5-2006

Approximate dynamic programming for anemia management.

Mehmet K. Muezzinoglu
University of Louisville

APPROXIMATE DYNAMIC PROGRAMMING FOR ANEMIA MANAGEMENT

By

Mehmet K. Muezzinoglu
Ph.D., Dokuz Eylul University, 2003

A Dissertation
Submitted to the Faculty of the
Graduate School of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

Electrical Engineering Program
University of Louisville
Louisville, Kentucky

May 2006

APPROXIMATE DYNAMIC PROGRAMMING FOR ANEMIA MANAGEMENT

By

Mehmet K. Muezzinoglu
Ph.D., Dokuz Eylul University, 2003

A Dissertation Approved on

April 12, 2006

by the following Dissertation Committee:

ACKNOWLEDGMENTS

I would first like to thank my advisor, Dr. Jacek M. Zurada, Director of the Computational Intelligence Laboratory at the University of Louisville, for his guidance and encouragement. I would also like to thank the members Drs. Mike Brier, Joseph Cole, James Graham, John Lilly, and the reader Dr. Tamer Inanc for agreeing to serve on my dissertation committee.

The motivation and medical background generously provided by the Kidney Diseases Program, University of Louisville, are greatly appreciated. Special thanks to Dr. Adam Gaweda, who set up the experimental framework of this study.

I am grateful to Dr. Andrew Barto and the members of Autonomous Learning Laboratory, University of Massachusetts, Amherst, for their hospitality, interest in this work, and constructive input during the summer of 2005.

I am also indebted to my friends and colleagues, Dr. Greg Boratyn, Piotr Habas, Yanfeng Hou, and Maciej Mazurowski for their unselfish help.

This dissertation came into being thanks to my wife Tulay. I am thankful for her patience, unconditional support, and, beyond all, her faith in me.

ABSTRACT

APPROXIMATE DYNAMIC PROGRAMMING FOR ANEMIA MANAGEMENT

Mehmet K. Muezzinoglu

April 12, 2006

The focus of this dissertation work is the formulation and improvement of anemia management process involving trial-and-error. A two-stage method is adopted toward this objective. Given a medical treatment process, a discrete Markov representation is first derived as a formal translation of the treatment process to a control problem under uncertainty. A simulative numerical solution of the control problem is then obtained on-the-fly in the form of a control law maximizing the long-term benefit at each decision stage.

Approximate dynamic programming methods are employed in the proposed solution. The motivation underlying this choice is that, in reality, some patient characteristics, which are critical for the sake of treatment, cannot be determined through diagnosis and remain unknown until early stages of treatment, when the patient demonstrates them upon actions by the decision maker. A review of these simulative control tools, which are studied extensively in reinforcement learning theory, is presented.

Two approximate dynamic programming tools, namely SARSA and Q-learning, are introduced. Their performance in discovering the optimal individualized drug dosing policy is illustrated on hypothetical patients made up as fuzzy models for simulations. As an addition to these generic reinforcement learning methods, a state abstraction scheme for the considered application domain is also proposed. The control methods of this study, capturing the essentials of a drug delivery problem, constitutes a novel computational framework for model-free medical treatment.

Experimental evaluation of the dosing strategies produced by the proposed methods against the standard policy, which is being followed actually by human experts in Kidney Diseases Program, University of Louisville, shows the advantages for use of reinforcement learning in the drug dosing problem in particular and in medical decision making in general.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE PAGE

CHAPTER I
INTRODUCTION

Motivation

Trial-and-error is a fundamental component of reasoning. The role of this heuristic in animal learning has been first conjectured rigorously by Edward Thorndike in (Thorndike, 1911) by the *Law of Effect*. His view initiated a mainstream approach in learning theory, namely Reinforcement Learning (RL), as the prominent alternative to the Pavlovian explanation. His approach suggests that developing a strategy to cope with the environment is possible due to two characteristic features: First, this form of learning is selective, as the learning system selects an action among alternatives towards a goal. Second, it is associative as these alternatives are corresponded to particular situations.

Mimicking trial-and-error learning in animals constitutes a major challenge in artificial intelligence (Russell and Norvig, 2002). Due to the external information gained by an agent, *the learner*, along its interactions with its environment, this form of learning has long been confused with supervised learning, the environment being considered as the supervisor, in artificial intelligence literature. In fact, an external supervision telling the agent what to do explicitly barely exists. It is true that there is a signal available for the agent incurred upon each decision, and this signal is really correlated to the agent's performance upon each interaction. However, unlike in conventional supervised learning, it does not convey an example for the learner. Observing such a signal gives rise to a non-trivial internal evaluation in the agent, involving learning lessons from its previous choices and reflecting this gained experience in planning pragmatically its future. Due to this distinctive effort, modern approaches in artificial intelligence study trial-and-error learning exclusively under the category of RL. Another computational aspect differing RL from conventional supervised learning schemes, which have been founded on Newtonian methods and gradient descent iterations, is that RL is applicable on dynamic and stochastic problems very effectively. With its computational roots fed by the theory of stochastic approximation (Benveniste et al., 1990), RL has developed into a major research field in computational intelligence for more than two decades (Barto et al., 1983; Kaelbling et al., 1996).

Formally, the process of trial-and-error is a systematic way of eliminating non-beneficial attempts in a learning task, thus is strongly linked to the computational framework of Dynamic Programming (DP) studied in control theory (Bellman, 1957) and also in theoretical computer science (Baase and van Gelder, 1999). However, for this powerful methodology to be directly applicable, a critical piece of information, namely the exactly-quantified utility of each possible action at each situation that the learning system can ever encounter, is essential (c.f. the association feature of trial-and-error). Unfortunately, these data become available in many real-life applications only after sufficient amount of interaction of the learning system with its environment. For example, in order to apply the Dijkstra algorithm, a popular shortest-path solver employing DP ideas, the complete graph information in the form of nodes and arc lengths is required in advance. In many learning tasks, however, it is too optimistic to expect that complete information be available to the learning system in advance. To solve the shortest path problem under such a constraint, the learning system would have choice but to explore both the nodes and the arcs of the graph by a number of trials, most of them leading possibly to non-optimal solutions. The information gained by exploring the graph is stored in the DP table on-the-fly, i.e. in parallel with the solution.

The initial lack of information about the problem, which hinders or simply blocks a direct DP solution, has given rise to Approximate Dynamic Programming (ADP) methods, which combine the simulation-based information gathering techniques and the well-posed

DP methods. The author's interest in the approximate dynamic programming is due to the observation that they have been remarkably successful in overcoming real-life multistage decision making problems under uncertainty, yet employing simple mechanisms (Connell and Mahadevan, 1993; Tesauro, 1995; Barto et al., 1995; Bertsekas and Tsitsiklis, 1996).

Another (and the most celebrated) characteristic of ADP is its utilizing a parametric approximator to summarize the DP table, the only means for storing gained experience during the entire solution process. This crucial set of information grows to unfeasibly large sizes as the dimension of the state space is increased linearly. In particular, its size is an exponential of the state-space dimension, a phenomenon addressed in the computer science jargon as *the curse of dimensionality*. To maintain tables cursed by dimensionality, they are typically parameterized in ADP framework by a reasonable number of variables, which converts the conventional DP solution from DP table updates to parameter updates at each solution stage. As parametric universal approximators, feedforward neural networks are primarily employed in ADP solutions for this purpose.

A majority of medical decision making instances have traditionally been based on trial-and-error, which aims at keeping certain patient variables quantified by (possibly erroneous) measurements. The basic motive that validates the trial-and-error heuristic in this process is the difficulty in (or in many cases simply the impossibility of) modeling the patient. Therefore, a medical process is sometimes a model-free control task subject to uncertainty, thus cannot be handled by the classical control theory, which assumes typically a given plant on which the control designer can rely to a certain extent.

Drug administration in chronic conditions is a typical recurrent trial and error process: A physician selects an initial drug dose based on a standard reference and observes the patient for specific response and/or side effects. Following the observed state of the patient, the dose is adjusted to improve the response and/or to minimize dangerous side effects. The adjustment continues until a desired response is achieved. Here the physician can be viewed as an agent performing goal-oriented learning.

Scope and Contributions of This Study

This thesis work reviews ADP tools to reveal their applicability in medical treatment processes, which involve inherently trial-and-error heuristics. An automated procedure that is applicable on certain treatment instances constitutes the broader objective of this study.

To achieve a valid formulation and optimal control of a given treatment process, a two-step procedure is implemented in this work.

1. Translating a given treatment process into a Markov Decision Problem (MDP) constitutes the foremost challenge towards solution. This includes quantifying and quantizing the control variables as well as declaring internal state variables and all inputs/outputs of the considered system. Another essential task performed at this step is the introduction of reward formulation in order to reflect the high-level control objective to rank the state transitions and underlying control actions. It is important to discriminate this phase from conventional modeling procedures for control (Astrom and Wittenmark, 1989), where the system dynamics need to be delineated in the form of a legitimate state representation.

2. Having obtained a MDP formulation for the considered treatment, the second step is selecting and customizing an ADP method to extract actions that maximize the cumulative reward. This method operates on sample trajectories actually generated by the patient and is assumed to have access to the patient in order to demonstrate actions and observe their consequences. Based on these observations, the learning system of interest ranks the states and the control actions based on their utility with respect to the ultimate control goal. In some cases, this is even achieved in real time, together with the actual system operation by evaluating the consequences derived from the samples, which are generated under the control of the learning system itself. In other words, the considered framework enables carrying out both learning and control simultaneously. This second phase of the procedure involves particularly the

temporal difference methods of ADP augmented by connectionist approximators, particularly Radial Basis Function (RBF) networks that implement the control law being developed online as a locally parameterized input-output relation.

This study views the broad field of medical treatment under a fundamental assumption, namely the process' compatibility with the MDP framework. In particular, the methods of this study are applicable on treatment processes which can be accurately accommodated in a fully-observable Markovian domain with well-defined finite sets of states and actions. The scope of this work is limited also by fictitious patients implemented on digital computers that are *believed* to be mimicking real patient behavior in the considered problems. Consequently, this study does not claim any direct contribution to real-life treatment of any kind, although some practical concerns will be addressed in experiments, to an extent that they can be reflected in the design of virtual patients. Extensions of ADP-based methods in medical treatment are left to succeeding studies aiming primarily a safe clinical practice of trial-and-error heuristics.

A drug-dosing problem, namely anemia management is selected as the test bed for the methods adopted and/or developed in this study. All experimental results reported in this work address this problem described briefly and formulated in the following chapters. The experimental findings are due to a joint effort maintained with the Kidney Diseases Program, University of Louisville, which contributed to this work by providing the medical background, specifics of the benchmark problem, and continuous feedback. A particular contribution of this work, which has yet become the subject matter for (Gaweda et al., 2005c; Gaweda et al., 2005b; Gaweda et al., 2005a; Gaweda et al., 2006a), is a computational ground for the trial-and-error heuristics in anemia management. The author, however, would like to stress at this point that having focused on this particular problem should not constrain the validity of the general two-stage approach to the selected test bed.

Among other model-free control approaches in medical decision making, which are much smaller in number as compared to model-based methods (Khoo, 1999), this study constitutes apparently the pioneering attempt in utilizing approximate dynamic programming for treatment purpose, although its outcomes are regarding only the virtual patients currently.

Outline of the Dissertation

An introduction to Markov Decision Processes is made in the following chapter. Staged decision making terminology is presented together with examples illustrating how real-world problems are accommodated in this framework. A minimum set of conditions on the considered problem to achieve such an accommodation are listed therein. The practical value of this set is of utmost importance since it actually draws the crisp boundaries for the (medical) decision problems that can be handled using the general strategy of this study. Chapter II presents also the conventional Dynamic Programming methodology, an elegant solver for MDP, and a discussion of its limitations. Alternative simulative sub-procedures, which are compatible with the general DP strategy but capable of eliminating the demand for complete information about the process, are brought to the reader's attention. The role of parametric approximators in realizing these flexible solutions is finally explicated in Chapter II.

Chapter III sets the test bed of this study. The problem of Anemia Management is introduced and formulated within the MDP framework. As a critical component of anemia management, the patient response is described and possibilities of mimicking a real patient's behavior are discussed (although the primary motivation of this work is, indeed, avoiding the modeling process). A fuzzy patient model is advocated as a virtual patient that can sufficiently demonstrate critical features of a real patient with reasonably simple parameter adjustments. This model constitutes the object of the computing experiments presented in the remainder of this document.

Chapter IV is devoted to customization the two ADP methods, namely SARSA and Q-learning for anemia management. The performances of these approaches are evaluated experimentally on virtual patients with diverse response characteristics. A statistical comparison of the methods is also provided therein.

Chapter V proposes and implements a state abstraction scheme based on an inherent feature of the valid dosing strategies for anemia patients. In particular, the monotonicity of a valid dosing strategy with respect to patient variables is utilized to extend the reasoning triggered by observing a state transition in the patient to other (and possibly yet unobserved) states. This constitutes an illustration of utilizing problem-specific information in reducing the computational demands drastically. It is shown with extensive simulations that such an abstraction can substantially improve extraction of the optimal dosing strategy using trials. Improvement on the performance due to incorporating the suggested mechanism to the previous approaches is also accounted for.

Conclusions and selected future research directions motivated by this work are highlighted in Chapter VI.

CHAPTER II
DECISION MAKING IN STAGES

This study deals with problems where decisions are made in stages. This chapter describes a constructive framework to handle such problems, including many medical treatment instances under reasonable assumptions as formulated in the following chapters. The considered framework stands on two legs: A Markovian setting to cast the given decision problem - *the representation*, and a DP-based methodology - *the solver.*

In what follows, the representation is introduced together with the basic probabilistic notions. Dynamic programming methodology is then based on these notions. Some computational limitations of conventional DP approach are highlighted and alternative procedures are introduced.

Markov Decision Problem

Finite-State Stationary Markov Chains

Markov Chains (MC) is a unique representational domain for dynamical behavior. It adopts the elementary concept of *state* in system theory: *the summary of the past that acts on the future.* MC representation is applicable on (time-indexed) processes, which (i) occupy one and only one state at any given time and (ii) possess the essential Markovian property: *the necessity and sufficiency* of quantifying state variables at a fixed time t_0 in producing their values in any future time $t > t_0$.

An MC consists of a countable set S of states, which are usually enumerated by positive integers: $S = \{1, 2, \ldots, N\}$. Although N may be infinite, the case where it is finite, i.e. the finite-state MC, is of special interest.

The key idea of MC representation is treating the evolution of states of a dynamical process as a discrete-time[1] sequence of random variables $\{x_k\}_{k=0}^{\infty}$ on S, x_0 being the initial state.[2] This sequence (also called a trajectory) is assumed to be demonstrating the Markovian property and is governed by an $N \times N$ state transition matrix $\mathbf{P}[k]$, where the component $p_{ij}[k]$ denotes the probability that, given the current state is i, the following state will be j at time instant k:

$$p_{ij}[k] = P\left\{x_{k+1} = j | x_k = i\right\}. \tag{1}$$

This interpretation implies clearly that $\mathbf{P}[k]$ is a stochastic matrix for all k:

Definition 1 *A square matrix* $\mathbf{P}$ *is said to be stochastic matrix if all its elements are non-negative and the sum of the elements on each row is equal to* 1.

The final (yet the most restrictive) assumption of this study regarding MCs is that the state transition probabilities governing all dynamics are time-invariant: $\mathbf{P}[k] = \mathbf{P}$.

Definition 2 *A finite-state stationary Markov Chain is a Markovian process given the two-tuple* $\mathcal{M} = (S, \mathbf{P})$, *where* S *is the finite state set and* $\mathbf{P}$ *is the stochastic matrix inducing dynamics by providing the transition probabilities.*

Note that $(S, \mathbf{P})$ is an alternative to the conventional state representation, which is praised by the classical dynamical system and control theories for providing system analysis and design efforts with powerful tools of linear algebra and functional analysis. In fact, the state equations of a discrete-time dynamical process, with an input sequence specified, can always be considered as a declaration of the general term of a sequence,

[1]There are also continuous-time Markov processes. The word *chain* is reserved here for the discrete-time case, which is assumed throughout this study.

[2]As a note on notation, the time index k can be appended to a variable x either as a subscript x_k or as $x[k]$, whichever is appropriate in the current context.

reducing the representation to an MC. In other words, it is always possible to switch from a given state representation to an equivalent and unique MC. On the other hand, the reverse direction does not always exist, i.e. it may not be possible to obtain a state representation that governs the dynamical system exactly in the same way as a given MC. Therefore, the generic MC representation has a broader scope than the state representation in the sense that MC is capable of accounting for stochasticity that might be involved in state transitions in an explicit way, by means of the state transition matrix.

The Markov chain, even with the assumptions of stationarity and finiteness standing, is not only capable of accommodating a fairly broad class of dynamical processes but also brings novel solution techniques to the dynamical systems domain, especially when there is no state representation available. As will be demonstrated in this study, it is even possible to extract a partial MC representation of a stochastic system and still achieve a certain qualitative performance measure. Moreover, these two can be performed simultaneously. These useful features of MCs are shared by conventional state representation only in very special cases, e.g. in the case of state equations involving additive and Gaussian noise.

Markovian Processes Driven by Decisions

Staged decision processes in Markovian setting form a critical class of dynamical systems for modern engineering sciences. There are two characteristic features of these generally stochastic processes. First, the state transitions occur due to an external effect, called action. In particular, an action is assumed to be applied by a decision maker as an input to the process at each transition and has a non-neglectable effect on the determination of the next state. In a Markovian setting, the next state is produced not only by the current state but also on the action applied. The other characteristic of staged decision processes is that upon each state transition an immediate consequence is incurred. In real-life problems, such a consequence is usually an immediate cost or benefit regarding the decision made on

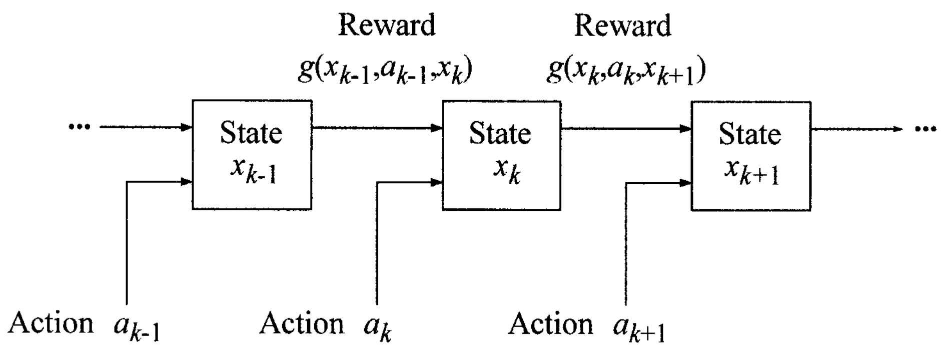

Figure 1. Three stages of a decision process in Markovian setting.

the preceding state. The reward (benefit) approach is adopted in this study. An illustration of a staged decision process including the actions and reward is given in Figure 1.

