2.997 Decision-Making in Large-Scale Systems — February 23
MIT, Spring 2004 — Handout #9

Lecture Note 6

1 Application to Queueing Networks

In the first part of this lecture, we will discuss the application of dynamic programming to the queueing network introduced in [1], which illustrates several issues encountered in the application of dynamic programming in practical problems. In particular, we consider the issues that arise when value iteration is applied to problems with a large or infinite state space.

The main points in [1], which we overview today, are the following:

- Naive implementation of value iteration may lead to slow convergence and, in the case of infinite state spaces, policies with infinite average cost in every iteration step, even though the iterates $J_k(x)$ converge pointwise to $J^*(x)$ for every state x;
- Under certain conditions, with proper initialization J_0, we can have a faster convergence and stability guarantees;
- In queueing networks, proper J_0 can be found from well-known heuristics such as fluid model solutions.

We will illustrate these issues with examples involving queueing networks. For the generic results, including a proof of convergence of average-cost value iteration for MDPs with infinite state spaces, refer to [1].

1.1 Multiclass queueing networks

Consider a queueing network as illustrated in Fig. 1.

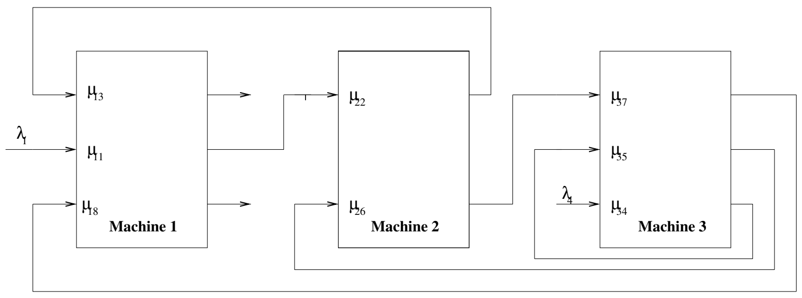

Figure 1: A queueing system

We introduce some notation:

N	the number of queues in the system
λ_i	probability of exogenous arrival at queue i
μ_i	probability that a job at queue i is completed if the job is being served
x_i	state, length of queue i
$g(x) = \sum_{i=1}^{N} x_i$	cost function, in which state $x = (x_1, \dots, x_N)$
$a \in \{0,1\}^N$	$a_i = 1$ if a job from queue i is being served, and $a_i = 0$ otherwise.

The interpretation is as follows. At each time stage, at most one of the following events can happen: a new job arrives at queue i with probability λ_i, a job from queue i that is currently being served has its service completed, with probability μ_i, and either moves to another queue or leaves the system, depending on the structure of the network. Note that, at each time stage, a server may choose to process a job from any of the queues associated with it. Therefore the decision a encodes which queue is being processed at each server. We refer to such a queueing network as *multiclass* because jobs at different queues has different service rates and trajectories through the system.

As seen before, an optimal policy could be derived from the differential cost function h^*, which is the solution of Bellman's equation:

$$\lambda^* + h^* = Th^*.$$

Consider using value iteration for estimating h^*. This requires some initial guess h_0. A common choice is $h_0 = 0$; however, we will show that this can lead to slow convergence of h^*. Indeed, we know that h^* is equivalent to a quadratic, in the sense that there is a constant γ and a solution to Bellman's equation such that $\gamma^{-1} \sum_i x_i^2 \le h^*(x) \le \gamma \sum_i x_i^2$[1]. Now let $h_0 = 0$. Then

$$T^k h_0(x) = \min_u \mathrm{E}\left[\sum_{t=0}^{k-1} \sum_{i=1}^{N} x_i(t) \,\middle|\, x_0 = x \right].$$

Since

$$\mathrm{E}[x_i(t)] = \mathrm{E}[x_i(t-1)] + \underbrace{\mathrm{E}[A_i(t)]}_{=\lambda_i \text{ (arrival)}} \underbrace{-\mathrm{E}[D_i(t)]}_{\le 0 \text{ (departure)}},$$

we have

$$\mathrm{E}[x_i(t) - x_i(t-1)] \le \lambda_i \tag{1}$$

By (1), we have

$$\begin{aligned}
\mathrm{E}[x_i(1)] &\le \mathrm{E}[x_i(0)] + \lambda_i \\
\mathrm{E}[x_i(2)] &\le \mathrm{E}[x_i(1)] + \lambda_i \le \mathrm{E}[x_i(0)] + 2\lambda_i \\
&\vdots \\
\mathrm{E}[x_i(t)] &\le \mathrm{E}[x_i(0)] + t\lambda_i
\end{aligned}$$

[1]You will show this for the special case of a single queue with controlled service rate in problem set 2.

