a particular preference if the Q-values of those actions are very similar. In the medical setting, one may prefer to allow the physician or patient to break ties if outcomes under different treatments are deemed to be "close." We note two criteria for "closeness" that deserve further study.

Statistical Significance One reason for recommending a set of treatments arises when there is insufficient evidence that one action is actually superior to another. Ideally, one would like to know if an observed difference in Q-values for different actions is true for the population or if it is present in the data we have simply by chance. The methods we have presented do not provide uncertainty information about the learned value function or policy, and although the algorithm is based on linear regression which itself has well-established methods for statistical inference, it is known that even standard single-reward fitted-Q iteration requires specially tailored statistical methods in order to obtain valid confidence measures (Laber et al., 2009; Shortreed et al., 2010). These methods, based on the bootstrap data re-sampling procedure, can be very computationally intensive even for one reward function; thus it will be crucial to combine them with new approximations to the problem in order to produce analyses in a reasonable amount of time. Methods for mitigating the bias induced by having partially missing data can be computationally intensive as well (Shortreed et al., 2010), and should be investigated concurrently with methods for producing confidence information.

Practical Significance Even if we have strong statistical evidence that one action has a higher Q-value than another, we may still wish to recommend a set of actions if that difference is too small to be practically meaningful. Methods for mathematizing the idea of a "clinically meaningful

difference" are under investigation (Laber et al., 2012); we see promise for integrating them with our methods.

7.4 $D > 3$ Basis Rewards

To allow more than 3 basis rewards, we need methods that can represent and manipulate piecewise linear functions in higher dimensions. One avenue would be to use *extended algebraic decision diagrams*, which have been successfully applied to MDPs (Zamani et al., 2012). It is not obvious whether XADD methods provide us with a computationally feasible solution for $D > 3$, but their use is worthy of future study.

8. Conclusion

We have presented a general and explicit development of finite-horizon fitted-Q iteration with an arbitrary number of reward signals and linear value function approximation using an arbitrary number of state features. This included a detailed treatment of the 3-reward function case using triangulation primitives from computational geometry and a method for identifying globally dominated actions under linear function approximation. We also presented an example of how our methods can be used to construct real-world decision aid by considering symptom reduction, weight gain, and quality of life in sequential treatments for schizophrenia. Finally, we have discussed future directions in which to take this work that will further enable our methods to make a positive impact on the field of evidence-based clinical decision support.

Acknowledgments

We extend our sincere thanks to our reviewers, whose detailed comments have enabled us to substantially improve our work. We acknowledge support from Natural Sciences and Engineering Research Council of Canada (NSERC) and the National Institutes of Health (NIH) grants R01 MH080015 and P50 DA10075. Data used in the preparation of this article were obtained from the limited access data sets distributed from the NIH-supported "Clinical Antipsychotic Trials of Intervention Effectiveness in Schizophrenia" (CATIE-Sz). The study was supported by NIMH Contract N01MH90001 to the University of North Carolina at Chapel Hill. The ClinicalTrials.gov identifier is NCT00014001. This manuscript reflects the views of the authors and may not reflect the opinions or views of the CATIE-Sz Study Investigators or the NIH.

Appendix A. Proofs

Note that Lemmas 3 and 4 are known (or deemed "obvious") in the computational geometry literature, but are proved here for completeness.

A.1 Proof of Lemma 3

Over each $\mathcal{R}_i$, $f_{\max} = f_i$ which is linear. Each $\mathcal{R}_i$ is an intersection of the convex polytope $\mathcal{R}$ with an intersection of half-spaces of the form $\{\delta : f_i(\delta) \geq f_j(\delta)\}$, which are also convex polytopes. Thus each $\mathcal{R}_i$ is a convex polytope.