Finite-state MCs from an extended perspective provide the necessary mathematical rigor and constitute a constructive representational domain for staged decision processes. Two additional components to the basic MC setting are required to capture the characteristics mentioned above.

The first addition is a component addressing the actions which the target process. This will be denoted by the finite set A augmenting $\mathcal{M} = (S, \mathbf{P})$:

$$A = \{a_1, \ldots, a_M\}, \tag{2}$$

which contains all possible actions that can be applied at each state. Note that, setting a single set A for all possible states does not really restrict the actions to be applicable on each state. In a valid decision process, it may well be the case that the decision maker faces two different sets of options of actions A_i and A_j for two different states $i, j \in S$. In such cases, the global action set is determined as the union of applicable options, e.g. $A_i \cup A_j$ in this example.

The action $a \in A$ applied at state $i \in S$ can be viewed as specifying N probabilities regarding all possible transitions originating at i. From this perspective, the role of applying an action for i among the elements of A is actually equivalent to picking uniquely the i-th row of $\mathbf{P}$ from m possible candidates (each adding up to unity).

Definition 3 *Any function $\pi(\cdot) : S \longrightarrow A$ mapping states to actions is called policy.*

Under the finiteness assumptions above, it is easy to see that there are $|S| \cdot |A| = NM$ possible policies associated with the process.

Definition 4 *A given MC $\mathcal{M} = (S, A, \mathbf{P})$ is said to be following a policy π if the decision maker always performs as the action $\pi(i)$ at state i, for all $i \in S$.*

Note that specifying a policy to follow forces out the component A in the description of the process in the sense that there is no need to state the set of options A, once which option to pick is clearly stated. Specifying a particular π also fixes all rows of $\mathbf{P}$, thus determining uniquely all state transitions. The MC on the policy π is therefore denoted by $\mathcal{M}^{\pi} = (S, \mathbf{P}^{\pi})$.

The second addition to MC to handle decision processes is the real function $g(\cdot, \cdot, \cdot) : S \times A \times S \longrightarrow \Re$, called the reward functional. The evaluation $g(i, a, j)$ quantifies the immediate consequence of taking the action a at state i and attaining j as the next state. Note that the considered reward formulation is compatible with the probabilistic nature of the process in the sense that taking action a at state i may result in a variety of states along which the incurred reward may vary. Note that, when following a particular policy π, the reward function can be tabulated as an $N \times N$ matrix G^{π}, since the reward value is dependent only on i and j in this case: $g(i, a, j) = g(i, \pi(i), j)$.

Although the considered reward formulation is fairly general, there are two implicit assumptions underlying it: the functional $g(\cdot, \cdot, \cdot)$ is deterministic and time-invariant. In other words, the same reward is ensured in all cases whenever state j occurs upon taking action a at state i, irrespective of the time this happens.[3] The author would like to note at

[3]This neither implies that *state j always occurs when action a is taken at state i*, nor that *the action a must be taken at state i.*

this point that these assumptions may be dangerously restrictive in casting some real-life problems, although they are accepted by the canonical problems handled in this study.

Problem Statement

Given a Markovian decision process $\mathcal{M} = (S, A, \mathbf{P}, g)$ and an initial state $i \in S$, the Markov Decision Problem (MDP) is defined over the policy space and is formally stated as follows (Puterman, 1994).

Problem 1 *Find the policy π^* which maximizes the expected discounted sum of immediate rewards over the time interval $[0, t_f]$:*

$$\arg\max_{\pi} J^{\pi}(i) \triangleq E\left\{ \sum_{k=0}^{t_f-1} \alpha^k \cdot g\left(x_k, \pi(x_k), x_{k+1}\right) \,\middle|\, x_0 = i \right\}. \tag{3}$$

where the sequence $x_0, x_1, \ldots, x_{t_f}$ is produced by the chain $\mathcal{M}^{\pi}$, $\alpha \in (0.1]$ is a specified discount factor and the expectation is taken with respect to the probability distribution specified by the state transition matrix $\mathbf{P}$ governing the Markovian decision process.

As stated formally in Problem 1, the MDP formulation seeks the optimal way of assigning actions to states that maximizes the accumulated immediate rewards over a specified time interval. The problem is stated for an arbitrary state $i \in S$ to be taken as the initial state of the decision process $\mathcal{M}^{\pi}$. Note that, also implicit in this formulation is the sequence of states generated by $\mathcal{M}^{\pi}$ as a consequence of the optimal action sequence $\{\pi(x_0), \pi(x_1), \pi(x_2) \ldots\}$.

The kernel α of the power sequence that weights the immediate rewards is called the discount factor. This parameter of the decision problem tells to what extent the rewards due to future state transitions should contribute to the sum with respect to their distances to the initial state in time. The meaning of $\alpha < 1$ is that future rewards matter to us less than the same costs incurred at the present time. The marginal case of $\alpha = 1$ corresponds to the *undiscounted* case which sets the goal simply as maximizing the total reward accumulated over the specified time interval.

The problem is called an *infinite-horizon* MDP when $t_f \to \infty$. This case induces the most general formulation. It addresses even problems with a terminal state, i.e. a goal state which finalizes the state transitions. In fact, such problems with $\alpha = 1$ are called *the (stochastic) shortest path problems.* Analysis of these problems, which is skipped in this study, provides an foundation of the other types of problems accommodated by the infinite-horizon formulation. The author recommends the text (Bertsekas, 2000) for an insightful treatment of this topic.

The focus of this study will be on the discounted ($\alpha < 1$) infinite horizon problems with a bounded reward function, i.e. $g(\cdot, \cdot, \cdot) \leq M < \infty$. In this case, the decreasing geometric profile $\{\alpha^k M\}$ makes $J^\pi(i)$ well-defined for all π and i, thus a solution to Problem 1 always exists in this case. These problems constitute the second level of generality in the classification of MDPs and their solution methods are outlined in the following section.

The remaining two classes of MDPs are classified by the cases where there is no terminal state, $\alpha = 1$, and/or the reward $g(\cdot, \cdot, \cdot)$ is unbounded. These types of problems are out of the scope of this study.

Dynamic Programming

Dynamic Programming (DP) is a methodology to solve MDPs. It was manifested in (Bellman, 1957) as a numerical solver. The presentation begins with a case study analyzing a discrete-time but continuous-state optimal control problem, namely inventory control. Although the problem utilizes the common state representation, the material then extends the discussion and the algorithm's application field to general Markovian representations. Due to the mathematical rigor and the generality of the perspective achieved by Bellman in his text, DP is among the few engineering tools whose computational benefits and limitations have been known almost completely since their conception.

The application field of DP, on the other hand, has grown enormously in the last five decades. Modern approaches to DP are no longer confined to the classical control theory arguments, but rather evaluate it as a general class of heuristics that minimize non-beneficial attempts in the solution of a staged decision problem, thus saving resources and time. Along its 50 years of history, numerous problems have been formulated and solved within this framework. There is a significant diversity among real-life engineering problems, especially in the broad field of operations research, where the only reasonable solutions can be obtained by DP-based methods.

It can be argued from an algorithmic point of view that DP methods balance the computational burden and the memory requirement in the solution of a problem in an alternative way: DP provides generally a faster exact solution in contrast to other known heuristics, such as classical depth-first or breadth-first search strategies, at the expense of a larger amount of (dynamic) memory that also needs to be maintained. A typical example that demonstrates DP characteristics is the well-known Dijkstra algorithm (Baase and van Gelder, 1999), which computes shortest paths in a given graph. Along with its algorithmic aspect, there is a variety of interpretations of DP and a voluminous literature on the analysis and synthesis of its methods.

The key element of a DP-based solution procedure is the DP table, which contains *values* of the states already visited in the search space. A value is the unique measure of the benefit in including the associated state in the optimal trajectory, which would be generated by the optimal policy (i.e. the solution of MDP). The term *dynamic* refers to the updates on the DP table and the time-varying (usually growing) memory requirements to maintain this table along the solution, as the procedure explores all elements of the state space and ascertain their values.

The formal definition of value has already been made implicitly in the MDP formulation in (3):

Definition 5 *The value of state i in a MC following the policy π is evaluated by the value* *function $J^{\pi}(\cdot) : S \to \Re$ defined by*

$$J(i) \triangleq E\left\{ \sum_{k=0}^{\infty} \alpha^k \cdot g\left(x_k, \pi(x_k), x_{k+1}\right) \;\middle|\; x_0 = i \right\}. \tag{4}$$

Determining the value function is a critical sub-goal in the DP strategy. Once the value function is known for each π, the remainder of the solution turns out to be picking merely the policy π^* which maximizes $J^{\pi}(i)$ evaluated at the provided initial state i: arg $\max_p\{J^p(i) : p \in \Pi\}$, where Π is the (finite) set of all possible policies. Therefore, in many applications, DP is viewed only as a method for determining the value function only, although value iteration is only a part of its role in staged decision making.

Principle of Optimality

All DP strategies are based on the principle of optimality:

Definition 6 *Let π^* be an optimal policy for the MDP in (3) and assume that when using π^*, a given state $i \in S$ occurs at time $t > 0$ with positive probability. Consider the truncated version initiated at $k = t$:*

$$\arg\max_{\pi} E\left\{ \sum_{k=t}^{t_f-1} \alpha^k \cdot g\left(x_k, \pi(x_k), x_{k+1}\right) \;\middle|\; x_t = i \right\}. \tag{5}$$

Then, the principal of optimality holds if and only if the policy π^* is still an optimal policy for the truncated problem for all $1 \leq t \leq t_f$.

The notion of this principle is intuitively very sound. Humans apply this principle routinely in making daily-life inferences such as:

Since I-64 is the shortest route from Louisville to Lexington and since this route passes through Frankfort, then I-64 is also the shortest route between Frankfort and Lexington.

Using the definition (4) of $J^{\pi}(\cdot)$ directly, one obtains

$$\begin{aligned} J^{\pi}(i) &= E\left\{ \sum_{k=0}^{\infty} \alpha^k \cdot g\left(x_k, \pi(x_k), x_{k+1}\right) \middle| \; x_0 = i \right\} \\ &= E\left\{ g(x_0, \pi(x_0), x_1) + \sum_{k=1}^{\infty} \alpha^k \cdot g\left(x_k, \pi(x_k), x_{k+1}\right) \middle| \; x_0 = i \right\} \\ &= \sum_{j=1}^{N} p_{ij} \left[g(i, \pi(i), j) + E\left\{ \sum_{k=1}^{\infty} \alpha^k \cdot g\left(x_k, \pi(x_k), x_{k+1}\right) \middle| \; x_1 = j \right\} \right] \\ &= \sum_{j=1}^{N} p_{ij} \left[g(i, \pi(i), j) + \alpha J^{\pi}(j) \right], \end{aligned} \tag{6}$$

which relates the values of two consecutive states i and j in a given process. An immediate result that can be derived from (6) is that, if the values of all possible successors of a state i is known (or guessed), then the value of i can be exactly computed. Of course, the transition probabilities p_{ij} are critical in this calculation and they are yet assumed to be known in the reasoning.

Application of the principle of optimality on (6) then yields the crucial property of the *optimal* value function, namely $J^{\pi^*}(\cdot)$ is a solution of N linear equalities involving N unknowns:

$$J^{\pi^*}(i) = \sum_{j=1}^{N} \left[p_{ij} g(i, \pi(i), j) + \alpha J^{\pi^*}(j) \right] \; , \; i \in S = \{1, \ldots, N\} \tag{7}$$

called Bellman's equation. By adopting the matrix forms **P** and **G** for the transition probabilities and the reward function, respectively, they can be written in the compact form

$$\begin{bmatrix} J^{\pi^*}(1) \\ J^{\pi^*}(2) \\ \vdots \\ J^{\pi^*}(N) \end{bmatrix} = \operatorname{diag}\left(\mathbf{P}\mathbf{G}^T\right) + \alpha \mathbf{P} \begin{bmatrix} J^{\pi^*}(1) \\ J^{\pi^*}(2) \\ \vdots \\ J^{\pi^*}(N) \end{bmatrix}, \tag{8}$$

where $\operatorname{diag}(\cdot)$ functional gives the diagonal of its argument as a column vector.

The transition probabilities $\mathbf{P}$ driving the process and the rewards $\mathbf{G}$ are critical in this calculation. Once they are known, the solution to the evaluation problem is obtained by

$$\begin{bmatrix} J^{\pi^*}(1) & J^{\pi^*}(2) & \cdots & J^{\pi^*}(N) \end{bmatrix}^T = (\mathbf{I} - \alpha\mathbf{P})^{-1} \operatorname{diag}\left(\mathbf{P}\mathbf{G}^T\right). \tag{9}$$

And once the states are evaluated in this optimal way, the solution to the MDP is extracted by

$$\pi^*(i) \leftarrow \{a \in A : a \text{ maximizes the probability } p_{ij^*}\} \tag{10}$$

where j^* is the index of maximum element of $\begin{bmatrix} J^{\pi^*}(1) & J^{\pi^*}(2) & \cdots & J^{\pi^*}(N) \end{bmatrix}^T$. The resulting policy π^* is clearly the desired one as it maximizes the chance of switching to the most beneficial state, maximizing the cumulative reward (3).

There is a variety of DP algorithms, each leading to the solution (9) iteratively. Their common approach is starting by the initial guesses of state values and a particular initial policy. This corresponds to initializing the DP table by these guesses, possibly randomly. The adopted initial policy may also be random, or a best-guess one based on the prior information on the considered problem.

The solution procedure is a loop containing the two particular sub-procedures:

a. Value Iteration The following two steps constitute *the value iteration* sub-procedure of DP.

1. Apply the policy in effect and observe the next state and the incurred reward.

2. Using the relation (6) between the values of new and previous states, the value of the new state in the DP table is updated assuming that the value of the preceding state was correct.

Note that, in the light of the discussion above, performing these steps sufficiently many times on a fixed policy π is equivalent to numerically solving (6) for the exact values $J^{\pi}(\cdot)$.

b. Policy Iteration Note that any update on the DP table may result in an inconsistency between the current status of the DP table and the policy that is being applied, and the value iteration is no exception. In other words, the policy in effect may no longer yield the most beneficial states with respect to the current values after performing the two steps above for any number of times.

To achieve consistency between the policy in effect and the current content of the DP table, a *policy iteration* step is essential. This is achieved simply by applying (10) for each state, where, instead of J^{π^*}, one has now the current values J^{π^k} represented in the DP table:

$$\pi^{k+1}(i) \leftarrow \{a \in A : a \text{ maximizes the probability } p_{ij^*}\} \tag{11}$$

where j^* is the index of maximum element of the current DP table.

The point where DP algorithms differ is actually the point where they perform the policy iteration step. The Gauss-Seidel iteration (Bertsekas and Tsitsiklis, 1996) for instance performs policy update upon each value iteration step (i.e. right after Steps 2 above). Extracting the policy from the current DP table may alternatively be performed after a certain amount of repeating steps 1 and 2 within a loop. A third scheme may be iterating the policy upon encountering all states.

This interaction between the policy in effect and the values in the DP table continues until both of them convergence. Under mild assumptions, including $\alpha < 1$, all such procedures can be shown to be really converging along iterations to a policy which is consistent with the limit values in DP table, where the limit policy is the optimal policy: $\pi^k \to \pi^*$ as $k \to \infty$.

The general DP strategy outlined here constitutes the basis of the approximate methods, which will be utilized for the problems of the following chapters.

Limitations

Despite theoretical guarantees on their convergence to an optimal policy, the computational demands of DP solutions may be overwhelming. The obvious reason for this is that a DP algorithm handles the states individually and builds relations between their values. As a result, the number N of states is a direct parameter of the computational load: N real numbers in the DP table to be maintained, each requiring $N(N-1)$ additions (c.f. (6)) in the value iteration step. This is, by itself, a heavy burden for a DP algorithm and is especially a major problem when dealing with problems with continuous state-spaces, such as the ones of interest to us in this study. Such problems require a preprocessing which quantizes their states into a finite state space in order for DP methods be applicable. The density of this finite space is usually proportional to the accuracy of the approximate representation. This means that for accurate results by DP, more and more states should be augmented, which in turn may load the solution algorithm with unacceptably many computations.

Taking into account that the dimension of the state space is a logarithmic function of the number of states, linear growth in the dimension gives rise to exponential increase in the space requirements of any DP algorithm. This effect has been first named by Bellman as the curse of dimensionality. It constitutes the most serious obstacle for application DP methods; because many real-life decision problems can only be modeled on larger dimensional state-spaces than any available processor can accommodate to perform a DP solution in reasonable amount of time. It should be noted that the drastic improvement on processor speeds in the last few decades made accommodating many problems within this framework possible (Powell, 2005). Although the borders of the application field of DP algorithms seem to be drawn by the technology from this point of view, a certain technological level that would infallibly overcome the curse of dimensionality can never be given.

The second major problem of DP methods is that the process must be modeled by exact state transition probabilities. That is knowing the process model exactly is essential to carry out a conventional DP solution. With any amount of inaccuracy in representing these parameters, one can no longer guarantee the optimality of the resulting policy. Needless to say, the lack of information on some or all of these parameters is absolutely intolerable since the algorithm requires their explicit values in the value iteration step. As a result, conventional DP methods are certainly classified as model-based methods, which contradict with the point of view to the medical decision problems in this study.

These two issues, common to all DP solutions, can only be resolved by replacing two fundamental sub-procedures, namely value and policy iteration, by their approximate versions. As in any approximation, the benefits of this substitution come at the expense of settling with a sub-optimal policy instead of the exact solution π^*, which would really be ensured by the conventional DP solution.

Approximate Dynamic Programming

The role of approximate DP methods in solving MDP is interpreted as twofold in this study, as previously mentioned and described in the remainder of this chapter. Therefore the treatment of ADP in this study is broader than the views that consider ADP solely as a method of parameterizing the DP table to overcome the scaling problem. Though addressed rigorously in the following sections, such a parameterization constitutes a secondary goal for us that will even be disregarded in some of the demonstrations on the following chapters.

