Proc. of 2010 Conference on Decision and Control, Atlanta, Ga., Dec. 2010

Pathologies of Temporal Difference Methods in Approximate Dynamic Programming

Dimitri P. Bertsekas

***Abstract*—Approximate policy iteration methods based on temporal differences are popular in practice, and have been tested extensively, dating to the early nineties, but the associated convergence behavior is complex, and not well understood at present. An important question is whether the policy iteration process is seriously hampered by oscillations between poor policies, roughly similar to the attraction of gradient methods to poor local minima. There has been little apparent concern in the approximate DP/reinforcement learning literature about this possibility, even though it has been documented with several simple examples. Recent computational experimentation with the game of tetris, a popular testbed for approximate DP algorithms over a 15-year period, has brought the issue to sharp focus. In particular, using a standard set of 22 features and temporal difference methods, an average score of a few thousands was achieved. Using the same features and a random search method, an overwhelmingly better average score was achieved (600,000-900,000). The paper explains the likely mechanism of this phenomenon, and derives conditions under which it will not occur.**

I. Introduction

In this paper we discuss some phenomena that hamper the effectiveness of approximate policy iteration methods for finite-state stochastic dynamic programming (DP) problems. These are iterative methods that are patterned after the classical policy iteration method, with each iteration involving evaluation of the cost vector of a current policy, and a policy improvement process aiming to obtain an improved policy.

We focus on the classical discounted finite-state Markovian Decision Problem (MDP) as described in textbooks such as [Ber07] and [Put94]. Here, for a given policy μ, the policy evaluation phase involves the approximate solution of the Bellman equation

$$J = g_\mu + \alpha P_\mu J, \tag{1}$$

where $g_\mu \in \Re^n$ is the one-stage cost vector of the policy μ, P_μ is its $n \times n$ transition probability matrix, and $\alpha \in (0,1)$ is a discount factor.[1] The unique solution of Eq. (1) is denoted J_μ and is the cost vector of policy μ.

In the most common form of approximate policy iteration, we introduce an $n \times r$ matrix Φ whose columns can be viewed as basis functions, and we approximate the cost vectors of policies by vectors in the range of Φ:

$$S = \{\Phi r \mid r \in \Re^s\}.$$

Thus, given μ, an approximation of J_μ that has the form Φr_μ is used to generate an (approximately) improved policy $\overline{\mu}$ via the equation

$$\overline{\mu} = \arg \min_{\mu' \in M} \big[g_{\mu'} + \alpha P_{\mu'} \Phi r_\mu\big], \tag{2}$$

where M is the set of admissible policies, and the minimization is done separately for each state/component. This approach is described in detail in the literature, has been extensively tested in practice, and is one of the major methodologies for approximate DP (see the books by Bertsekas and Tsitsiklis [BeT96], Sutton and Barto [SuB98], Gosavi [Gos03], Cao [Cao07], Chang, Fu, Hu, and Marcus [CFH07], Meyn [Mey07], Powell [Pow07], and Borkar [Bor08]; the textbook [Ber07] together with its on-line chapter [Ber10a] provide a recent treatment and up-to-date references). The main advantage of using approximations of the form Φr is that the algorithm can be implemented using Monte-Carlo simulation using exclusively low-dimensional linear algebra calculations (of order s rather than n).

An important question is whether the policy iteration process will converge or whether it will cycle between several policies. Convergence is guaranteed in the case where the policy evaluation is exact (Φ is the identity matrix), or is done using an aggregation method (to be described later; see Section IV). However, the process may also be cycling among several (possibly poor) policies. There has been little apparent concern in the approximate DP/reinforcement learning literature about this possibility, even though it was documented with several relatively simple examples in the book [BeT96]. Unfortunately recent research developments with the game of tetris, a popular testbed for reinforcement learning algorithms for more than 15 years ([Van95], [TsV96], [BeI96], [Kak02], [FaV06], [SzL06], [DFM09], [ThS09]), have brought the issue to sharp focus and suggest that policy oscillations can prevent the attainment of good performance. In particular, using a set of 22 features introduced by Bertsekas and Ioffe [BeI96],

Dimitri Bertsekas is with the Dept. of Electr. Engineering and Comp. Science, M.I.T., Cambridge, Mass., 02139. dimitrib@mit.edu

Dimitri Bertsekas was supported by NSF Grant ECCS-0801549, by the LANL Information Science and Technology Institute, and by the Air Force Grant FA9550-10-1-0412.

