An accelerated value/policy iteration scheme for optimal control problems and games

Alessandro Alla, Maurizio Falcone, Dante Kalise

RICAM-Report 2014-40

An accelerated value/policy iteration scheme for optimal control problems and games

Alessandro Alla[1], Maurizio Falcone[2], and Dante Kalise[3]

[1] University of Hamburg, Bundesstraße 55, Hamburg, Germany
alessandro.alla@uni-hamburg.de
[2] SAPIENZA - University of Rome, Ple. Aldo Moro 2, Rome, Italy
falcone@mat.uniroma1.it
[3] RICAM, Austrian Academy of Sciences, Altenberger Straße 69, Linz, Austria
dante.kalise@oeaw.ac.at

Abstract. We present an accelerated algorithm for the solution of static Hamilton-Jacobi-Bellman equations related to optimal control problems and differential games. The new scheme combines the advantages of value iteration and policy iteration methods by means of an efficient coupling. The method starts with a value iteration phase on a coarse mesh and then switches to a policy iteration procedure over a finer mesh when a fixed error threshold is reached. We present numerical tests assessing the performance of the scheme.

1 Introduction

The numerical solution of optimal control problems is a crucial issue for many industrial applications; usually, the final goal is to compute an optimal trajectory for the controlled system and its corresponding optimal control. To solve this problem, we focus our attention on the Dynamic Programming (DP) approach, introduced by Bellman [3] since it produces optimal control in feedback form. However, the synthesis of feedback requires the knowledge of the value function over the whole state space and this is the major bottleneck for the application of DP-based techniques.
It is well-known that the characterization of the value function for optimal control problems by means of DP is obtained in terms of a first order nonlinear Hamilton-Jacobi-Bellman (HJB) partial differential equation. In the last twenty years, this approach has been pursued for all the classical control problems in the framework of viscosity solutions introduced by Crandall and Lions (see [2] for a comprehensive illustration), since the value function of an optimal control problem is known to be only Lipschitz continuous even when the data is regular. Several approximation schemes have been proposed for this class of equations, ranging from finite differences to semi-Lagrangian and finite element methods. Some of these methods converge to the value function but their convergence is slow. Moreover, for grid-based methods, the storage requirements are huge and the complexity of the algorithm increases fast due to the so-called *curse of the dimensionality*; this explains why the problem is very challenging from a computational point of view.

Our main contribution in this article, is a new accelerated algorithm which can produce an accurate approximation of the value function in a reduced amount of time, in comparison to the currently available methods. Furthermore, the proposed scheme can be used over a wide variety of problems connected to static HJB equations, such as infinite horizon optimal control, minimum time control and some cases of pursuit-evasion games. The new method couples two ideas already existing in the literature: the value iteration method (VI), and the policy iteration method (PI) for the solution of Bellman equations. The first is known to be slow but convergent for any initial guess, while the second is known to be fast when it converges (but if not initialized correctly, convergence might be as slow as the value iteration). The approach that we consider relates to multigrid methods (we refer to Santos [11] for a brief introduction to subject in this context), as the coupling that we introduce features an unidirectional, two-level mesh. The work by Chow and Tsitsiklis [5] exploits a similar idea with a value iteration algorithm. However, as far as we know the efficient coupling between the two methods has not been investigated.
To set this paper into perspective, we must recall that algorithms based on the iteration in the space of controls (or policies) for the solution of HJB equations has a rather long history, starting more or less at the same time of dynamic programming. The PI method, also known as Howard's algorithm [7], has been investigated by Kalaba [8], and Pollatschek and Avi-Itzhak [9], who proved that it corresponds to the Newton method applied to the functional equation of dynamic programming. Later, Puterman and Brumelle [10], gave sufficient conditions for the rate of convergence to be either superlinear or quadratic. More recent contributions on the policy iteration method, and some extensions to games can be found in Santos and Rust [12], and Bokanowski et al. [4] (a more complete list of references is given in [1]).
This paper is structured as follows. In Section 2, we describe some model problems and the basic building blocks of our approach. In Section 3, an accelerated scheme for dynamic programming equations is introduced, and Section 4 presents numerical examples assessing its performance.