Instead, the focus here is mostly on addressing the problem of unknown (or partially known) transition probabilities here. This is viewed as the primary shortcoming of conventional DP methods, because obtaining the patient behavior (i.e. the process model) in a medical treatment may be too costly from many aspects. The medical decision maker thus faces the difficulty of extracting the process model (i.e. how the patient responds to medical decisions) simultaneously with achieving the therapeutic goal. He/she does not usually have a patient model to begin the control directly, and without a model conventional DP methods are simply inapplicable. This is a typical chicken and egg problem.

As will be shown shortly, a solution to this dilemma brings another form of approximation scheme into the big picture of staged decision making. It deserves an analysis still under the ADP topic, although this new approximator serves for another purpose than the standard view to ADP suggests.

Coping with Model Uncertainty within MDP

Making decisions on a MC with unknown transition probabilities and rewards is troublesome but common matter in critical tasks, including medical treatment processes. Moreover, in many of these instances, one has no access to the process (c.f. the black box phenomenon), which renders the simulation as the only method to gain information on the dynamics. In particular, the system must be probed by a set of inputs at certain states and the parameters of the dynamics must be determined based on the observed outcomes from the black box, i.e. state transitions. The goal here is to *explore* the system.

Reaching the optimal policy, on the other hand, is a serious control task, and a constant improvement in the state values, as guaranteed by the conventional DP algorithms, must be *exploited* to solve the problem. Since the two goals conflict in general, a standard approach in handling unknown (or partially known) processes is to isolate exploration from exploitation (Barto et al., 1995).[4] The former, also called the modeling process, is independent of the upcoming control efforts, and the sole objective is to represent the dynamics as accurate as possible. The latter comes in effect strictly after the modeling is over (or suspended) and during its operation, the obtained model on the preceding step is assumed to be perfectly accurate (i.e. certainty equivalence principle).

In some most critical instances, however, there is no time to perform exploration and

[4]This trade-off has deep roots in control theory. It is a fundamental problem of adaptive control (Astrom and Wittenmark, 1989; Kumar, 1985).

exploitation separately, thus it is further necessary to combine the two (usually conflicting) efforts. Again, most medical decisions are subject to this requirement, due to limitations on treatment time.

Simulative approximate dynamic programming methods combine simulative modeling (i.e. *learning* the model) with the continuous tendency to encountering advantageous states towards the objective (i.e. MDP solution). These methods include various forms of reinforcement learning (Sutton, 1988) that will shelter also the methods utilized/developed in the sequel. The unique feature utilized in these methods is their ability to maintain both goals simultaneously and reasonably accurately.

The viewpoint adopted in this study is augmented by computational intelligence components. The focus is specifically on the design and analysis of learning systems that maintain a particular DP method to solve such a decision problem, given only sequences of data generated by the dynamic process. Since the dynamical form governing the process is disregarded in this setting, the objective could be classified as *a model-free control task.*

The basic strategy common to the methods of interest here is to make use of the information gain obtained by the state transition $i \longrightarrow j$ and the corresponding reward $g\left(i, \pi(i), j\right)$ in updating the values in the DP table. In fact, such an observation contains a novelty as an aid in coping with the two unknowns, i.e. the value $J(i)$ and the transition probability p_{ij}.

Method of Temporal Differences

Being an alternative to the value iteration step of conventional DP algorithms, Temporal Difference (TD) methods, parameterized by $\lambda \in [0, 1]$ is an iterative way to calculate the state values under a specified policy in effect (Sutton, 1988).

TD iterations are applied on a sample trajectory generated by the actual process, whose transition probabilities are typically unknown. TD(λ) is an episodic (approximate) value iteration process in the sense that the TD iteration is repeated within a loop, for each state transition observed in the sample trajectory. When the sample trajectory is finished, the stage is left for a policy iteration procedure, where the policy in effect is improved by examining the resulting value function, which was prepared by the last TD iteration step. The learning proceeds with a new trajectory generated by the new policy.

A single iteration of TD performs partial correction on the current state values based on a comparison between the immediate reward observed and the values of the last and the current state. Note that these three quantities are strictly linked to each other by (6).

In the basic case of $\lambda = 0$, a backup toward the correction is performed by

$$J^{\pi}(x_k) \leftarrow J^{\pi}(x_k) + \delta[k]\left(g(x_k, \pi(x_k), x_{k+1}) + \alpha J^{\pi}(x_{k+1}) - J(x_k)\right), \qquad (12)$$

where $\delta[\cdot]$ is a positive learning rate sequence. Using a well-known result from the stochastic approximation theory (Poljak and Tsypkin, 1973), $\delta[k]$ is selected in this iteration (and in almost all variants of TD) as a diminishing step-size satisfying $\lim_{k\to\infty} \delta[k] \to 0$ and $\sum_{k=0}^{\infty} \delta[k] < \infty$.

For $\lambda > 0$, the immediate correction is propagated towards the values of future states, which will be examined later in the sample trajectory:

$$J^{\pi}(x_k) \leftarrow J^{\pi}(x_k) + \delta[k] \sum_{m=k}^{l-1} \lambda^{k-m}\left(g(x_m, \pi(x_m), x_{m+1}) + \alpha J^{\pi}(x_{m+1}) - J(x_m)\right), \quad (13)$$

where l is the episode length. The rationale behind introducing this propagation is that the values of the future states will effective in the current state's value due to the basic relation (6).

Validated by the convergence results (Sutton, 1988; Jaakkola et al., 1994), tabular TD methods constitute one of the most efficient ways of learning from trajectories. On the other hand, TD is a stochastic process that is biased on the selected samples, so the method may not be fast enough in deciding on the optimal policy to be applied in many critical applications.

For a parametric approximator $\phi(\cdot, \mathrm{r})$ utilized to represent the state values, TD(λ) iteration can be converted to the update rule of the parameter vector r as

$$\begin{aligned}\mathrm{r}[k+1] \quad = \quad & \mathrm{r}[k] + \delta[k] \nabla\phi(x_k, \mathrm{r}[k]) \sum_{m=0}^{l-1} \lambda^{k-m} \left(g(x_k, \pi(x_k), x_{k+1})\right. \\ & \left. + \alpha\phi(x_{k+1}, \mathrm{r}[k]) - \phi(x_k, \mathrm{r}[k])\right) \qquad (14)\end{aligned}$$

which is actually an incremental gradient update of r based upon the observation of $g(x_k, \pi(x_k), x_{k+1})$ (Bertsekas and Tsitsiklis, 1996).

The convergence of the last iteration has not been established for all approximation models, i.e. for all forms of $\phi(\cdot, \mathrm{r})$. In fact, it has been shown in (Tsitsiklis and van Roy, 1997) that it diverges for a particular closed-form nonlinear model. Available constructive results (Dayan, 1992; Bradtke and Barto, 1996; de Farias and van Roy, 2000; Boyan, 2002; Bertsekas et al., 2004) on the performance of TD with value function approximation are currently limited to the basic case of linear approximation of $J(\cdot)$.

The primary benefit of value function approximation is the reduced computational demand of maintaining a parameter set compared to the tabular TD. In fact, parametric approximation appears to be the unique opportunity to cope with the curse of dimensionality, the serious thread on applicability of tabular DP methods, as mentioned above.

Although the nature of TD does not allow for eliminating completely the adaptive nature of the procedure on the presented trajectory (as this data constitutes the sole information that needs to be presented sequentially to the method), it is indeed possible to reduce the expected time for the procedure to finalize the parameter updates on the approximation scheme (Bertsekas et al., 2004). Unlike the tabular methods, several state values are affected by the parameter update performed based on a single state transition. This should be viewed as a side-benefit of dynamically approximating the value function.

Value function approximation is the most popular way of incorporating connectionist schemes into this learning task. An approximation scheme employing an RBF network is suggested in the Chapter IV on a case study.

Policy Approximation

Another point where parametric approximation could aid ADP methods is the policy representation. In the original ADP setting, the policy is maintained in the form of a explicit lookup table relating states to the control actions. It should be noted that there are other approaches which take the concept of policy in different ways, such as probability distribution of actions given each state (Sutton and Barto, 1998). However, in any case, this table identifies an algebraic relation, which could really be implemented as an input-output relation of a parametric approximator.

Although such an approximator has nothing to do with the curse of dimensionality, there are still two important reasons to employ it: First, as in the case of value function approximation, the computational burden of updating a tabular policy (by policy iteration methods) grows exponentially with the dimension and cardinality of the state space. Second, the discrete nature of MDP results in a policy which is valid only on discrete points (states), and there may be a practical necessity to extend the policy to a continuum sampled by these points. In the latter case, the role of the connectionist tool integrated in the learning process goes beyond just approximation and turns out to be an interpolator of a general continuous control law as illustrated in Figure 2.

RBF networks have been proven to be effective tools for generalizing the control law (Sanner and Slotine, 1992). An RBF-based policy approximator is proposed also in Chapter IV.

Control-Oriented Variants

DP table is a novel and very useful tool, which is processed based on observations of the actual system behavior and then used indirectly to develop a control strategy. As

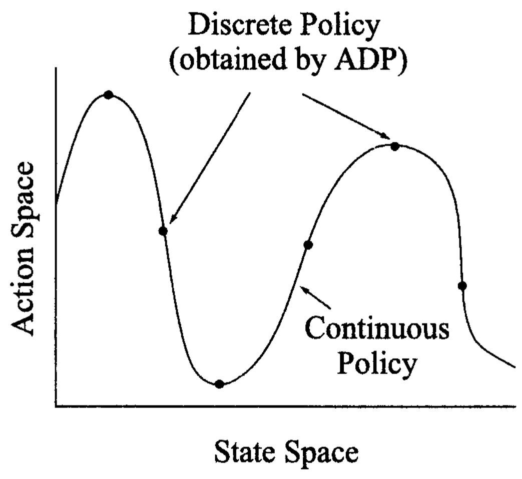

Figure 2. Policy interpolation.

mentioned in the MDP definition, correct evaluation of states according to (1) is equivalent to solving the problem. Having obtained the values of the states, it is straightforward to pick actions that yield the most advantageous states, constructing an optimal policy. However, even after correct evaluation, the trivial maximization of the state values over the control actions may be costly, especially when there are too many alternative actions to take.

At the cost of populating the DP table, it is possible to simplify this final maximization stage by evaluating each state-action pair, instead of leaving the effect of actions on the state values implicit in the state values, as done in (1). With this extension the DP table becomes a 2D array with entries pointing to the value of state-action pairs:

$$Q(s,a) \triangleq E\left[\sum_{k=0}^{\infty} \alpha^k g\left(x_k, a_k, x_{k+1}\right) \,\middle|\, x_0 = s, a_0 = a\right]. \tag{15}$$

The Q-values are then updated according to generalized versions of TD methods, called Q-learning methods. In particular, the generalized version of TD(λ) (c.f. (5)) to maintain the new DP table is given by

$$Q(s_k,a_k) \leftarrow Q(s_k,a_k)+\delta[k]\sum_{m=k}^{l-1} \lambda^{k-m}\left(g(s_m,a_m,s_{m+1}) + \alpha Q(s_{m+1},a_{m+1}) - Q(s_m,a_m)\right), \tag{16}$$

l being the episode length. The convergence of this iteration to real Q-values is established in (Watkins and Dayan, 1992; Tsitsiklis, 1994) under mild statistical assumptions.

Using Q-values instead of J's trivializes the policy iteration step as the best action for a state s is the index of the maximum element of the row indexed by s, which is evident in the Q-table. Since control actions are addressed explicitly in this formulation, Q value representation is in general preferable when dealing with control problems. To cope with medical decision problems, which are considered as control problems in this study, this particular view is adopted and the considered ADP methods are picked among Q-learning methods and their variations. The discussion made in the preceding section on approximation is still valid in this new learning scheme.

A critical issue in the choice of the suitable ADP tool is to decide whether to learn on-policy or off-policy. Off-policy methods enable learning by observing the effects of a policy other than the one processed, so that probing the process (i.e. the patient in medical problems) with inadequate policies may be avoided. This feature makes off-policy methods appropriate tools for medical applications considered in this study. On the other hand, on-policy methods are mathematically tractable and, hence, attract the majority of the research efforts in ADP.

As a final note, the TD and Q-learning iterations presented above constitute a general framework of many simulation-based control methods and is dependent only on the observed state transition and the resulting reward. Therefore, they do not actually point to a specific type of the learning method in the above terms. What categorizes a simulative ADP method as an on- or off-policy is the source of the policy being applied to generate sample trajectory, which is then processed by TD or Q-learning iterations.

CHAPTER III
ANEMIA MANAGEMENT PROBLEM

In this chapter, the problem of anemia management in patients with End-Stage Renal Disease (ESRD) is introduced as the test problem on which the methods of this study will be demonstrated.

Anemia due to End Stage Renal Disease (ESRD) is a common chronic condition in patients receiving hemodialysis (Eschbach and Adamson, 1985). The reason is known as the insufficiency of a hormone called erythropoietin (EPO), which stimulates the production of red blood cells (erythropoiesis). In fact, the hemoglobin (HGB) is the building block of these cells and its derivation in blood known to be correlated with that of EPO. The preferred treatment of renal anemia consists of external administration of recombinant human erythropoietin (rHuEPO, or EPO). Treatment process aims at maintaining the HgB level between 11 and 12 g/dL. This problem is formulated in the sequel as a MDP with unknown transition probabilities.

Individualization of Chronic Pharmacotherapy

Drug administration in chronic conditions is a recurrent trial and error process. Typically, a physician selects an initial drug dose based on a standard reference and observes the patient for specific response and/or side effects. Following the observed state of the patient, the dose is adjusted to improve the response and/or to minimize dangerous side effects. The adjustment continues until a desired response is achieved. Therefore, the physician can be viewed as an agent performing goal-oriented learning.

Oftentimes, the relationship between the drug dose and the patient's response is complex. Practitioners attempt to use protocols and algorithms to simplify this relationship. However, protocols and algorithms are developed from average responses to treatment in populations of patients. Individualization of drug dosing is complicated by the patient's response to the drug and to other concurrent medications.

The application of DP methods in pharmacotherapy has been advocated by Bellman (Bellman, 1983). A pioneering example for drug delivery optimization can be found in (Buell et al., 1970). Other examples include the works of Hu et al. (Hu et al., 1994b; Hu et al., 1994a). Very recently, Shaeffer et al. (Schaeffer et al., 2004) provided a review of modeling medical treatments using MDP. Moore et al. (Moore et al., 2004) demonstrated a successful application of the Q-learning algorithm to closed-loop control of patient sedation in an intensive care unit.

Anemia is an almost universal sequel in an ESRD patient. Until the introduction of recombinant human EPO, ESRD patients faced severe cardiovascular risk factors due to multiple transfusions. However, the use of EPO creates additional challenges to the physician. The National Kidney Foundation's Dialysis Outcomes Quality Initiative recommends maintaining HGB levels within a narrow range of $11 - 12$ g/dL. To achieve this, protocol-based strategies exist for EPO administration.

Based on the population response, these strategies adjust the dose amount or the dosing frequency based on the HGB level. The dosing of EPO is labor intensive and requires trained personnel to assess monthly HGB and iron levels and to make adjustments or assessments every two or four weeks. Having computational tools support the medical personnel in this difficult task would be a major step forward.

Problem Statement

The problem of anemia management for a given patient is a typical staged decision problem under uncertainty. The quantity to be kept under control is the HGB, whereas the control input is the amount of EPO administered by the physician. The iron stores in the patient, determined by Transferrin Saturation (TSAT), have an impact on the process of red blood cell creation and are considered as a state component together with HGB. In this setting, the patient is viewed as a discrete-time dynamic system with the state space $\mathcal{H} \times \mathcal{S}$, where $\mathcal{H}$ and $\mathcal{S}$ are sets of valid HGB and TSAT levels, respectively. The control space, i.e. the set of valid EPO amounts, is denoted by $\mathcal{E}$. As the measurements are performed monthly, the time index k in the state representation denotes the months.

Patient Model

In the classical pharmacological framework, a patient's response is analyzed using a PK/PD compartment model containing a set of differential equations. In the case of the red blood cell production, called erythropoiesis, regular measurement of EPO concentration would be required to acquire all the information necessary to build a PK/PD model. Due to the high cost of this procedure, alternative modeling methods, such as Artificial Neural Networks (Zurada, 1992), become a feasible option. In (Gaweda et al., 2003), a population-based neural network was proposed for dose-response modeling in anemia management.

For the purpose resembling a patient in generating simulated trajectories, a sub-population approach (Brier et al., 2006) is used to make up a patient. The underlying principle for this approach is the existence of several characteristic response groups within a patient population. Each one of these groups is assumed to bear a unique dose-response relationship. Using fuzzy rules, a patient's response is first classified and subsequently a one-step-ahead prediction of HGB level is obtained using the following second-order model:

$$\begin{aligned} x_1[k+1] &= \theta_1 a[k-1] + \theta_2 a[k] + \theta_3 a[k+1] + \\ &\quad \theta_4 x_1[k-1] + \theta_5 x_1[k] + \theta_6 x_2[k] + \theta_0 \end{aligned} \tag{17}$$

where a is the control input (EPO), x_1 is the HGB, and x_2 is the TSAT. The response is classified based on the six month average levels of HGB, TSAT, and EPO. The proposed approach can be conveniently implemented using Takagi-Sugeno (TS) fuzzy model (Takagi and Sugeno, 1985).

Records of 186 patients at the Division of Nephrology, University of Louisville, were used in this study to perform data-driven estimation of the TS model. The data were randomly divided into equally sized estimation (training) and evaluation (testing) sets, containing data of 93 patients each. For consistency, a total of 100 model estimations were performed using different patient selections for estimation and evaluation. Eventually, the following three-rule TS model was obtained:

$$\begin{aligned}
R_1 :\ & \text{If } (avg\ \mathrm{EPO}_{6\mathrm{m}}, target\ \mathrm{HGB}_{6\mathrm{m}}, norm\ \mathrm{TSAT}_{6\mathrm{m}}) \\
& \text{Then } \mathrm{HGB}[k+1] = \Theta_1\,\zeta \\
R_2 :\ & \text{If } (avg\ \mathrm{EPO}_{6\mathrm{m}}, target\ \mathrm{HGB}_{6\mathrm{m}}, low\ \mathrm{TSAT}_{6\mathrm{m}}) \\
& \text{Then } \mathrm{HGB}[k+1] = \Theta_2\,\zeta \\
R_3 :\ & \text{If } (high\ \mathrm{EPO}_{6\mathrm{m}}, low\ \mathrm{HGB}_{6\mathrm{m}}, low\ \mathrm{TSAT}_{6\mathrm{m}}) \\
& \text{Then } \mathrm{HGB}[k+1] = \Theta_3\,\zeta
\end{aligned}$$

In these rules, the subscript 6m denotes the six month average of the corresponding quantity, Θ_i are the parameter vectors of the predictive model (17), and ζ is the regressor vector:

$$\begin{aligned}
\zeta = [\,&\mathrm{EPO}[k-1],\ \mathrm{EPO}[k],\ \mathrm{EPO}[k+1], \\
&\mathrm{HGB}[k-1],\ \mathrm{HGB}[k],\ \mathrm{TSAT}[k],\ 1\,]
\end{aligned}$$

Two rules (R_1, R_2) specify the HGB response for *normal responders*, i.e. the patients who achieve target HGB levels upon administration of average EPO amount (ca. $12,000$ Units per week). These two rules cover normal responders with low and normal TSAT, respectively. The third rule (R_3) specifies the HGB response function for a group of patient, called *poor responders*. These are patients who receive high amounts of EPO yet their HGB level stays low. The reason for using fuzzy sets to represent the response groups is due to the fact that patients in real life exhibit features typical for both groups to a certain degree. In other words, only very few patients can be classified strictly as a normal or poor responder.

