September 2010

Williams-Baird Counterexample for Q-Factor Asynchronous Policy Iteration

Dimitri P. Bertsekas †

Abstract

A counterexample due to Williams and Baird [WiB93] (Example 2 in their paper) is transcribed here in the context and notation of two papers by Bertsekas and Yu[BeY10a], [BeY10b], and it is also adapted to the case of Q-factor-based policy iteration. The example illustrates that cycling is possible in asynchronous policy iteration if the initial policy and cost/Q-factor iterations do not satisfy a certain monotonicity condition, under which Williams and Baird [WiB93] show convergence. The papers [BeY10a], [BeY10b] show how asynchronous policy iteration can be modified to circumvent the difficulties illustrated in this example. The purpose of the transcription given here is to facilitate the understanding of this theoretically interesting example, thereby providing motivation and illustration of the methods proposed in [BeY10a], [BeY10b]. The example has not been altered in any material way.

1. COUNTEREXAMPLE

We consider the standard asynchronous version of policy iteration for Q-factors in a discounted problem, where the updates of the policy μ and the Q-factors Q are executed selectively, for only some of the states and state-control pairs (Eqs. (3.1) and (3.2) in [BeY10a]). There are two types of iterations: those corresponding to an index subset K_Q where Q is updated, and those corresponding to the complementary subset K_μ where μ is updated. The algorithm generates a sequence of pairs (Q_k, μ_k), starting from an arbitrary pair (Q_0, μ_0) as follows:

$$Q_{k+1}(i,u) = \begin{cases} (F_{\mu_k} Q_k)(i,u) & \text{if } (i,u) \in R_k, \\ Q_k(i,u) & \text{if } (i,u) \notin R_k, \end{cases} \qquad \forall\, k \in K_Q, \tag{1.1}$$

$$\mu_{k+1}(j) = \begin{cases} \arg\min_{v \in U(j)} Q_k(j,v) & \text{if } j \in S_k, \\ \mu_k(j) & \text{if } j \notin S_k, \end{cases} \qquad \forall\, k \in K_\mu, \tag{1.2}$$

† The author is with the Dept. of Electr. Engineering and Comp. Science, M.I.T., Cambridge, Mass., 02139. His research was supported by the AFOSR under Grant FA9550-10-1-0412.

where F_μ is the mapping given by

$$(F_\mu Q)(i,u) = \sum_{j=1}^{n} p_{ij}(u)\Big(g(i,u,j) + \alpha Q\big(j,\mu(j)\big)\Big), \qquad \forall\ (i,u),$$

and R_k and S_k are subsets of state-control pairs and states, respectively. The notation of the Williams and Baird [WiB93] second counterexample is adapted below, showing that this algorithm may cycle and never converge to the optimal Q-factors.

The states are $i = 1,2,3,4,5,6$, arranged in a cycle in clockwise order, and the controls are denoted h (high cost) and ℓ (low cost). The system is deterministic, and transitions are counterclockwise by either one state or two states.

At states $i = 1,3,5$, only control h is available. The transition from i is to state $(i-1)\mathrm{mod}6$ at cost $g(i,h) = -1$.

At states $i = 2,4,6$, both controls h and ℓ are available. Under h, the transition from i is to state $(i-1)\mathrm{mod}6$ at cost $g(i,h) = -1$. Under ℓ, the transition from i is to state $(i-2)\mathrm{mod}6$ at cost $g(i,\ell) = -3$.

The example works only when the discount factor α satisfies $\alpha > \frac{\sqrt{5}-1}{2}$, e.g., $\alpha = 0.9$.

A sequence of operations of the form (1.1) and (1.2) is exhibited, that regenerates the initial conditions. This sequence consists of three iteration blocks. At the end of each block, the values $Q(i,\mu(i))$ have been shifted clockwise by two states, i.e., are equal to the values of $Q((i-2)\mathrm{mod}6, \mu((\mathrm{i}-2)\mathrm{mod}6))$ prevailing at the beginning of the block. Thus after three blocks the original values $Q(i,\mu(i))$ are repeated for all i.

A peculiarity of this example is that the values $Q(i,\mu(i))$ at the end of an iteration block depend only on the initial values $Q(i,\mu(i))$ at the beginning of the block, so we don't have to specify the initial μ and $Q(i,u)$ for $u \neq \mu(i)$. This fact also allows the adaptation of the example to the case where costs rather than Q-factors are iterated on (cf. [BeY10b]).

The initial values $Q(i,\mu(i))$ at the beginning of the 1st iteration block are

$$Q(i,\mu(i)) = \begin{cases} -\frac{1}{1-\alpha} & \text{if } i = 1, \\ -\frac{1}{1-\alpha} & \text{if } i = 2, \\ -\frac{1+2\alpha}{1-\alpha} & \text{if } i = 3, \\ -\frac{3}{1-\alpha} & \text{if } i = 4, \\ -\frac{1+2\alpha}{1-\alpha} & \text{if } i = 5, \\ -\frac{1}{1-\alpha} & \text{if } i = 6. \end{cases} \tag{1.3}$$

Define an *I-step at node i* to be an update of all the Q-factors of node i via Eq. (1.1), followed by an update of $\mu(i)$ via Eq. (1.2) (this is like a full value iteration/policy improvement).

Define an *E-step at pair* (i,u) to be an update of $Q(i,u)$ based on Eq. (1.1) (this is like a Q-factor policy evaluation based on a single iteration).

The iteration block consists of the following:

1) I-step at $i = 6$.

2) I-step at $i = 4$.

3) I-step at $i = 3$, or equivalently an E-step at $(i, u) = (3, h)$ (since only h is available at $i = 3$).