A.2 Proof of Lemma 4

For any point δ in a set $\mathcal{U} \cap \mathcal{V}$ as above, we have $g_1(\delta) = \delta^\mathsf{T} \mathbf{w}_\mathcal{U}$ and $g_2(\delta) = \delta^\mathsf{T} \mathbf{w}_\mathcal{V}$. Therefore, for such δ, we have

$$\begin{aligned}\alpha_1 \cdot g_1(\delta) + \alpha_2 \cdot g_2(\delta) &= \alpha_1 \cdot (\delta^\mathsf{T} \mathbf{w}_\mathcal{U}) + \alpha_2 \cdot (\delta^\mathsf{T} \mathbf{w}_\mathcal{V}), \\ &= \delta^\mathsf{T} (\alpha_1 \cdot \mathbf{w}_\mathcal{U} + \alpha_2 \cdot \mathbf{w}_\mathcal{V}), \\ &= \delta^\mathsf{T} \mathbf{w}_{\mathcal{U} \cap \mathcal{V}}.\end{aligned}$$

Therefore within each set given by the intersections above, both g_1 and g_2 are linear.

A.3 Proof of Lemma 5

Suppose a is optimal for some (ψ_{s_T}, δ) in the domain but is not optimal for any (ψ_{s_T}, δ) on the boundary of the domain. Further suppose that a is not optimal at any triple-point. Then the region where a is optimal must be completely enclosed by the region where a *single* other action a' is optimal. However, by Equation (8), the boundary between the regions where a is superior to a' and vice-versa is a hyperbola composed of two sets (sheets) that are each continuous and have infinite extent in both ψ_{s_T} and δ. The set must therefore intersect the boundary of the domain of (ψ_{s_T}, δ) and thus there must exist a certificate for a on the boundary. Thus we have a contradiction.

A.4 Proof of Proposition 7

Assume there is a region inside the domain where a is optimal. Assume a is not optimal at any $(p+D+1)$-tuple point. Since a is not optimal at a point where $p+D+1$ actions are optimal, the region where a is optimal must have on its boundary points where k actions are simultaneously optimal for some $k < p+D+1$. Choose the maximum k for which this is true. This boundary is defined by a system of $k-1$ polynomial equations on $p+D$ variables of the form (10); call the variables $\zeta_1, \zeta_2, ..., \zeta_{p+D}$. Since we assume the region where a is optimal is in the interior of the domain, there exists an interior point ζ^* that is a solution to the system of equations. Create a new system of $k-1$ equations and $k-1$ unknowns by fixing the last $(p+D)-(k-1)$ variables to $\zeta_k = \zeta_k^*$, $\zeta_{k+1} = \zeta_{k+1}^*$ ..., $\zeta_{p+D} = \zeta_{p+D}^*$. The point $(\zeta_1^*, ..., \zeta_{k-1}^*)$ is a solution to this reduced system. Suppose the solution of the reduced system is a continuous function of ζ_k^*, which holds if Conjecture 6 is true. Then if we move ζ_k^* toward a boundary from its original value, either we will find a point satisfying the original system with ζ_k^* on the boundary and the remaining variables in the interior of the domain, or another variable or variables will reach its boundary first, and the remainder of the variables will be in the interior of the domain. In either case, there exists a point on the boundary of the domain of interest where action a is optimal.

References

D. B. Allison, J. L. Mentore, M. Heo, L. P. Chandler, J. C. Cappelleri, M. C. Infante, and P. J. Weiden. Antipsychotic-induced weight gain: A comprehensive research synthesis. *American Journal of Psychiatry*, 156:1686–1696, November 1999.

L. Barrett and S. Narayanan. Learning all optimal policies with multiple criteria. In *Proceedings of the 25th International Conference on Machine Learning*, pages 41–47, 2008.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*, chapter 2.1, page 12. Athena Scientific, 1996.

O. Bonnichsen. Elicitation of ostomy pouch preferences: a discrete-choice experiment. *Patient*, 4 (3):163–175, 2011.

C. Boutilier. A POMDP formulation of preference elicitation problems. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 239–246, 2002.

P. Brass. On the size of higher-dimensional triangulations. In J. E. Goodman, J. Pach, and E. Welzl, editors, *Combinatorial and Computational Geometry*, volume 52, pages 147–152. MSRI Publications, 2005.

A. Castelletti, S. Galelli, M. Restelli, and R. Soncini-Sessa. Tree-based reinforcement learning for optimal water reservoir operation. *Water Resources Research*, 46(W06502), 2010.

CGAL. CGAL, Computational Geometry Algorithms Library, 2011. URL `http://www.cgal.org`.

