Stochastics and Statistics

Approximate dynamic programming via direct search in the space of value function approximations

E.F. Arruda a,⇑, M.D. Fragoso b, J.B.R. do Val c

a FENG/PUCRS. Av. Ipiranga, 6681. Porto Alegre, RS 90619-900, Brazil
b CSC/ENCC. Av. Getúlio Vargas, 333. Petrópolis, RJ 25651-075, Brazil
c DT/FEEC/UNICAMP. Cidade Universitária Zeferino Vaz. Av. Albert Einstein, 400. CP 6101. Campinas, SP 13083-852, Brazil

ARTICLE INFO

Article history:
Received 4 January 2010
Accepted 17 November 2010
Available online 21 November 2010

Keywords:
Dynamic programming
Markov decision processes
Convex optimization
Direct search methods

ABSTRACT

This paper deals with approximate value iteration (AVI) algorithms applied to discounted dynamic programming (DP) problems. For a fixed control policy, the span semi-norm of the so-called Bellman residual is shown to be convex in the Banach space of candidate solutions to the DP problem. This fact motivates the introduction of an AVI algorithm with local search that seeks to minimize the span semi-norm of the Bellman residual in a convex value function approximation space. The novelty here is that the optimality of a point in the approximation architecture is characterized by means of convex optimization concepts and necessary and sufficient conditions to local optimality are derived. The procedure employs the classical AVI algorithm direction (Bellman residual) combined with a set of independent search directions, to improve the convergence rate. It has guaranteed convergence and satisfies, at least, the necessary optimality conditions over a prescribed set of directions. To illustrate the method, examples are presented that deal with a class of problems from the literature and a large state space queueing problem setting.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

The rise of dynamic programming (DP) (Bellman, 1957) was a major breakthrough in the treatment and solution of deterministic and stochastic sequential decision problems. There is nowadays a wide variety of applications for this framework (Puterman, 1994), ranging from scheduling problems, e.g. (Sox et al., 1999) to complex and network models as presented by Swarts and Ferreira (1993), Meyn (2008). The elegant DP recursion is efficient because it allows an implicit comparison between a combinatorial number of scenarios by enumerating the states, i.e. possible configurations, of the system. However, the number of states in a system increases exponentially with the system dimension, thus making standard DP algorithms prohibitively demanding for problems with a moderately large number of dimensions. Detailed treatments of DP techniques can be found in the classical works by Puterman (1994) and Bertsekas (1995). For an interesting study on the convergence properties of standard dynamic programming algorithms we refer to Zobel and Scherer (2005). A related study on the effectiveness of action elimination in value iteration algorithms was conducted in Jaber (2008).

The approximate dynamic programming (ADP) framework comprises a body of theory and computational tools developed to tackle situations where standard DP algorithms become computationally too demanding, see for example (Bertsekas and Tsitsiklis, 1996; Si et al., 2004; Sutton and Barto, 1998, or Powell, 2007). For further results and applications of ADP methods, we refer to Bertsekas and Yu (2009), Yu and Bertsekas (2009), Borkar et al. (2009), Choi and Van Roy (2006), Menache et al. (2005). A popular ADP approach involves incorporating an arbitrary parametric approximation scheme into the original DP problem and consists in seeking sub-optimal solutions in a lower dimensional subset of the standard value iteration algorithm search space. Although this approach has proven successful in real-world applications (e.g. Tesauro, 1992), some measure of refinement is needed, for it may lead to unstable and possibly divergent algorithms (Boyan and Moore, 1995). It was the possibility of erratic behavior that led to the development of convergent ADP algorithms specifically tailored for certain types of approximation schemes (architectures), e.g. (Baird, 1995; Gordon, 1995).

Convergent ADP algorithms often rely on specific properties of the approximation architecture and/or the projection/fitting operator, i.e. the operator that converts elements in the Banach space of value function candidates into elements (approximate solutions) in the approximation space. The non-expansion based algorithm introduced in Gordon (1995) applies to non-expansive projection...
mappings. Residual and gradient descent algorithms, e.g. (Baird, 1995; Baird and Moore, 1999) perform gradient descent search with respect to the mean squared Bellman residual, thereby requiring that each value function in the approximation space be differentiable with respect to the parameters, i.e. requiring the existence of a differential mapping from approximate value functions to parameters. Although convergence is guaranteed, not much can be inferred about the nature of the accumulation point. A convergent algorithm for fairly general approximation architectures under a class of expansive projection mappings was introduced in Arruda and do Val (2006). It was shown to converge to the projection of a local solution to the DP problem in the approximation space under suitable conditions.

