A New Adaptive Aggregation Algorithm for Infinite Horizon Dynamic Programming*

Chang Zhang and John S. Baras
Department of Electrical and Computer Engineering
and the Institute for Systems Research
University of Maryland, College Park, MD 20742 USA
baras@isr.umd.edu

***Abstract*— Dynamic programming suffers the "curse of dimensionality" when it is employed for complex control systems. State aggregation is used to solve the problem and accelerate computation by looking for a sub-optimal policy. In this paper, a new method, which converges much faster than conventional aggregated value iteration based on TD(0), is proposed for computing the value functions of the aggregated system. Preliminary results show that the new method increases the speed of convergence impressively. Aggregation introduces errors inevitably. An adaptive aggregation scheme employing the new computation method is also proposed to reduce the aggregation errors.**

I. Introduction

Dynamic Programming (DP) plays an important role in intelligent control and optimal control. The "curse of dimensionality" is a major obstacle to deploying DP algorithms for complex control systems. To make the controller carry out the computation and achieve an acceptable performance within a finite time frame, simplifications to the DP algorithms are greatly needed. Many approximate DP algorithms have been developed in the past [1]- [6], [8], [11]- [13], [16], [18], [21]- [24], etc. The techniques include approximate policy evaluation, policy approximation, approximate value iteration by state aggregation, etc.

A natural idea to deal with the "curse of dimensionality" is to use a low-dimensional system to approximate the original system by state aggregation. A continuous-state system can be approximated by a finite number of discrete states, and the system equation is replaced by a set of transition equations between these states. A discrete infinite-state system can be approximated by a finite-state system with dimension N, and we let $N \mapsto \infty$ [17] to reduce errors. A large-dimension finite-state system can be approximated by a smaller dimension finite-state system. State aggregation for continuous-state systems can be found in [16] and [8]. Computational complexity of the aggregated system is less than the original system since less number of "states" are involved. Because after discretization, a continuous-state system can be transformed into a discrete-state system, we will focus on the state aggregation of discrete-state systems in this paper.

*Research supported in part by NSF grant ECS 9727805 and ARO MURI grant GC 169368NGD

A. Problem Formulation

Consider a discrete-time finite state system $S = \{1, 2, ..., n\}$. The action space $U(i)$ at state $i \in S$ is also finite. For a positive integer T, given a policy $\pi = \{\mu_0, \mu_1, ..., \mu_T\}$, the sequence of states becomes a Markov chain with transition probabilities

$$P(x_{t+1} = j | x_t = i, ..., x_0 = i_0, \mu_t(i), ..., \mu_0(i_0)) = P_{i,j}(\mu_t(i)) \quad (1)$$

The infinite horizon cost function is

$$J^\pi(i) = \lim_{T \mapsto \infty} \mathbf{E}\{\sum_{t=0}^{T-1} \alpha^t g_t(x_t, \mu_t(x_t), x_{t+1}) | x_0 = i\} \quad (2)$$

where $0 < \alpha < 1$ is the discount factor and $g_t(x_t, \mu_t(x_t), x_{t+1})$ is the transition cost for taking action $\mu_t(x_t)$ at state x_t and making a transition to state x_{t+1}.

The DP problem is to solve:

$$J^*(i) = \min_\pi (J(i)) \quad (3)$$

The state aggregation method is to aggregate states by certain principles into a smaller number of macro-states (clusters). The state space $S = \{1, ..., n\}$ is divided into disjoint subsets S_1, ..., S_K, with $S = S_1 \cup S_2 \cup ... \cup S_K$ and $K < n$. The value functions of the original states are approximated by the value functions of the macro-states.

B. State Aggregation Algorithms

For a special system with strong and weak interactions, the states can be aggregated by the transition probability matrix [1], [9], [15]. The system equations can be transformed into standard singular perturbation forms. The transformed system has typically a slow part and a fast part with the fast part converging quickly. The dimension of the slow part is small and the optimal control problem can be solved sub-optimally. However, for general systems, such aggregation methods will fail.

