Thesis Proposal

Approximate Dynamic Programming Using Bellman Residual Elimination

Brett Bethke

November 25, 2008

Thesis Committee: Jonathan P. How, Dimitri Bertsekas, Asuman Ozdaglar

Abstract

The overarching goal of the thesis is to devise new strategies for multi-agent planning and control problems, especially in the case where the agents are subject to random failures, maintenance needs, or other health management concerns, or in cases where the system model is not perfectly known. We argue that dynamic programming techniques, in particular Markov Decision Processes (MDPs), are a natural framework for addressing these planning problems, and present an MDP problem formulation for a persistent surveillance mission that incorporates stochastic fuel usage dynamics and the possibility for randomly-occurring failures into the planning process. We show that this problem formulation and its optimal policy lead to good mission performance in a number of real-world scenarios. Furthermore, an on-line, adaptive solution framework is developed that allows the planning system to improve its performance over time, even in the case where the true system model is uncertain or time-varying. Motivated by the difficulty of solving the persistent mission problem exactly when the number of agents becomes large, we then develop a new family of approximate dynamic programming algorithms, called *Bellman Residual Elimination* (BRE) methods, which can be employed to approximately solve large-scale MDPs. We analyze these methods and prove a number of desirable theoretical properties about them, including reduction to exact policy iteration under certain conditions. Finally, we apply these BRE methods to large-scale persistent surveillance problems and show that they yield good performance, and furthermore, that they can be successfully integrated into the adaptive planning framework.

1 Introduction

Problems in which decisions are made in stages are ubiquitous in the world: they arise in engineering, science, business, and more generally, in life. In these problems, the ability to make good decisions relies critically on balancing the needs of the present with those of the future. To present a few examples: the farmer, when choosing how many seeds to plant, must weigh the short-term profits of a larger harvest against the long-term risk of depleting the soil if too many plants are grown. The chess player, when considering a particular move, must weigh the short-term material gain of that move (tactical advantage) with its long-term consequences (strategic advantage). The student must weigh the short-term opportunity cost of obtaining an education against the long-term prospects of more desirable career opportunities.

Most such sequential decision-making problems are characterized by a high degree of complexity. This complexity arises from two main sources. First, the long-term nature of these problems means that the tree of all possible decision sequences and outcomes is typically extraordinarily large (for example, the number of possible games of chess has been estimated at $10^{10^{50}}$ [46]). Second, some degree of uncertainty is often present in these problems, which makes prediction of the future difficult.

In the face of these formidable challenges, humans are often remarkably good at solving many sequential decision-making problems. That humans are able to play chess, make sound business decisions, and navigate through life's sometimes daunting complexities is a testament to our critical-thinking and problem-solving skills. Motivated by our own success at dealing with these challenges, it is natural to ask whether it is possible to teach a computer to perform as well or perhaps even better than a human in these complex situations. This difficult question, which is central to the field of Artificial Intelligence [78], offers the tantalizing possibilities of not only being able to create "intelligent" machines, but also of better understanding the ways that humans think and solve problems.

1.1 Motivation: Multi-agent Robotic Systems

Developing planning architectures that make good decisions in real-world, sequential decision-making problems is the topic of this thesis. The motivating application, which will be developed and used in much of the thesis work, is drawn from the rapidly evolving field of multi-agent robotics.

Robotic systems are becoming increasingly sophisticated in terms of hardware capabilities. Advances in sensor systems, onboard computational platforms, energy storage, and other enabling technologies have made it possible to build a huge variety of air-, ground-, and sea-based robotic vehicles for a range of different mission scenarios [2, 65]. Many of the mission scenarios of interest, such as persistent surveillance, search and rescue, weather pattern monitoring, etc, are inherently long-duration and require sustained coordination of multiple, cooperating robots in order to achieve the mission objectives. In these types of missions, a high level of autonomy is desired due to the logistical complexity and expense of direct human control of each individual vehicle. Currently, autonomous mission planning and control for multi-agent systems is an active area of research [68, 43, 55, 75, 50]. Some of the issues in this area are similar to questions arising in manufacturing systems [14, 47] and air transportation [17, 3, 76, 8, 32, 45, 49]. While these efforts have made significant progress in understanding how to handle some of the complexity inherent in multi-agent problems, there remain a number of open questions in this area.

One important question is referred to as the *health management problem for multi-agent systems* [57, 58]. Designs of current and future robotic platforms increasingly incorporate a large number of sensors for monitoring the health of the robot's own subsystems. For example, sensors may be installed to measure the temperature and current of electric motors, the effectiveness of the robots's control actuators, or the fuel consumption rates in the engine. On a typical robot, sensors may provide a wealth of data about a large number of vehicle subsystems. By making appropriate use of this data, a health-aware autonomous system may be able to achieve a higher level of overall mission performance, as compared to a non-health-aware system, by making decisions that account for the current capabilities of each agent. For example, in a search and track mission, utilization of sensor health data may allow an autonomous system to assign the robots with the best-performing sensors to the search areas with the highest probability of finding the target.

Utilization of the current status of each robot is an important aspect of the health management problem. Another important aspect is the ability not only to react to the current status, but to consider the implications of future changes in health status or failures on the successful outcome of the mission.

This predictive capability is of paramount importance, since it may allow an autonomous system to avoid an undesirable future outcome. For example, if a robot tracking a high value target is known to have a high probability of failure in the future, the autonomous system may be able to assign a backup robot to track the target, ensuring that the tracking can continue even if one of the robots fails.