2 Model problems and building blocks

In this section, we introduce our model problems and summarize the basic results for the two methods which will constitute the building blocks for our new algorithm.
Let the system dynamics be given by

$$\begin{cases} \dot{y}(t) = f(y(t), \alpha(t)) \\ y(0) = x\,, \end{cases} \tag{1}$$

where $y \in \mathbb{R}^n$, $\alpha \in \mathbb{R}^m$ and $\alpha(t) \in \mathcal{A} \equiv L^\infty([0, +\infty[, A)$, and A is a compact subset of $\mathbb{R}^m$. If f is Lipschitz continuous with respect to the state variable

and continuous with respect to (x,α), the classical assumptions for the existence and uniqueness result for the Cauchy problem (1) are satisfied.
Let us consider the minimum time problem where we want to compute the time of arrival of the dynamics (1) to a given target $\mathcal{T}$, denoted by $T(x)$. Note that, without additional assumptions, it is not guaranteed that T is finite everywhere, so a crucial role is played by the reachable set $\mathcal{R} = \{x \in \mathbb{R}^n : T(x) < +\infty\}$. By applying the Dynamic Programming Principle, it is possible to obtain the Bellman equation giving the characterization of the solution $T(x)$ on $\mathcal{R}$ and, via the change of variable $v(x) = 1 - \exp(-T(x))$ we extend it to $\mathbb{R}^n$. In conclusion, v is the unique viscosity solution of the Dirichlet problem

$$\begin{cases} v(x) + \sup\limits_{a\in A}\{-f(x,a)\cdot Dv(x)\} = 1 & \text{in } \mathbb{R}^n\backslash\mathcal{T} \\ v(x) = 0 & \text{on } \partial\mathcal{T}\,. \end{cases} \tag{2}$$

A similar equation will appear for differential games where two players are acting on the dynamics: player-a wants to hit the target in minimal time, whereas player-b wants to keep the system away from $\mathcal{T}$. Under appropriate assumptions, one can obtain a characterization of the v in terms of the following Dirichlet problem

$$\begin{cases} v(x) + \sup\limits_{a\in A}\inf\limits_{b\in B}\{-f(x,a,b)\cdot Dv(x)\} = 1 & \text{in } \mathbb{R}^n\backslash\mathcal{T} \\ v(x) = 0 & \text{on } \partial\mathcal{T}\,. \end{cases} \tag{3}$$

Value iteration. A natural way to solve (2) in the context of semi-Lagrangian schemes, is to consider a pseudotime parameter Δt for the discretization of the system dynamics, to apply a discrete version of the DP, and to consider an approximation in space of the resulting HJB equation. The resulting nonlinear equation is solved by means of a fixed point iteration over the discrete value function, as described in Algorithm 1.

Algorithm 1: Value Iteration for the minimum time problem **(VI)**

```
Data: Mesh $G$, $\Delta t$, initial guess $V^0$, tolerance $\epsilon$.
Define $V^0 = 0$ on $\mathcal{T}$, $V^0 = 1$ elsewhere
while $||V^{k+1} - V^k|| \geq \epsilon$ do
    forall the $x_i \in G$ do
        $V_i^{k+1} = \min_{a\in A}\{e^{-\Delta t} I\left[V^k\right](x_i + \Delta t f(x_i,a))\} + 1 - e^{-\Delta t}$    (4)
    end
    $k = k + 1$
end
```

Here, we denote by V_i^k the value function at a node x_i of the grid at the k-th iteration and I is an interpolation operator acting on the values of the grid; without loss of generality, throughout this paper we will assume that the numerical grid G is a regular equidistant array of points with mesh spacing denoted by Δx, and that the interpolant I corresponds to a multilinear interpolation operator. A drawback of this approach resides in the fact that the convergence of the iteration is governed by a contraction constant is given by $e^{-\Delta t}$, thus higher accuracy will dramatically increase the number of iterations. We consider an alternative solution method which circumvents this problem.