Based on policy iteration, Bertsekas [5] developed an adaptive aggregation scheme for policy evaluation. i.e., given a fixed policy, using state aggregation to find the corresponding cost-to-go. However, its selection of the aggregation matrix is based on the assumption of ergodic processes and its number of aggregated groups is fixed. Furthermore, no aggregation is

applied to the policy improvement step, which may limit the speed of the algorithm.

Tsitsiklis and Van Roy derived a TD(0) based two-level state aggregation for infinite horizon DP [6]. For all states in a cluster $S_k, k = 1, ..., K$, the value functions of the states are approximated by a value function $J(S_k)$. Let $\gamma_t > 0$ be the learning rate that satisfies the condition $\sum_{t=0}^{\infty} \gamma_t = \infty$ and $\sum_{t=0}^{\infty} \gamma_t^2 < \infty$. The value iteration algorithm with look-up table is as follows. If $i \in S_k$,

$$J_{t+1}(S_k) := (1 - \gamma_t) J_t(S_k) + \gamma_t \min_u \sum_{j=1}^{n} P_{i,j}(u)[g_t(i, u, j) + \alpha J_t(S_{k(j)})], \quad (4)$$

otherwise, $J_{t+1}(S_k) := J_t(S_k)$.

The following result is well-known for the above iteration.

Theorem 1 [6]

If each cluster is visited infinitely often, then the iteration (4) converges to a limit $J^*(S_k)$. Let

$$\varepsilon(k) = \max_{i,j \in S_k} |J^*(i) - J^*(j)| \quad (5)$$

where $J^*(i)$ is the optimal cost-to-go at state i, and let

$$\varepsilon = \max_k \varepsilon(k), \quad (6)$$

then the error bound for the iteration is:

$$|J^*(S_k) - J^*(i)| \leq \frac{\varepsilon}{1 - \alpha} \text{ with Prob. 1, for all } i \in S_k. \quad (7)$$

Theorem 1 gives a theoretical framework for computing the value functions of the aggregated system based on TD(0). Unfortunately, TD(0) based iteration often suffers the drawback of slow convergence, which limits its practical application. The slow convergence is caused by the diminishing step size (learning rate) γ_t. Moreover, the authors assumed that the aggregation structure is fixed and did not give an efficient aggregation scheme. State aggregation inevitably introduces errors. Inappropriate aggregation methods can cause large aggregation errors.

In this paper, we derive a method to calculate the value functions of the aggregated system much faster than iteration (4). An adaptive aggregation scheme is also proposed to reduce the aggregation errors.

II. A Fast Way to Calculate the Value Functions of the Aggregated System

One important parameter in state aggregation is the sampling probability $q(x|S_k)$. It is the conditional probability that state x will be chosen if the current cluster is S_k. Given a fixed aggregation, different sampling probabilities can lead to different aggregated value functions for iteration (4). Therefore, to ensure the value iteration (4) converges, it is required that the sampling probability be consistent. It means that there exists a positive number B, such that for iterations $t > B$, $q_t(x|S_k) = C$, where C is a constant, for all $x \in S$ and partitions $S_1, ..., S_K$. In general, for a fixed aggregation, the sampling probability is also fixed.

The key idea of fast computation of the value functions for the aggregated system is to avoid using TD(0) based iteration in (4). The following theorem shows that there are other ways to calculate the approximate value functions.

Theorem 2

If we build a new system with state space $X = \{S_1, S_2, ..., S_K\}$, transition probabilities $P_{S_k,S'_k}(u(S_k)) = \sum_{i \in S_k} \sum_{j \in S'_k} q(i|S_k) P_{i,j}(u(i))$ and transition costs $g(S_k, u(S_k), S'_k) = \frac{\sum_{i \in S_k} \sum_{j \in S'_k} q(i|S_k) P_{i,j}(u) g(i,u,j)}{P_{S_k,S'_k}(u(S_k))}$, where $q(i|S_k)$ is the sampling probability used in iteration (4), then employing DP algorithms on the new system will produce the same optimal value functions and optimal policy as those obtained by iteration (4).