A peculiar feature of the present paper is that it introduces a class of ADP algorithms with value function approximation that seeks to minimize the span semi-norm of the Bellman residual. The rationale behind the proposed approach is similar to that behind the residual algorithms of Baird (1995) and Baird and Moore (1999), namely, to find a low residual approximate solution while relying on the fact that the Bellman residual is an estimate of the distance between any value function candidate and the true value function. Nevertheless, whereas the main concern of residual algorithms is to ensure convergence by applying a gradient descent procedure, the approach proposed here, while also ensuring convergence, is additionally concerned with unveiling the nature of the accumulation point. The span semi-norm is chosen, instead of the generally applied supremum norm, to account for the fact that adding a constant to an approximate value function does not alter the corresponding greedy control policy and, consequently, the quality of the approximation (Bertsekas, 1995). Accordingly, the span semi-norm of both an approximate value function and its Bellman residual remain unchanged with the addition of a constant scalar (Bertsekas, 1995; Puterman, 1994).

A novel contribution of this work is to prove that, for a fixed control policy, an appropriate function of the Bellman residual, namely the span semi-norm, is convex in the Banach space of real-valued functions. Such a conclusion implies the existence of a single global optimizer for that function under a convex approximation architecture. In addition, convex programming theory, e.g. (Bazaraa et al., 1993), is applied to derive necessary and sufficient local optimality conditions, enabling the user to unveil the nature of the accumulation point. This motivates the introduction of an ADP algorithm with local search and guaranteed convergence. The proposed algorithm seeks the minimum Bellman residual in the approximation space and converges, at a minimum, to an accumulation point that satisfies optimality conditions in a set of prescribed directions within the approximation space.

Another interesting and distinguishing feature of the proposed ADP algorithm is that it is inspired by direct search (Lewis et al., 2000) and derivative free unidimensional search (Bazaraa et al., 1993) optimization procedures and does not make use of gradient information. Therefore, in contrast to residual gradient algorithms, the proposed algorithm does not require the existence of a differential mapping from approximate value functions to parameters. As a result, the proposed approach can be applied to fairly general convex approximation architectures and can be viewed as a generalization of the residual gradient approach. Preliminary results of the present study were presented in Arruda et al. (2008).

This paper is organized as follows. Section 2 gives a general description of exact and approximate discounted dynamic programming problems. Section 3 presents the proposed formulation of the ADP problem. Section 4 investigates convexity properties of the proposed objective function and characterizes local optimality. A pair of ADP algorithms that incorporates concepts of convex optimization and direct search methods is presented in Section 5.

Numerical experiments are presented in Sections 6 and 7 concludes the paper.

2. Preliminaries

Consider a discrete dynamic programming problem P whose controlled dynamics are described by a Markov chain $X_k$, $k \geq 0$. Let $S$ denote the state space of the problem and $U(x)$, $x \in S$ denote the set of available control actions at state $x$. At any period $k$, with $X_k = x \in S$, a control action $u \in U(x)$ is taken, an instantaneous non-negative cost $c(x,u)$ is incurred and the system moves to some state $y \in S$ with probability $p_{xy}^u$. Note that this formulation encompasses the class of deterministic problems, for which we have $p_{xy}^u = 1$ for some $y \in S$ and nil otherwise.

A stationary deterministic control policy $\pi : S \rightarrow U$ is a mapping that prescribes a single control action $u = \pi(x)$ for any $x \in S$ that can be taken each time the system visits state $x$. Let $\Pi$ be the class of feasible stationary policies and $\pi(0,1)$ be a discount factor. Associated to any policy $\pi \in \Pi$ is a discounted long term cost $V^\pi(x) = \mathbb{E}_x \left[ \sum_{k=0}^{\infty} \gamma^k c(X_k, \pi(X_k)) \right]$, $X_0 = x$, (1) where $\mathbb{E}_x$ denotes the conditional expectation given that policy $\pi$ is applied. The objective of solving problem P is to find a stationary policy $\pi^*$ such that $V^*(x) \geq V^\pi(x)$, $\forall \pi \in \Pi$ and $x \in S$.

Such a policy exists and is unique whenever $U(x)$ is finite for each $x \in S$ and $S$ is countable (Puterman, 1994, Theorem 6.2.10). For more general conditions for the existence of an optimal stationary policy, we refer to Harrison (1972) or Bertsekas and Shreve (2007).

Let $V$ be the space of non-negative real valued functions $V : S \rightarrow \mathbb{R}$ and define mappings $T^u : V \rightarrow V$ and $T : V \rightarrow V$, respectively, such that for each $x \in S$ and $u \in U(x)$,

$$T^u V(x) := c(x,u) + \alpha \mathbb{E}_x[V(X_{k+1}) | X_k = x] = c(x,u) + \alpha \sum_{y \in S} p_{xy}^u V(y),$$

$$T V(x) = \min_{u \in U(x)} T^u V(x).$$

Standard DP theory states that $T$ is a contraction mapping with respect to the supremum norm, denoted in this paper as $\| \cdot \|_\infty$, and its unique fixed point ($V^*$) coincides with the solution to the problem P. Moreover, $V^*$ can be computed recursively by the value iteration (VI) algorithm

$$V_0 \in V,$n$$

$$V_{k+1} = T V_k.$$ (4)

Definition 1. Let $A$ be a subset of $V$. For any function $f : A \rightarrow \mathbb{R}$, we say that $\delta := V_2 - V_1$, $V_1, V_2 \in A$, is a descent direction if $f(V_2) < f(V_1)$.