Algorithm 2: Policy Iteration for the minimum time problem **(PI)**

```
Data: Mesh $G$, $\Delta t$, initial guess $V^0$, tolerance $\epsilon$.
Define $V^0 = 0$ on $\mathcal{T}$, $V^0 = 1$ elsewhere
while $||V^{k+1} - V^k|| \geq \epsilon$ do
    Policy evaluation step: forall the $x_i \in G$ do
        $V_i^k = 1 - e^{-\Delta t} + e^{-\Delta t} I\left[V^k\right]\left(x_i + \Delta t f\left(x_i, a_i^k\right)\right)$    (5)
    end
    Policy improvement step:
    forall the $x_i \in G$ do
        $a_i^{k+1} = \arg\min_a \left\{e^{-\Delta t} I\left[V^k\right](x_i + \Delta t f(x_i, a))\right\}$    (6)
    end
    $k = k + 1$
end
```

Policy iteration. In the *approximation over policy space* presented in Algorithm 2, we start from an initial guess for the control for every point in the state space. Once the control has been fixed, the Bellman equation becomes linear (no search for the minimum in the control space is performed), and it is solved as an advection equation. Then, an updated policy is computed and a new iteration starts. The resulting sequence of value functions V^k is monotone decreasing at every node of the grid. At a theoretical level, policy iteration can be shown to be equivalent to a Newton method, so under appropriate assumptions, it converges locally with quadratic speed.

3 The accelerated scheme for HJB equations

We present an accelerated iterative algorithm which is constructed upon the building blocks previously introduced. We aim at an efficient formulation exploiting the main computational features of both value and policy iteration algorithms. As it has been stated in [10], there exists a theoretical equivalence between both algorithms, which guarantees a rather wide convergence framework. However, from a computational perspective, there are significant differences between both implementations. A first key factor can be observed in Figure 1, which shows, for a two-dimensional minimum time problem, the typical situation arising with the evolution of the error measured with respect to the optimal solution, when comparing value and policy iteration algorithms. To achieve a similar error level, policy iteration requires considerably fewer iterations than the value iteration scheme, as quadratic convergent behavior is reached faster for any number of nodes in the state-space grid.

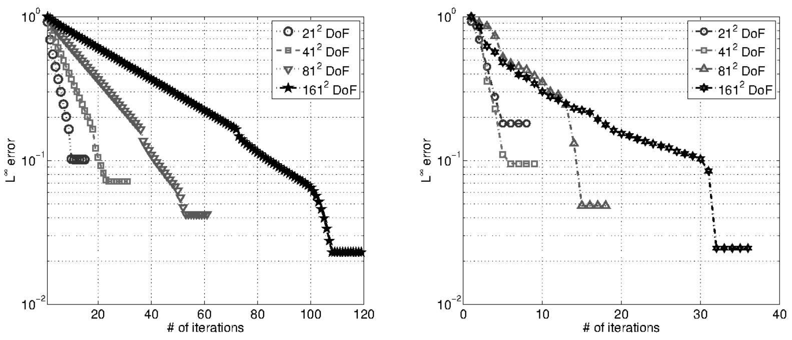

Fig. 1. Error in a 2D problem: value iteration (left) and policy iteration (right).

However, a known drawback of policy iteration is its dependence on a good initial guess in order to yield such an efficient behavior. Whereas some guesses will produce quadratic convergence from the beginning of the iterative procedure, others can lead to an underperformant value iteration-like evolution of the error. A final relevant remark can be made from Figure 1, where it can be observed that for coarse meshes, the value iteration algorithm generates a fast error decay up to a higher global error. This, combined with the fact that value iteration algorithms are rather insensitive to the choice of the initial guess for the value function, are crucial points for the construction of our accelerated algorithm. The accelerated policy iteration (API) Algorithm 3, is based on a robust initialization of the policy iteration procedure via a coarse value iteration which will yield to a good guess of the initial control field. The aforementioned accelerated algorithm can lead to a considerably improved performance when compared to value iteration and naively initialized policy iteration algorithms. However, it naturally contains trade-offs that need to be carefully handled in order to obtain a correct behavior. We present some general guidelines for a correct initialization.

Coarse and fine meshes. For a good behavior of the PI phase, a good initialization is required, but this should be obtained without deteriorating the overall performance. If we denote by Δx_c and by Δx_f the mesh parameters associated to the coarse and fine grids respectively, numerical findings reported in [1], suggest that for minimum time problems and infinite horizon optimal control, a good balance is achieved with $\Delta x_c = 2\Delta x_f$.