The state variable x_2 is assumed to be a random variable with normal distribution around the mean $\bar{x}_2$ and variance σ^2_{TSAT}. The random variation of TSAT emulates the uncertainty in the process dynamics. The main control objective is to drive the HGB level to and maintain within the target range $11 - 12$ g/dL. For simplicity, it is assumed that maintaining HGB within target range is equivalent to keeping it as close as possible to the median, i.e. 11.5 g/dL.

The author would like to emphasize at this point that the sole purpose of the patient model specified above is to generate realistic sequences of trajectory. In the learning scheme described below, the qualitative properties of this model are never taken into account by the learning system, which actually assumes that it has access to a real patient from its own perspective.

MDP Setting

The state space of the considered MDP is first reduced to 28 representative states by quantizing the HGB and TSAT intervals with medians fixed at $\mathcal{H} = \{5, 10, 11, 11.5, 12.33, 13, 15\}$ and $\mathcal{S} = \{10, 25, 40, 70\}$. The admissible set of discrete actions for each state is defined as $\mathcal{E} = \{0, 5, 10, \ldots, 60\}$. There is a number of alternatives on the choice of the reward function as defined next.

Having decided on the state and the action spaces, the second step in the formulation stage is to reflect the control objective, i.e. stabilization of x_1 at 11.5. The designer is absolutely free in choosing any reward formulation as long as it is consistent with the specified goal. This study considers two reward functions associated to the state transition $\mathbf{x}[k] \rightarrow \mathbf{x}[k+1]$.

The first one is chosen merely as the negative of absolute difference in the first state variable from the target level:

$$g_1\left(\mathbf{x}[k], \mathbf{x}[k+1]\right) = -\left|x_1[k+1] - 11.5\right|. \tag{18}$$

The second reward formulation is a finite-valued function defined by

$$g(\mathbf{x}_k, \mathbf{x}_{k+1}) = \begin{cases} -1 & , \quad 11.5 \geq x_1[k] > x_1[k+1] \\ & \quad \vee \;\; x_1[k+1] > x_1[k] \geq 11.5 \\ 0.5 & , \quad x_1[k+1] > 11.5 > x_1[k] \\ & \quad \vee \;\; x_1[k] > 11.5 > x_1[k+1] \\ 1 & , \quad x_1[k+1] = 11.5 \\ 0 & , \quad \text{otherwise} \end{cases} \tag{19}$$

Although the general ADP framework permits, these choices does not include the action value $a[k]$ as a parameter of the reward.[5]

The task of drug dosing as an MDM is then posed as follows: For all possible initial conditions $\mathbf{x}[0] \in \mathcal{H} \times \mathcal{S}$, determine the best control (action) $a[0]$ such that the expected cumulative discounted reward

$$Q\left(\mathbf{x}[0], a[0]\right) = E\left[\sum_{k=0}^{\infty} \alpha^k g\left(\mathbf{x}[k], \mathbf{x}[k+1]\right) \,\middle|\, a[0]\right] \tag{20}$$

is maximized, where $0 < \alpha < 1$ is the discount factor.

[5]Omitting the drug dose from reward formulation implies that the resulting learning mechanism will not demonstrate any tendency of minimizing the drug dose, although this tendency may be obvious in real clinical practice.

CHAPTER IV
LEARNING THE DOSING POLICY

Reinforcement Learning (RL) is a computational framework under ADP that mimics the goal-oriented knowledge and skill acquisition in humans and animals. Within this framework, an agent interacts with its environment by performing actions and observing the states. Based on the state transitions, each action is rewarded or penalized, conveying a critic to the agent on the immediate benefit of its preceding decision. Approaching the drug delivery problem from this point of view, the physician can be viewed as the agent, drug dose - as the action, patient - as the environment, and patient's response - as the state. Several authors have advocated the use of techniques underlying the RL to medical decision making (Bellman, 1983), (Hu et al., 1994b), (Hu et al., 1994a) over the years. However, applications of RL to drug administration have surfaced in the literature only very recently (Moore et al., 2004).

This chapter presents two conventional RL applied on the drug-dosing problem in order to extract the optimal dosing policy for individuals without any specified dose-response model. In the first approach, beginning with a common-sense policy, the patient is simulated along episodes, each demonstrating the policy in effect on the unknown patient model. The current policy is improved by using the gained experience at the end of each simulation episode. This is a training-oriented approach which emphasizes *exploration*. The second approach presented here is an application of the conventional Q-learning strategy (Watkins and Dayan, 1992) on anemia management, which explores and controls concurrently. Here, the policy improvements occur at each state transition, so the policy is continuously changing. Therefore, there is no particular policy in effect that is being fully demonstrated on the patient. It is shown that, when there is a time constraint on the treatment process, which is naturally the case in clinical practice, the latter method performs as well as the Anemia Management Protocol, the standard dosing procedure of Kidney Diseases Program, University of Louisville.

Episodic On-Policy Learning of Dosing Policy

The first attack toward the problem has adopted episodic and on-policy interpretation of the drug-dosing process, where a learning episode includes a trajectory simulated with a random initial state and by using a particular policy. Here gaining experience on the model (by examining the state transitions) and improving the policy (by TD iterations) are separate and sequential subtasks in a loop. In particular, the SARSA(λ) algorithm, an episodic on-policy ADP process, is employed here to develop a drug dosing policy for an individual patient. In this setting, the learning occurs by reflecting the experience gained within an episode to improve the policy in effect. Such a learning scheme offers a better monitoring of the improvement in policy along the proposed approach.

SARSA Algorithm

The learning progresses along the episode by evaluating each transition observed within the episode, the incurred immediate reward, and then by correcting the Q values of these transitions. In particular, it can be shown that the quantity

$$\delta[k] = g_1(\mathrm{x}[k], \mathrm{x}[k+1]) + \gamma Q(\mathrm{x}[k+1], a[k+1]) - Q(\mathrm{x}[k], a[k]) \tag{21}$$

associated to the state transition $\mathrm{x}[k] \rightarrow \mathrm{x}[k+1]$ due to the action $a[k]$ is a correction on the estimate $Q(\mathrm{x}[k], a[k])$ of the state/action pair $(\mathrm{x}[k], a[k])$. For each transition $\mathrm{x}[k] \rightarrow \mathrm{x}[k+1]$ encountered in an episode due to $a[k]$, the SARSA(λ) algorithm performs the update

$$Q(\mathrm{x}[k], a[k]) \leftarrow Q(\mathrm{x}[k], a[k]) + \nu\delta(1 + e(\mathrm{x}[k], a[k])), \tag{22}$$

where ν is a sufficiently small learning rate and $e(\mathbf{x}[k], a[k]) \geq 0$ denotes the eligibility of the state/action pair $(\mathbf{x}[k], a[k])$ in this correction. After this correction, before proceeding with the next transition, eligibility of the current state/action pair is first updated as

$$e(\mathbf{x}[k], a[k]) \leftarrow 1 + e(\mathbf{x}[k], a[k]) \tag{23}$$

and then the entire e table is iterated as

$$e \leftarrow \nu \lambda e, \tag{24}$$

where $\lambda \in [0, 1]$ is a parameter of the algorithm (c.f. (13)). When λ is small the state/action pairs looses rapidly their eligibilities to update the Q entries. So the frequency of encountering a particular state/action pair in the trajectory becomes a less important effect in the update of the associated Q entry. For $\lambda = 1$, all encountered state/action pairs are treated equi-eligible in the update of Q table. Note that the introduction of e provides an efficient implementation of the summation appearing in the original temporal difference definition (13).

After the Q and e updates for each state transition observed in the sample trajectory, the final step performed by the algorithm to complete the episode is the update of the policy based on the resulting Q table:

$$p(\mathbf{x}) = \arg \max_{a \in \mathcal{E}} Q(\mathbf{x}, a). \tag{25}$$

This particular policy determined merely as the maximum element of Q is called the greedy policy.

In order to apply SARSA(λ) on the drug dosing problem, the initial trajectory (of the first episode) of length 24 is generated using the linear model (17), where p is computed using the first six measurements from the considered patient. The corresponding reward sequence is then obtained by evaluating the reward function for each state transition observed in the generated trajectory. Note that there are 28 states and 13 possible actions for each state, so the Q-table has 364 entries.

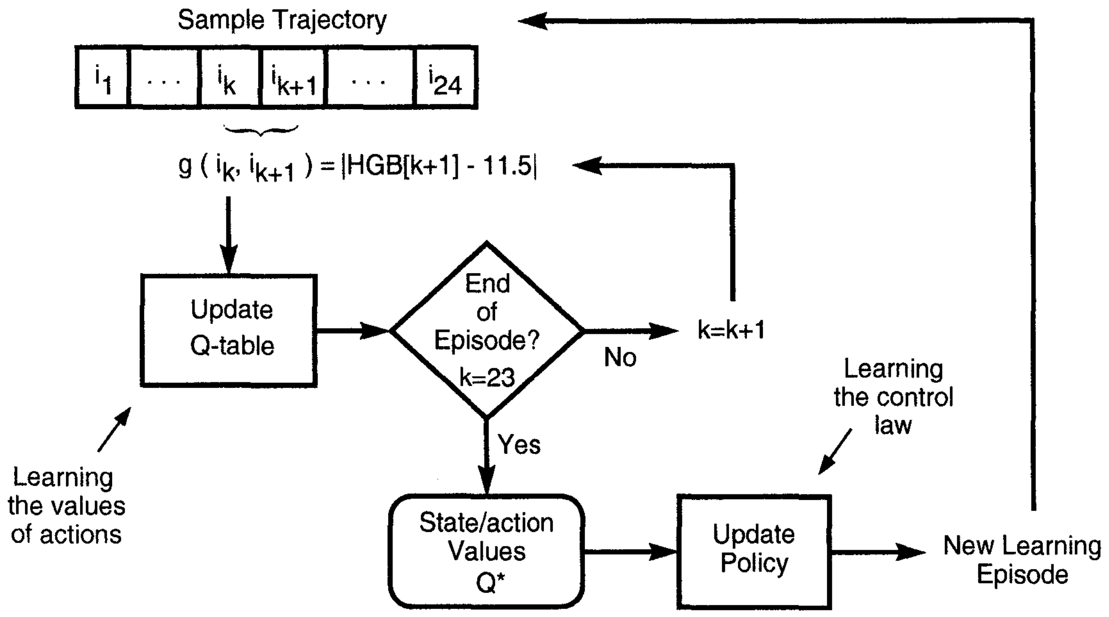

Figure 3. Proposed episodic anemia management scheme employing SARSA.

For diminishing learning constant ν and $\lambda \in [0, 1]$, the iteration on the policy performed at the end of the episodes based on the generated Q-table converges to an optimal policy, where the algorithm terminates, provided that all state/action pairs are visited frequently enough (Sutton and Barto, 1998).

The block diagram of the learning scheme proposed in this section is illustrated in Figure 3.

Approximating Q-table using RBF Network

Storing and updating the values of all possible state/action pairs explicitly in the Q-table necessitate exponentially larger amounts of memory and computation power as the cardinality of the state space expands. In such cases, a compact parametric representation of the Q values in the drug dosing problem turns out to be essential. Note that, the required approximation would not be in the traditional form of interpolating a finite set of static data, but rather constitute a representation that adapts itself incrementally to the updates performed by the ADP procedure upon each state transition.

Approximating the dynamic programming table using artificial neural networks has been proven to be an effective way of handling large decision making problems (Bertsekas and Tsitsiklis, 1996). Although it is computationally feasible to maintain explicitly the Q array in this problem setting with 364 entries in the setting above, in order to shed light onto the expanded versions of the considered problem, a connectionist approximation scheme is incorporated with the original SARSA method.

An RBF-based approximation scheme is considered here, because such algebraic networks enable localized parameterizations (Park and Sandberg, 1991) in the sense that each RBF node in the network and its parameters are related to the approximation performance on a particular sub-region of the input space only. This would be a useful feature in updating only the Q values of the observed state/action pairs along the SARSA iteration, without modifying the values of irrelevant pairs.

In the approximation scheme proposed here, the finite state space augmented by the action space ($\mathcal{H} \times \mathcal{S} \times \mathcal{E}$) is first partitioned into ℓ nonempty partitions and then the representative state for each sub-region is determined. Such a partitioning could be achieved effectively by a clustering procedure, such as the k-means algorithm (Bishop, 1995). Then each of these representative states is assigned as the center of a Gaussian RBF node. The widths of the RBF nodes are fixed. The considered RBF network here has 3 inputs, namely $x_1[k]$, $x_2[k]$, and $a[k]$. The output layer consists of a single linear unit with the real weight vector $\mathbf{w}$. The algebraic function implemented by the network is given by

$$\varphi(\mathbf{z}) = \mathbf{w}^T \begin{bmatrix} \exp\left(-\frac{\|\mathbf{c}_1 - \mathbf{z}\|_2^2}{2\sigma_1^2}\right) \\ \vdots \\ \exp\left(-\frac{\|\mathbf{c}_\ell - \mathbf{z}\|_2^2}{2\sigma_\ell^2}\right) \end{bmatrix} \tag{26}$$

where $\mathbf{c}_i$ and σ_i^2 are the center and the width parameter of the i-th RBF node, respectively.

The output weights are adjusted in compliance with the original Q-update mechanism (22):

$$w^* \leftarrow w^* + \nu\delta(1 + e(\mathbf{x}[k], a[k])) \tag{27}$$

where w^* is the output weight parameter from the RBF node whose center is closest to the observed state/action pair $(\mathbf{x}[k], a[k])$.

Experimental Results

To perform an experimental evaluation of the proposed method, an artificial group of 200 patients was used. Out of this group, the first 100 were typical for normal responders, while the remaining 100 were typical for poor responders. For each individual patient, a trajectory of EPO, TSAT, and HGB was generated over 6 months.

Experiment 1 A normal responder is considered in this experiment. After setting the Q-table and the eligibility matrix to zero and picking a random initial policy, a trajectory of length 30 was generated using the TS model described in Section III.C. The variance of TSAT was $\sigma^2_{\mathrm{SAT}} = 100$. It is important to note that there is no indication about the patient's membership in the response group other than the first six entries of EPO, TSAT, and HGB, so the patient model is absolutely unknown to the learning system.

The SARSA(λ) procedure was then applied as described above with $\lambda = 0.1$, $\gamma = 0.9$, and $\nu = 0.99$. Following each Q-update along the trajectory, ν was multiplied by 0.9. The state variable x_1 settled within the 5% band of the target value 11.5 after 5 episodes. The Euclidean norm of the difference between the actual HGB level and the target value 11.5 for each episode are shown Figure 4. This figure also presents the variability of HGB level, expressed in terms of standard deviation.

The resulting policy was finally on a normal responder. The amounts of EPO applied according to the policy and the resulting HGB level are shown in Figure 5. The effectiveness of the policy obtained by SARSA iteration is evident.

Experiment 2 The subject in this experiment was a patient whose HGB level does not

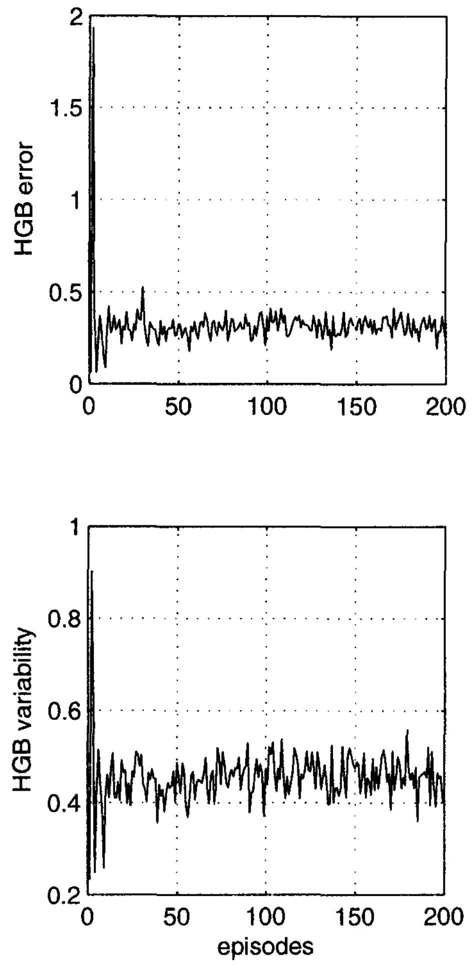

Figure 4. Variation of HGB level from the target 11.5 g/dL (top) and variability of HGB level (bottom) on simulated trajectories of a normal responder after 200 episodes of SARSA procedure.