Proof: 1) Iteration (4) can be shown to be a Robbins-Monro stochastic approximation of the equation [6]:

$$J^*(S_k) = \sum_{i \in S_k} q(i|S_k) \min_u \sum_{j=0}^{n} P_{i,j}(u)[g(i, u, j) + \alpha J^*(S'_k)] \quad (8)$$

They have the same limit $J^*(S_k)$.

2) For the new system, the Bellman equation is:

$$J^*(S_k) = \min_{u(S_k)} \sum_{S'_k} P_{S_k,S'_k}(u(S_k))[g(S_k, u(S_k), S'_k) + \alpha J^*(S'_k)] \quad (9)$$

It can be changed into:

$$\begin{aligned} J^*(S_k) &= \min_{u(S_k)} \sum_{S'_k} P_{S_k,S'_k}(u(S_k)) \\ &\frac{\sum_{i \in S_k} \sum_{j \in S'_k} q(i|S_k) P_{i,j}(u) g(i,u,j)}{P_{S_k,S'_k}(u(S_k))} + \alpha J^*(S'_k) \\ &= \min_{u(S_k)} \sum_{S'_k} [\sum_{i \in S_k} \sum_{j \in S'_k} q(i|S_k) P_{i,j}(u) g(i,u,j) + \\ &\qquad \alpha P_{S_k,S'_k}(u(S_k)) J^*(S'_k)] \\ &= \min_{u(S_k)} [\sum_{i \in S_k} \sum_{j=1}^{n} q(i|S_k) P_{i,j}(u) g(i,u,j) + \\ &\qquad \alpha \sum_{S'_k} P_{S_k,S'_k}(u(S_k)) J^*(S'_k)] \end{aligned} \quad (10)$$

since

$$\begin{aligned} &\sum_{S'_k} P_{S_k,S'_k}(u(S_k)) J^*(S'_k) \\ &= \sum_{S'_k} \sum_{i \in S_k} \sum_{j \in S'_k} q(i|S_k) P_{i,j}(u) J^*(S'_k) \\ &= \sum_{i \in S_k} q(i|S_k) \sum_{j=1}^{n} P_{i,j}(u) J^*(S'_k) \end{aligned} \quad (11)$$

Substituting equation (11) into equation (10), we obtain:

$$J^*(S_k) = \sum_{i \in S_k} q(i|S_k) \min_u \sum_{j=1}^{n} P_{i,j}(u)[g(i, u, j) + \alpha J^*(S'_k)], \quad (12)$$

which is the same as equation (8). Hence, the theorem holds.

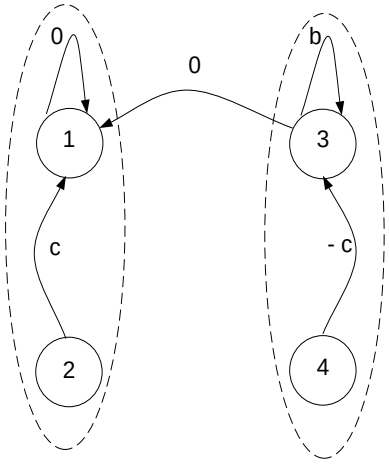

Fig. 1. Example of a state aggregation

In the theorem, $u(S_k)$ is a combination of control actions $u(i), ..., u(j)$ for $i, ..., j \in S_k$. This theorem gives us a foundation to build a new system according to the sampling probability used in iteration (4). Employing DP algorithms for the new system will give the same optimal value functions and optimal policy as those in iteration (4). We can use policy iteration, Gauss-Seidel value iteration or successive over-relaxation methods for the new system directly instead of TD(0) based iteration in (4). For example, we can use the following value iteration to calculate the optimal value functions directly:

$$J_{t+1}(S_k) := \min_{u(S_k)} \sum_{S'_k} P_{S_k,S'_k}(u(S_k))[g(S_k, u(S_k), S'_k) + \alpha J_t(S'_k)] \quad (13)$$

Since these DP algorithms usually converge faster than the iteration based on TD(0), we can achieve significant performance improvement. The error bound in inequality (7) still holds for the solutions of the new system since the approximate value functions will be the same. The following example demonstrates the effectiveness of this method.