Part of the work of this thesis addresses these health management issues and develops a general framework for thinking about the health management problem. It then specializes the discussion to the persistent surveillance problem with a focus on group fuel management when there is uncertainty in the fuel consumption dynamics. This work builds on previous health management techniques developed for the persistent surveillance problem [57]. While the previous work focused on embedding health-aware heuristics into an already-existing mission management algorithm, this thesis develops a new formulation of the problem in which health-aware behaviors emerge automatically.

1.2 Background

Dynamic programming is a natural mathematical framework for addressing the sequential decision-making problems that are the subject of this thesis [14]. First proposed by Bellman [11], the basic dynamic programming framework incorporates two main elements. The first of these elements is a discrete-time, dynamic model of the system in question. The dynamic model captures the evolution of the state of the system over time, under the influence of a given sequence of actions. Depending on the nature of the system, this model may incorporate either deterministic or stochastic state transitions. The second element is a cost function, depending on the system state and possibly the decision made in that state, that is additive over time. In order to "solve" a dynamic program, we seek to find a state-dependent rule for choosing which action to take (a *policy*) that minimizes the expected total cost incurred.

If the state space associated with a particular dynamic programming problem is discrete, either because the state space is naturally described in a discrete setting or because it results from the discretization of an underlying continuous space, the resulting problem formulation is referred to as a Markov Decision Process (MDP). This thesis considers the general class of infinite horizon, discounted MDPs. The MDP is specified by $(\mathcal{S}, \mathcal{A}, P, g)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is the (possibly stochastic) system dynamics model where $P_{ij}(u)$ gives the transition probability from state i to state j under action u, and $g(i,u)$ gives the immediate cost of taking action u in state i. Future costs are discounted by a factor $0 < \alpha < 1$. A policy of the MDP is denoted by $\mu : \mathcal{S} \to \mathcal{A}$. Given the MDP specification, the problem is to minimize the so-called cost-to-go function $J_\mu : \mathcal{S} \to \mathbb{R}$ over the set of admissible policies Π:

$$\min_{\mu \in \Pi} J_\mu(i_0) = \min_{\mu \in \Pi} \mathbb{E}\left[\sum_{k=0}^{\infty} \alpha^k g(i_k, \mu(i_k))\right]. \tag{1}$$

Here, $i_0 \in \mathcal{S}$ is an initial state and the expectation $\mathbb{E}$ is taken over the possible future states $\{i_1, i_2, \dots\}$, given i_0 and the policy μ.

In solving the MDP, the primary object of interest is the policy $\mu^\star$ which achieves the minimum in (1). The optimal cost associated with $\mu^\star$, denoted by $J^\star \equiv J_{\mu^\star}$, satisfies the Bellman equation

$$J^\star(i) = \min_{u \in \mathcal{A}} \left(g(i,u) + \alpha \sum_{j \in \mathcal{S}} P_{ij}(u) J^\star(j) \right) \quad \forall i \in \mathcal{S}. \tag{2}$$

If $J^\star$ can be found by solving (2), then the optimal policy $\mu^\star$ is given by

$$\mu^\star(i) = \arg\min_{u \in \mathcal{A}} \left(g(i,u) + \alpha \sum_{j \in \mathcal{S}} P_{ij}(u) J^\star(j) \right). \tag{3}$$

(3) establishes a relationship between the policy and its associated cost function. Usually, the minimization (3) is easier to perform than solving the nonlinear, coupled system of equations (2). Therefore, the bulk of the work in solving an MDP normally involves computing the optimal cost function. One approach to computing the optimal cost is to solve (2) in an iterative fashion using a procedure known as *value iteration*. Once the cost is known, the associated policy can be computed.

An alternative approach to solving an MDP arises from the observation that if a policy μ is fixed, then the nonlinear system (2) reduces to a linear system which is easier to solve:

$$J_\mu(i) = g(i, \mu(i)) + \alpha \sum_{j \in \mathcal{S}} P_{ij}(\mu(i)) J_\mu(j) \quad \forall i \in \mathcal{S}. \tag{4}$$

Solving (4) is known as *policy evaluation*. The solution yields J_μ, the cost-to-go of the fixed policy μ. Once the policy's cost-to-go function is known, a new, better policy can be constructed by performing a *policy improvement* step analogous to (3). By iteratively performing policy evaluation followed by policy improvement, a sequence of policies that are guaranteed to converge to the optimal policy $\mu^\star$ is obtained. This procedure is known as *policy iteration*.

Approximate Methods

While MDPs are a powerful and general framework, there are two significant challenges, known as the "twin curses" of dynamic programming, that must be overcome in order to successfully apply them to real-world problems:

The curse of dimensionality refers to the fact that as the problem size increases, the size of the state space $|\mathcal{S}|$, and therefore the amount of computation necessary to solve (2), grows exponentially rapidly. For typical problems, this exponential growth quickly overwhelms the available computational resources, making it difficult or impossible to solve the problem exactly in reasonable time.

The curse of modeling refers to the potential difficulty of accurately modeling the dynamics of the system in question. Many systems exhibit *parametric uncertainty*. In this scenario, the basic character of the governing dynamic equations is known, but the exact values of one or more parameters are not. For example, in a spacecraft control problem, the basic equations of motion (rigid-body dynamics) are well-understood, but the various moments of inertia may not be known exactly. In more extreme cases, even the basic form of the dynamic equations may not be fully understood. Inaccuracies in the model used to solve the MDP may lead to poor performance when the resulting control policy is implemented on the real system.