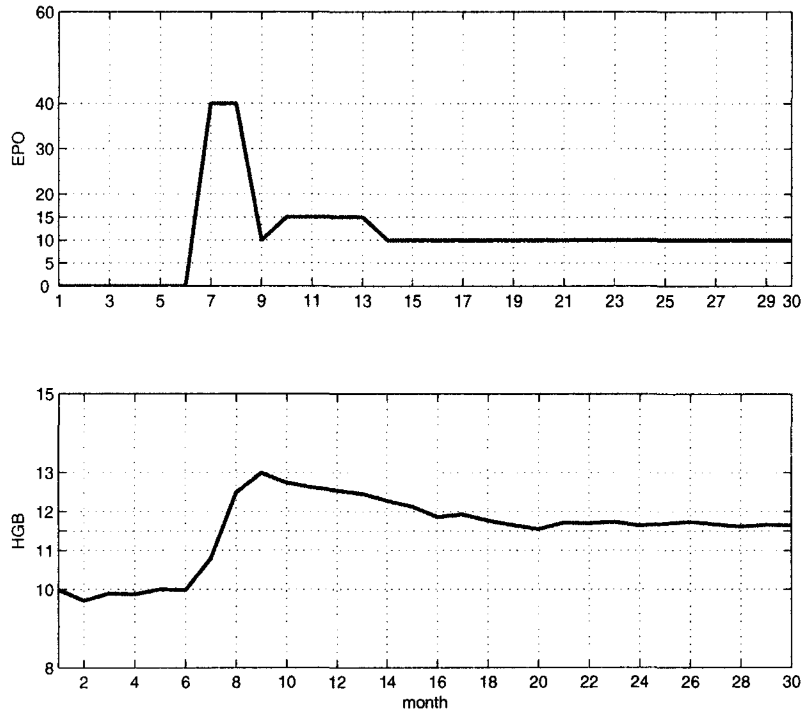

Figure 5. The sequence of administered EPO doses (top) and the resulting HGB trajectory (bottom) on the normal responder policy.

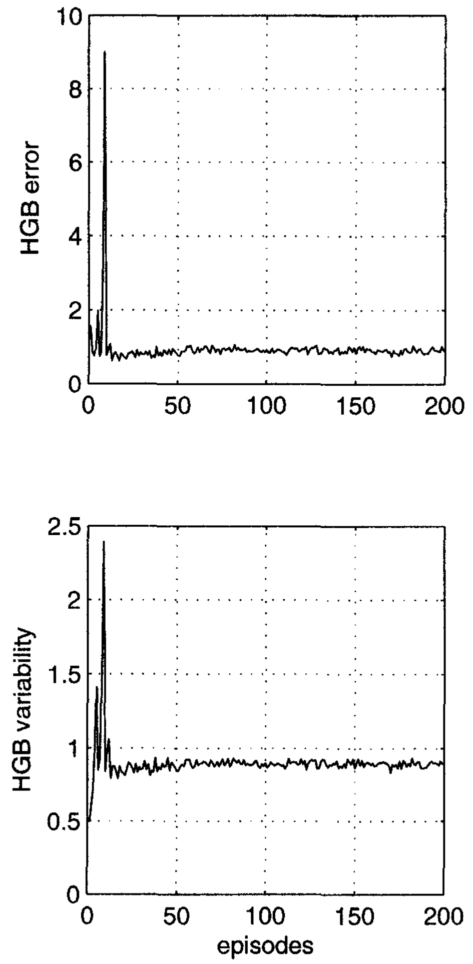

Figure 6. Variation of HGB level from the target 11.5 g/dL (top) and variability of HGB level (bottom) on simulated trajectories of a poor responder after 200 episodes of SARSA procedure.

adequately change in response to the EPO applied. Such a patient is classified as a poor responder. Experiment 1 was repeated for this patient with the same σ^2_{TSAT}, γ, ν, and λ values. The difference between the actual HGB level and the target value over 200 episodes and the HGB variability are shown in Figure 6.

The resulting policy was tested on a poor responder in a similar vein as it was done for the normal responder. The EPO doses administered to the patient following the policy and the corresponding HGB levels are shown in Figure 7.

The magnitude of the action sequence a (i.e. amount of EPO) needed to drive the

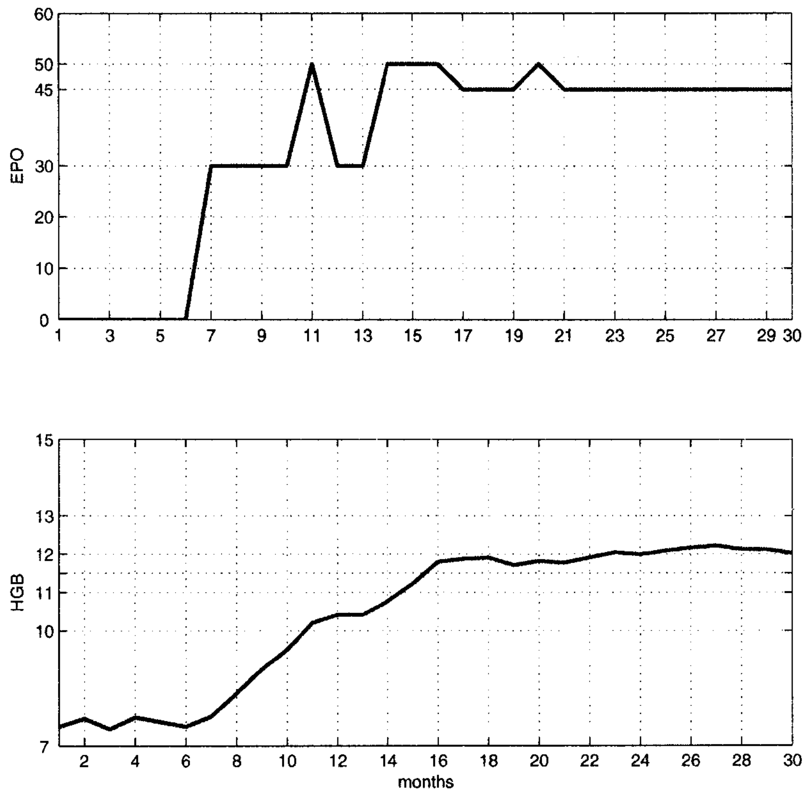

Figure 7. The sequence of administered EPO doses (top) and the resulting HGB trajectory (bottom) on the poor responder policy.

Table 1

Simulation Statistics of SARSA Learning

Responders	HGB level (g/dL)	HGB variability	EPO dose (1,000 U)	EPO variability
Normal	11.45 ± 0.36	0.44 ± 0.12	14.7 ± 4.5	7.6 ± 2.0
Poor	11.42 ± 0.66	0.91 ± 0.27	43.6 ± 5.8	7.5 ± 6.5

HGB level to the target range and to fix it there was expected to be higher than that of a normal responder. This phenomenon can be verified from Figures 5 and 7 by comparing the administered EPO amounts.

The results of the two experiments given above show that the original SARSA(λ) algorithm can be used to adjust the drug dose in both types of responders.

Experiment 3 Having established the ability of the proposed method to discover individual EPO dosing policies, a series 200 simulations over the whole population of artificially generated normal and poor responders was performed, where the same SARSA parameters as in Experiments 1 and 2 were used. The results of this simulation are summarized in Table 1. The HGB level and variability (defined as Standard Deviation) are in g/dL, whereas the EPO dose and variability are in $1,000$ Units per week. The entries in the table are mean $\pm$ standard deviation (computed over 100 individuals within each response group)

The results summarized in Table 1 confirm the findings of the first two experiments. The mean HGB levels close to 11.5 for both groups, as well as a relatively low HGB variability show that the proposed method consistently generates adequate EPO administration policies for both types of responders.

The RBF-based approximation scheme proposed in Section IV.B.1 was finally applied to parameterize the Q-table. The first entry of the RBF centers is associated with the HGB level (x_1) and the second one to the EPO (a). In this simulation, the TSAT state component was discarded, because the policy entries were mostly independent of the TSAT variable. The number ℓ of RBF nodes was then picked as 91 and their centers were assigned to the HGB/EPO pairs in $\mathcal{H} \times \mathcal{E}$. Note that neglecting TSAT value in the determination centers is equivalent to assigning all states with the same HGB and EPO values to the same point in the input space. (There are $|\mathcal{S}| = 4$ such states for each HGB and EPO in this setting). The width parameters $\sigma_1, \ldots, \sigma_{91}$ have been set equal to 0.01 and initial weights to zero. The RBF-based SARSA algorithm was performed by replacing (22) with (27) to the individuals considered in Experiments 1 and 2.

The Euclidean norm of the difference between the policies obtained by the original SARSA (pol_Q) and its approximate version (pol_{RBF}) at each episode is shown in Figure 8. This plot was generated for the normal responder and the poor responder policy exhibited almost identical convergence. As can be observed, the approximate version of the SARSA algorithm converges to a policy similar to the one obtained by the tabular algorithm.

Off-policy Learning of Dosing Policy

To demonstrate an off-policy learning tool on the test-bed, the control-oriented ADP method introduced in Section 2.C.5 is utilized, where gaining experience and improving the policy are considered as integrated subtasks to be achieved simultaneously. The learning occurs in the form of immediate improvements (i.e. Q-value updates) reflected to the policy due to the experience gained by observing the HGB and EPO sequences of an individual patient. As delineated below, the learning system is particularly responsible for calculating optimum actions (EPO in $1,000$ Unit steps) to be taken at each representative HGB level. The amount of EPO to be applied is then calculated based on the actual (non-quantized) HGB level of the patient by an RBF network, which interpolates now the drug dosing policy

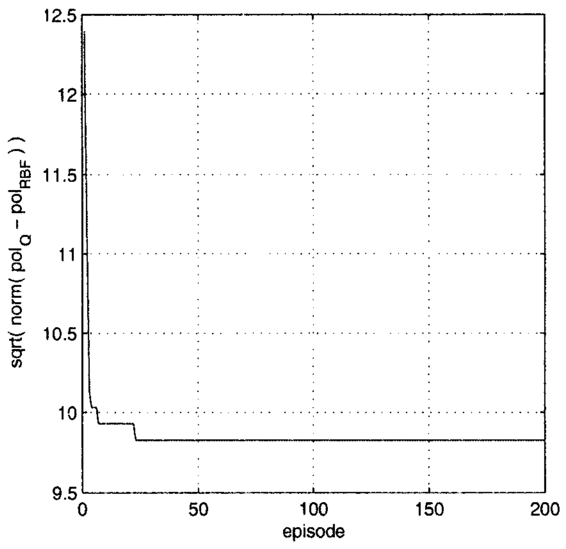

Figure 8. The difference between the two policies obtained by plain SARSA and its approximated version using RBF network.

from the finite samples produced by the learning system.

Q-Learning System

Unlike the MDP formulation adapted for episodic learning scheme of the preceding section, the patient dynamics (c.f. Section III.C) is considered as operating on a continuous state-space, $\mathcal{H} = [9, 15]$, $\mathcal{S} = [0, 40]$, $\mathcal{E} = [0, 60]$, but the quantities presented to the learning system are quantized to some finite representative values denoted by $\hat{\mathcal{H}}$, $\hat{\mathcal{E}}$. The learning system observes the quantized current state (HGB level) $\hat{x}_1[k]$, the quantized action (EPO dose) $\hat{a}[k]$, and the quantized successor state $\hat{x}_1[k+1]$. Given this information, the rewarding mechanism embedded in the learning system evaluates $g(\hat{x}_1[k], \hat{x}_1[k+1])$ based on the immediate contribution of the current state transition toward the control goal. The finite-valued reward formula given by (19) is used in the considered learning task. The proposed control scheme is illustrated in Figure 9.

The update equations in this form of Q-learning is the same as (21)-(24). The

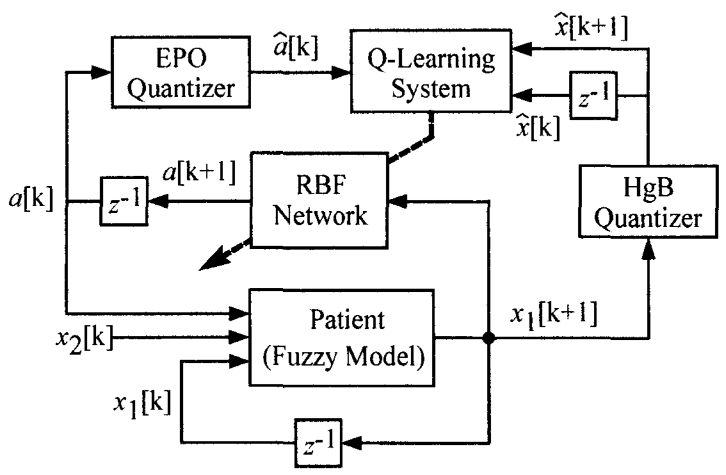

Figure 9. Block diagram of the Q-learning-based decision process.

action $\hat{a}[k]$ is then related to the current quantized state $\hat{x}_1[k]$ through the algebraic policy $p(\cdot) : \hat{\mathcal{H}} \to \hat{\mathcal{E}}$. The learning scheme employs a slightly modified version of the greedy policy (25), namely ϵ-greedy policy defined by

$$p_\epsilon(\hat{x}) = \begin{cases} \arg\max_{a\in\mathcal{E}} Q(\hat{x}, a) & , \quad z > \epsilon \\ \text{an arbitrary element of } \mathcal{E} & , \quad \text{otherwise} \end{cases} , \tag{28}$$

where z is a random variable distributed uniformly within $[0, 1]$, and $\epsilon \in [0, 1]$ is a parameter of the learning algorithm. Note that the policy updated in this way contributes to the exploration effort toward the optimal policy in a different way than the dynamic uncertainty x_2 does.

The fundamental difference of this learning scheme from the episodic method of the preceding section is that the policy update is performed immediately after the Q-table update. Hence, at any given time, there is no prescribed policy in effect, which makes the learning off-policy.

RBF Policy Network

Since the resulting policy (28) obtained by the $Q(\lambda)$ algorithm is valid only for the quantized states $\hat{x} \in \hat{\mathcal{H}}$, in order to be applicable to the considered patient model, $p(\cdot)$ needs to be generalized to the cover $\mathcal{H}$ by means of an interpolator.

An RBF network with $|\hat{\mathcal{H}}|$ Gaussian RBF nodes centered at the representative states is proposed in this study. The actual EPO dose to be applied for a given HGB level $x_1[k]$ is then determined by this policy network as

$$a[k] = \left[p(s_1) \ \cdots \ p(s_{|\hat{\mathcal{H}}|})\right] \cdot \begin{bmatrix} \exp\left(-\frac{x_1[k]-s_1}{\sigma}\right) \\ \vdots \\ \exp\left(-\frac{x_1[k]-s_{|\hat{\mathcal{H}}|}}{\sigma}\right), \end{bmatrix} \tag{29}$$

where $\{s_i\}_{i=1}^{|\hat{\mathcal{H}}|}$, are the representative states, i.e. the elements of the quantized state space $\hat{\mathcal{H}}$.

Since the representative states are equally spaced in $\mathcal{H}$, the spreads σ of the Gaussian nodes can be picked as

$$\sigma = \frac{d}{2},$$

so that the outputs of all nodes add up to approximately 1 for all points in $\mathcal{H}$, where d is the distance between two consecutive representative HGB levels in $\hat{\mathcal{H}}$. This enables assigning valid degrees of membership to the representative levels, so that the network gives an acceptable EPO dose in the form of the weighted sum of the actions imposed by the discrete policy. In this way, the RBF network plays a critical role by implementing the algebraic policy in the proposed drug dosing scheme.

Experimental Results

To perform an experimental evaluation of the proposed method, an artificial group of 200 patients was created. Out of this group, the first 100 were typical for normal responders, while the remaining 100 were typical for poor responders. As was done in the testing of episodic learning, a trajectory of EPO, TSAT, and HGB was generated over 6 months for each individual patient.

Five years of anemia management was simulated for each patient group using the following two methods:

- Q-learning with RBF Policy Network
- Anemia Management Protocol (AMP)

Anemia Management Protocol is a numerical implementation of an EPO administration protocol which is currently used at the Division of Nephrology. This last simulation was performed to establish a *gold standard* to which the results obtained by Q-learning can be compared. It must be pointed out that the AMP uses a mechanism for determination of EPO dose which is quite different and more involved than the one used in the Q-learning based scheme. Furthermore, the administration strategy implemented by AMP is fixed *a priori*, as opposed to the one used in Q-learning, which evolves in time. The dose selection procedure, as implemented in AMP, can be shortly described by the following expression:

$$\begin{aligned}\Delta\text{EPO}\,[k] &= F\,[\text{HGB}[k-1],\ \text{HGB}[k-2],\\ &\qquad \text{HGB}[k-3],\ \text{EPO}[k-1]\,]\end{aligned}$$

This is a higher order dynamic system, as opposed to a simple algebraic policy representation in ADP.

To apply Q-learning, the state (HGB) was quantized into 5 equally sized intervals with medians at: $9.5, 10.5, 11.5, 12.5, 13.5$ g/dL. These four values constituted the finite set $\hat{\mathcal{H}}$ explained above. The finite action set used by the learning system was set as $\hat{\mathcal{E}} = \{0,\ 5,\ 10,\ \ldots,\ 60\,\}$. Due to clinical constraints, the action values in $\mathcal{E}$ were also rounded to the nearest integer.[6]

[6]$1,000$ EPO Units is the lowest dose increment currently used.

In each simulation, the treatment was started at the seventh month. The Q-table was initialized using a best guess method such that the most viable policy was used in the first step. The best guess policy used in the simulations was as follows:

HGB (g/dL)	9.5	10.5	11.5	12.5	13.5
EPO (1,000 U.)	60	30	15	5	0

When a new patient comes in, they cannot be classified immediately to a response group. This information is obtained as the treatment progresses. Consequently, using one sound and common initial policy and tailoring it for the individual patient during treatment is a viable solution. Thus, the same initial policy was used for both response groups. Furthermore, all updates to the policy entries at the extreme states 9.5 and 13.5 were inhibited. The motivation underlying this decision is that when HGB reaches a dangerously low level, the maximum EPO dose 60 is the only feasible action. On the other hand, when HGB level is too high, EPO should be withheld. These extreme states are undesirable and it is expected that the system avoids them.

Due to the nature of the problem, the policy exploration was limited to time instances when the system was visiting the target state. In other words, ϵ is nonzero only when $11.0 \leq x_1 \leq 12.0$. In this case, the exploration probed how decreasing EPO affects the patient's response. Such an exploration aims at minimizing the patient exposition to the drug, as well as the total EPO administered.

In the simulation involving the Q-learning procedure, λ was 0.1, the diminishing learning rate was $\nu = 1/k$, the discount factor was $\gamma = 0.9$, and the exploration parameter was $\epsilon = 0.3$ when the system encountered the target states and $\epsilon = 0$, otherwise. The spreads of the RBF nodes were picked as $\sigma = 0.5$.