The recursion in (4) can be deemed as an unconstrained subgradient algorithm that takes at any iteration $k$ a descent direction $\delta_k = TV_k - V_k$ with respect to the Bellman residual. Both the convergence and uniqueness of the fixed point of the recursion in (4) follow from the contraction property, that reads

$$\|TV - TV^*\|_\infty \leq \lambda \|V - V^*\|_\infty, \forall V, V^* \in V.$$ (5)

When the state space is prohibitively large, an alternative is to substitute the recursion in (4) for an approximate value iteration (AVI) algorithm that seeks an approximate solution in a parametric approximation space $A \subseteq V$. The approximate algorithm applies mapping $T$ to a subset of $S$ and projects the samples thus obtained.
into the approximation space by means of a projection operator \( P_A : V \to A \). Such an algorithm can be defined by the recursion below
\[
\begin{align*}
V_0 & \in A, \\
V_{k+1} & = V_k + d_k, \quad d_k := P_A(TV_k) - V_k,
\end{align*}
\]
where \( d_k \) is a search direction. The goal is to find the solution to the auxiliary problem
\[
\min_{V \in A} f(V) := \min_{V \in V} d(V, V'),
\]
where \( d : A \to \mathbb{R} \) is some distance function defined a priori. Hence, the approximate algorithm seeks the best available approximation (in \( A \)) to \( V' \), according to some error criterion defined a priori. Nevertheless, considering that \( V' \) is not known a priori, the evaluation of \( f \) in (7) is impractical, and therefore the optimality of any element in \( A \) cannot be verified unless the original DP problem can be optimally solved.

In contrast to pure DP algorithms, that converge to the optimal solution and monotonically decrease a normed function of the so-called 'Bellman' residual (\( (TV - V, V \in V \)), AVI algorithms can present a divergent behavior, e.g., (Boyan and Moore, 1995). Moreover, the Bellman residual is no longer guaranteed to monotonically decrease. In order to overcome this difficulty, residual gradient algorithms, introduced in Baird (1995), alter the update in (6) to make sure that the mean squared Bellman residual is decreased at each update, assuming that the gradient of each iterate with respect to the approximation parameters is available. However, the properties of the accumulation point of such an algorithm remain poorly understood and not much can be inferred about the quality of the approximate solution obtained.

This work investigates an alternative direct search based AVI algorithm that seeks to minimize a semi-norm function of the Bellman residual, which can be viewed as an upper bound to \( f \) in (7), and is indicative of the quality of a given value function approximation. The rationale behind the approach is to search for a descent direction by sweeping a sufficiently large set of directions in the approximation space at each iteration. This procedure is inspired by properties of pattern search algorithms (Torusczon, 1997; Lewis et al., 1998). These algorithms obtain a descent direction at each iteration by sweeping a sufficient number of directions as to make sure that at least one of the directions has a positive inner product with the gradient, therefore being a descent direction.

3. Formulation and definitions

In this section we seek to reformulate Problem (7) in terms of the span semi-norm. The problem could also be formulated in terms of any standard norm. In fact, we can point out that the results in the following sections could be analogously derived for other standard norms. The decision to use the span semi-norm is taken partly because the addition of a constant vector to any element \( V \in V \) does not alter the ranking of the control policies (Puterman, 1994). To see that, it suffices to note that expression (3) yields \( TV(V + c) = V + c \). That means that the control policy that yields the minimum in (3) for some \( V \in V \) also minimizes (3) for any \( W \in V : W = V + c \), where \( c \) is a constant scalar. Hence, it can be argued that two approximate value functions that differ by a constant scalar yield the same quality of approximation, for they lead to the same control policy. Accordingly, the residuals of \( V \) and \( W = V + c \), respectively \( R_1 = (TV - V) \) and \( R_2 = (TW - W) \), also differ by a constant \( d \). As such, they possess different norms. Hence, an evaluation of the norms of such residuals may lead us to believe that one approximation is better than the other, when they are, in fact, equivalent. On the other hand, the span semi-norm of both \( (TV - V) \) and \( (TW - W) \) coincide. Consequently, it can be argued that the span semi-norm offers a better notion of the quality of a given value function approximation than standard norms. Another reason for choosing the span semi-norm is based on the fact that span semi-norm based VI stopping criteria can reportedly accelerate convergence (Puterman, 1994, Section 6.5).

Let \( P \) be a discounted DP problem and \( V \) be the Banach space of value function candidates for problem \( P \). Let \( S \) denote the state space of problem \( P \) and
\[
R(V) := TV - V, \quad V \in V
\]
be the Bellman residual associated to \( V \). For any \( n \)-dimensional vector \( W \), let \( w_i \) denote its \( i \)-th component. We define the span semi-norm of vector \( W \) as
\[
\|W\|_{sp} = \max_{1 \leq i \leq n} w_i - \min_{1 \leq i \leq n} w_i.
\]
The subscript \( sp \) in the span semi-norm notation is included to differentiate it from a general norm.