Thus,

$$\begin{aligned} T^k h_0 &\le \sum_{t=0}^{k-1}\sum_{i=1}^{N}(x_i(0)+t\lambda_i) \\ &= \sum_{i=1}^{N} kx_i(0)+\frac{k(k-1)}{2}\lambda_i \end{aligned}$$

This implies that $h_k(x)$ is upper bounded by a linear function of the state x. In order for it to approach a quadratic function of x, the iteration number k must have the same magnitude as x. It follows that, if the state space is very large, convergence is slow. Moreover, if the state space is infinite, which is the case if queues do not have finite buffers, only pointwise convergence of $h_k(x)$ to $h^*(x)$ can be ensured, but for every k, there is some state x such that $h_k(x)$ is a poor approximation to $h^*(x)$.

Example 1 (Single queue length with controlled service rate) *Consider a single queue with*

$$\begin{aligned} &\textit{State } x \textit{ defined as the queue length} \\ &P_a(x,x+1)=\lambda, \qquad \textit{(arrival rate)} \\ &P_a(x,x-1)=\mu_1+a\mu_2, \ \textit{where action } a\in\{0,1\} \\ &P_a(x,x)=1-\lambda-\mu_1-a\mu_2. \end{aligned}$$

Let the cost function be defined as $g_a(x)=(1+a)x$.

The interpretation is as follows. At each time stage, there is a choice between processing jobs at a lower service rate μ_1 *or at a higher service rate* μ_2. *Processing at a higher service rate helps to decrease future queue lengths but an extra cost must be paid for the extra effort.*

Suppose that $\lambda>\mu_1$. *Then if policy* $u(x)=0$ *for all* $x\ge x_0$, *whenever the queue length is at least* x_0, *there are on average more job arrivals than departures, and it can be shown that eventually the queue length converges to infinity, leading to infinite average cost.*

Suppose that $h_0(x)=0, \forall x$. *Then in every iteration* k, *there exists an* x_k *such that* $h_k=T^k h_0(x)=cx+d$ *for all* $x\ge x_k$. *Moreover, when* $h_k=cx+d$ *in a neighborhood of* $\bar{x}$, *the greedy action is* $u_k(\bar{x})=0$, *which is the case that the average cost goes to infinity.*

As shown in [1], using the initial value $h_0(x)=\frac{x^2}{\mu_1+\mu_2-\lambda}$ *leads to stable policies for every iterate* h_k, *and ensures convergence to the optimal policy. The choice of* h_0 *as a quadratic arises from problem-specific knowledge. Moreover, appropriate choices in the case of queueing networks can be derived from well-known heuristics and analysis specific to the field.*

2 Simulation-based Methods

The dynamic programming algorithms studied so far have the following characteristics:

- infinitely many value/and or policy updates are required at every state,
- perfect information about the problem is required, i.e., we must know $g_a(x)$ and $P_a(x,y)$,
- we must know how to compute greedy policies, and in particular compute expectations of the form

$$\sum_y P_a(x,y)J(y). \tag{2}$$

In realistic scenarios, each of these requirements may pose difficulties. When the state space is large, performing updates infinitely often in every state may be prohibitive, or even if it is feasible, a clever order of visitation may considerably speed up convergence. In many cases, the system parameters are not known, and instead one has only access to observations about the system. Finally, even if the transition probabilities are known, computing expectations of the form (2) may be costly. In the next few lectures, we will study simulation-based methods, which aim at alleviating these issues.

2.1 Asynchronous value iteration

We describe the asynchronous value iteration (AVI) as

$$J_{k+1}(x_k) = (TJ_k)(x_k), \quad \forall x_k \in \mathcal{S}_k$$

We have seen that, if every state has its value updated infinitely many times, then the AVI converges (see arguments in Problem set 1). The question remains as to whether convergence may be improved by selecting states in a particular order, and whether we can dispense with the requirement of visiting every state infinitely many times.

We will consider a version of AVI where state updates are based on actual or simulated trajectories for the system. It seems reasonable to expect that, if the system is often encountered at certain states, more emphasis should be placed in obtaining accurate estimates and good actions for those states, motivating performing value updates more often at those states. In the limit, it is clear that if a state is never visited, under any policy, then the value of the cost-to-go function at such a state never comes into play in the decision-making process, and no updates need to be performed for such a state at all. Based on the notion that state trajectories contain information about which states are most relevant, we propose the following version of AVI. We call it *real-time value iteration* (RTVI).

1. Take an arbitrary state x_0. Let $k = 0$.
2. Choose action u_k in some fashion.
3. Let $x_{k+1} = f(x_k, u_k, w_k)$ (recall from lecture 1 that f gives an alternative representation for state transitions).
4. Let $J_{k+1}(x_{k+1}) = (TJ_k)(x_{k+1})$.
5. Let $k = k + 1$ and return to step 2.

2.2 Exploration x Exploitation

Note that there is still an element missing in the description of RTVI, namely, how to choose action u_k. It is easy to see that, if for every state x and y there is a policy u such that there is a positive probability of reaching state y at some time stage, starting at state x, one way of choosing u_k that ensures convergence of RTVI is to select it randomly among all possible actions. This ensures that all states are visited infinitely often, and the convergence result proved for AVI holds for RTVI. However, if we are actually applying RTVI as we run the system, we may not want to wait until RTVI converges before we start trying to use good policies; we would like to use good policies as early as possible. A reasonable choice in this direction is to take an action u_k that is greedy with respect to the current estimate J_k of the optimal cost-to-go function.