The question of how to address these challenges has been the subject of a great deal of research, leading to the development of fields known as *approximate dynamic programming*, *reinforcement learning*, and *neuro-dynamic programming* [16, 92]. Semantically, these fields are essentially equivalent (having different names due only to the fact that they were originally developed by different communities), and for the sake of clarity, we shall refer to all of them as approximate dynamic programming (abbreviated ADP).

A wide variety of methods for dealing with the twin curses of dynamic programming have been proposed in the ADP literature. Since many of these methods are significantly different from each other, it is helpful to understand a few general properties that allow us to categorize and compare different methods. Broadly speaking, there are three main properties of interest:

Choice of approximation architecture A central idea for addressing the curse of dimensionality is the use of a *function approximation architecture* to represent the cost function. Functions that are representable within the chosen architecture are typically described by a set of parameters, where the number of parameters N_p is much smaller than the size of the state space $|\mathcal{S}|$. In this way, the problem of finding the cost-to-go function is reduced from a search in an $|\mathcal{S}|$-dimensional to one in an N_p-dimensional space, where $N_p \ll |\mathcal{S}|$. A large number of approximation architectures are possible and may involve, for example, neural networks, polynomials, radial basis functions, wavelets, kernel-based techniques, etc. The type of approximation architecture employed is a key property of any ADP method.

Model availability The amount of system model information assumed to be available is another key property of any ADP method. There are two main classes of methods with regard to model availability. *Model-free* methods attempt to address the curse of modeling by assuming no knowledge of the system model whatsoever. In order to learn the cost function, model-free methods rely on observed state transition data, obtained from either the real system or a simulation thereof. In contrast, *model-based* methods assume that some form of system model information is available and exploit this information in order to learn the cost function.

Training technique The question of how to select the best cost function from the chosen approximation architecture (for example, choosing the weights in a neural network approximation architecture) is called "training". Generally, training is formulated as an optimization problem. The type of optimization problem (i.e. linear, quadratic, convex, non-convex, etc) and method of solution (i.e. gradient descent, analytic, linear programming, etc) are further important characteristics of an ADP method.

1.3 Literature Review

Some of the earliest investigations into ADP methods can be traced back to Shannon and Samuel in the 1950's. Shannon was perhaps the first to propose the approximation architecture now known as the *linear combination of basis functions* in his work with computer chess [86]. In this work, both important features of the chess game (such as material advantage, piece mobility, etc) and their corresponding weights were hand-selected and used to evaluate potential moves. Samuel, working with the game of checkers, went a step further and devised a automated training method for the computer to learn the weights [1]. Eventually, Samuel's checkers program was able to consistently beat its human creator.

The development of artificial neural networks [52, 24, 48] led to a great deal of interest in using them as approximation architectures in ADP problems. One of the most notable successes in this area was Tesauro's computer backgammon player, which was able to achieve world-class play [94, 95]. Tesauo's work utilized the method of temporal differences (TD) [93, 77, 16, 92] as the training mechanism. The success of TD methods has since encouraged development of other methods such as Least Squares Policy Evaluation (LSPE) [15, 64] and Least Squares Temporal Differences (LSTD) [29, 28, 56]; a summary of these methods is given in [14, Vol. 2, Chap. 6].

Another group of ADP methods arise from the observation that finding the optimal cost function can be cast as a linear programming problem [38, 25, 61, 51]. The resulting linear program suffers from the curse of dimensionality; it has as many decision variables as states in the MDP, and an even larger number of constraints. De Farias and Van Roy proposed an approach, known as approximate linear programming (ALP), that reduces the dimensionality of the linear program by utilizing a linear combination of basis functions architecture combined with a mechanism to sample only a small subset of the constraints [34, 35, 36, 37]. An extension to the ALP approach that automatically generates the basis functions in the linear architecture was developed by Valenti [97].

Recently, the use of kernel-based architectures for pattern classification and function approximation has generated considerable excitement in the machine learning community. Building on results from statistical learning [99, 98, 33, 62] and the theory of reproducing kernels [5, 13, 44, 79, 81, 100], initial progress in this area focused on the development of kernel-based classifiers such as Support Vector Machines (SVMs) [26, 82, 12, 31]. Much research has been done on efficient SVM training, even when the size of the problem is large [70, 69, 54, 67, 101, 27]. Similar kernel-based techniques can also be applied to the function approximation (regression) problem, yielding a family of regression algorithms such as Support Vector Regression [88, 84, 83], Gaussian Process Regression [72, 89, 30], and Kernel Ridge Regression [80]. Compared with function approximation architectures such as the neural networks used in much of the previous ADP work, kernel-based architectures enjoy a number of advantages, such as being trainable via convex (quadratic) optimization problems that do not suffer from local minima, and allowing the use of powerful, infinite-dimensional data representations.

Motivated by these recent advances, some research has been done to apply powerful kernel-based techniques to the ADP problem. Dietterich and Wang investigated a kernelized form of the linear programming approach to dynamic programming [39]. Ormoniet and Sen presented a model-free approach for doing approximate value iteration using kernel smoothers, under some restrictions on the structure of the state space [66]. Using Gaussian processes, Rasmussen and Kuss derived analytic expressions for the approximate cost-to-go of a fixed policy, in the case where the system state variables are restricted to evolve independently according to Gaussian probability transition functions [71]. Engel, Mannor, and Meir applied a Gaussian process approximation architecture to TD learning [42, 41, 40], and Reisinger, Stone, and Miikkulainen subsequently adapted this framework to allow the kernel parameters to be estimated online [74]. Tobias and Daniel proposed a LSTD approach based on SVMs [96]. Several researchers have investigated designing specialized kernels that exploit manifold structure in the state space [90, 91, 59, 60, 10, 9, 87]. This work represents exciting progress; however, the field of kernel-based ADP has developed only recently, and there remain numerous possibilities that are yet unexplored. A