Algorithm 3: Accelerated Policy Iteration **(API)**

```
Data: Coase mesh $G_c$ and $\Delta t_c$ , fine mesh $G_f$ and $\Delta t_f$, initial coarse guess
      $V_c^0$, coarse-mesh tolerance $\epsilon_c$, fine-mesh tolerance $\epsilon_f$.
begin
    Coarse-mesh value iteration step: perform Algorithm 1
    Input: $G_c$, $\Delta t_c$, $V_c^0$, $\epsilon_c$
    Output: $V_c^*$
    forall the $x_i \in G_f$ do
        $V_f^0(x_i) = I_1[V_c^*](x_i)\ A_f^0(x_i) = \underset{a \in A}{argmin}\,\{e^{-\lambda \Delta t} I_1[V_f^0](x_i + f(x_i, a))\}$
    end
    Fine-mesh policy iteration step: perform Algorithm 2
    Input: $G_f$, $\Delta t_f$, $V_f^0$, $A_f^0$, $\epsilon_f$
    Output: $V_f^*$
end
```

Accuracy. Both VI and PI algorithms require a stopping criterion for convergence. Following [12], the stopping criteria is given by $||V^{k+1} - V^k|| \leq C\Delta x^2$, which relates the error to the resolution of the state-space mesh. The constant C is set to $C = \frac{1}{5}$ for the fine mesh, and for values ranging from 1 to 10 in the coarse mesh, as we do not strive for additional accuracy that usually does not improve the initial guess of the control field.
We now assess the performance of the scheme by presenting two numerical tests related to minimum time optimal control and differential games.

4 Numerical tests

Test 1: Zermelo navigation problem. We consider a minimum time problem to the target $\mathcal{T} = \{x \in \mathbb{R}^2 : ||x||_2 <= 0.2\}$, for states inside the spatial domain $\Omega = [-1, 1]^2$, with system dynamics and parameters given by

$$f(x, y, a) = \begin{pmatrix} 1 + V_b \cos(a) \\ V_b \sin(a) \end{pmatrix}, \quad A = [-\pi, \pi], \quad \Delta t = 0.8\Delta x ,$$

where set A is uniformly discretized into 72 values.
We study two cases, setting $V_b = 0.6$ and $V_b = 1.4$, generating solutions with different reachable sets. In the case $V_b = 1.4$, the reachable set corresponds to Ω, whereas for $V_b = 0.6$ there exists a sharp restriction of the region of the state space able to reach the target in finite time, as shown in Figure 2. For every case, we implement both VI and PI algorithms, and our API iteration procedure. Comparisons of CPU time for different schemes are presented in Table 1; a speedup of 7-8x is observed in the finest meshes for $V_b = 1.4$, and we report that a similar acceleration is achieved in the case $V_b = 0.6$, i.e., in this case the speedup is independent of the regularity of the solution.

Mesh				API		
# nodes	Δx	VI	PI	VI($2\Delta x$)	PI(Δx)	Total
81^2	0.025	32.85 (240)	6.02 (17)	3.1 (36)	0.99 (4)	4.09
161^2	0.0125	184.11 (468)	45.52 (70)	15.47 (115)	4.37 (6)	19.84
321^2	0.00625	1.18e+03 (904)	458.03 (49)	116.9 (339)	33.81 (7)	150.71

Table 1. Test 1: CPU time (iterations) for different algorithms with $Vb = 1.4$.

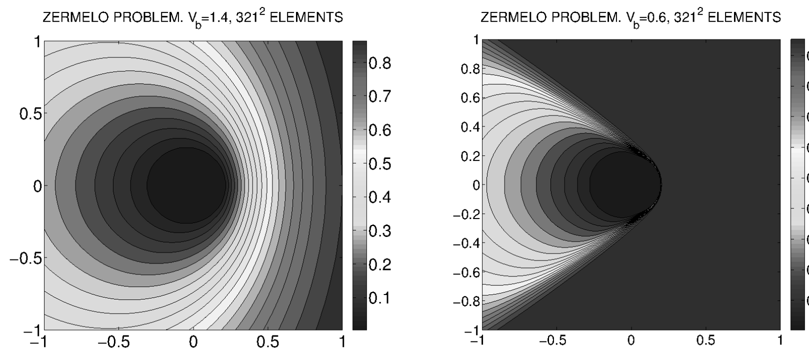

Fig. 2. Zermelo navigation problem: different reachable sets for different choices of the parameter V_b.