Many thanks are due to Huizhen (Janey) Yu for many helpful discussions and suggestions.

[1]In our notation $\Re^n$ is the n-dimensional Euclidean space, all vectors in $\Re^n$ are viewed as column vectors, and a prime denotes transposition. The identity matrix is denoted by I.

and temporal difference methods (the LSPE method) an average score of a few thousands was achieved (similar results were obtained with the LSTD method [LaP03]). Using the same features and a random search method in the space of weight vectors r, an average score of 600,000-900,000 was achieved [SzL06], [ThS09]. These case studies, together with the oscillatory computational results of [BeI96] (also reproduced in [BeT96], Section 8.3), suggest that in the tetris problem, policy iteration using the projected equation is seriously hampered by oscillations between poor policies, roughly similar to the attraction of gradient methods to poor local minima.

Moreover related phenomena may be causing similar (and hard to detect) difficulties in other related approximate DP methodologies: approximate policy iteration with the Bellman error method, policy gradient methods, and approximate linear programming. In particular, the tetris problem, using the same 22 features, has been addressed by approximate linear programming [FaV06], [DFM09], and with a policy gradient method [Kak02], also with a grossly suboptimal achieved average score of a few thousands.

This paper has a dual purpose. The first purpose is to draw attention to the significance of policy oscillations, as an issue that has been underestimated, and has the potential to alter in fundamental ways our thinking about approximate DP. The second purpose is to propose approximate policy iteration methods with guaranteed convergence/termination. In Section II we provide some background on approximate policy iteration, while in Section III, we describe the mechanism for oscillations. In Section IV, we describe some methods to prevent oscillations and the restrictions that they require. This leads to specific methods, some new and some known, which not only don't exhibit oscillations but also achieve a more favorable order of error bound.

II. Approximate Policy Iteration

Let us first introduce some notation. We introduce the mapping $T : \Re^n \mapsto \Re^n$ defined by

$$(TJ)(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J(j)\big),$$

for $i = 1, \ldots, n$, where $p_{ij}(u)$ is the transition probability from state i to state j, as a function of the control u, and $g(i,u,j)$ is the one-stage cost. For a policy μ we introduce the mapping $T_\mu : \Re^n \mapsto \Re^n$ defined by

$$(T_\mu J)(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\big(g(i,\mu(i),j) + \alpha J(j)\big),$$

for $i = 1, \ldots, n$. The optimal cost vector J^* is the unique fixed point of T, while J_μ, the cost vector of μ, is the unique fixed point of T.

We consider a standard approximate policy iteration method with linear cost function approximation. In this method, given Φr_μ, an approximation of the cost vector of the current policy μ, we generate an "improved" policy $\overline{\mu}$ using the formula $T_{\overline{\mu}}(\Phi r_\mu) = T(\Phi r_\mu)$, i.e., for all i

$$\overline{\mu}(i) = \arg\min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j)+\alpha\phi(j)'r_\mu\big), \quad (3)$$

where $\phi(j)'$ is the row of Φ that corresponds to state j. The method terminates with μ if $T_\mu(\Phi r_\mu) = T(\Phi r_\mu)$. We then repeat with μ replaced by $\overline{\mu}$.

The theoretical basis for the preceding approximate policy iteration method is given in Prop. 6.2 of [BeT96] (also Prop. 1.3.6 of [Ber07]), where it was shown that if the policy evaluation is accurate to within δ (in the sup-norm sense $\|\Phi r_\mu - J_\mu\|_\infty \leq \delta$), then the method will yield a sequence of policies $\{\mu^k\}$ such that

$$\limsup_{k\to\infty} \|J_{\mu^k} - J^*\|_\infty \leq \frac{2\alpha\delta}{(1-\alpha)^2}. \quad (4)$$

Experimental evidence indicates that this bound, while tight in theory ([BeT96], Example 6.4), tends to be conservative in practice. Furthermore, often just a few policy evaluations are needed before the bound is attained. Still, however, given that $\alpha \approx 1$, and that the sup-norm error of the policy evaluation process (and hence δ) is potentially large, the bound (4) is far from reassuring.