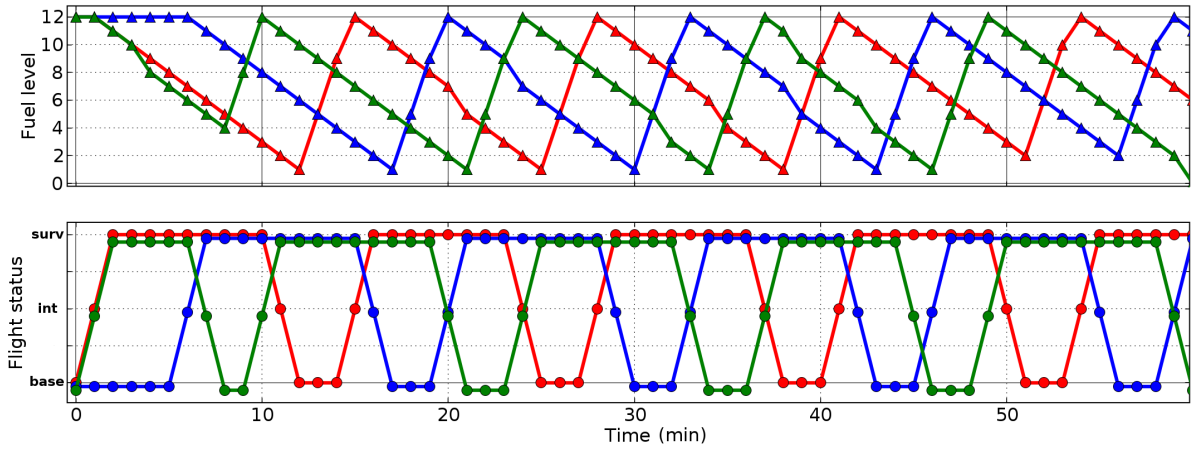

Figure 1: Persistent surveillance mission simulation, under the optimal policy [23]. Fuel levels of each of the three UAVs (shown in red, blue, and green) are shown in the upper plot, and the locations of each UAV (which can be one of the base location, intermediate location, or surveillance location) are shown in the lower plot. Note that the UAVs must coordinate their actions in order to provide continuous coverage in which two UAVs are on-station in the surveillance location.

central goal of this thesis is to investigate a few of these possibilities by developing a family of new kernel-based ADP algorithms.

2 Work to Date

To date, work has been accomplished in three main areas related to the central goal of expanding the capabilities of multi-agent planning systems. The three areas are: 1) modeling and analysis of multi-agent health management problems, 2) approximate dynamic programming algorithm development and implementation, and 3) adaptive MDP-based planning. The work in each of these areas is described below.

2.1 Modeling and Analysis of Multi-Agent Health Management Problems [23]

The first area's focus is on developing models of real-world, multi-agent planning problems which incorporate health management considerations, such as vehicle failures or maintenance needs. In [23], we argued that these planning problems are naturally modeled as MDPs and formulated a particular MDP, based on the requirement of using a group of UAVs to provide persistent surveillance coverage over a region of interest, to illustrate our approach. The persistent surveillance problem formulation includes both the possibility of random vehicle failures as well as uncertainty in the fuel usage dynamics. The optimal control policy for this problem, which was initially computed using value iteration, demonstrates a number of desirable properties. These properties include cooperation between the individual UAVs to establish continuous surveillance coverage, and proactive hedging against the fuel usage uncertainty, resulting in good coverage performance while minimizing the probability of vehicle loss due to fuel exhaustion. Figure 1 shows simulation results of the persistent surveillance problem under the optimal policy.

Hardware flight experiments have shown that the persistent surveillance problem formulation and optimal policy lead to good overall performance in real-world mission scenarios, while proactively managing the health of the UAV fleet. In our flight experiments to date, we have tested missions involving up to three UAVs at a time. In order to carry out missions with a larger number of UAVs, it is necessary to utilize approximate solution techniques, since the time required to solve the problem exactly becomes prohibitively large. To address this problem, the second major area of work to date involves construction and theoretical analysis of a new class of approximate dynamic programming algorithms.

2.2 Approximate Dynamic Programming Algorithm Development and Implementation [6, 22, 21]

As discussed, the need to solve large instances of health-aware, multi-agent planning problems has led to the development of new approximate dynamic programming algorithms. These algorithms are similar in spirit to the class of Bellman residual minimization methods [85, 7, 63, 4], which perform approximate policy evaluation by minimizing the Bellman error, given by

$$\left(\sum_{i \in \widetilde{\mathcal{S}}} |\widetilde{J}_\mu(i) - T_\mu \widetilde{J}_\mu(i)|^2 \right)^{1/2}, \tag{5}$$

over a set of candidate cost-to-go functions $\widetilde{J}_\mu(\cdot)$. Here, $\widetilde{\mathcal{S}}$ is a set of representative sample states in the MDP, and the individual terms $(\widetilde{J}_\mu(i) - T_\mu \widetilde{J}_\mu(i))$ are referred to as the *Bellman residuals*. The motivation behind our algorithms is the observation that, given the Bellman error (5) as the objective function to be minimized, we should seek to find a solution for which the objective is identically zero, the smallest possible value. By exploiting the flexibility of nonparametric, kernel-based function approximation architectures, we have developed a new class of algorithms that can be proven to always find a cost-to-go solution whose Bellman residuals are exactly zero at the sample states. Because of this property, we have named our approach *Bellman Residual Elimination* (BRE) [6, 22, 21]. A number of desirable features of the algorithms can be proved because of the BRE property, including the fact that they reduce to exact policy iteration in the limit of sampling the entire state space.