Figures 10 and 11 show the progress of anemia management for a selected representative normal responder. The top plots in each figure depict the HGB trajectory obtained as a result of administering EPO as shown in the plots second from the top. As an indicator of convergence of the Q-learning process, the third plot from the top of Fig. 10 presents

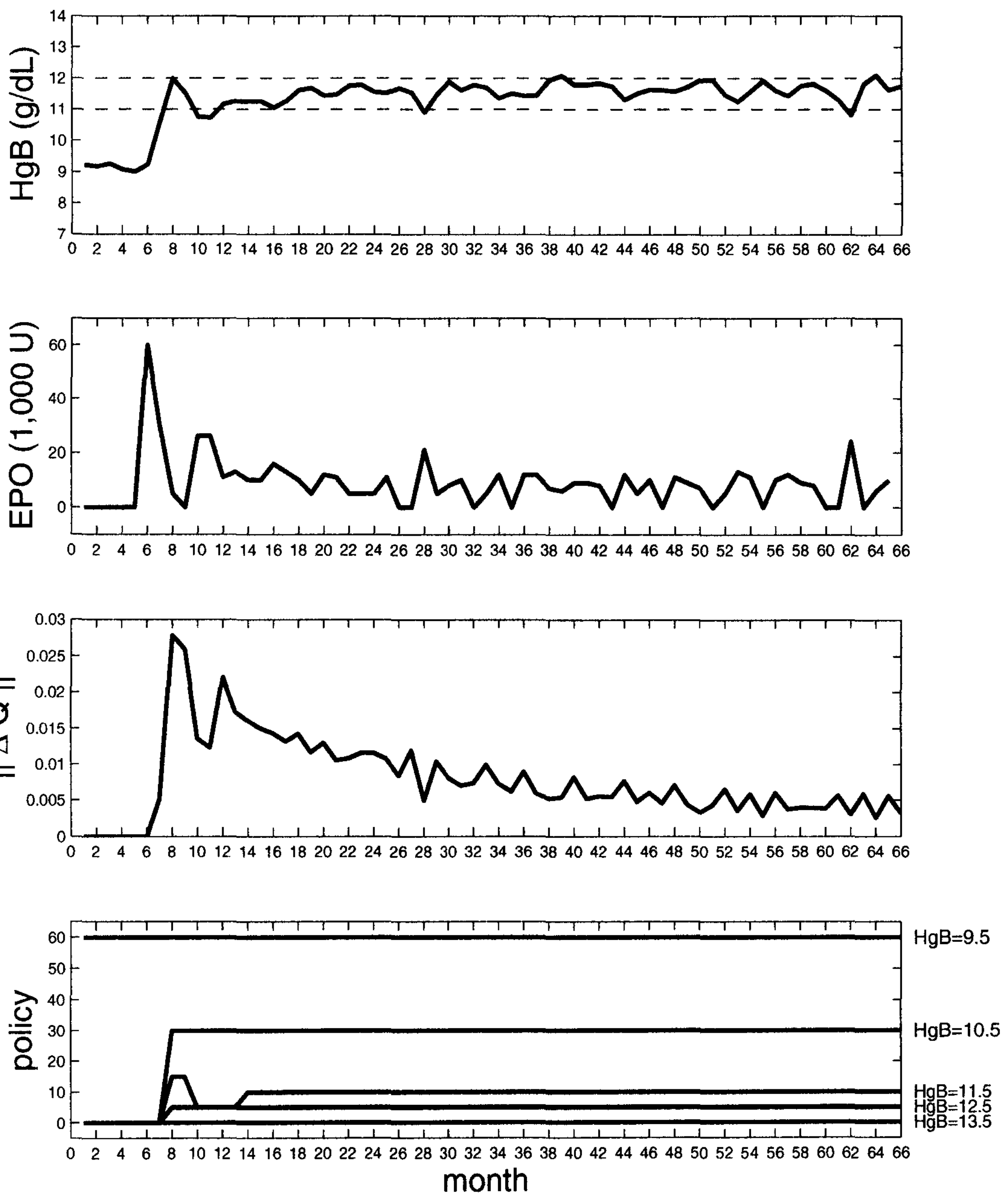

Figure 10. HGB level (top), EPO dose (second from top), the magnitude of the Q-table updates (third from the top), and policy evolution (bottom) for an individual normal responder as performed by Q-learning with RBF Policy Network.

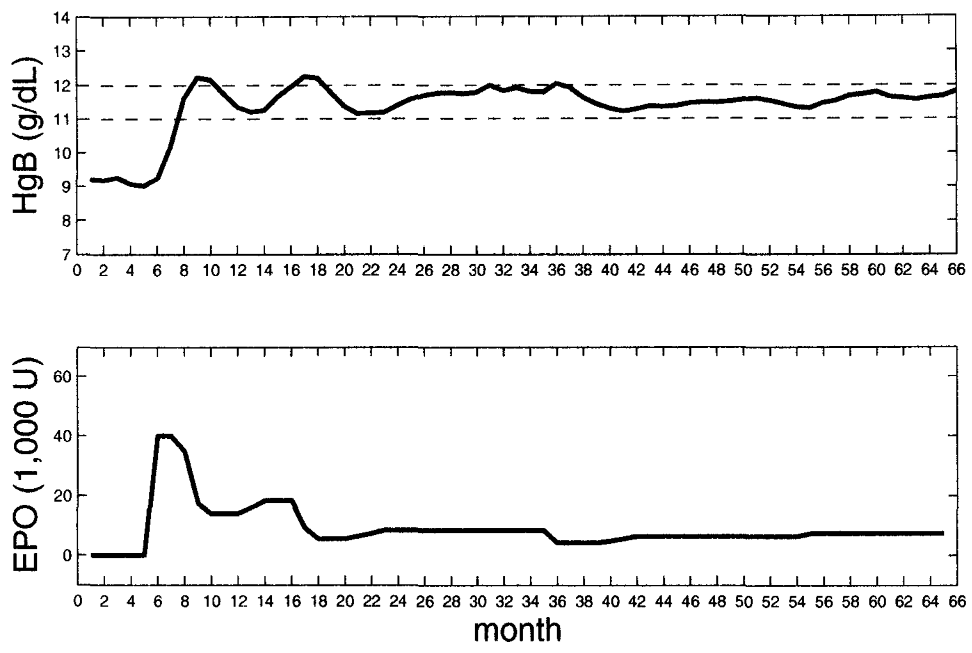

Figure 11. HGB level (top) and EPO dose (bottom) for an individual normal responder as performed by AMP.

the maximum norm of the deviation in the Q-table along the treatment. The bottom plot in Figure 10 shows the policy evolution. Each curve in the policy plot represents an action for the corresponding state as marked to the right of the plot. By analyzing the HGB trajectories, it can be concluded that Q-learning achieves the therapeutic goal.

This observation is also confirmed in Tables 2 and 3, where the statistics of the simulation are presented in terms of the mean value and the 95% confidence interval calculated over 100 patients for the following outcome measures:

- mean HGB level over the treatment period,
- standard deviation of HGB over the treatment period,
- total EPO used during the treatment.

The simulation statistics presented in these tables for normal responders show no significant clinical differences in terms of quality of anemia management between the two methods.

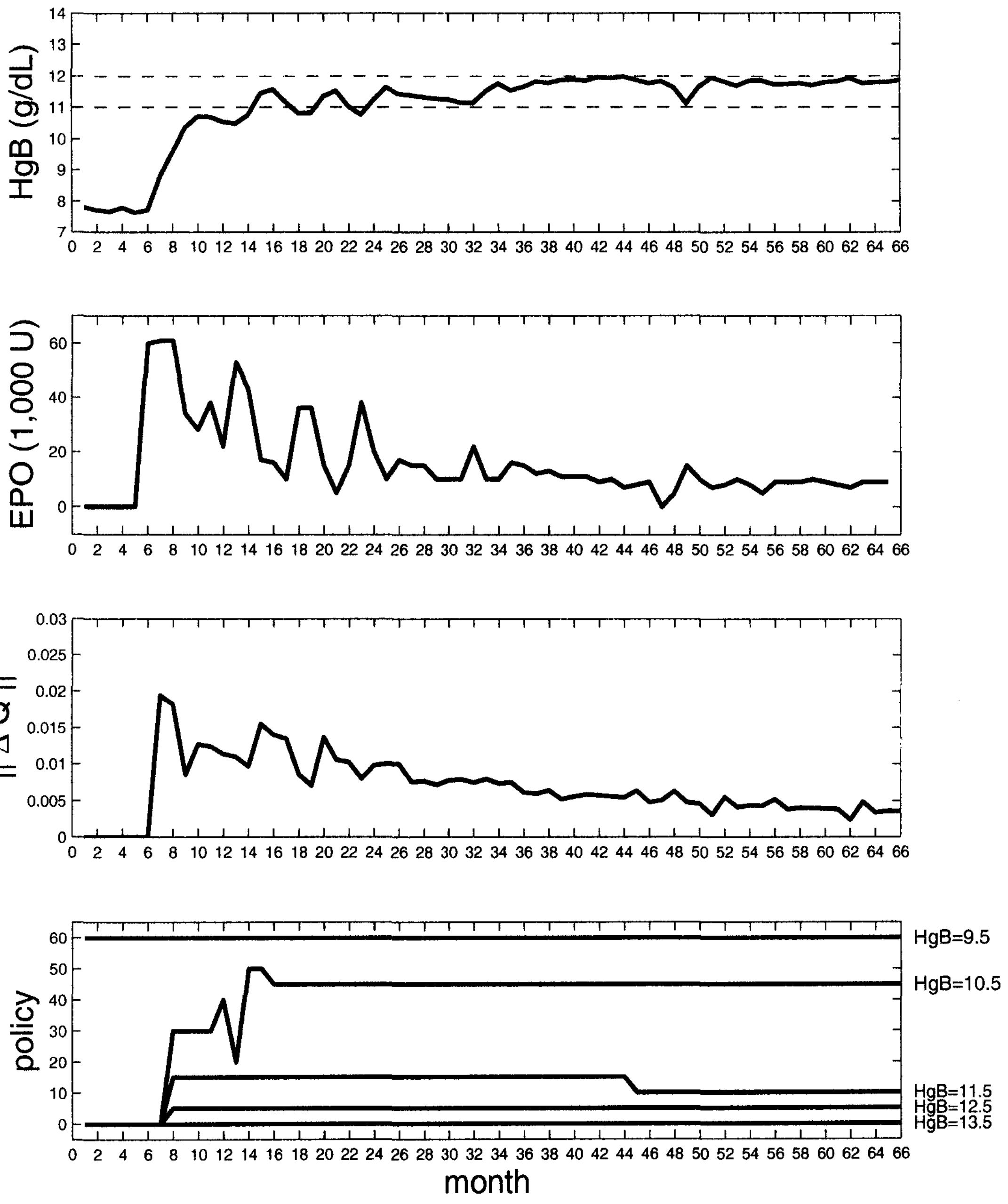

Figure 12. HGB level (top), EPO dose (second from top), the magnitude of the Q-table updates (third from the top), and policy evolution (bottom) for an individual poor responder as performed by Q-learning with RBF Policy Network.

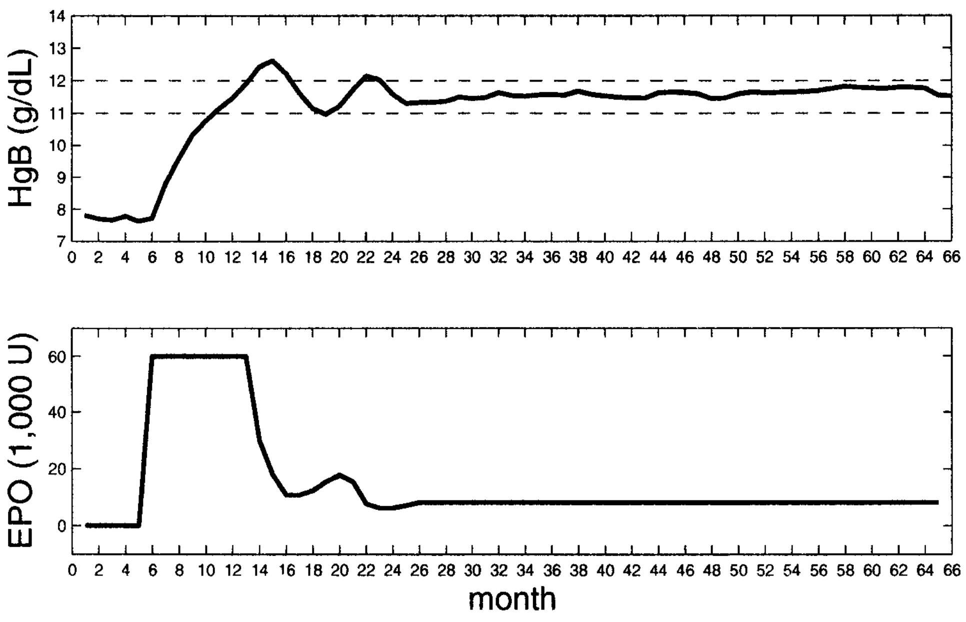

Figure 13. HGB level (top) and EPO dose (middle) for an individual poor responder as performed by AMP.

Figures 12 to 13 show the progress of anemia management for a selected representative poor responder. The most profound difference between the HGB trajectory of a poor responder and that of a normal one is the time to get to the target range. It takes an average of 2 months for the HGB to find the target range for a normal responder. For the poor responder, this period takes from 8 to 18 months. It can be observed that the Q-learning takes longer to get the HGB level of a poor responder to the target range, compared to the AMP. This phenomenon can be attributed to the policy update occurring at the beginning of the therapy. Evidently, the initial action for HGB = 10.5, namely $30,000$ Units of EPO administration, is not aggressive enough for a poor responder and causes a drop in HGB. As mentioned above, HGB below target is an undesired behavior, thus such an action receives a relatively low reward (or punishment) so that a different action is selected in the next step, based on the Q-table. This process contributes to increasing the time required to reach the target HGB range. Consequently, the AMP, as a prescribed policy, works faster for a poor responder, than Q-learning, which learns the policy on-the-fly. Nevertheless,

Table 2

Simulation Statistics for Q-learning

Response Group	Normal	Poor
HGB Level	11.59 (11.12, 12.04)	11.16 (10.76, 11.55)
HGB Variability	0.29 (0.15, 0.42)	0.74 (0.52, 0.95)
Total EPO (1, 000 U)	589.29 (344.56, 834.02)	1145.25 (926.61, 1363.88)

statistics presented in Tables 2 and 3 for poor responders show that the policy obtained by Q-learning and AMP achieve a comparable outcome.

Table 3

Simulation Statistics for AMP

Response Group	Normal	Poor
HGB Level	11.66 (11.56, 11.78)	11.51 (11.35, 11.67)
HGB Variability	0.32 (0.22, 0.41)	0.67 (0.49, 0.84)
Total EPO (1, 000 U)	610.57 (356.91, 864.23)	1075.39 (942.50, 1208.28)

CHAPTER V
GOAL-ORIENTED LEARNING DRIVEN BY CLINICAL KNOWLEDGE

This chapter presents an extension to the Q-learning method demonstrated in the preceding chapter. An alternative Q-value update scheme is proposed here that is based on a critical prior knowledge about the dose-response characteristic: For all patients, it is known that the dose-response curve of HGB vs. EPO is monotonically non-increasing. For example, if a patient's response is evaluated as insufficient for a particular dose at a particular state, then the physician knows that the optimal dose for that state should definitely be higher than the administered one. Consequently, there is no need to explore the benefit of lower doses for that state in future decision stages of treatment.

The conventional Q-learning algorithm does not possess any mechanism to utilize this type of prior information in learning. Hence, this learning method may lead to sub-optimal HGB levels for some patient populations (especially those with decreased EPO-response). Therefore, an additional mechanism is suggested here to be incorporated in the original Q-learning algorithm so that the information about monotonically increasing character of the HGB vs. EPO curve can be incorporated in the update procedure. It is shown in the sequel that such a modification will make the EPO dosing faster and, thus, more efficient.

This study suggests a novel mechanism to perform group updates in Q-table, which is now considered as a time-invariant but a nonlinear function of the state/action pair encountered. The group of state/action pairs to be updated for each state transition is defined a priori, with prior knowledge about the problem. Such an aid to the learning process is categorized as an *advice* in reinforcement learning theory.

Use of Advice in Reinforcement Learning

The idea of propagating the experience gained by the most recently-encountered state/action pair toward the value updates of other pairs is utilized in the general Q(λ) algorithm (Watkins and Dayan, 1992). The eligibility trace (Sutton and Barto, 1998) constitutes an efficient mechanism to implement this learning scheme. Eligibility of a state/action pair $e(x, a)$ has been defined below eq. (24) as a temporal variable, which attains its maximum of 1 as soon as (x, a) is encountered in the simulation and vanishes geometrically by the predefined constant rate $\lambda \in [0, 1]$ along further observations. The value updates of all state/action pairs are then performed in parallel upon each information gain, but weighted by their corresponding eligibility.

Eligibility trace performs specific time-varying abstraction temporal difference iteration. It has an accelerating effect on the convergence of the Q-table, since it gives rise to the update of multiple entries upon the observation of a single state transition. Given the fact that many real-life problems come indeed with a set of prior information or hints about their solution, there may be other forms of abstractions as well, and these may aid in the speed and accuracy of the learning process.

State Abstraction

The earliest work in reinforcement learning literature that utilized region-based updates to evaluate states exactly appears as (Yee et al., 1990). The definition of regions in this work were based on *concepts* organized as a hierarchy to classify the states efficiently. This hierarchy is optimized by an explanation-based method to yield large, useful regions.

Dietterich and Flann (Dietterich and Flann, 1997) proposed optimal state abstraction routines as components embedded in their value iteration algorithms for planning in both deterministic and stochastic domains. These methods assume perfect knowledge about the environment and create adaptively regions of states to be evaluated by Bellman backups. The inverse approach to the decision sequence and reflecting backups to value table only upon detecting an improvement in the processed state's value are both inherent features of conventional dynamic programming, e.g. all-pairs shortest path algorithm (Baase and van Gelder, 1999). These methods were demonstrated on a grid maze (i.e. 2D state representation) and the regions were rectangular. Experiments verified the expectation that regional backups accelerate learning.

An attempt to generalize Bellman backups to handle relational logic operators was made recently in (Kersting et al., 2004). In particular, this work defines abstract versions of the components of conventional reinforcement learning by means of logical queries and then derives a Bellman operator to evolve these rules to express efficiently the pre-image of an action that gives rise to a particular state.

These works aim at accelerating learning by making use of some sort of a built-in procedure to maximize the regions to be updated. However, they allow no manipulation from the exterior in forming these regions, which are presumably optimally large under some assumptions on the domain.

From the point of view of this work, such views to state abstraction tend to relate to hierarchical methods of reinforcement learning eventually, as they draw generalizations on the state space in unsupervised ways.