Numerical Example

The example in Figure 1 was used in [23] to demonstrate the tightness of the error bound in (7). Four states 1, 2, 3, and 4 are divided into two clusters $S_1 = \{1, 2\}$ and $S_2 = \{3, 4\}$. State 1 is a zero cost absorbing state. A transition from state 2 to state 1 with cost c is inevitable, and a transition from state 4 to state 3 with cost $-c$ always occurs. In state 3, two actions "*move*" and "*stay*" are allowed. The labels on the arcs in Figure 1 represent transition costs. All transition probabilities are one. The parameters in the example satisfy: $c > 0$, $b = \frac{2\alpha c - \delta}{1-\alpha}$ and $0 < \delta < 2\alpha c$.

The optimal value functions for cluster 1 and cluster 2 are calculated. We will compare the convergence speed for value iteration in (13) with the iteration in (4). Two different learning rates $\gamma_t = log(t)/t$ and $\gamma_t = 1/t$ are used in iteration (4). In the comparison, parameters are set as follows: $c = 5$, $b = 3$, $\alpha = 0.9$, sampling probability $q(x|S_1) = 1/2$ for $x = 1, 2$ and $q(x|S_2) = 1/2$ for $x = 3, 4$. The initial values for each cluster are $J(S_1) = J(S_2) = 1$.

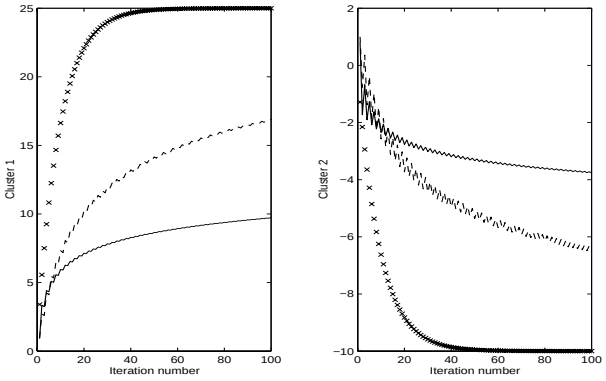

Fig. 2. Value iteration for the aggregated system in 100 runs

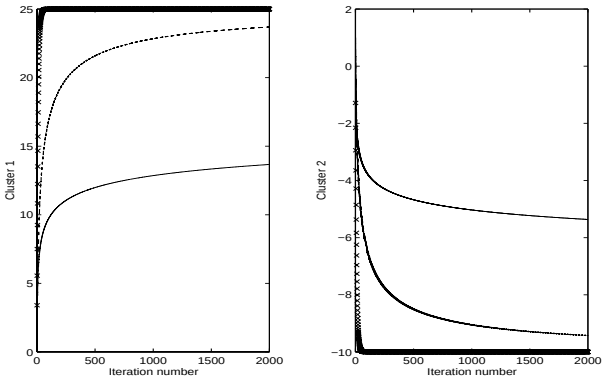

Fig. 3. Value iteration for the aggregated system in 2000 runs

Figure 2 and 3 show the simulation results. In the figures, the curve with symbols "×" is for the iteration (13), the solid line is for the iteration (4) with step size $\gamma_t = 1/t$ and the dashed line is for the iteration (4) with step size $\gamma_t = log(t)/t$.

It can be seen from Figure 2 that the value iteration (13) based on the aggregated system converges to the limits within 60 runs, while the value iterations based on (4) have not converged to the limits in 100 runs. The iteration with step size $\gamma_t = log(t)/t$ performs a lot better than the iteration using step size $\gamma_t = 1/t$. Both of the iterations based on TD(0) converge much more slowly than the value iteration (13). Figure 3 shows the result of value iterations in 2000 runs. The value functions in iteration (4) with learning rate $\gamma_t = log(t)/t$ are close to the limits in run 2000, but the value functions in iteration (4) with learning rate $\gamma_t = 1/t$ are still far away from the limits. In this example, there are only two policies. Therefore, if we use policy iteration for the aggregated system, then we can obtain the optimal value functions within two policy iterations. However, we have to pay the cost of policy evaluation.