In addition to development of the algorithms and demonstration of these theoretical properties, the BRE algorithms have been shown to successfully solve several small test problems. Furthermore, we have demonstrated that the calculations necessary to carry out the algorithms can be naturally distributed over a cluster of networked computational resources, leading to faster solution times. To date, a software architecture for performing the BRE algorithms on a high-performance, scalable computer cluster has been developed and validated. Ongoing work is using this software on a 24-processor cluster to apply BRE to the large-scale planning problems discussed in the previous section.

2.3 Adaptive MDP-Based Planning [20, 19, 18, 73]

The third area of work to date addresses the problem of model uncertainty in MDP-based planning approaches. Specifically, since any control policy computed using an MDP-based approach relies on an underlying specification of a dynamic model, the performance achieved by the policy when implemented on the real system may be degraded if the assumed model is inaccurate. Furthermore, in many cases, an accurate model may be be available until the real system begins operation, at which point observations of the system become available and it is possible to begin estimating relevant model parameters. To cope with these challenges, we have been investigating a number of off-line and on-line strategies. The off-line strategies deal with ways of computing policies that are robust to modeling uncertainties [18]. In this approach, the uncertainty in the model is quantified (using, for example, covariance estimates of the important model parameters), and a robust policy is found using a minimax-like approach. In contrast, the on-line strategies are focused on simultaneous model estimation (using observations of the true system) and re-solving the MDP to compute the optimal policy with respect to the updated model estimate [20, 19]. Flight results of this on-line, adaptive approach are shown in Figure 2. The results demonstrate the value of the adaptive approach; as the mission progresses, the model estimate improves, and thus the updated policy is able to achieve improved mission performance. Ongoing work is also investigating the possibility of active learning in the on-line setting, where the control policy actively selects actions that are likely to lead to faster convergence of the model estimate [73].

3 Proposed Work

To complete the thesis work, we propose the following tasks, which build upon the areas of previous work discussed above:

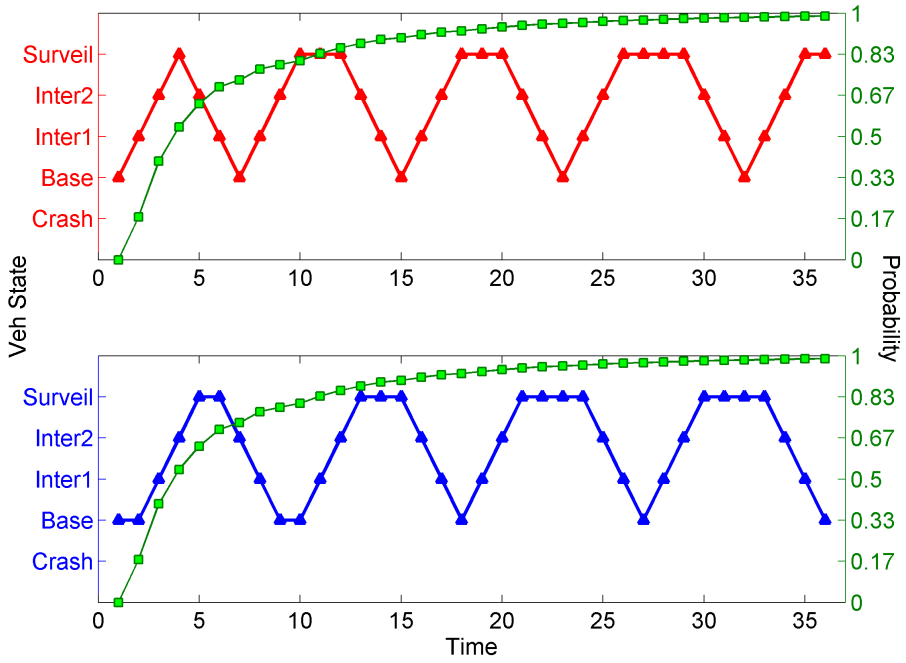

Figure 2: Persistent surveillance mission adaptive flight results. As the system is running, a model estimator continuously updates its estimate of an important fuel flow parameter (shown in green), which directly influences the time each UAV (shown in red and blue) can spend on-station. As the mission progresses, the estimate converges to the true value, and the control policy adjusts to allow the UAVs to stay on-station longer, resulting in improved mission performance.

3.1 Further BRE Algorithm Development/Extension

First, we propose a number of tasks related to further development and extension of the BRE approach developed to date.

"n-stage BRE" The BRE algorithms developed to date effectively solve the single-stage, fixed-policy Bellman equation, given by

$$T_\mu J_\mu = J_\mu,$$

over a set of representative sample states. However, this Bellman equation can be expressed in an equivalent form by simply applying the dynamic programming operator T_μ multiple times, leading to an equation of the form

$$T_\mu^n J_\mu = J_\mu. \tag{6}$$

It is possible to devise BRE algorithms that effectively solve (6) for general n in addition to the special case $n = 1$, which has been developed to date. These n-stage BRE algorithms may be interesting from both a theoretical standpoint (providing a better understanding of the general approach) and from an applications point of view (since they may lead to better cost approximations and policies). We propose to develop these n-stage BRE algorithms, apply them to test problems, and compare their performance with single-stage BRE.