Let
\[
g(W) := \|R(W)\|_{sp}, \quad W \in V.
\]
It is known that the solution to \( P \) is unique, e.g., (Bertsekas, 1995), and that a necessary and sufficient condition for the optimality of \( V' \in V \) is \( R(V') = 0 \). Therefore, problem \( P \) can be equivalently formulated as
Minimize \( g(V), \quad V \in V \).

In (10) the residual that should be set to zero in the original optimization problem is replaced by the minimization of its semi-norm. Unicity is lost in a semi-norm evaluation, but only to some constant, as the definition of the span semi-norm entails.

The contraction property of mapping \( T \) also follows for the span semi-norm, e.g., (Puterman, 1994, Corolary 6.6.8, page 204). Hence, it follows that
\[
\|TV' - V\|_{sp} \leq \lambda \|TV' - V\|_{sp}
\]
Given that \( V' \) is unknown, one can apply an indirect method to find an upper bound on the distance between any element in \( V \) and \( V' \). Consider the expressions below,
\[
\|V' - V\|_{sp} \leq \|TV' - TV\|_{sp} + \|TV - V\|_{sp} \leq \lambda \|V' - V\|_{sp} + \|TV - V\|_{sp},
\]
where the first relation results from the triangle inequality, noting that \( V' = TV' \), and the last inequality follows from (11). The above expressions imply
\[
\|V' - V\|_{sp} \leq \frac{\|TV - V\|_{sp}}{1 - \lambda}.
\]
Therefore \( g(V) \) is indicative of the proximity between \( V \) and the exact solution \( V' \) to the DP problem.

In this work, we address a constrained version of the DP problem, which can be formulated as:
Minimize \( g(V), \quad V \in V \)
subject to \( V \in A \).

The concepts below, extracted from Bazaraa et al. (1993), will be useful in the definition of local optimality that follows.

Definition 2. Given a point \( V \in V \) and \( \epsilon > 0 \), the set \( N_{\epsilon}(V) = \{ V : \|V - V\|_{sp} \leq \epsilon \} \) is called an \( \epsilon \)-neighborhood of \( V \).
4. Optimality characterization

The linear and monotone convergence rate of the VI algorithm to the optimal solution of problem (10), e.g. (Bertsekas, 1995), indicates that the Bellman residual is always a descent direction for this problem. This suggests that problem (10) is a well-defined one. Indeed, we prove in this section that function \( g: \mathbb{V} \to \mathbb{R} \) in (9) is convex when a single control policy is available. In such a trivial case, any local minimum to (12) is also a global minimum, provided that \( A \) is convex; however, the general problem studied inherits a local optimality characterization, as we will point out.

Let \( d \in \mathbb{V} \) be an arbitrary direction, \( x \in S, V_1 \in \mathbb{V} \) and \( x \in (0,1) \) be the discount factor of the VI algorithm. Then, expression (2) in Section 2 implies, for any \( d \in \mathbb{V} \),

\[
T^\epsilon(V_1 + d)(x) = T^\epsilon V_1(x) + 2\epsilon x \delta V_1(x)\left[\delta X_1\right]_{X_0 = X} - \delta(x) \quad \forall x \in S.
\]

(14)

For any stationary policy \( \pi: S \to U \), denote \( T^\pi V(x) = T^\pi V(x) \) with \( \pi = \pi(x) \), and define the function

\[
g^\pi V = \|T^\pi V - V\|_{\text{sp}} \quad \forall V \in \mathbb{V}.
\]

Lemma 4. Function \( g^\pi V \) is convex in \( \mathbb{V} \).

Proof. Define

\[
R^0(V) = T^0 V - V.
\]

It follows from (14) that

\[
R^0(V_1 + (1 - \lambda)\delta)(x) = R^0(V_1(x) + (1 - \lambda)\epsilon e_{X_1}[\delta X_1]_{X_0 = X} - \delta(x)) = R^0(V_1(x) + (1 - \lambda)R^0(V_1(x) + \delta(x)) - R^0(V_1(x)) = \lambda R^0(V_1(x) + (1 - \lambda)R^0(V_1(x) + \delta(x)) \quad \forall x \in S.
\]

(15)

Hence, for \( \delta = V_2 - V_1 \), the equality above yields

\[
R^0(\lambda V_1 + (1 - \lambda)V_2)(x) = \lambda R^0(V_1(x)) + (1 - \lambda)R^0(V_2(x)).
\]

(16)

From the triangular inequality for the span semi-norm (Puterman, 1994, p. 196), it follows that

\[
\|R^\pi(\lambda V_1 + (1 - \lambda)V_2)\|_{\text{sp}} \leq \lambda\|R^\pi(V_1)\|_{\text{sp}} + (1 - \lambda)\|R^\pi(V_2)\|_{\text{sp}}
\]

(17)

\]