Proposition 1

Given an infinite horizon DP problem with discount factor $0 < \alpha < 1$*, we aggregate the states with the same optimal value functions into the same cluster. Let a new system have the transition probabilities* $P_{S_k,S'_k}(u(S_k)) = \sum_{i \in S_k} \sum_{j \in S'_k} q(i|S_k) P_{i,j}(u(i))$ *and transition costs*

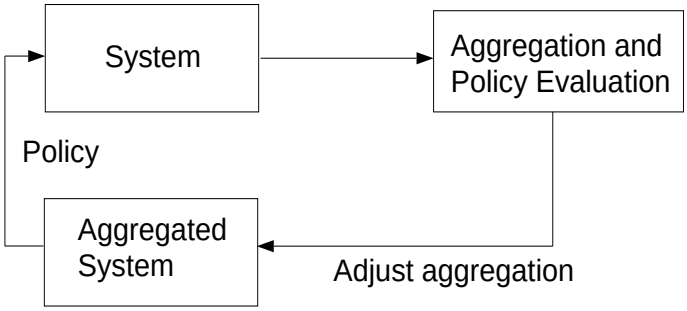

Fig. 4. Block diagram of the adaptive aggregation

$g(S_k, u(S_k), S'_k) = \frac{\sum_{i\in S_k}\sum_{j\in S'_k} q(i|S_k)P_{i,j}(u)g(i,u,j)}{P_{S_k,S'_k}(u(S_k))}$. *Solve the optimal decision problem for the new system using DP, then the states in the new system will have the same optimal value functions as the states in the old system, i.e.,* $J^*(i) = ... = J^*(j) = J^*(S_k)$*, for all* $i, ..., j \in S_k$.

Proof: 1) For the discounted system, if we do value iteration according to (4), then by the error bound in inequality (7), we have $J^*(S_k) = J^*(i) = ... = J^*(j)$, for all $i, ..., j \in S_k$.

2) From Theorem 2, the conclusion follows immediately.

By this proposition, if all the states in each cluster have the same optimal value functions for an aggregated system, then the aggregation will not introduce any error. Such an aggregation is called a perfect aggregation. According to the error bound in inequality (7) and this proposition, for general systems, we should aggregate the states according to their value functions. The states with the same optimal value functions should be aggregated together. However, it usually happens that we do not know the optimal cost-to-go value for each state *apriori*. A solution is to aggregate states according to the current estimated value functions. It leads to the following adaptive aggregation scheme we propose.

III. Adaptive Aggregation for Infinite Horizon DP

The basic idea of adaptive aggregation can be thought as an actor-critic structure. It is shown in Figure 4. The aggregated system can be thought of as an actor that calculates the approximate value functions and derives a sub-optimal policy for the original system. The block of aggregation and policy evaluation can be thought as a critic that evaluates the effect of aggregation and policy. According to the evaluation from the critic, we adjust the aggregation and re-calculate the approximate value functions.

Conventional aggregation methods do not have the critic part. When the actor finishes iteration, the whole algorithm stops and the value functions of the states in a cluster are simply approximated by the value function of the cluster:

$$J^*(i) := ... =: J^*(j) =: J^*(S_k) \tag{14}$$

for $i, ..., j \in S_k$. The critic part can reduce the aggregation error significantly. It is realized by disaggregating the macro-states based on the value functions of the macro-states. It will be introduced in detail in the next section.

A. Adaptive Aggregation Algorithm for Infinite Horizon DP

Consider an ergodic process whose stationary probability distributions are positive for a given policy. (The algorithm applies to general processes. We choose an ergodic process to show how to calculate the sampling probability in this case. Selection of sampling probability for general systems will be given in the next section.) The following algorithm is proposed for a two-level aggregation. Multi-level aggregation can be developed similarly. The algorithm is as follows.