Other Forms of Advice

Guiding reinforcement learning with external knowledge has been a major issue for over a decade. Many researchers have adopted the term advice to name this knowledge provided/imposed by an external source. Two natural problems arising in dealing with advice are how to represent it and where to incorporate it in the learning system.

Maclin and Shavlik touched these problems (arguably the former one more than the latter) in (Maclin and Shavlik, 1996). They actually proposed a reinforcement learning system, called RATLE, which *requests* advice from an external observer and assimilates the provided information in its internal connectionist structure. In a parallel direction, an actor/critic learning scheme augmented with an explicit supervisor is presented in (Rosenstein and Barto, 2004). This study addresses the issue of combining supervisor knowledge with reinforcement signal in an optimal way. These two works present and delineate very sound scenarios of utilizing advice, which is provided on-the-fly, not a priori.

The works (Ng et al., 1999) and (Wiewiora et al., 2003) propose a computational method to utilize prior knowledge in Q-learning. This is achieved by adding a static and real-valued *potential function* of the encountered state/action pair to the right-hand side of the conventional Q-value iteration. Ng et. al. enlighten in (Wiewiora et al., 2003) the mild conditions under which this modification does not alter the optimal policy so that the effect of potential function is limited to injecting prior knowledge. The latter work generalizes the potential function to accommodate state transitions, i.e. the 4-tuple (s, a, s', a'), rather than state/action pairs.

Using potential function is an elegant way of guiding the search of useful actions with prior knowledge. Employing a fixed value representing prior knowledge in the recursion is a significant extension of setting an initial Q-landscape, which can convey the prior knowledge only temporarily. However, introducing a bias to the iteration as suggested by this function still necessitates a careful fine-tuning of the iteration parameters, because it involves the magnitude of Q-updates.

Suggested Abstraction Scheme

This study suggests a predefined partition of state/action space into *groups* such that the experience gained by an instance (in the form of state transition $x_k \xrightarrow{a_k} x_{k+1}$ and corresponding reward) routes the value updates of all elements in the group. Note that it is actually the form of partitioning of the state/action pairs that represents the prior knowledge about the problem.

The following example demonstrates the basic idea underlying the suggested state abstraction scheme, which is customizes next for the anemia management problem.

An Example

Consider Markov decision process with finite state and action spaces $\mathcal{S}$ and $\mathcal{A}$, but with incomplete transition information, and Q-learning will be applied to discover the optimal policy.

Suppose also that $\mathcal{S}$ and $\mathcal{A}$ are both partially ordered sets and that it is known in advance that the policy $p(\cdot) : \mathcal{S} \rightarrow \mathcal{A}$ governing this process optimally is non-increasing. This information inherent in the problem implies that, if the action part of an encountered state/action pair (x^*, a^*) is evaluated as being unfavorable because of its *insufficiency* with respect to the control objective in effect at some point, then the action portions of all pairs $\mathcal{G}(x^*, a^*) = \{(x^*, a), a \in \mathcal{A} : a < a^*\}$ could be classified in this way. It is then reasonable to apply the value iteration to all members of $\mathcal{G}(x^*, a^*)$ in parallel with the one for the individual pair (x^*, a^*), resulting in decrements in all their values by amounts determined by the temporal difference iteration. Excluding the exploration mechanism from the discussion, such a group update would reduce the chance of actions $a < a^*$ being selected in future visits to x^* (as it is already known -due to the prior knowledge- that those actions would really be non-beneficial).

State Abstraction Due to Monotonicity

The particular information utilized in the drug dosing problem is that any admissible drug dosing policy exploits a *monotonically non-decreasing* relationship between the EPO dose and the HGB level. This fact validates routing the Q-value updates upon the state transition $x_k \rightarrow x_{k+1}$ due to the action a_k in this way:

- If $11.5 > x_k \geq x_{k+1}$ or $x_k = 11.5 > x_{k+1}$, then add the temporal difference, i.e. the second term on the right-hand side (16), not only to $Q(x_k, a_k)$ but also to all

$\{Q(x, a) : x < x_k, a < a_k\}$

- Else, if $x_{k+1} \geq x_k > 11.5$ or $x_{k+1} > x_k = 11.5$, then add the temporal difference, i.e. the second term on the right-hand side (16), not only to $Q(x_k, a_k)$ but also to all $\{Q(x_k, a) : x > x_k, a > a_k\}$
- Else, perform the basic Q-update (16) on $Q(x_k, a_k)$ only.

Note that the new update scheme *punishes* a group of state/action pairs whenever the system exploits an undesired state transition so that the group of actions, which are certainly unfavorable in the light of the monotonicity information. As a result, such actions are less likely to be taken in further decision instances, so the learning speed is expected to improve with the proposed modification.

Experimental Evaluation

The proposed extension was evaulated using an experimental setup similar to the ones used in Section IV.B. An artificial population of 100 normal responders and 100 poor responders was created first. For each artificial patient, a set of four initial HGB values was generated based on the statistics from the above mentioned patient data from 186 individuals. The considered drug dosing mechanism was the one used in Section IV.B, i.e. the discrete-time control system in Figure 9. The difference from the plain Q-learning method here is the suggested abstraction scheme developed above.

In the first set of simulations, plain Q-learning, i.e. the one without group Q-updates, was applied. Subsequently, the simulations were repeated using the proposed extended Q-learning algorithm that includes group updates using the same experimental conditions. The evaluation process was completed by simulating the anemia treatment with a numerical implementation of the Anemia Management Protocol.

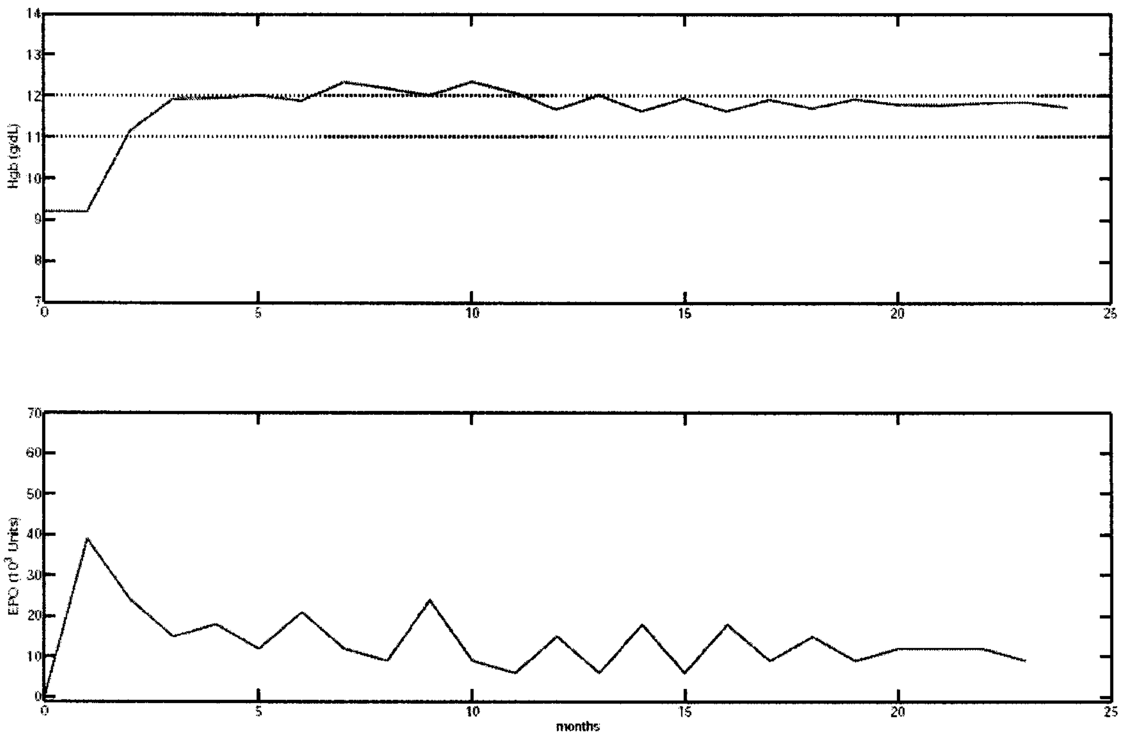

Figure 14. HGB level and EPO dose for an individual normal responder as performed by plain Q-learning.

Results on Individuals

Figures 14 and 15 show examples of a simulated anemia treatment performed by the plain Q-learning for a representative normal and poor responder, respectively. These figures, similar to the ones in Figures 10 and 12, demonstrate that the standard Q-learning exhibits a tendency to maintain the HGB level close to the upper bound of the target range in normal responders and close to the lower bound of the target range in poor responders.

Figures 16 and 17 show examples of a simulated anemia treatment performed by the extended Q-learning method with group updates for a representative normal and poor responder, respectively. These figures show that the addition of group updates allows for much better control of HGB level. For both, normal and poor responder, the HGB level is now much closer to the median of the target range. Hence, one would expect this method to be more effective than the classical Q-learning in a real clinical environment.

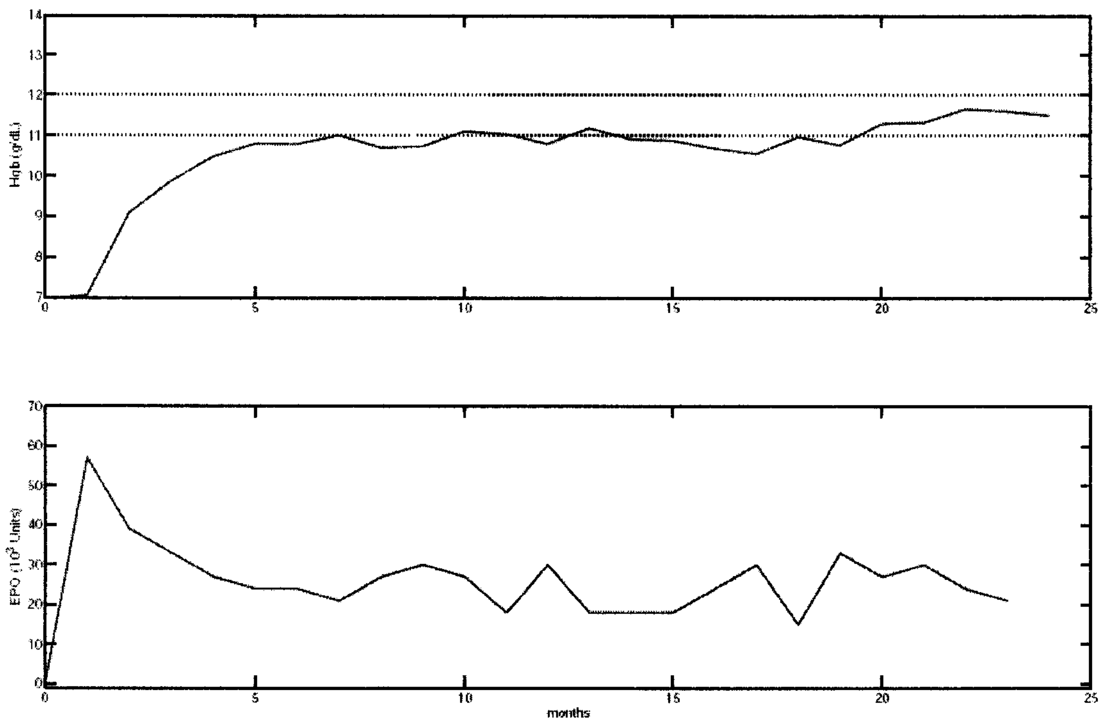

Figure 15. HGB level and EPO dose for an individual poor responder as performed by plain Q-learning.

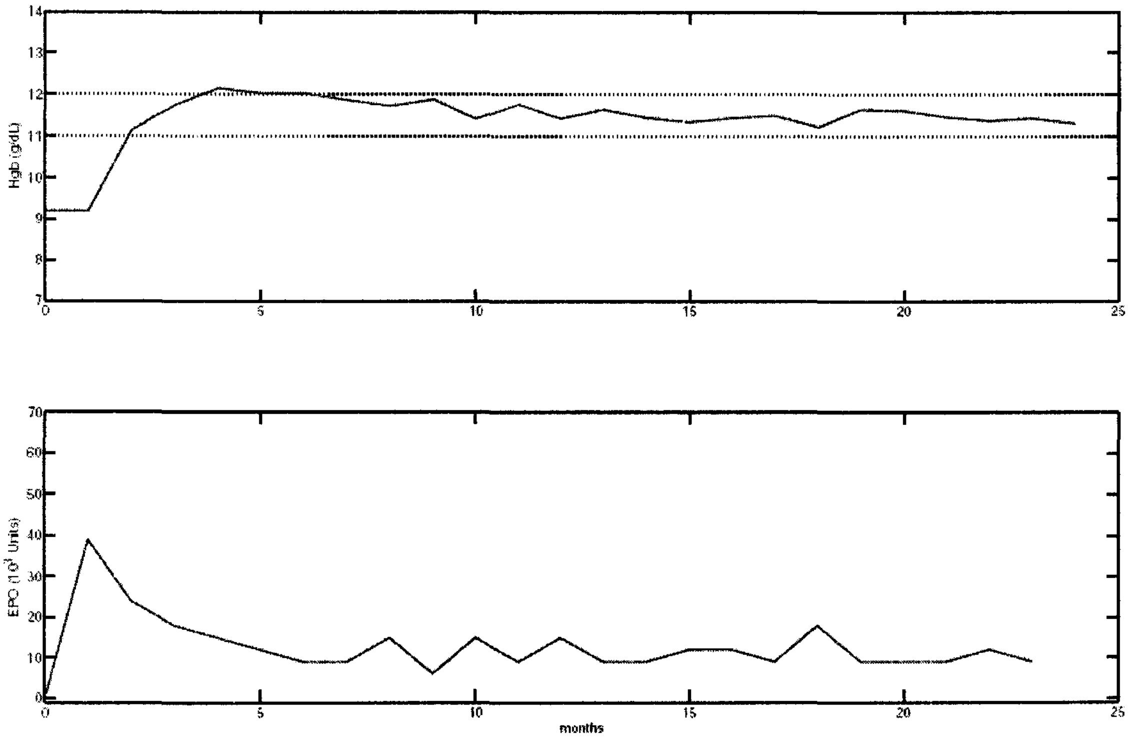

Figure 16. HGB level and EPO dose for an individual normal responder as performed by Q-learning with group updates.

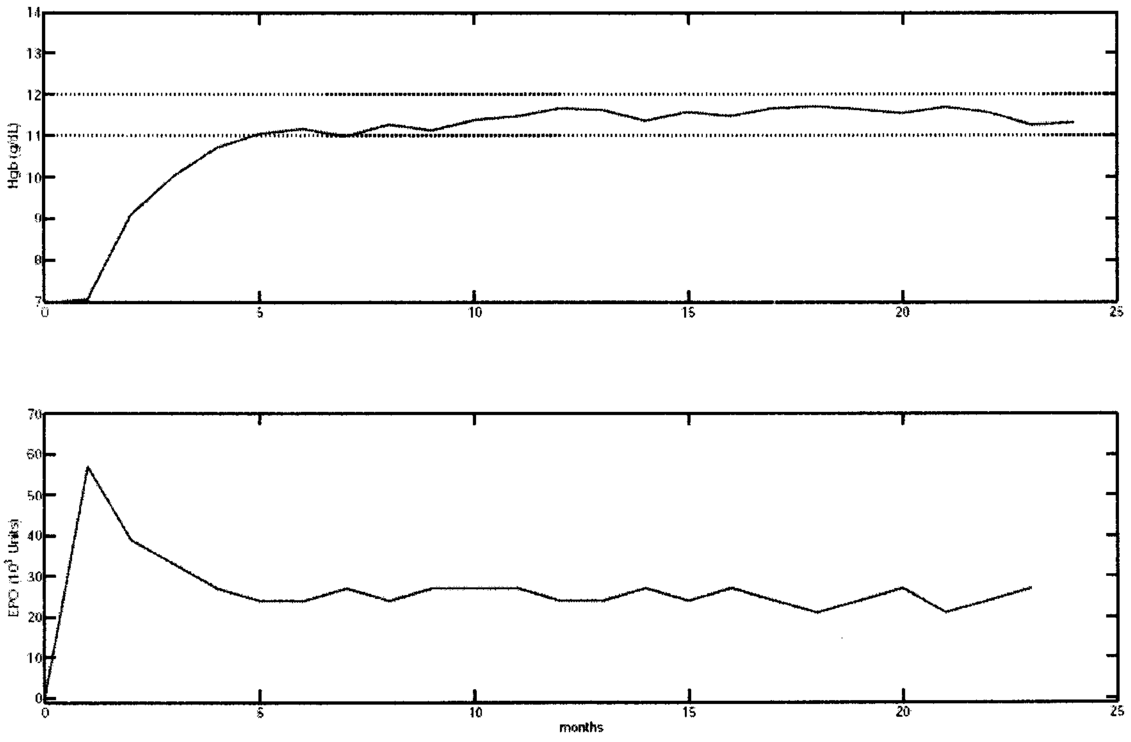

Figure 17. HGB level and EPO dose for an individual poor responder as performed by Q-learning with group updates.

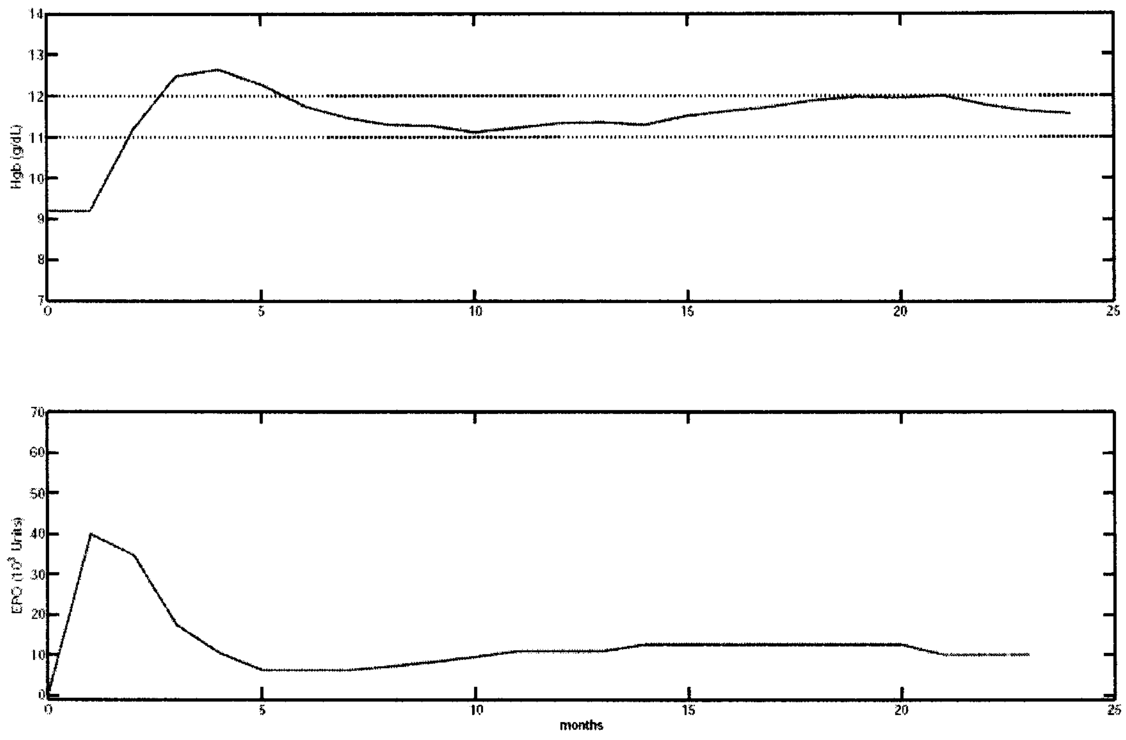

Figure 18. HGB level and EPO dose for an individual normal responder as performed by AMP.