Test 2: a two-player pursuit-evasion game. Although there are cases where the application of policy iteration algorithms for differential games can fail to converge, as shown in [4], there exists a class of pursuit-evasion games where such an approach can lead to convergence to the correct solution in a faster way. We consider a reduced-coordinate system for a two-player pursuit-evasion game in two dimensions. In $\Omega = [-1, 1]^2$, we aim at solving a Hamilton-Jacobi-Bellman-Isaacs equation of the form (3) with

$$f(x, y, a, b) = \begin{pmatrix} \sin(b) - 2\sin(a) \\ \cos(b) - 2\cos(a) \end{pmatrix}, \quad A = \left[-\frac{3}{4}\pi, \frac{3}{4}\pi\right], \quad B = [-\pi, \pi].$$

The capture set is defined $\mathcal{T} = \{x \in \mathbb{R}^2 : ||x||_2 <= 0.2\}$, and the implementation procedure follows the same guidelines as for minimum time problem, except that in a single policy iteration, both control fields (for player A and B) are fixed, and the policy update is performed via an *argmaxmin* search. We discretize both control fields into uniform sets of 36 values, and the time step is set $\Delta t = 0.8\Delta x$. Table 2 shows that in this case, the VI is slow even in a coarse mesh, leading to underperformant results for the overall API algorithm. By replacing the coarse VI phase by a coarse PI phase, a speedup of 7x with respect to the fine mesh VI is recovered.

Mesh				API		
# nodes	Δx	VI	PI	VI($2\Delta x$)	PI(Δx)	Total
41^2	0.05	67 (55)	20.08 (19)	21.06 (21)	7.51 (19)	28.57
81^2	0.025	2.29E02(106)	65.41 (35)	58.19 (48)	15.56 (35)	73.75
# nodes	Δx	VI	PI	PI($2\Delta x$)	PI(Δx)	Total
41^2	0.05	67 (55)	20.08 (19)	10.17 (10)	7.46 (19)	17.63
81^2	0.025	2.29E02(106)	65.41 (35)	20.21 (19)	13.32 (35)	33.53

Table 2. Test 2: CPU time (iterations) for different algorithms.

Acknowledgements. This research was supported by the ITN-Marie Curie Grant SADCO, and the FWF project Y432-N15 START-Preis.

References

1. A. Alla, M. Falcone and D. Kalise, *An Efficient Policy Iteration Algorithm for Dynamic Programming Equations*, submitted, available at `arXiv:1308.2087`.
2. M. Bardi and I. Capuzzo-Dolcetta,*Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Birkhäuser, 1997.
3. R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.
4. O. Bokanowski, S. Maroso and H. Zidani, *Some convergence results for Howard's algorithm*, SIAM J. Numer. Anal., **47** (2009), pp. 3001–3026.
5. C. Chow and J. Tsitsiklis, *An optimal one-way multigrid algorithm for discrete-time stochastic control*, IEEE Trans. Aut. Control, **36** (1991), pp. 898–914.
6. M. Falcone and R. Ferretti, *Semi-Lagrangian schemes for linear and Hamilton-Jacobi equations*. SIAM, Philadelphia, 2014.
7. R.A. Howard, *Dynamic programming and Markov processes*. Wiley, 1960.
8. R. Kalaba, *On nonlinear differential equations, the maximum operation and monotone convergence*, J. of Math. Mech., **8** (1959), pp. 519–574.
9. M. Pollatschek and B. Avi-Itzhak, *Algorithms for Stochastic Games with Geometrical Interpretation*, Manage. Sci. **15** (1969), pp. 399–415.
10. M.L. Puterman and S.L. Brumelle, *On the convergence of Policy iteration in stationary Dynamic Programming*, Math. Oper. Res., **4** (1979), pp. 60–69.
11. M.S. Santos, *Numerical solution of dynamic economic models*, in Handbook of Macroeconomics, J.B. Taylor and M. Woodford eds., Elsevier Science, 1999, pp. 311–386.
12. M.S. Santos and J. Rust, *Convergence properties of policy iteration*, SIAM J. Control Optim., **42** (2004), pp. 2094–2115.