Investigation of manifold-based kernels, and their relationship to n-stage BRE A number of researchers have proposed using kernels that exploit manifold structure on the state space as a means of devising cost approximation algorithms [90, 91, 59, 60, 10, 9, 87]. We believe that these kernels are particularly appropriate for use in our BRE algorithms, and propose to test these manifold-based kernels in several BRE test problems. Furthermore, we believe that there may be an interesting connection between manifold-based kernels and the n-stage BRE algorithms discussed

previously. This connection arises because both manifold-based kernels and n-stage BRE examine the local structure of the state space. We propose to investigate this connection.

Decentralization of BRE We have already demonstrated that the calculations necessary to perform BRE can be carried out in parallel on a distributed computer cluster. Related to this, there is interest in devising algorithms that can be carried out in a fully decentralized sense, using a team of agents with limited communication bandwidth. We propose to investigate methods for running BRE in this decentralized sense.

Extension of BRE to model-free learning The BRE algorithms developed to date are model-based. By using the system model, the BRE algorithms completely avoid the need to perform trajectory simulations and thus avoid the problem of simulation noise. However, nominal BRE is not applicable in situations where the model is unknown. We propose to investigate ways to construct model-free BRE algorithms for use in these situations.

3.2 Large-Scale Health Management Flight Demonstrations

The second area of development focuses on application of the BRE approach to a large-scale, multi-agent planning problem with health management. We plan to use the MIT RAVEN flight testbed [53] to carry out a number of flight experiments which demonstrate the applicability of both the persistent surveillance MDP problem formulation and the BRE approximate dynamic programming approach to interesting real-world problems. These demonstrations will incorporate a number of UAVs and ground vehicles cooperating to provide mission services such as searching for and tracking targets of interest. The missions will be performed on time-scales longer than the flight times of any individual UAV, resulting in a need for the proactive planning capabilities provided by our MDP problem formulation.

A major goal of the flight experiments will be to demonstrate that our BRE algorithms yield policies that are near-optimal for the complex persistent surveillance problem. In addition, we propose to carry out adaptive flight experiments similar to the ones performed in [20, 19], except that we will use our approximate BRE techniques as the solution mechanism, rather than the exact technique (value iteration) which has been used to date. With these results, we hope to show that BRE is an effective technology for multi-agent planning problems, and can be used successfully even in large, complex problems and in cases where the system model is not perfectly known.

References

[1] A. Samuel. Some studies in machine learning using the game of checkers. *IBM J. of Research and Development*, 3:210–229, 1959.

[2] American Institute of Aeronautics and Astronautics). Worldwide UAV Roundup. Available at http://www.aiaa.org/images/PDF/WilsonChart.pdf, 2007.

[3] K. Andersson, W. Hall, S. Atkins, and E. Feron. Optimization-Based Analysis of Collaborative Airport Arrival Planning. *Transportation Science*, 37(4):422–433, November 2003.

[4] A. Antos, C. Szepesvári, and R. Munos. Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71(1):89–129, 2008.

[5] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

[6] B. Bethke, J. How, A. Ozdaglar. Approximate Dynamic Programming Using Support Vector Regression. In *Proceedings of the 2008 IEEE Conference on Decision and Control*, Cancun, Mexico, 2008.

[7] L. C. Baird. Residual algorithms: Reinforcement learning with function approximation. In *ICML*, pages 30–37, 1995.

[8] C. Barnhart, F. Lu, and R. Shenoi. Integrated Airline Scheduling. In G. Yu, editor, *Operations Research in the Airline Industry*, volume 9 of *International Series in Operations Research and Management Science*, pages 384–422. Kluwer Academic Publishers, 1998.

[9] M. Belkin and P. Niyogi. Semi-supervised learning on riemannian manifolds. *Machine Learning*, 56(1-3):209–239, 2004.

[10] M. Belkin and P. Niyogi. Towards a theoretical foundation for laplacian-based manifold methods. In Peter Auer and Ron Meir, editors, *COLT*, volume 3559 of *Lecture Notes in Computer Science*, pages 486–500. Springer, 2005.

[11] R. Bellman. The theory of dynamic programming. *Bull. Amer. Math. Soc.*, 60:503–515, 1954.

[12] K. Bennett and C. Campbell. Support vector machines: Hype or hallelujah ? *SIGKDD Explorations*, 2(2), 2000.

[13] C. Berg, J. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups*. Springer-Verlag, New York, 1984.

[14] D. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA, 2007.

[15] D. Bertsekas and S. Ioffe. Temporal Differences-Based Policy Iteration and Applications in Neuro-Dynamic Programming. http://web.mit.edu/people/dimitrib/Tempdif.pdf, 1996.

[16] D. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[17] D. Bertsimas and S. S. Patterson. The Air Traffic Flow Management Problem with Enroute Capacities. *Operations Research*, 46(3):406–422, May-June 1998.

[18] L. Bertuccelli, B. Bethke, and J. How. Robust adaptive markov decision processes in multi-vehicle applications. In *Proceedings of the American Control Conference (to appear)*, 2009.

[19] B. Bethke, L. Bertuccelli, and J. How. Real-time adaptive mdp-based planning. *IEEE Robotics and Automation Magazine (to appear)*, 2008.

[20] B. Bethke, L. Bertuccelli, and J. P. How. Experimental Demonstration of MDP- Based Planning with Model Uncertainty. In *AIAA Guidance Navigation and Control Conference*, Aug 2008. AIAA-2008-6322.

[21] B. Bethke and J. How. Approximate dynamic programming using bellman residual elimination and gaussian process regression. In *Proceedings of the American Control Conference (to appear)*, 2009.