When the policy sequence $\{\mu^k\}$ terminates with some $\overline{\mu}$, the much sharper bound

$$\|J_{\overline{\mu}} - J^*\|_\infty \leq \frac{2\alpha\delta}{1-\alpha} \quad (5)$$

holds. To show this, let $\overline{J}$ be the cost vector obtained by policy evaluation of $\overline{\mu}$, and note that it satisfies $\|\overline{J} - J_{\overline{\mu}}\|_\infty \leq \delta$ (by assumption) and $T\overline{J} = T_{\overline{\mu}}\overline{J}$ (since μ^k converges to $\overline{\mu}$). We write

$$\begin{aligned} TJ_{\overline{\mu}} &\geq T(\overline{J} - \delta e) = T\overline{J} - \alpha\delta e \\ &= T_{\overline{\mu}}\overline{J} - \alpha\delta e \geq T_{\overline{\mu}}J_{\overline{\mu}} - 2\alpha\delta e \\ &= J_{\overline{\mu}} - 2\alpha\delta e, \end{aligned}$$

where e is the unit vector, from which by repeatedly applying T to both sides, we obtain

$$J^* = \lim_{m\to\infty} T^m J_{\overline{\mu}} \geq J_{\overline{\mu}} - \frac{2\alpha\delta}{1-\alpha}e,$$

thereby showing the error bound (5). A comparison of the two bounds (4) and (5) suggests an important advantage for methods that guarantee policy convergence.

Let us now discuss algorithmic approaches for approximate policy evaluation. The most popular are:

(a) *Projected equation approach*: Here we solve the projected equation

$$\Phi r = \Pi T_\mu(\Phi r),$$

where Π denotes projection onto the subspace $S = \{\Phi r \mid r \in \Re^s\}$. The projection is with respect to a weighted Euclidean norm $\|\cdot\|_\xi$, where $\xi = (\xi_1,\ldots,\xi_n)$ is a probability distribution with positive components (i.e., $\|x\|_\xi^2 = \sum_{i=1}^n \xi_i x_i^2$, where $\xi_i > 0$ for all i).

(b) *Aggregation approach*: Here we solve an equation of the form

$$\Phi r = \Phi D T_\mu(\Phi r),$$

where Φ and D are matrices whose rows are restricted to be probability distributions. The vector r has an interpretation as a cost vector of an aggregate problem that has s states and is defined by Φ and D. Note that there is a probabilistic structure restriction in the form of Φ, while there is no such restriction in the projected equation approach.

Generally, the projected equation approach is associated with *temporal difference* (TD) methods, which originated in reinforcement learning with the works of Samuel [Sam59], [Sam67] on a checkers-playing program. The papers by Barto, Sutton, and Anderson [BSA83], and Sutton [Sut88] proposed the TD(λ) method, which motivated a lot of research in simulation-based DP, particularly following an early success with the backgammon playing program of Tesauro [Tes92]. The original papers did not make the connection of TD methods with the projected equation, and for quite a long time it was not clear which mathematical problem TD(λ) was aiming to solve! The convergence properties of TD(λ) and its connections with the projected equation were clarified in the mid 90s through the works of several authors, see e.g., Tsitsiklis and Van Roy [TsV97]. More recent works have focused on the use of least squares-based TD methods, such as the LSTD (Least Squares Temporal Differences) method (Bradtke and Barto [BrB96]), and the LSPE (Least Squares Policy Evaluation) method (Bertsekas and Ioffe [BeI96]).

The aggregation approach also has a long history in scientific computation and operations research. It was introduced in the simulation-based approximate DP context, mostly in the form of value iteration; see Singh, Jaakkola, and Jordan [SJJ94], [SJJ95], Gordon [Gor95], and Tsitsiklis and Van Roy [TsV96] (for textbook presentations of aggregation, see [Ber07], [Ber10a]). Currently the aggregation approach seems to be less popular, but as we will argue in this paper, it has some interesting advantages over the projected equation approach, even though there are restrictions in its applicability.