In general, choosing u_k greedily does not ensure convergence to the optimal policy. One possible failure scenario is illustrated in Figure 2. Suppose that there is a subset of states B which is recurrent under an optimal policy, and a disjoint subset of states A which is recurrent under another policy. If we start with a guess J_0 which is high enough at states outside region A, and always choose actions greedily, then an action that never leads to states outside region A will be selected. Hence RTVI never has a chance of updating and correcting the initial guess J_0 at states in subset B, and in particular, the optimal policy is never achieved.

It turns out that, if we choose initial value $J_0 \leq J^*$, then the greedy policy selection performs well, as shown in Fig 2(b). We state this concept formally by the following theorem.

The previous discussion highlights a tradeoff that is fundamental to learning algorithms: the conflict of exploitation versus exploration. In particular, there is usually tension between exploiting information accumulated by previous learning steps and exploring different options, possibly at a certain cost, in order to gather more information.

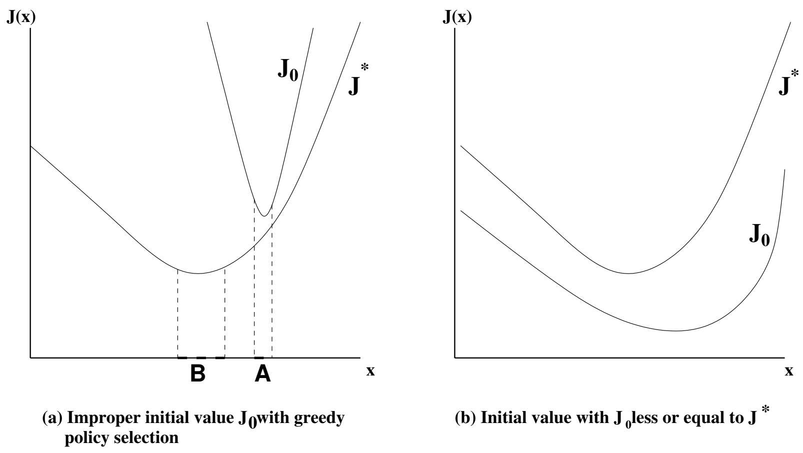

Figure 2: Initial Value Selection

Theorem 1 *If $J_0 \leq J^*$ and all states are reachable from one another, then the real time value iteration algorithm (RTVI) with greedy policy u_t satisfies the following*

(a) $J_k \to J_\infty$ for some J_∞,

(b) $J_\infty = J^*$ for all states visited infinitely many times,

(c) after some iterations, all decisions will be optimal.

Proof: Since T is monotone, we have

$$(TJ_0)(x) \leq (TJ^*)(x),\ \forall x \Rightarrow J_1(x_1) \leq J^*(x_1) \text{ and } J_1(x) = J_0(x) \leq J^*(x), \forall x \neq x_1.$$

We thus conclude that $J_0 \leq J^*$ implies $J_k \leq J^*$ for all k. Let A be the set of states visited infinitely many times. Assume without loss of generality that only states in A are visited and they are visited infinitely many times. Define

$$(T^A J)(x) = \min_a \left\{ g_a(x) + \alpha \sum_{y \in A} P_a(x,y) J(y) + \alpha \sum_{y \notin A} P_a(x,y) J_0(y) \right\},\ \forall\, x \in A.$$

Hence one could regard J as a function from the set A to $\Re^{|A|}$. So T^A is "similar" to DP operator for the subset A of states and

$$||T^A J - T^A \bar{J}||_\infty \leq \alpha ||J - \bar{J}||_\infty.$$

Therefore, RTVI is AVI over A, with every state visited infinitely many times. Thus,

$$J_k(x) \to J_\infty(x) = \begin{cases} J_\infty(x), & \text{if } x \in A, \\ J_0(x), & \text{if } x \notin A. \end{cases}$$

Since the states $x \notin A$ are never visited, we must have

$$P_a(x,y) = 0, \forall\, x \in A, y \notin A,$$

where a is greedy with respect to J_∞. Let u_∞ be the greedy policy of J_∞. Then

$$J_\infty(x) = g_{u_\infty}(x) + \alpha \sum_{y \in A} P_{u_\infty}(x,y) J_\infty(y) = g_{u_\infty}(x) + \alpha \sum_{y \in \mathcal{S}} P_{u_\infty}(x,y) J_\infty(y), \,\forall\, x \in A.$$

Therefore, we conclude

$$J_\infty(x) = J_{u_\infty}(x) \geq J^*(x), \quad \forall\, x \in A.$$

By hypothesis $J_0 \leq J^*$, we know that

$$J_\infty(x) = J^*(x), \quad \forall\, x \in A.$$

$\square$

References

[1] R-R Chen and S.P. Meyn, *Value Iteration and Optimization of Multiclass Queueing Networks*, Queueing Systems, 32, pp. 65–97, 1999.