Definition 5. Let \( A \subseteq \mathbb{V} \) be the span of a parametric approximation architecture. \( A \) is said to be convex if, for any \( V_1, V_2 \in A \), \( \lambda V_1 + (1 - \lambda) V_2, \lambda \in (0,1) \) also belong to \( A \). DP theory (Puterman, 1994, Chapter 6) yields that the direction \( d = R(V) \) is a descent direction of function \( g \) at each \( V \in \mathbb{V} \). We also note that a necessary and sufficient condition for local optimality of \( V \) in Problem (10) is that no descent direction exists, i.e. \( d = R(V) = 0 \), which means that \( g(V) = 0 \) if \( V \) solves (10). In addition, since \( g(\cdot) \) is a semi-norm function, \( g(W) \geq 0, W \in \mathbb{V} \), hence \( g(V) = 0 \) implies that \( V \) is a global optimum of Problem (10). Consequently, \( g(V) = 0 \) is a necessary and sufficient condition for global optimality in Problem (10). Analogously, a local minimum to (13) is attained whenever no descent direction exists in \( A \subseteq \mathbb{V} \).

5. Deriving AVI algorithms

This section seeks to exploit the results in Section 4 and improve the convergence properties of the algorithm in (6). We seek to introduce methods combining elements of dynamic programming, convex optimization and pattern search. These methods are implemented in two algorithms. Each algorithm employs a unidimensional search procedure (Bazaraa et al., 1993) to sweep a set of directions in \( A \) in search of a descent direction. The idea is to exploit a sufficiently large set of directions in \( A \) so that a descent direction can be identified whenever it exists. For each direction, a derivative-free unidimensional search is performed to identify
the best point in the prescribed direction. The algorithms terminate when no further improvement is attained.

In Algorithm 1, the set of directions to be considered for the unidimensional search is arbitrarily chosen at each iteration of the method. Algorithm 2 differs from Algorithm 1 in that it selects descent directions based on the projection in $A$ of the application of the dynamic programming operator $T$ to randomly selected sample sets, each with arbitrary cardinality $n$, of states in $S$ (sets $Y_i = \{x_1, \ldots, x_n\} \subset S$ in Step 2). In our experiments, the states in $Y_i$ are drawn according to a uniform distribution in $S$. Naturally, different distributions can be tried in other applications. For each sample set, a search direction is identified and unidimensional search is performed. If the target direction is not a descent direction, the sample states are added to an extended sample set, the union of the sets $Y_i$ applied at the current iteration, that is subsequently used to identify a second search direction by projecting the application of mapping $T$ to the extended set into the approximation space $A$.

Algorithm 2 is inspired by the Dantzig–Wolfe decomposition algorithm, e.g. (Lasdon, 1970), which is very popular in the domain of optimization theory. The proposed algorithm sweeps residual states is generated.

Algorithm 1 (Approximate Value Iteration with Direct Search).

Step 1: (Initialization)
- Choose $v_0 \in A$, $n$ and $tol$
- $k \leftarrow 0$, $\epsilon_0 = \|TV_0 - V_0\|_p$

Step 2: (Projection)
- $V = P(A(TV_k))$
- $\epsilon_{k+1} = \|TV - V\|_p$

Step 3: (Unidimensional Search)
- Select Feasible Directions $\{d_1, \ldots, d_m\}$, $n < \infty$
- $d_1 \leftarrow \delta$
- $F_0 = \{d_1, \ldots, d_m\}$
- $j \leftarrow 1$, $\lambda \leftarrow 0$
- While ($\lambda = 0$) AND $j < n$
  - $\lambda \leftarrow \arg \min_{d_1, \ldots, d_m} \|VR(V_k + \lambda d_j)\|_p$
  - $j \leftarrow j + 1$
- End While
- $V = V_k + \lambda d_j$
- $V_{k+1} = V$

Step 4: (Convergence Test)
- If $\|V_{k+1} - V_k\| < tol$
  - STOP
- Else
  - $k \leftarrow k + 1$, return to step 2
- End If

Algorithm 2 (Decomposed Approximate Value Iteration).

Step 1: (Initialization)
- Choose $v_0 \in A$, $n$ and $tol$
- $k \leftarrow 0$

Step 2: (Projection and Direction Generation)
- $i \leftarrow 0$
- $j \leftarrow 0$
- $Y \leftarrow \emptyset$
- While ($i = 0$) AND $j < m$
  - Randomly select a set $Y_i = \{x_1, \ldots, x_n\}$
  - $V = Y \cup Y_i$
  - $V' = P(A(TV_k(Y_i)))$
  - $V'' = P(A(TV_k(Y_i)))$
  - $d_1 = V' - V_k$, $d_2 = V'' - V_k$
  - $\lambda = \arg \min_{d_1, d_2} \|VR(V_k + \lambda d_1)\|_p$
  - $V = V_k + \lambda d_1$
- If ($i = 0$)
  - $i \leftarrow \arg \min_{d_1, d_2} \|VR(V_k + \lambda d_2)\|_p$
- End If
- $V = V_k + \lambda d_2$
- $V_{k+1} = V$
- $j \leftarrow (j + 1)$
- End While

Step 3: (Convergence Test)
- If $\|V_{k+1} - V_k\| < tol$
  - STOP
- Else
  - $k \leftarrow k + 1$, return to step 2
- End If

The approach proposed here could certainly benefit from further developments. The design of specific procedures for the selection of search directions seems to be a very promising and challenging problem. Indeed, the design of sub-routines to search for descent directions, perhaps based on Simulated Annealing or Evolutionary Algorithms, seems to be a promising area for future research. Other promising area involves deriving performance bounds for such procedures.