An important fact, to be shown later, is that when the aggregation approach is used for policy evaluation, the policy sequence $\{\mu^k\}$ terminates with some $\overline{\mu}$, while this is generally not true for the projected equation approach.

III. Policy Oscillations

Projected equation-based variants of policy iteration methods are popular in practice, and have been tested extensively, dating to the early nineties (see e.g., the books [BeT96], [SuB08], and the references quoted there; for a sample of more recent experimental studies, see Lagoudakis and Parr [LaP03], Jung and Polani [JuP07], and Busoniu et al. [BED09]), but the associated convergence behavior is complex, and involves potentially damaging oscillations that are not well understood at present. This behavior was first described, together with the attendant policy oscillation and chattering phenomena, by the author at an April 1996 workshop on reinforcement learning [Ber96], and was subsequently discussed in Section 6.4 of [BeT96].

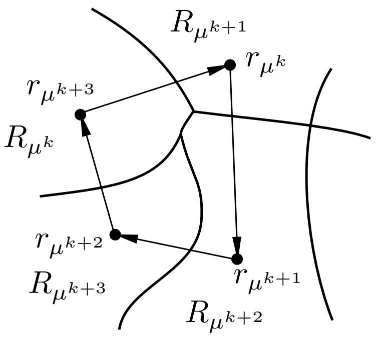


Fig. 1. Greedy partition and cycle of policies generated by nonoptimistic policy iteration. Thus μ yields $\overline{\mu}$ by policy improvement if and only if $r_\mu \in R_{\overline{\mu}}$. In this figure, the method cycles between four policies and the corresponding four parameters r_{μ^k} $r_{\mu^{k+1}}$ $r_{\mu^{k+2}}$ $r_{\mu^{k+3}}$.

To get a sense of this behavior, we introduce the so called *greedy partition*. This is a partition of the space $\Re^s$ of parameter vectors r into subsets R_μ, each subset corresponding to a stationary policy μ, and defined by

$$R_\mu = \left\{ r \;\middle|\; \mu(i) = \arg\min_{u\in U(i)} \sum_{j=1}^n p_{ij}(u) \right.$$
$$\left. \big(g(i,u,j) + \alpha\phi(j)'r\big),\ i = 1,\ldots,n \right\}.$$

Thus, R_μ is the set of parameter vectors r for which μ is greedy with respect to Φr.

For simplicity, let us assume that we use a policy evaluation method that for each given μ produces a unique parameter vector r_μ. Nonoptimistic policy iteration starts with a parameter vector r_0, which specifies μ^0 as a greedy policy with respect to Φr_0, and generates r_{μ^0} by using the given policy evaluation method. It then finds a policy μ^1 that is greedy with respect to Φr_{μ^0}, i.e., a μ^1 such that $r_{\mu^0} \in R_{\mu^1}$. It then repeats the process with μ^1 replacing μ^0. If some policy μ^k satisfying $r_{\mu^k} \in R_{\mu^k}$ is encountered, the method keeps generating that policy. This is the necessary and sufficient condition for policy convergence in the approximate policy iteration method. Of course, the mere fact that a policy iteration method is guaranteed to converge is not in itself a guarantee of good performance, beyond the fact that a better error bound holds in this case [cf. Eqs. (5) and (4)].

In the case of a lookup table representation where the parameter vectors r_μ are equal to the cost-to-go vector J_μ, the condition $r_{\mu^k} \in R_{\mu^k}$ is equivalent to $r_{\mu^k} = Tr_{\mu^k}$, and is satisfied if and only if μ^k is optimal. When there is cost function approximation, however, this condition need not be satisfied for any policy. In this case, since there is a finite number of possible vectors r_μ, one generated from another in a deterministic way, the algorithm ends up repeating some cycle of policies $\mu^k, \mu^{k+1}, \ldots, \mu^{k+m}$ with

$$r_{\mu^k} \in R_{\mu^{k+1}}, r_{\mu^{k+1}} \in R_{\mu^{k+2}}, \ldots, \\ r_{\mu^{k+m-1}} \in R_{\mu^{k+m}},\ r_{\mu^{k+m}} \in R_{\mu^k};$$

(see Fig. 1). Furthermore, there may be several different cycles, and the method may end up converging to any one of them depending on the starting policy μ^0. The actual cycle obtained depends on the initial policy μ^0. This is reminiscent of gradient methods applied to minimization of functions with multiple local minima, where the limit of convergence depends on the starting point. Furthermore, the policies obtained may be quite bad, subject only to the error bound (4) The following example illustrates policy oscillation.