L. P. Chew. Constrained delaunay triangulations. In *Proceedings of the Third Annual Symposium on Computational geometry*, SCG '87, pages 215–222, New York, NY, USA, 1987.

R. D. Cook and S. Weisberg. *Applied Regression Including Computing and Graphics*. Wiley, August 1999.

D. A. Cox, D. O'Shea, and J. B. Little. *Ideals, Varieties and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, 1997.

J. A. Cramer, R. Rosenheck, W. Xu, J. Thomas, W. Henderson, and D. S. Charney. Quality of life in schizophrenia: A comparison of instruments. *Schizophrenia Bulletin*, 26(3):659–666, 2000.

C. C. Davis, M. Claudius, L. A. Palinkas, J. B. Wong, and L. K. Leslie. Putting families in the center: Family perspectives on decision making and ADHD and implications for ADHD care. *Journal of Attention Disorders*, Oct 2011. E-pub ahead of print.

M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry Algorithms and Applications*. Springer, 3 edition, 2008.

M. Ehrgott. *Multicriteria Optimization*, chapter 3. Springer, second edition, 2005.

Z. Gábor, Z. Kalmár, and C. Szepesvári. Multi-criteria reinforcement learning. In *Proceedings of the 15th International Conference on Machine Learning*, pages 197–205, 1998.

G. H. Golub and C. F. Van Loan. *Matrix Computation*. John Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 1996. ISBN 0471935271.

B. Grünbaum. *Convex Polytopes*, volume 221 of *Graduate Texts in Mathematics*. Springer-Verlag, 1967. ISBN 0387004246;.

D. W. Heinrichs, T. E. Hanlon, and W. T. C. Jr. The Quality of Life Scale: An instrument for rating the schizophrenic deficit syndrome. *Schizophrenia Bulletin*, 10(3):388–398, 1984.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(12):99–134, 1998.

S. R. Kay, A. Fiszbein, and L. A. Opfer. The Positive and Negative Syndrome Scale (PANSS) for schizophrenia. *Schizophrenia Bulletin*, 13(2):261–276, 1987.

P. W. Keller, S. Mannor, and D. Precup. Automatic basis function construction for approximate dynamic programming and reinforcement learning. In *Proceedings of the 23rd International Conference on Machine learning*, ICML '06, pages 449–456, 2006.

A. Y. Kuk, J. Li, and A. J. Rush. Recursive subsetting to identify patients in the STAR*D: a method to enhance the accuracy of early prediction of treatment outcome and to inform personalized care. *Journal of Clinical Psychiatry*, 71(11):1502–8, November 2010.

E. B. Laber, M. Qian, D. J. Lizotte, and S. A. Murphy. Statistical inference in dynamic treatment regimes. Technical Report 50, Univ. of Michigan Statistics Department, 2009.

E. B. Laber, D. J. Lizotte, and B. Ferguson. Set-valued dynamic treatment regimes for competing outcomes. `arXiv:1207.3100v2 [stat.ME]`, 2012.

D. J. Lizotte, M. Bowling, and S. A. Murphy. Efficient reinforcement learning with multiple reward functions for randomized controlled trial analysis. In J. Fürnkranz and T. Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning*, pages 695–702, Haifa, Israel, June 2010. Omnipress.

S. Mannor and N. Shimkin. A geometric approach to multi-criterion reinforcement learning. *Journal of Machine Learning Research*, 5:325–260, 2004.

S. Mannor and J. N. Tsitsiklis. Mean-variance optimization in Markov decision processes. In *Proceedings of the 28th International Conference on Machine Learning*, pages 177–184, 2011.

M. S. McDonagh, K. Peterson, S. Carson, R. Fu, and S. Thakurta. Drug class review: Atypical antipsychotic drugs. Technical report, Oregon Health & Science University, July 2010. Drug Effectiveness Review Project, Update 3.

J. R. McKay. *Treating Substance Use Disorders with Adaptive Continuing Aftercare*. American Psychological Association, 2009.

A. L. Miller, J. A. Chiles, and J. K. C. et al. The Texas Medication Algorithm Project (TMAP) schizophrenia algorithms. *Journal of Clinical Psychiatry*, 60:649–657, 1999.

S. A. Murphy. An experimental design for the development of adaptive treatment strategies. *Statistics in Medicine*, 24:1455–1481, 2005.