5.1. Performance evaluation

As pointed out in Section 4, an insight about the quality of a candidate solution can be obtained by considering a set of $|S| + 1$ linearly independent directions in $V$. This alternative, which offers a hint into the complexity of the problem, is a function of the dimension of the state space, regardless of the complexity of the set of actions $U$. For very large state spaces, the alternative is to examine a smaller number of alternatives, depending upon the computational resources available.

Both Algorithms 1 and 2 make use of Corollary 7 to search for a satisfactory solution to (13). The number of search directions is a user-defined parameter. The projection of the Bellman residual direction, $P(A(TV_k))$, is included in the search directions set both because it is a promising direction and because it conveys local information of the DP recursion. In the next section a few simulations are presented with a view toward providing insights on the sensitivity of the algorithm with respect to the direction set examined in each step of Algorithm 1. Algorithm 2 seems more adequate to large problems, since the potentially large set of search directions is automatically generated by the algorithm. Hence, Algorithm 2 is applied to moderately large, albeit simple queueing problems to provide insights on the efficiency of the proposed method in large DP settings.
6. Numerical examples

In this section, we replicate the chain-walk problem, as it is employed by Lagoudakis and Parr (2003). The problem consists of a chain of N states, labeled 1 to N. At each state there are two actions available, “go left” (L) and “go right” (R). The actions succeed with probability 0.9 and fail with probability 0.1; when an action fails, the complementary action is the one actually executed; the two boundaries of the chain are dead-ends. The “reward” function is nil at the boundaries and unitary otherwise and the discount factor is 0.9. The objective is to maximize the cumulative discounted return at the boundaries and unitary otherwise and the discount factor is 0.9. The “reward” function is nil at the boundaries and unitary otherwise and the discount factor is 0.9. The projection operator $P_A$ chosen is linear regression in w with least-squares minimization. Our experiments showed that Algorithm 1 converges to the value function for the example in Fig. 1. In that particular setting, it outperforms its counterpart (Koller and Parr, 1998) which encounters difficulty to converge.

To verify the behavior of Algorithm 1 when the value function does not belong to the approximation domain, experiments with a 5-state and a 10-state chain-walk examples were carried out. In these examples, approximate solutions with non-zero Bellman residual have to be pursued. In each experiment, $v_0 = 0$ and the set $F_0$ in Step 3 of Algorithm 1 was varied. Since the constant part of $v$ does not affect $g(v)$, the search directions can be specified in the set $\phi w$, $w \in \mathbb{R}^3$, $w_i = 0$. For convenience, the search directions are specified as unitary vectors and described in terms of polar coordinates in the plane $w_2 \times w_3$. Hence, an angle in the interval $[0, 2\pi)$ suffices to describe a direction in $F_0$. In the experiments, $F_0$ is a set of uniformly spaced vectors in the unitary circle, with the angle of the first direction being $\theta_1 = 0$ and $\theta_i = \theta_{i-1} + \frac{2\pi}{n}$, $i = 1, \ldots, n$. To this set $F_0$, the residual direction $P_A(TV_k) - v_k$ is appended and a choice among these $n+1$ directions is made in favor of the one that attains the smallest residual semi-norm; see step 3 of Algorithm 1. The algorithm stops when two consecutive iterates are within a prescribed tolerance.

The 5-state example was firstly solved using the classical AVI (CAVI) recursion in (6). After that, the number of search directions in Step 3 of Algorithm 1 was varied and the residual direction $d_1$ removed. The algorithm did not improve upon the initial value for $n < 62$. The result with 62 search directions is conveyed in Table 1 with the mnemonic (DSAVI). Finally, Algorithm 1 (RDSAVI) was applied with $n = 5$ and $d_1$ included.

As shown in Table 1, a pure direct search algorithm (DSAVI) may require the evaluation of a large set of directions in order to ensure optimality. As expected, the classical AVI algorithm

\[ \phi = [\phi(1)^T \ldots \phi(N)^T]^T, \]

where $T$ indicates the transpose of $v$.

The 5-state example was firstly solved using the classical AVI (CAVI) recursion in (6). After that, the number of search directions in Step 3 of Algorithm 1 was varied and the residual direction $d_1$ removed. The algorithm did not improve upon the initial value for $n < 62$. The result with 62 search directions is conveyed in Table 1 with the mnemonic (DSAVI). Finally, Algorithm 1 (RDSAVI) was applied with $n = 5$ and $d_1$ included.