4) I-step at $i = 1$, or equivalently an E-step at $(i, u) = (1, h)$ (since only h is available at $i = 1$).

5) E-step at $(i, \mu(i))$, where $i = 4$.

Note that steps 1)-4) are ordinary value iterations where all Q-factors of i are updated. The 5th step is a "modified policy iteration" step and causes the difficulty (an algorithm that consists of just I-steps is convergent, as it is just asynchronous value iteration).

It can be verified that the values $Q(i, \mu(i))$ at the end of an iteration block depend only on the initial values $Q(i, \mu(i))$ at the beginning of the block. The reason is that the results of I-steps depend only the prevailing values of the values $Q(i, \mu(i))$, and the last E-step depends only on the prevailing values of $\mu(4)$ (which is set at Step 2).

It is calculated in [WiB93] (p. 19) that at the end of the iteration block the values $Q(i, \mu(i))$ are:

$$Q(i, \mu(i)) = \begin{cases} -\frac{1+2\alpha}{1-\alpha} & \text{if } i = 1, \\ -\frac{1}{1-\alpha} & \text{if } i = 2, \\ -\frac{1}{1-\alpha} & \text{if } i = 3, \\ -\frac{1}{1-\alpha} & \text{if } i = 4, \\ -\frac{1+2\alpha}{1-\alpha} & \text{if } i = 5, \\ -\frac{3}{1-\alpha} & \text{if } i = 6. \end{cases}$$

so the values are shifted clockwise by two states relative to the beginning of the iteration block. For example (allowing for differences in notation and context) [WiB93] calculate the results of iterations as follows:

1) I-step at $i = 6$. Sets $Q(6, h) = -\left(1 + \alpha\frac{1+2\alpha}{1-\alpha}\right) = -\frac{1+2\alpha^2}{1-\alpha}$, $Q(6, \ell) = -\left(3 + \alpha\frac{3}{1-\alpha}\right) = -\frac{3}{1-\alpha}$, and $\mu(6) = \ell$.

2) I-step at $i = 4$. Sets $Q(4, h) = -\left(1 + \alpha\frac{1+2\alpha}{1-\alpha}\right) = -\frac{1+2\alpha^2}{1-\alpha}$, $Q(4, \ell) = -\left(3 + \alpha\frac{1}{1-\alpha}\right) = -\frac{3-2\alpha}{1-\alpha}$, and $\mu(4) = h$ (because $\alpha > \frac{\sqrt{5}-1}{2}$).

3) I-step at $i = 3$. Sets $Q(3, h) = -\left(1 + \alpha\frac{1}{1-\alpha}\right) = -\frac{1}{1-\alpha}$.

4) I-step at $i = 1$. Sets $Q(1, h) = -\left(1 + \alpha\frac{3}{1-\alpha}\right) = -\frac{1+2\alpha}{1-\alpha}$.

5) E-step at $(4, \mu(4))$. Here $\mu(4) = h$ because of step 2, so $Q(4, \mu(4)) = -1 + \alpha Q(3, \mu(3)) = -\left(1 + \alpha\frac{1}{1-\alpha}\right) = -\frac{1}{1-\alpha}$.

Note that only the initial values of $Q(i, \mu(i))$, as given in Eq. (1.3), matter in these calculations.

The second iteration block consists of the following:

1) I-step at $i = 2$.

2) I-step at $i = 6$.

3) I-step at $i = 5$, or equivalently an E-step at $(i, u) = (3, h)$.

4) I-step at $i = 3$, or equivalently an E-step at $(i, u) = (1, h)$.

5) E-step at $(i, \mu(i))$, where $i = 6$.

The third iteration block consists of the following:

1) I-step at $i = 4$.

2) I-step at $i = 2$.

3) I-step at $i = 1$, or equivalently an E-step at $(i, u) = (3, h)$.

4) I-step at $i = 5$, or equivalently an E-step at $(i, u) = (1, h)$.

5) E-step at $(i, \mu(i))$, where $i = 2$.

At the end of this third block the initial conditions $Q(i, \mu(i))$ should be recovered.

The difference between the above calculations and the ones of [WiB93] is just notation: we assume minimization instead of maximization, hence a change of signs in the calculations. Moreover, in [BeY10a] we consider an asynchronous modified policy iteration for Q-factors, while [WiB93] consider an asynchronous modified policy iteration for costs, and $J(i)$ in place of $Q(i, \mu(i))$. With this notational change we also obtain a counterexample for the convergence of algorithm (1.5) of the paper [BeY10b].

Our remedy in the papers [BeY10a], [BeY10b] is to modify the dangerous E-steps so that an oscillation of the type of this example is prevented [e.g., by using the minimum of $\min\{J(j), Q(j, \mu(j))\}$ in place of $Q(j, \mu(j))$ in our Q-learning paper, we introduce a mechanism for guarding against dangerously large increases of $Q(i, u)$].

2. REFERENCES

[BeY10a] Bertsekas, D. P., and Yu, H., 2010. "Q-Learning and Enhanced Policy Iteration in Discounted Dynamic Programming," Lab. for Information and Decision Systems Report LIDS-P-2831, MIT, April, 2010 (revised October 2010).

[BeY10b] Bertsekas, D. P., and Yu, H., 2010. "Distributed Asynchronous Policy Iteration in Dynamic Programming," Proc. of 2010 Allerton Conference on Communication, Control, and Computing, Allerton Park, ILL.

[WiB93] Williams, R. J., and Baird, L. C., 1993. "Analysis of Some Incremental Variants of Policy Iteration: First Steps Toward Understanding Actor-Critic Learning Systems," Report NU-CCS-93-11, College of Computer Science, Northeastern University, Boston, MA.