Step 1. Initialize values for stationary probability distribution and cost-to-go functions for the system with state space S and give an aggregation of the system with disjoint subsets $S_1, S_2, ..., S_K$, $K < |S|$ where $|S|$ is the cardinality of space S.

Step 2. Calculate sampling probability

$$q(i|S_k) = \frac{\pi_t(i)}{\sum_{i\in S_k}\pi_t(i)}, \text{for } i \in S_k \tag{15}$$

and calculate $P_{S_k,S'_k}(u(S_k))$ and $g(S_k, u(S_k), S'_k)$ according to Theorem 2.

Step 3. Employ DP algorithm on the aggregated system. We have two choices in this step.

a) Use Gauss-Seidel asynchronous iteration to do one-step or several-step value iteration for the macro states.

$$J_{t+1}(S_k) := \min_{u(S_k)} \sum_{S'_k=0}^{n_1} P_{S_k,S'_k}(u(S_k))[g(S_k, u, S'_k) + \alpha J_t(S'_k)] \tag{16}$$

b) Use policy evaluation/policy improvement to solve the DP equation for the aggregated system directly. Since policy iteration usually finds the optimal policy with a small number of iterations, it can reduce the iteration number of the whole algorithm significantly. The cost is the increasing complexity because of the policy evaluation step. Because the policy evaluation is employed on the aggregated system, the computation load is much less than the policy evaluation on the un-aggregated system.

Step 4. Disaggregate the aggregation and evaluate the current aggregation. This corresponds to the critic part in Figure 4. Let $J_t(i) := J_{t+1}(S_k)$, for all $i \in S_k$. Disaggregate the system by

$$J_{t+1}(i) := \min_u\{\sum_{j=1}^{i-1} P_{i,j}(u)[g(i,u,j) + \alpha J_{t+1}(j)] + \sum_{j=i}^{n} P_{i,j}(u)[g(i,u,j) + \alpha J_t(j)]\} \tag{17}$$

Re-group states according to the new estimate of value functions $J_{t+1}(i)$. There are two ways to re-group the states. The simplest way is to re-order the states according to the current estimates of value functions, and then put adjacent states in one cluster. The number of states in a cluster is fixed for this approach. The drawback is that we do not have the control over the error. It is possible that two adjacent states having a large difference of value functions are put in the same cluster, so that the aggregation error can be large. An improved

approach is as follows. Let $L_t = \max_{i,j \in S} |J_t(i) - J_t(j)|$, and the cardinality of S be N. Define the aggregation distance $\epsilon_t = n\frac{L_t}{N-1}$, where $n \geq 1$. Then we can aggregate states by the distance of ϵ_t. From $\min_{i \in S} J_t(i)$ to $\max_{i \in S} J_t(i)$, we divide the states by the distance of ϵ_t. If there are no states falling into a division, then there is no such macro state corresponding to the division. Therefore, when the algorithm converges, the error of aggregation is bounded by $\frac{\epsilon_\infty}{1-\alpha}$. By letting $\epsilon_{t+1} \leq \epsilon_t$, the aggregation error can be reduced. In practice, the second approach will be employed.

Step 5. According to the optimal policy obtained from step 4, we obtain the new transition probability matrix between states. If the optimal policy is different from before, then the transition probability matrix is usually different too. Therefore, the stationary probability distributions have to be recalculated. When it happens, we can calculate new stationary probability distributions $\pi_{t+1}(i)$ according to Takahashi's algorithm [7] or the Seelen algorithm [19]. Both methods are used to calculate the stationary probability distribution by state aggregation. Given a partition of the system, they can find the solution in an impressively short time period. Due to the limitation of space, readers are referred to read related material in [7], [19].