Example 1: Consider a discounted problem with two states, 1 and 2, illustrated in Fig. 2(a). There is a choice of control only at state 1, and there are two policies, denoted μ^* and μ. The optimal policy μ^* when at state 1 stays at 1 with probability $p > 0$ and incurs a negative cost c. The other policy is μ and cycles between the two states with 0 cost. We consider cost approximation of the form $i\,r$, so $\Phi = (1,2)'$.

Let us construct the greedy partition. We have

$$R_\mu = \Big\{ r \mid p\big(c + \alpha(1\cdot r)\big) + (1-p)\alpha(2\cdot r) \ge \alpha(2\cdot r) \Big\} \\ = \{r \mid c \ge \alpha r\}, \\ R_{\mu^*} = \{r \mid c \le \alpha r\}.$$

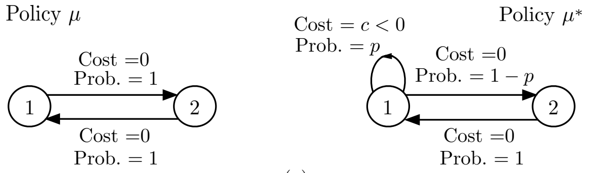
We next calculate the points r_μ and r_{μ^*} that solve the projected equations $C_\mu r_\mu = d_\mu$ and $C_{\mu^*} r_{\mu^*} = d_{\mu^*}$, which correspond to μ and μ^*, respectively. We have

$$C_\mu = \Phi'\Xi_\mu(1-\alpha P_\mu)\Phi \\ = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & -\alpha \\ -a & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = 5 - 9\alpha,$$

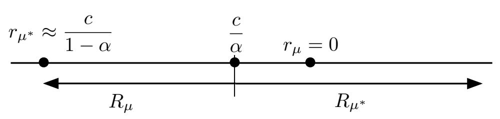
$$d_\mu = \Phi'\Xi_\mu g_\mu = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix} = 0,$$

so $r_\mu = 0$. Similarly, with some calculation,

$$C_{\mu^*} = \Phi'\Xi_{\mu^*}(1-\alpha P_{\mu^*})\Phi \\ = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \begin{pmatrix} \frac{1}{2-p} & 0 \\ 0 & \frac{1-p}{2-p} \end{pmatrix} \begin{pmatrix} 1-\alpha p & -\alpha(1-p) \\ -a & 1 \end{pmatrix} \begin{pmatrix} c \\ 0 \end{pmatrix} \\ = \frac{c}{2-p},$$



(a)



(b)

Fig. 2. The problem of Example 1. (a) Costs and transition probabilities for the policies μ and μ^*. (b) The greedy partition and the solutions of the projected equations corresponding to μ and μ^*. Nonoptimistic policy iteration oscillates between r_μ and r_{μ^*}.

$$d_{\mu^*} = \Phi'\Xi_{\mu^*} g_{\mu^*} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \begin{pmatrix} \frac{1}{2-p} & 0 \\ 0 & \frac{1-p}{2-p} \end{pmatrix} \begin{pmatrix} c \\ 0 \end{pmatrix} = \frac{c}{2-p},$$

so

$$r_{\mu^*} = \frac{c}{5 - 4p - \alpha(4-3p)}.$$

We now note that since $c < 0$, $r_\mu = 0 \in R_{\mu^*}$, while for $p \approx 1$ and $\alpha > 1 - \alpha$, we have

$$r_{\mu^*} \approx \frac{c}{1-\alpha} \in R_\mu;$$

cf. Fig. 2(b). In this case, approximate policy iteration cycles between μ and μ^*.