[22] B. Bethke, J. How, and A. Ozdaglar. Kernel-based reinforcement learning using bellman residual elimination. *Journal of Machine Learning Research (to appear)*, 2008.

[23] B. Bethke, J. How, and J. Vian. Group health management of UAV teams with applications to persistent surveillance. In *Proceedings of the American Control Conference*, 2008.

[24] C. M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995.

[25] V.S. Borkar. A convex analytic approach to Markov decision processes. *Probability Theory and Related Fields*, 78(4):583–602, 1988.

[26] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Computational Learing Theory*, pages 144–152, 1992.

[27] L. Bottou. *Large scale kernel machines*. MIT Press, Cambridge, MA, 2007.

[28] Justin A. Boyan. Technical update: Least-squares temporal difference learning. *Machine Learning*, 49(2-3):233–246, 2002.

[29] Steven J. Bradtke and Andrew G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1-3):33–57, 1996.

[30] J. Candela and C. Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

[31] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[32] L. Clarke, E. Johnson, G. Nemhauser, and Z. Zhu. The Aircraft Rotation Problem. *Annuls of Operations Research*, 69:33–46, 1997.

[33] Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39(1), 2002.

[34] D. P. de Farias. *The Linear Programming Approach to Approximate Dynamic Programming: Theory and Application*. PhD thesis, Stanford University, June 2002.

[35] D.P. de Farias and B. Van Roy. Approximate linear programming for average-cost dynamic programming. *Advances in Neural Information Processing Systems*, 15:1587–1594, 2003.

[36] DP de Farias and B. Van Roy. The Linear Programming Approach to Approximate Dynamic Programming. *Operations Research*, 51(6):850–865, 2003.

[37] D.P. de Farias and B. Van Roy. A Cost-Shaping LP for Bellman Error Minimization with Performance Guarantees. *Advances in Neural Information Processing Systems 17: Proceedings Of The 2004 Conference*, 2005.

[38] E.V. Denardo. On linear programming in a Markov decision problem. *Management Science*, 16(5):282–288, 1970.

[39] T. Dietterich and X. Wang. Batch value function approximation via support vectors. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *NIPS*, pages 1491–1498. MIT Press, 2001.

[40] Y. Engel. *Algorithms and Representations for Reinforcement Learning*. PhD thesis, Hebrew University, 2005.

[41] Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with gaussian processes. In Luc De Raedt and Stefan Wrobel, editors, *ICML*, volume 119 of *ACM International Conference Proceeding Series*, pages 201–208. ACM, 2005.

[42] Y. Engel, S. Mannor, and Ron Meir. Bayes meets bellman: The gaussian process approach to temporal difference learning. In Tom Fawcett and Nina Mishra, editors, *ICML*, pages 154–161. AAAI Press, 2003.

[43] P. Gaudiano, B. Shargel, E. Bonabeau, and B. Clough. Control of UAV SWARMS: What the Bugs Can Teach Us. In *Proceedings of the 2nd AIAA Unmanned Unlimited Systems, Technologies, and Operations Aerospace Conference*, San Diego, CA, September 2003.

[44] F. Girosi. An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10(6):1455–1480, 1998.

[45] R. Gopalan and K. Talluri. The Aircraft Maintenance Routing Problem. *Operations Research*, 46(2):260–271, March-April 1998.

[46] G.H. Hardy. *Ramanujan: Twelve Lectures on Subjects Suggested by His Life and Work, 3rd ed.* Chelsea, New York, 1999.

[47] J. C. Hartman and J. Ban. The series-parallel replacement problem. *Robotics and Computer Integrated Manufacturing*, 18:215–221, 2002.

[48] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan, New York, 1994.

[49] J. Higle and A. Johnson. Flight Schedule Planning with Maintenance Considerations. Submitted to OMEGA, The International Journal of Management Science, 2005.

[50] M. J. Hirsch, P. M. Pardalos, R. Murphey, and D. Grundel, editors. *Advances in Cooperative Control and Optimization. Proceedings of the 7th International Conference on Cooperative Control and Optimization*, volume 369 of *Lecture Notes in Control and Information Sciences*. Springer, Nov 2007.

[51] A. Hordijk and LCM Kallenberg. Linear programming and Markov decision chains. *Management Science*, 25(4):352–362, 1979.

[52] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[53] J. How, B. Bethke, A. Frank, D. Dale, and J. Vian. Real-Time Indoor Autonomous Vehicle Test Environment. *IEEE Control Systems Magazine*, April 2008.

[54] S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. http://citeseer.ist.psu.edu/244558.html, 1999.

[55] A. Kott. *Advanced Technology Concepts for Command and Control*. Xlibris Corporation, 2004.

[56] M. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

[57] M. Valenti, B. Bethke, J. How, D. Pucci de Farias, J. Vian. Embedding Health Management into Mission Tasking for UAV Teams. In *American Controls Conference*, New York, NY, June 2007.

[58] M. Valenti, D. Dale, J. How, and J. Vian. Mission Health Management for 24/7 Persistent Surveillance Operations. In *Submitted to the 2007 AIAA Guidance, Control and Navigation Conference*, Myrtle Beach, SC, August 2007.

[59] S. Mahadevan. Proto-value functions: Developmental reinforcement learning. In *International Conference on Machine Learning*, 2005.

[60] S. Mahadevan and M. Maggioni. Value function approximation with diffusion wavelets and laplacian eigenfunctions. In *NIPS*, 2005.

[61] A.S. Manne. Linear programming and sequential decisions. *Management Science*, 6(3):259–267, 1960.

[62] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. In *IEEE Neural Networks*, number 12(2), pages 181–201, 2001.