Step 6. If aggregation adjustment happens in step 4, then update $J_{t+1}(S_k)$ by

$$J_{t+1}(S_k) := \sum_{i \in S_k} \frac{\pi_t(i) J_{t+1}(i)}{\sum_{i \in S_k} \pi_t(i)}; \tag{18}$$

otherwise, skip to next step. When states are re-aggregated, the aggregated value functions are adjusted according to equation (18).

Step 7. Check final errors. If the result is not satisfactory, then let $t := t + 1$ and go back to step 2; otherwise, stop.

In step 3 b), Bertsekas's policy evaluation method can be combined to accelerate the computation [5]. The adaptive aggregation algorithm not only dynamically adjusts clusters according to current estimate of the value functions, but also adjusts the sampling probabilities according to the new policy. The adaptive aggregation algorithm is guaranteed to converge by limiting the number of adjustments on the sampling probability and aggregation. Since when there is no adjustment of the sampling probability and aggregation, the adaptive aggregation becomes fixed aggregation and Theorem 2 applies. The upper bound of state aggregation in (7) applies to the approximate value functions of the adaptive aggregation. In practice, the adaptive scheme is expected to introduce less error than fixed aggregation schemes because of the adaptive feature.

In real life, there are many systems whose states have explicit values. Such systems usually have a very good property: their value functions change monotonically or smoothly with respect to the values of the states. Typical examples are semiconductor processes, machine repair problems and queueing systems, et al. For these systems, initial aggregation is straightforward: adjacent states should be aggregated together. Then the adaptive aggregation algorithm can be used to reduce the aggregation error.

B. Selection of Sampling Probability

How to sample the states in each aggregation is an important problem. Different sampling schemes can lead to different value functions for the same aggregation. Inappropriate sampling methods can lead to large errors. We made the following observations about sampling probability. 1) Under a non-consistent sampling probability, the approximate value iteration (4) may not converge. Therefore, we are interested in consistent sampling schemes which can minimize the approximation error. 2) For a perfectly aggregated system where the states in each cluster have the same optimal value functions, sampling probability has no effect on the aggregation since no error is introduced. 3) For a biased sampling probability where some states are sampled with 0 probability, the error bound in (7) still applies. 4) For an ergodic process whose stationary probability distribution in each state is positive, sampling by equation (15) gives satisfactory result. 5) For a general process, a fair sampling probability is desired. In a cluster, a state with higher probability to be visited should be given a larger sampling probability compared with a state with a low probability to be visited. However, how to select the sampling probability is still an open question and more theoretical work is greatly needed.

C. Numerical Example

The example in Figure 1 is used here to show the effectiveness of the adaptive aggregation algorithm. In the beginning of the algorithm, we still form two clusters as in Figure 1. For the fixed aggregation scheme, the approximation error reaches the upper bound in (7). It is easy to verify that the optimal cost-to-go values at state 1, 2, 3 and 4 are 0, c, 0 and $-c$ respectively. The fixed aggregation method gives the approximate values: $J^*(S_1) = \frac{c}{1-\alpha}$ and $J^*(S_2) = \frac{-c}{1-\alpha}$.

The following is the result using the adaptive aggregation scheme. Because state 1 is an absorbing state and a transition from state 2 to state 1 is inevitable, there are more chances to visit state 1 than state 2 in cluster S_1. The sampling probabilities in cluster S_1 are set as: $q(1|S_1) = 0.9$ and $q(2|S_1) = 0.1$. For a similar reason, we set $q(3|S_2) = 0.8$ and $q(4|S_2) = 0.2$. There are only two possible policies in this example. Define $u_m = \{$action "move" is chosen in state $3\}$ and $u_s = \{$action "stay" is chosen in state $3\}$. Then the corresponding transition probabilities and transition costs for the aggregated system are: $P_{S_1,S_1} = 1$, $P_{S_1,S_2} = 0$, $P_{S_2,S_1}(u_m) = 0.8$, $P_{S_2,S_2}(u_m) = 0.2$, $P_{S_2,S_1}(u_s) = 0$, $P_{S_2,S_2}(u_s) = 1$, $g(S_1, S_1) = 0.9 \times 0 + 0.1 \times c = 0.1c$, $g(S_2, u_m, S_1) = 0$, $g(S_2, u_m, S_2) = -c$ and $g(S_2, u_s, S_2) = 0.8b - 0.2c$.