Notice that it is hard to predict when and what kind of oscillation will occur. For example if $c > 0$, we have $r_\mu = 0 \in R_\mu$, while for $p \approx 1$ and $\alpha > 1 - \alpha$, we have

$$r_{\mu^*} \approx \frac{c}{1-\alpha} \in R_{\mu^*}.$$

In this case approximate policy iteration will converge to μ (or μ^*) if started with r in R_μ (or R_{μ^*}, respectively).

IV. Conditions for Policy Convergence

The preceding analysis has indicated that it is desirable to avoid policy oscillation in approximate policy iteration. Moreover, as mentioned earlier, when policies converge there is a more favorable error bound associated with the method [cf. Eq. (5) versus Eq. (4)]. It is therefore interesting to investigate conditions under which the sequence of policies will converge.

To this end, consider a method that evaluates the cost vector J_μ of a policy μ by a vector Φr_μ that is defined as a solution of the fixed point equation

$$(MT_\mu)(\Phi r) = \Phi r \tag{6}$$

where $M : \Re^n \mapsto S$ is a mapping (independent of μ). Let us assume the following conditions:

(a) M is monotone in the sense that $My \le M\overline{y}$ for any two vectors $y, \overline{y} \in \Re^n$ such that $y \le \overline{y}$
(b) For each μ, there is a unique solution of Eq. (6), denoted Φr_μ.

(c) For each μ and starting point r_0, the iteration

$$\Phi r_{k+1} = MT_\mu(\Phi r_k)$$

converges to Φr_μ.

Note that conditions (b) and (c) are satisfied if MT_μ is a contraction on S.

Proposition 1. *Let conditions (a)-(c) hold. Consider the approximate policy iteration method that uses Eq. (6) for policy evaluation, and is operated so that it terminates when a policy $\overline{\mu}$ is obtained such that $T_{\overline{\mu}}(\Phi r_{\overline{\mu}}) = T(\Phi r_{\overline{\mu}})$. Then the method terminates in a finite number of iterations.*

Proof: Similar to the standard proof of convergence of (exact) policy iteration, we use the policy improvement equation $T_\mu(\Phi r_\mu) = T(\Phi r_\mu)$, the monotonicity of M, and the policy evaluation Eq. (6) to write

$$(MT_{\overline{\mu}})(\Phi r_\mu) = (MT)(\Phi r_\mu) \le (MT_\mu)(\Phi r_\mu) = \Phi r_\mu.$$

Since $MT_{\overline{\mu}}$ is monotone by condition (a), by iterating with $MT_{\overline{\mu}}$ and by using condition (c), we obtain

$$\Phi r_{\overline{\mu}} = \lim_{k\to\infty} (MT_{\overline{\mu}})^k(\Phi r_\mu) \le \Phi r_\mu.$$

There are finitely many policies, so we must have $\Phi r_{\overline{\mu}} = \Phi r_\mu$ after a finite number of iterations, which implies that $T_{\overline{\mu}}(\Phi r_{\overline{\mu}}) = T(\Phi r_{\overline{\mu}})$, and that the algorithm terminates with $\overline{\mu}$. $\square$

When the projected equation approach is used ($M = \Pi$ where Π is a Euclidean projection matrix), the monotonicity assumption is satisfied if Π is independent of the policy μ and has nonnegative components. It turns out that *this is true when Φ has nonnegative components, and has linearly independent columns that are orthogonal with respect to the inner product $< x_1, x_2 >= x_1'\Xi x_2$*. This follows from the projection formula $\Pi = \Phi(\Phi'\Xi\Phi)^{-1}\Phi'\Xi = \Phi\Phi'\Xi$ and the fact that $\Phi'\Xi\Phi$ is positive definite and diagonal.

An important special case where Prop. 4.1 applies and policies converge is when $M = \Phi D$, and Φ and D are matrices whose rows are probability distributions. This is policy evaluation by aggregation, noted in Section II. Briefly, there is a finite set $\mathcal{A}$ of aggregate states, and D and Φ are the matrices that have as rows disaggregation and aggregation probability distributions, respectively, whose components relate the original system states with the aggregate states and are defined as follows:

(1) For each aggregate state x and original system state i, we specify the *disaggregation probability* d_{xi} [we have $\sum_{i=1}^n d_{xi} = 1$ for each $x \in \mathcal{A}$]. Roughly, d_{xi} may be interpreted as the "degree to which x is represented by i."