[63] R. Munos and C. Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 1:815–857, 2008.

[64] A. Nedic and Dimitri P. Bertsekas. Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamic Systems*, 13(1-2):79–110, 2003.

[65] Office of the Secretary of Defense. Unmanned Aircraft Systems Roadmap, available online http://www.acq.osd.mil/usd/Roadmap Final2.pdf. Technical report, OSD, 2005.

[66] D. Ormoneit and Ś. Sen. Kernel-Based Reinforcement Learning. *Machine Learning*, 49(2):161–178, 2002.

[67] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop*, pages 276–285, Sep 1997.

[68] H. Paruanak, S. Brueckner, and J. Odell. Swarming Coordination of Multiple UAV's for Collaborative Sensing. In *Proceedings of the 2nd AIAA Unmanned Unlimited Systems, Technologies, and Operations Aerospace Conference*, San Diego, CA, September 2003.

[69] J. Platt. Using sparseness and analytic QP to speed training of support vector machines. In *Advances in Neural Information Processing Systems*, pages 557–563, 1999.

[70] John Platt. *Fast training of support vector machines using sequential minimal optimization in Advances in Kernel Methods — Support Vector Learning*. MIT Press, 1999.

[71] C. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. *Advances in Neural Information Processing Systems*, 16:751–759, 2004.

[72] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[73] J. Redding, B. Bethke, L. Bertuccelli, and J. How. Experimental demonstration of exploration toward model learning under an adaptive mdp-based planner. In *Proceedings of the AIAA Infotech Conference (to appear)*, 2009.

[74] J. Reisinger, P. Stone, and R. Miikkulainen. Online kernel selection for bayesian reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[75] Richard M. Murray. Recent research in cooperative control of multi-vehicle systems. *ASME Journal of Dynamic Systems, Measurement, and Control (submitted) special issue on the ?Analysis and Control of Multi Agent Dynamic Systems?*, 2007.

[76] J. Rosenberger, E. Johnson, and G. Nemhauser. Rerouting Aircraft for Airline Recovery. *Transportation Science*, 37(4):408–421, November 2003.

[77] R.S. Sutton. Learning to Predict by the Methods of Temporal Differences. *Machine Learning*, 3:9–44, 1988.

[78] Stuart Jonathan Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall series in artificial intelligence. Prentice Hall, Upper Saddle River, 2nd edition, 2003.

[79] S. Saitoh. *Theory of Reproducing Kernels and its Applications*. Longman Scientific and Technical, Harlow, England, 1988.

[80] Craig Saunders, Alexander Gammerman, and Volodya Vovk. Ridge regression learning algorithm in dual variables. In Jude W. Shavlik, editor, *ICML*, pages 515–521. Morgan Kaufmann, 1998.

[81] B. Schölkopf. Support Vector Learning. Available from http://www.kyb.tuebingen.mpg.de/-bs, 1997.

[82] B. Schölkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. In *First International Conference on Knowledge Discovery and Data Mining*, Menlo Park, 1995. AAAI Press.

[83] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2002.

[84] B. Schölkopf, A. Smola, R. Williamson, and P. Bartlett. New support vector algorithms, 2000.

[85] P. Schweitzer and A. Seidman. Generalized polynomial approximation in Markovian decision processes. *Journal of mathematical analysis and applications*, 110:568–582, 1985.

[86] C. E. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41:256–275, 1950.

[87] W. Smart. Explicit manifold representations for value-function approximation in reinforcement learning. In *AMAI*, 2004.

[88] A. Smola and B. Schölkopf. A Tutorial on Support Vector Regression. *Statistics and Computing*, 14:199–222, 2004.

[89] E. Solak, Roderick Murray-Smith, William E. Leithead, Douglas J. Leith, and Carl Edward Rasmussen. Derivative observations in gaussian process models of dynamic systems. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *NIPS*, pages 1033–1040. MIT Press, 2002.

[90] M. Sugiyama, H. Hachiya, C. Towell, and S. Vijayakumar. Geodesic gaussian kernels for value function approximation. In *Workshop on Information-Based Induction Sciences*, 2006.

[91] M. Sugiyama, H. Hachiya, C. Towell, and S. Vijayakumar. Value function approximation on non-linear manifolds for robot motor control. In *Proc. of the IEEE International Conference on Robotics and Automation*, 2007.

[92] R. Sutton and A. Barto. *Reinforcement learning: An introduction*. MIT Press, 1998.

[93] R. S. Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts, Dept. of Comp. and Inf. Sci., 1984.

[94] G. Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 8:257–277, 1992.

[95] G. Tesauro. Temporal difference learning and TD-Gammon. *Commun. ACM*, 38(3):58–68, 1995.

[96] J. Tobias and P. Daniel. Least squares svm for least squares td learning. In Gerhard Brewka, Silvia Coradeschi, Anna Perini, and Paolo Traverso, editors, *ECAI*, pages 499–503. IOS Press, 2006.

[97] M. Valenti. *Approximate Dynamic Programming with Applications in Multi-Agent Systems*. PhD thesis, Massachusetts Institute of Technology, 2007.

[98] V. N. Vapnik. An overview of statistical learning theory. *Neural Networks, IEEE Transactions on*, 10(5):988–999, 1999.

[99] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[100] G. Wahba. Spline Models for Observational Data. In *Vol. 59 of CBMS-NSF Regional Conference Series in Applied Mathematics*, SIAM, Philidelphia, 1990.

[101] Jian Xiong Dong, Adam Krzyzak, and Ching Y. Suen. Fast svm training algorithm with decomposition on very large data sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(4):603–618, 2005.