Finally, Figures 18 and 19 show examples of a simulated anemia treatment performed by the numerically implemented Anemia Management Protocol. The most striking phenomenon that can be observed in these figures is the HGB level fluctuation within the target range. This fluctuation occurs for both types of responders. This observation in the simulated environment is consistent with actual data from the clinical environment.

Statistical Results

Tables 4 and 5 provide a quantitative comparison between the three simulated methods. The results are reported as means and 95% Confidence Intervals. Comparing the mean HGB levels of normal responders between the three methods, one can observe that Q-learning has a tendency to overshoot the HGB level, whereas Q-learning with group updates and the Anemia Management Protocol are capable of driving the HGB level to the

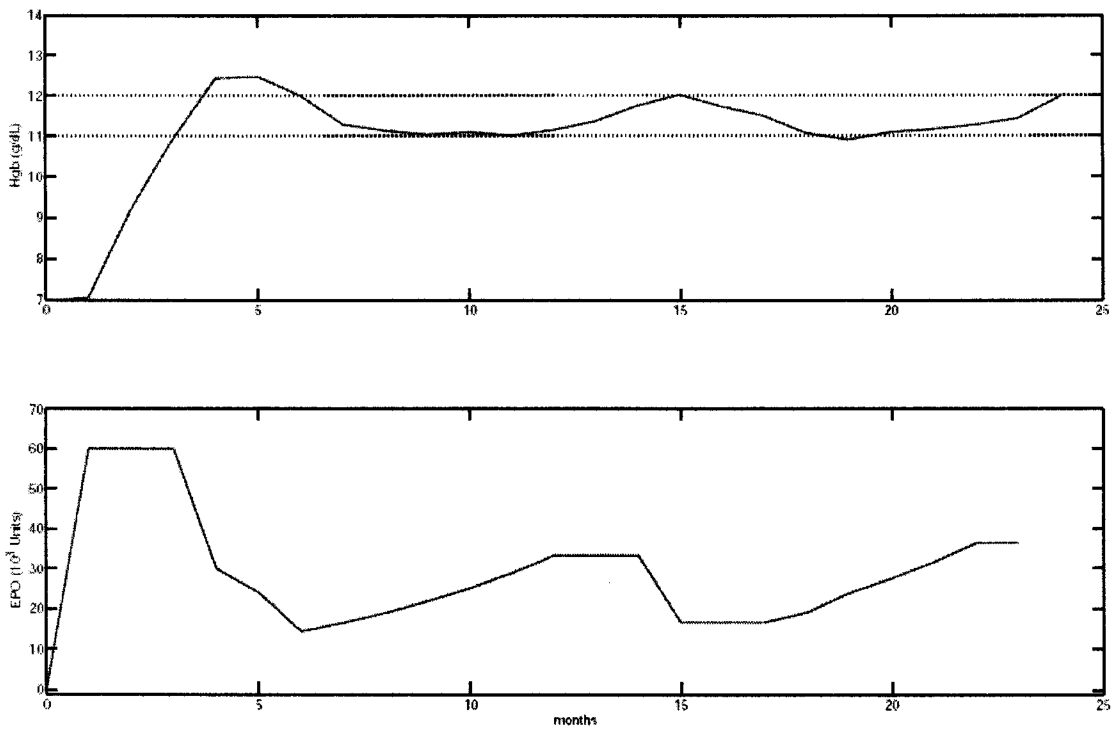

Figure 19. HGB level and EPO dose for an individual poor responder as performed by AMP.

target range. Comparing the standard deviations of HGB levels for normal responders between the three methods, one can observe that both Q-learning methods provide for more stable HGB control than the Anemia Management Protocol. Due to the inability of Q-learning to maintain the HGB level within the target range, the third criterion (number of times HGB out of target range) has a much larger value for this method, compared to the other two. The amounts of administered EPO are not significantly different between the three methods.

Comparing the mean HGB levels of poor responders between the three methods, one can observe that all three methods are capable of driving the HGB level to the target range. Comparison between the standard deviations of HGB levels reveals that, similarly as in the case of normal responders, Q-learning methods provide more stable HGB control than the Anemia Management Protocol. In terms of the number of times the HGB level was out of target range, Q-learning with group updates outperforms the other two competitors.

Table 4

Statistical Comparison of Anemia Management Methods on Normal Responders

Treatment Method	Plain Q-learning	Q-learning Group Update	AMP
Mean HGB (g/dL)	12.21 (11.77, 12.64)	11.77 (11.51, 12.02)	11.75 (11.47, 12.02)
Std Dev HGB (g/dL)	0.16 (0.06, 0.26)	0.25 (0.09, 0.40)	0.50 (0.30, 0.70)
Total EPO (1, 000 U)	286.3 (200.2, 372.5)	227.8 (122.3, 333.3)	223.2 (116.9, 329.6)

Table 5

Statistical Comparison of Anemia Management Methods on Poor Responders

Treatment Method	Plain Q-learning	Q-learning Group Update	AMP
Mean HGB (g/dL)	11.44 (10.69, 11.91)	11.46 (11.18, 11.73)	11.56 (11.34, 11.77)
Std Dev HGB (g/dL)	0.26 (0.14, 0.37)	0.26 (0.12, 0.39)	0.58 (0.28, 0.87)
Total EPO (1, 000 U)	468.1 (351.2, 585.1)	469.7 (334.3, 605.3)	474.9 (351.2, 585.1)

The amounts of administered EPO are again not significantly different between the three methods.

CHAPTER VI
CONCLUSIONS

This study is presumably the first attempt in formulating a medical treatment process as a model-free approximate dynamic programming instance, and casting reinforcement learning methods for solution.

Having chosen the anemia management as the test-bed, the optimal drug-dosing policies of individual virtual patients have been retrieved successfully by using two well-known reinforcement learning methods SARSA and Q-learning. To achieve the control task using these methods a particular patient model is not required. The methods process only simulated trajectories obtained from patients. For the purpose of generating these sample trajectories, a Takagi-Sugeno fuzzy model of an anemia patient has been used in the simulative treatment.

On-policy temporal difference procedure SARSA has been employed first to extract the optimal drug dosing policies from sample trajectories generated by the patient. The results show that the episodic SARSA procedure generates adequate dosing strategies for representative individuals from two different response groups, namely normal and poor responders. Statistics derived over repeated simulations have confirmed that the obtained results were consistent with the clinical goals. To facilitate compact parametric representation of the Q-table, the use of an RBF network as an approximator has also been explored. Initial results showed that it is possible to obtain a convergent RBF approximator for the Q-table.

Implementation of plain Q-learning with RBF network for policy interpolation has also been demonstrated. Experimental evaluation has allowed for comparing this method against the Anemia Management Protocol, regarded here as the gold standard. Statistical and clinical analysis of the test results have showed that Q-learning is actually capable of performing adequate anemia treatment in real time, comparable to the Anemia Management Protocol.

Finally, as an extension to the suggested reinforcement learning methods, a novel state-abstraction mechanism has been proposed and tested experimentally. The suggested approach has incorporated a useful clinical knowledge, namely the monotonicity of dose-response characteristic, into the simulative solution process. This constitutes a typical illustration of how problem-specific hints could aid substantially in the solution. The benefit gained by this addition has revealed itself as a significant improvement in the learning time, as simulated in the experiments.

Results of this study obtained using artificial patients require elaborate inspections and many clinical constraints taken into account before they become clinically meaningful in practice. Such a medical oriented extension is necessary as it would obviously yield a fair assessment of the practical value of this research effort.

As a final word, the author would like to stress his viewpoint that neither model-free nor model-based approaches alone can be successful enough in simulating and improving medical treatment processes, such as anemia management. Therefore, although inaccuracies of patient models in many medical applications were leading the motivations of this study, the approaches presented here could be utilized best in collaboration with mainstream biomedical approaches that focus on modeling patients. A useful decision mechanism could then be constructed from their constructive trade-off. As an immediate application of this approach, a hybrid method weighting the decisions from model-free and model-based decision makers is suggested in (Gaweda et al., 2006b), where weights assigned to their advices depend on the inaccuracy of the patient model along treatment.

REFERENCES

Astrom, K. J. and Wittenmark, B. (1989). *Adaptive Control.* Addison-Wesley, Reading, MA.

Baase, S. and van Gelder, A. (1999). *Computer Algorithms: Introduction to Design and Analysis.* Addison-Wesley Longman Publishing Co., Boston, MA, 3rd ed. edition.

Barto, A. G., Bradtke, S. J., and Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138.

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:835–846.

Bellman, R. E. (1957). *Dynamic Programming.* Princeton University Press, Princeton, NJ.

Bellman, R. E. (1983). *Mathematical Methods in Medicine.* World Scientific Publishing, Singapore.

Benveniste, A., Metivier, M., and Priouret, P. (1990). *Adaptive Algorithms and Stochastic Approximations.* Springer-Verlag, Berlin.

Bertsekas, D. P. (2000). *Dynamic Programming and Optimal Control, vol. 1 & 2.* Athena Scientific, Belmont, MA, 2nd edition.

Bertsekas, D. P., Borkar, V., and Nedic, N. A. (2004). Improved temporal difference methods with linear function approximation. In Si, J., Barto, A., Powell, W., and Wunsch, D., editors, *Handbook of Learning and Approximate Dynamic Programming*. Wiley-IEEE Press.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neurodynamic Programming.* Athena Scientific, Belmont, MA.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, England.

Boyan, J. A. (2002). Technical update: Least-squares temporal difference learning. *Machine Learning*, 49:1–15.

Bradtke, S. J. and Barto, A. G. (1996). Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22:33–57.

Brier, M. E., Jacobs, A. A., and Gaweda, A. E. (2006). A pharmacodynamic model of erythropoietin using fuzzy classification of hemoglobin response. Submitted to Pharmaceutical Research.

Buell, J., Jelliffe, R., Kalaba, R., and Sridhar, R. (1970). Modern control theory and optimal drug regimens II: Combination therapy. *Mathematical Biosciences*, 6:67–74.

Connell, J. and Mahadevan, S. (1993). *Robotic Learning*. Kluwer Academic, Boston, MA.

Dayan, P. D. (1992). The convergence of TD(λ) for general λ. *Machine Learning*, 8:341–362.

de Farias, D. P. and van Roy, B. (2000). On the existence of fixed points for approximate value iteration and temporal-difference learning. *Journal of Optimization Theory and Applications*, 105:589–608.

Dietterich, T. G. and Flann, N. S. (1997). Explanation-based learning and reinforcement learning: A unified view. *Machine Learning*, 28:169–210.

Eschbach, J. and Adamson, J. (1985). Anemia of end-stage renal disease (ESRD). *Kidney Int.*, 28:1–5.

Gaweda, A. E., Jacobs, A. A., Brier, M. E., and Zurada, J. M. (2003). Pharmacodynamic population analysis in chronic renal failure using artificial neural networks - A comperative study. *Neural Networks*, 16:841–845.

Gaweda, A. E., Muezzinoglu, M. K., Aronoff, A. G., Jacobs, A. A., Zurada, J. M., and Brier, M. E. (2006a). Using clinical information in goal-oriented learning for anemia management. *IEEE Engineering in Medicine and Biology Magazine*. In print.

Gaweda, A. E., Muezzinoglu, M. K., Aronoff, G. R., Jacobs, A. A., Zurada, J. M., and Brier, M. E. (2005a). Incorporating prior knowledge into Q-learning for drug delivery individualization. In *Proceedings of the 4th International Conference on Machine Learning and Applications (ICMLA'05)*, pages 207–214, Los Angeles, CA.

Gaweda, A. E., Muezzinoglu, M. K., Aronoff, G. R., Jacobs, A. A., Zurada, J. M., and Brier, M. E. (2005b). Individualization of pharmacological anemia management using reinforcement learning. *Neural Networks*, 18:826–834.

Gaweda, A. E., Muezzinoglu, M. K., Aronoff, G. R., Jacobs, A. A., Zurada, J. M., and Brier, M. E. (2005c). Reinforcement learning approach to individualization of chronic pharmacotherapy. In *Proceedings of the 2005 International Joint Conference on Neural Networks (IJCNN'05)*, pages 3290–3295, Montreal, Canada.

Gaweda, A. E., Muezzinoglu, M. K., Aronoff, G. R., Jacobs, A. A., Zurada, J. M., and Brier, M. E. (2006b). Fusion of goal-oriented and model-based advices for anemia management. In preparation.

Hu, C., Lovejoy, W. S., and Shafer, S. L. (1994a). Comparison of some control strategies for three-compartment PK/PD models. *Journal of Pharmacokinetics and Biopharmaceutics*, 22:525–550.

Hu, C., Lovejoy, W. S., and Shafer, S. L. (1994b). An efficient strategy for dose regimens. *Journal of Pharmacokinetics and Biopharmaceutics*, 22:73–92.

Jaakkola, T., Jordan, M., and Singh, S. (1994). On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6:1185–1201.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.

Kersting, K., van Otterlo, M., and de Readt, L. (2004). Bellman goes relational. In *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada.

Khoo, M. C. K. (1999). *Physiological Control Systems*. Wiley - IEEE Press, New Jersey.

Kumar, P. R. (1985). A survey of some results in stochastic adaptive control. *SIAM Journal on Control and Optimization*, 23:329–380.

Maclin, R. and Shavlik, J. W. (1996). Creating advice-taking reinforcement learners. *Machine Learning*, 22:251–281.

Moore, B. L., Sinzinger, E. D., Quasny, T. M., and Pyeatt, L. D. (2004). Intelligent control of closed-loop sedation in simulated ICU patients. In *Proceedings of the 17th International Florida Artificial Intelligence Research Symposium*, Miami Beach, FL.

Ng, A. Y., Harada, D., and Russel, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning*, Bled, Slovenia.

Park, J. and Sandberg, I. W. (1991). Universal approximation using radial-basis-function networks. *Neural Computations*, 3:246–257.

Poljak, B. T. and Tsypkin, Y. Z. (1973). Pseudogradient adaptation and training algorithms. *Adaptive Systems*, 12:377–397.

Powell, W. B. (2005). Approximate dynamic programming for high dimensional resource allocation problems. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'05)*, Montreal, Canada.

Puterman, M. L. (1994). *Markov Decision Processes*. John Wiley Inc., New York.

Rosenstein, M. T. and Barto, A. G. (2004). Supervised actor-critic reinforcement learning. In Si, J., Barto, A., Powell, W., and Wunsch, D., editors, *Handbook of Learning and Approximate Dynamic Programming*, pages 359–380. Wiley-IEEE Press, New York.

Russell, S. J. and Norvig, P. (2002). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 2nd edition.

Sanner, R. M. and Slotine, J. J. E. (1992). Gaussian networks for direct adaptive control. *IEEE Transactions on Neural Networks*, 3:837–863.

Schaeffer, A. J., Bailey, M. D., Shechter, S. M., and Roberts, M. S. (2004). Modeling medical treatment using Markov decision processes. In Brandeau, M. L., Sainfort, F., and Pierskalla, W. P., editors, *Handbook of Operations Research and Health Care: Methods and Applications*. Springer.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning*. MIT Press, Cambridge, MA.

Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its applications to modelling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15:116–132.

Tesauro, G. (1995). Temporal difference learning and TD-gammon. *Communications of the ACM*, 38:58–68.

Thorndike, E. L. (1911). *Animal Intelligence*. Hafner, Darien, CT.

Tsitsiklis, J. N. (1994). Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 16:185–202.

Tsitsiklis, J. N. and van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42:674–690.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8:279–292.

Wiewiora, E., Cottrell, G., and Elkan, C. (2003). Principled methods for advising reinforcement learning agents. In *Proceedings of the 20th International Conference on Machine Learning*, Washington D.C.

Yee, R. C., Saxena, S., Utgoff, P. E., and Barto, A. G. (1990). Explaining temporal differences to create useful concepts for evaluating states. In *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI-90)*, pages 882–888, Boston, MA.

Zurada, J. M. (1992). *Introduction to Artificial Neural Systems*. West Publishing Co., St. Paul, MN.

CURRICULUM VITAE

NAME:	Mehmet Kerem Muezzinoglu
ADDRESS:	Electrical and Computer Engineering Department University of Louisville Louisville, KY 40292
EDUCATION & TRAINING	Ph.D., Dokuz Eylul University Izmir, Turkey 2000 – 2003
	M.S., Istanbul Technical University Istanbul, Turkey 1998 – 2000
	B.S., Istanbul Technical University Istanbul, Turkey 1994 – 1998
APPOINTMENTS	Assistant Professor, Electrical and Computer Engineering Dept., University of Louisville Louisville, KY 2006 –
	University Fellow, Computational Intelligence Lab, University of Louisville Louisville, KY 2003 – 2006
	Visiting Researcher Autonomous Learning Lab, University of Massachusetts Amherst, MA 2005

Visiting Researcher
Computational Intelligence Lab,
University of Louisville
Louisville, KY
2002

Research and Teaching Assistant,
Electrical and Electronics Engineering Dept.,
Dokuz Eylul University
Izmir, Turkey
2000 – 2003

Research and Teaching Assistant,
Electronics and Communication Engineering Dept.,
Istanbul Technical University
Istanbul, Turkey
1998 – 2000