(2) For each aggregate state y and original system state j, we specify the *aggregation probability* ϕ_{jy} (we have $\sum_{y\in\mathcal{A}} \phi_{jy} = 1$ for each $j = 1,\ldots,n$). Roughly, ϕ_{jy} may be interpreted as the "degree of membership of j in the aggregate state y."

It can be seen that for the aggregation case $M = \Phi D$, M is monotone and MT_μ is a sup-norm contraction (since M is nonexpansive with respect to the sup norm), so that conditions (a)-(c) are satisfied. The same is true in the more general case *$M = \Phi D$, where Φ and D are matrices with nonnegative components, and the row sums of M are less or equal to 1*, i.e.,

$$\sum_{m=1}^s \Phi_{im} D_{mj} \ge 0, \qquad \forall\ i,j = 1,\ldots,n,$$

$$\sum_{m=1}^s \Phi_{im} \sum_{j=1}^n D_{mj} \le 1, \qquad \forall\ i = 1,\ldots,n.$$

Note that even in this more general case, the policy evaluation Eq. (6) can be solved by using simulation and low order calculations (see [Ber10a] and the survey [Ber10b], which contains many references). An interesting special case where the aggregation equation $\Phi r = \Phi D T_\mu(\Phi r)$ is also a projected equation is hard aggregation (the state space is partitioned into subsets, and each column of Φ has a 1 for the states in the subset corresponding to the column, and a 0 for the remaining states), as discussed in [Ber10a], [Ber10b].

V. Conclusions

We have considered some aspects of approximate policy iteration methods. From the analytical point of view, this is a subject with a rich theory and interesting algorithmic issues. From a practical point of view, this is a methodology that can address very large and difficult problems, and yet challenge the practitioner with unpredictable behaviors, such as policy oscillations, which are not fully understood at present.

We have derived conditions under which policy oscillations will not occur, and showed that aggregation-based methods satisfy these conditions. It thus appears that these methods have more regular behavior, and offer better error bound guarantees that TD methods. On the other hand, aggregation methods are restricted in the choice of basis functions that they can use, and this can be a significant limitation for many problems.

References

[BED09] Busoniu, L., Ernst, D., De Schutter, B., and Babuska, R., 2009. "Online Least-Squares Policy Iteration for Reinforcement Learning Control," unpublished report, Delft Univ. of Technology, Delft, NL.

[BSA83] Barto, A. G., Sutton, R. S., and Anderson, C. W., 1983. "Neuronlike Elements that Can Solve Difficult Learning Control Problems," IEEE Trans. on Systems, Man, and Cybernetics, Vol. 13, pp. 835-846.

[BeI96] Bertsekas, D. P., and Ioffe, S., 1996. "Temporal Differences-Based Policy Iteration and Applications in Neuro-Dynamic Programming," LIDS-P-2349, MIT, Cambridge, MA.

[BeT96] Bertsekas, D. P., and Tsitsiklis, J. N., 1996. Neuro-Dynamic Programming, Athena Scientific, Belmont, MA.

[Ber96] Bertsekas, D. P., 1996. Lecture at NSF Workshop on Reinforcement Learning, Hilltop House, Harper's Ferry, NY.

[Ber07] Bertsekas, D. P., 2007. Dynamic Programming and Optimal Control, 3rd Edition, Vols. I and II, Athena Scientific, Belmont, MA.

[Ber10a] Bertsekas, D. P., 2010. Approximate Dynamic Programming, on-line at http://web.mit.edu/dimitrib/www/dpchapter.html.

[Ber10b] Bertsekas, D. P., 2010. "Approximate Policy Iteration: A Survey and Some New Methods," Report LIDS-P-2833, MIT; to appear in Journal of Control Theory and Applications.

[Bor08] Borkar, V. S., 2008. Stochastic Approximation: A Dynamical Systems Viewpoint, Cambridge Press.

[BrB96] Bradtke, S. J., and Barto, A. G., 1996. "Linear Least-Squares Algorithms for Temporal Difference Learning," Machine Learning, Vol. 22, pp. 33-57.

[CFH07] Chang, H. S., Fu, M. C., Hu, J., Marcus, S. I., 2007. Simulation-Based Algorithms for Markov Decision Processes, Springer, NY.