As shown in Table 1, a pure direct search algorithm (DSAVI) may require the evaluation of a large set of directions in order to ensure optimality. As expected, the classical AVI algorithm

\[ \phi = [\phi(1)^T \ldots \phi(N)^T]^T. \]

The projection operator $P_A$ chosen is linear regression in $w$ with least-squares minimization. Our experiments showed that Algorithm 1 converges to the value function for the example in Fig. 1. In that particular setting, it outperforms its counterpart (Koller and Parr, 1998) which encounters difficulty to converge.

To verify the behavior of Algorithm 1 when the value function does not belong to the approximation domain, experiments with a 5-state and a 10-state chain-walk examples were carried out. In these examples, approximate solutions with non-zero Bellman residual have to be pursued. In each experiment, $v_0 = 0$ and the set $F_0$ in Step 3 of Algorithm 1 was varied. Since the constant part of $v$ does not affect $g(v)$, the search directions can be specified in the set $\phi w$, $w \in \mathbb{R}^3$, $w_i = 0$. For convenience, the search directions are specified as unitary vectors and described in terms of polar coordinates in the plane $w_2 \times w_3$. Hence, an angle in the interval $[0, 2\pi)$ suffices to describe a direction in $F_0$. In the experiments, $F_0$ is a set of uniformly spaced vectors in the unitary circle, with the angle of the first direction being $\theta_1 = 0$ and $\theta_i = \theta_{i-1} + \frac{2\pi}{n}$, $i = 1, \ldots, n$. To this set $F_0$, the residual direction $P_A(TV_k) - v_k$ is appended and a choice among these $n+1$ directions is made in favor of the one that attains the smallest residual semi-norm; see step 3 of Algorithm 1. The algorithm stops when two consecutive iterates are within a prescribed tolerance.

The 5-state example was firstly solved using the classical AVI (CAVI) recursion in (6). After that, the number of search directions in Step 3 of Algorithm 1 was varied and the residual direction $d_1$ removed. The algorithm did not improve upon the initial value for $n < 62$. The result with 62 search directions is conveyed in Table 1 with the mnemonic (DSAVI). Finally, Algorithm 1 (RDSAVI) was applied with $n = 5$ and $d_1$ included.

As shown in Table 1, a pure direct search algorithm (DSAVI) may require the evaluation of a large set of directions in order to ensure optimality. As expected, the classical AVI algorithm

\[ \phi = [\phi(1)^T \ldots \phi(N)^T]^T. \]

The projection operator $P_A$ chosen is linear regression in $w$ with least-squares minimization. Our experiments showed that Algorithm 1 converges to the value function for the example in Fig. 1. In that particular setting, it outperforms its counterpart (Koller and Parr, 1998) which encounters difficulty to converge.

To verify the behavior of Algorithm 1 when the value function does not belong to the approximation domain, experiments with a 5-state and a 10-state chain-walk examples were carried out. In these examples, approximate solutions with non-zero Bellman residual have to be pursued. In each experiment, $v_0 = 0$ and the set $F_0$ in Step 3 of Algorithm 1 was varied. Since the constant part of $v$ does not affect $g(v)$, the search directions can be specified in the set $\phi w$, $w \in \mathbb{R}^3$, $w_i = 0$. For convenience, the search directions are specified as unitary vectors and described in terms of polar coordinates in the plane $w_2 \times w_3$. Hence, an angle in the interval $[0, 2\pi)$ suffices to describe a direction in $F_0$. In the experiments, $F_0$ is a set of uniformly spaced vectors in the unitary circle, with the angle of the first direction being $\theta_1 = 0$ and $\theta_i = \theta_{i-1} + \frac{2\pi}{n}$, $i = 1, \ldots, n$. To this set $F_0$, the residual direction $P_A(TV_k) - v_k$ is appended and a choice among these $n+1$ directions is made in favor of the one that attains the smallest residual semi-norm; see step 3 of Algorithm 1. The algorithm stops when two consecutive iterates are within a prescribed tolerance.

The 5-state example was firstly solved using the classical AVI (CAVI) recursion in (6). After that, the number of search directions in Step 3 of Algorithm 1 was varied and the residual direction $d_1$ removed. The algorithm did not improve upon the initial value for $n < 62$. The result with 62 search directions is conveyed in Table 1 with the mnemonic (DSAVI). Finally, Algorithm 1 (RDSAVI) was applied with $n = 5$ and $d_1$ included.

As shown in Table 1, a pure direct search algorithm (DSAVI) may require the evaluation of a large set of directions in order to ensure optimality. As expected, the classical AVI algorithm

\[ \phi = [\phi(1)^T \ldots \phi(N)^T]^T. \]
converges to a suboptimal solution. By combining unidimensional search and direct search algorithm concepts with the classical AVI algorithm, one obtains a much improved convergence rate and attains a better objective function while evaluating a smaller set of directions.