S. A. Murphy, S. W. Oslin, A. J. Rush, and J. Zhu. Methodological challenges in constructing effective treatment sequences for chronic psychiatric disorders. *Neuropsychopharmacology*, 32: 257–262, 2007.

S. Natarajan and P. Tadepalli. Dynamic preferences in multi-criteria reinforcement learning. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 601–608, 2005.

National Institutes of Health. *Clinical Guidelines on the Identification, and Treatment of Overweight and Obesity in Adults: The Evidence Report*. National Heart, Lung, and Blood Institute, Sept 1998. NIH Publication no. 98-4083.

A. Y. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning*, pages 663–670, 2000.

J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence*, pages 1025–1032, 2003.

J. Pineau, G. Gordon, and S. Thrun. Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research*, 27:335–380, 2006.

J. Pineau, M. G. Bellemare, A. J. Rush, A. Ghizaru, and S. A. Murphy. Constructing evidence-based treatment strategies using methods from computer science. *Drug and Alcohol Dependence*, 88 (Suppl 2):S52–S60, May 2007.

P. Poupart and C. Boutilier. Value-directed compression of POMDPs. *Advances in Neural Information Processing Systems*, 15:1547–1554, 2002.

D. L. Sackett. Evidence-based medicine: What it is and what it isn't. *British Medical Journal*, 312 (7023):71–72, 1996.

S. Shortreed, E. B. Laber, D. J. Lizotte, T. S. Stroup, J. Pineau, and S. A. Murphy. Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Machine Learning*, pages 1–28, 2010. ISSN 0885-6125.

T. S. Stroup, J. P. McEvoy, M. S. Swartz, M. J. Byerly, I. D. Glick, J. M. Canive, M. McGee, G. M. Simpson, M. D. Stevens, and J. A. Lieberman. The national institute of mental health clinical antipschotic trials of intervention effectiveness (CATIE) project: Schizophrenia trial design and protocol deveplopment. *Schizophrenia Bulletin*, 29(1):15–31, 2003.

B. Sturmfels. *Solving Systems of Polynomial Equations*. Regional conference series in mathematics. Published for the Conference Board of the Mathematical Sciences by the American Mathematical Society, 2002. ISBN 9780821832516.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

M. S. Swartz, D. O. Perkins, T. S. Stroup, J. P. McEvoy, J. M. Nieri, and D. D. Haal. Assessing clinical and functional outcomes in the clinical antipsychotic of intervention effectiveness (CATIE) schizophrenia trial. *Schizophrenia Bulletin*, 29(1):33–43, 2003.

C. Szepesvári. *Algorithms for Reinforcement Learning*. Morgan & Claypool, July 2010.

P. F. Thall. Some geometric methods for constructing decision criteria based on two-dimensional parameters. *Journal of Statistical Planning and Inference*, 138(2):516–527, 2008.

D. J. Uherka and A. M. Sergott. On the continuous dependence of the roots of a polynomial on its coefficients. *The American Mathematical Monthly*, 84(5):368–370, 1977. ISSN 0002-9890.

P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning*, 84:51–80, 2011. ISSN 0885-6125. 10.1007/s10994-010-5232-5.

T. Wang, P. Poupart, M. Bowling, and D. Schuurmans. Compact, convex upper bound iteration for approximate pomdp planning. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 1245–1251, 2006.

J. R. Weisz, B. C. Chu, and A. J. Polo. Treatment dissemination and evidence-based practice: Strengthening intervention through clinician-researcher collaboration. *Clinical Psychology: Science and Practice*, 11(3):300–307, 2004. ISSN 1468-2850.

Working Group for the Canadian Psychiatric Association and the Canadian Alliance for Research on Schizophrenia. Canadian clinical practice guidelines for the treatment of schizophrenia. *Canadian Journal of Psychiatry*, 43(suppl. 2):25–40S, 1998.

Z. Zamani, S. Sanner, and C. Fang. Symbolic dynamic programming for continuous state and action MDPs. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, 2012. To appear.

Y. Zhao, M. R. Kosorok, and D. Zeng. Reinforcement learning design for cancer clinical trials. *Statistics in Medicine*, 28:3294–3315, 2009.