[Cao07] Cao, X. R., 2007. Stochastic Learning and Optimization: A Sensitivity-Based Approach, Springer, NY.

[DFM09] Desai, V. V., Farias, V. F., and Moallemi, C. C., 2009. "Aproximate Dynamic Programming via a Smoothed Approximate Linear Program, Submitted.

[FaV06] Farias, V. F., and Van Roy, B., 2006. "Tetris: A Study of Randomized Constraint Sampling, in Probabilistic and Randomized Methods for Design Under Uncertainty, Springer-Verlag.

[Gor95] Gordon, G. J., 1995. "Stable Function Approximation in Dynamic Programming," in Machine Learning: Proceedings of the Twelfth International Conference, Morgan Kaufmann, San Francisco, CA.

[Gos03] Gosavi, A., 2003. Simulation-Based Optimization Parametric Optimization Techniques and Reinforcement Learning, Springer-Verlag, NY.

[JuP07] Jung, T., and Polani, D., 2007. "Kernelizing LSPE(λ)," in Proc. 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning, Honolulu, Hawaii. pp. 338-345.

[Kak02] Kakade, S., 2002. "A Natural Policy Gradient," in Advances in Neural Information Processing Systems, Vol. 14, MIT Press, Cambridge, MA.

[LaP03] Lagoudakis, M. G., and Parr, R., 2003. "Least-Squares Policy Iteration," J. of Machine Learning Research, Vol. 4, pp. 1107-1149.

[Mey07] Meyn, S., 2007. Control Techniques for Complex Networks, Cambridge University Press, NY.

[Pow07] Powell, W. B., 2007. Approximate Dynamic Programming: Solving the Curses of Dimensionality, Wiley, NY.

[Put94] Puterman, M. L., 1994. Markov Decision Processes: Discrete Stochastic Dynamic Programming, J. Wiley, NY.

[SJJ94] Singh, S. P., Jaakkola, T., and Jordan, M. I., 1994. "Learning without State-Estimation in Partially Observable Markovian Decision Processes," Proc. of the Eleventh Machine Learning Conference, pp. 284-292.

[SJJ95] Singh, S. P., Jaakkola, T., and Jordan, M. I., 1995. "Reinforcement Learning with Soft State Aggregation," in Advances in Neural Information Processing Systems 7, MIT Press, Cambridge, MA.

[Sam59] Samuel, A. L., 1959. "Some Studies in Machine Learning Using the Game of Checkers," IBM J. Res. and Development, pp. 210-229.

[Sam67] Samuel, A. L., 1967. "Some Studies in Machine Learning Using the Game of Checkers. II – Recent Progress," IBM J. Res. and Development, pp. 601-617.

[SuB98] Sutton, R. S., and Barto, A. G., 1998. Reinforcement Learning, MIT Press, Cambridge, MA.

[Sut88] Sutton, R. S., 1988. "Learning to Predict by the Methods of Temporal Differences," Machine Learning, Vol. 3, pp. 9-44.

[SzL06] Szita, I., and Lorinz, A., 2006. "Learning Tetris Using the Noisy Cross-Entropy Method," Neural Computation, Vol. 18, pp. 2936-2941.

[Tes92] Tesauro, G., 1992. "Practical Issues in Temporal Difference Learning," Machine Learning, Vol. 8, pp. 257-277.

[ThS09] Thiery, C., and Scherrer, B., 2009. "Improvements on Learning Tetris with Cross-Entropy," Intern. Computer Games Assoc. Journal, Vol. 32, pp. 23-33.

[TsV96] Tsitsiklis, J. N., and Van Roy, B., 1996. "Feature-Based Methods for Large-Scale Dynamic Programming," Machine Learning, Vol. 22, pp. 59-94.

[Van06] Van Roy, B., 2006. "Performance Loss Bounds for Approximate Value Iteration with State Aggregation," Math. of Operations Research, Vol. 31, pp. 234-244.

[YuB10] Yu, H., and Bertsekas, D. P., 2010. "New Error Bounds for Approximations from Projected Linear Equations," Mathematics of Operations Research, Vol. 35, 2010, pp. 306-329.