Fig. 2(a) plots the Bellman residual against the iteration number for the 5-state chain-walk example. It shows that the algorithm takes 4 iterations to converge. Note that the accumulation point is better than the value function projection \( P_A(V^*) \) in terms of Bellman residual. In Fig. 2(b), the exact solution to the problem is plotted against its projection in the approximation architecture and the solution obtained by Algorithm 1. Fig. 3 depicts the optimal control policy together with the approximate policy, which is derived from the approximate value function. It can be verified that the both policies coincide and the proposed AVI algorithms yields the optimal policy.

Fig. 4(a) depicts the Bellman residual for the 10-state chain-walk example. For this example, Algorithm 1 converges in 4 iterations. Once again, the Bellman residual is decreased considerably if compared to its counterpart at the value function projection \( P_A(V^*) \). However, one can observe in Fig. 4(b) that the solution to the proposed algorithm is more distant from the value function than \( P_A(V^*) \) in terms of the span semi-norm. This seemingly counter-intuitive fact may happen when the residual is not a tight bound to the distance to the value function, i.e. when the upper bound in Eq. (12) not a tight bound. Fig. 5 plots the optimal and approximate policies for the 10-state problem. Observe that three actions out of the 10 prescribed by the approximate policy do not match the corresponding optimal control action.

6.1. A queueing example

To provide an insight on the performance of the proposed method for larger DP problems, we applied Algorithm 2 to a queueing problem with a single control policy. At each discrete time a client enters the queue with probability \( p = 0.4 \) and a client leaves the line with probability \( q = 0.6 \). The queue length varies from 0 to a maximum allowed length of \( N \). When the queue reaches the maximum length, no new clients are allowed until the queue is reduced. Similarly, the empty queue remains empty with probability \( p \) and jumps to 1 with probability \( q \). The cost of holding \( l \) clients in queue is

\[ c(l) = l^2 + 5 \cdot l. \]

The discount factor is \( \alpha = 0.9 \). The probability of selecting any state to compose a sample set \( Y_j \) in Step 2 of Algorithm 2 is inversely proportional to the current value function approximation for the state. Such a selection allows one to better represent the value function in the low-cost region and, therefore, define a more refined control policy in such a region, which one can expect to visit more often in a long term cost minimization setting.

For all examples, the length of the sample set \( |n| \) in Algorithm 2 is \( 3 \) and the value function is approximated as a second order polynomial. This choice is deliberate, since with three sample points we can find a quadratic function that represents exactly the value function at each state in the sample set. The number of different sample sets generated (parameter \( m \) in Step 2 of Algorithm 2) is also maintained constant at \( 20 \). Such a number is chosen to verify the effect of searching over a set of directions which is relatively small compared to the cardinality of the state space \( S \).
Fig. 6 shows the Bellman residual evolution of a queue with $N = 200$. Observe that the Bellman residual obtained by Algorithm 2 is reduced when compared to the residual of the value function projection $P_A(V^*)$. We iterated the algorithm 1000 times and verified that the residual decreases rapidly in the early stages and then present slow variation until stabilizing after 300 iterations.

Fig. 7 shows the results for a 1000-state queueing problem. For this experiment, the residual obtained by the proposed algorithm is slightly inferior to the residual of the value function projection in the approximation space $A$. Note that the residual levels are increased when compared to the previous experiment.

We finally simulated a 100,000-state problem. The results are plotted in Fig. 8. Once again, we observe that the residual obtained by the proposed algorithm is slightly better than the residual of the value function projection in $A$.

7. Concluding remarks

This paper shows that, for a fixed control policy, the Bellman residual is a convex function in the Banach space of real-valued functions. This fact motivates the introduction of an approximate value iteration (AVI) algorithm, devised to solve large scale dynamic programming problems. By combining unidimensional search and direct search algorithm concepts with the classical AVI algorithm, one obtains a much improved convergence rate and attains a better objective function while evaluating a smaller set of directions. Moreover, the procedure has guaranteed convergence and satisfies, at a minimum, the necessary optimality conditions over a prescribed set of directions. It can be viewed as an extension of residual algorithms to the context of convex approximation spaces that are non-differentiable with respect to the parameters.

The proposed method is applied to a class of problems previously explored in the literature for the purpose of benchmarking. The solution to the proposed problems exemplifies the strengths and drawbacks of the method. The results illustrate the potential of the proposed algorithm to find low-residual approximate solutions. Considering that the objective function is an upper bound to the distance to the value function, the quality of the approximate solution is also a function of the tightness of this bound. Additionally, a set of queueing problems with increasingly large state space is simulated and one can verify that, despite searching only a small set of directions, one can obtain low-residual approximations of the value function in such a setting.

Acknowledgments

This work was partially supported by the Brazilian National Research Council-CNPq, under grants PCI 384.970/06-0, 301740/2007-0, and 304856/2007-0.

The authors would like to express their gratitude to the referees for their many suggestions and helpful comments, which have certainly improved the quality of the paper. Edilson F. Arruda wishes to thank Professor Paulo A. Valente Ferreira, from the School of Electrical and Computer Engineering at Unicamp for his very helpful guidance on Direct Search Methods.

References