We let $c = 5$, $b = 3$ and $\alpha = 0.9$ as before. Policy iteration is used here for simplicity and gives results in two iterations: $J_1(S_1) = 5$, $J_1(S_2) = -1.707$ and u_m should be taken in cluster S_2. Then we apply the aggregation evaluation step: Let $J_1(1) := J_1(2) := J_1(S_1)$ and $J_1(3) := J_1(4) := J_1(S_2)$, then

$$J_1(1) := 0 + \alpha J_1(1) = 4.5$$

$$J_1(2) := c + \alpha J_1(1) = 9.5$$

$$J_1(3) := \min\{0 + \alpha J_1(1), b + \alpha J_1(3)\} = 1.464$$

$$J_1(4) := -c + \alpha J_1(3) = -3.683$$

Therefore, the distance $L_1 = \max\{J_1(i) - J_1(j)\} = 13.183$, for $i, j \in S$. Define $\epsilon_1 = \frac{L_1}{|S|-1} = \frac{13.183}{3} = 4.4$. We regroup the states according to step 4 in the adaptive aggregation algorithm. The new clusters are: $S_1 = \{1, 3\}$, $S_2 = \{2\}$ and $S_3 = \{4\}$. Let the sampling probabilities be $q(1|S_1) = 0.6$ and $q(3|S_1) = 0.4$, and we have: $P_{S_1,S_1} = 1$, $P_{S_2,S_1} = 1$ and $P_{S_3,S_1} = 1$ and $g(S_1, u_m, S_1) = 0$, $g(S_1, u_s, S_1) = 0.4b = 1.2$, $g(S_2, S_1) = c = 5$, $g(S_3, S_1) = -c = -5$. Applying policy iteration again, we obtain: $J_2(S_1) = 0$, $J_2(S_2) = c = 5$ and $J_2(S_3) = -c = -5$. Evaluate the aggregation, and the result is $J_2(1) = J_2(3) = 0$, $J_2(2) = c = 5$ and $J_2(4) = -c = -5$. The disaggregated states have the same optimal cost-to-go values as the clusters containing them. Therefore, in this example, the adaptive aggregation algorithm finds the optimal solutions without error.

IV. Conclusions

In this paper, a new method is proposed to calculate the value functions for the aggregated system. Numerical examples show that it speeds up the convergence significantly over conventional value iterations based on TD(0). Inappropriate aggregation can cause large errors. Therefore, an adaptive aggregation scheme is proposed to reduce the errors. The adaptive scheme can be applied to general systems. Numerical examples show the effectiveness of the adaptive aggregation algorithm. In the selection of sampling probability, theoretical work is needed. Also, the example in this paper is rather simple. We plan to apply the algorithm to more complex systems.

References

[1] R. W. Aldhaheri and H. K. Khalil, "Aggregation of the policy iteration method for nearly completely decomposable Markov chains", IEEE Trans. Automatic Control, vol. 36, no. 2, pp. 178-187, 1991.

[2] S. Axsater, "State aggregation in dynamic programming - An application to scheduling of independent jobs on parallel processes", Operations Research Letters, vol. 2, no. 4, pp. 171-176, 1983.

[3] J. S. Baras and V. S. Borkar, "A learning algorithm for Markov Decision Processes with adaptive state aggregation", Technical Report, JB 99-12, 1999.

[4] J. C. Bean, J. R. Birge and R. L. Smith, "Aggregation in dynamic programming", Operations Research, vol. 35, pp. 215-220, 1987.

[5] D. P. Bertsekas and D. A. Castanon, "Adaptive aggregation methods for infinite horizon dynamic programming", IEEE Trans. Automatic Control, vol. 34, no. 6, pp. 589-598, 1989.

[6] D. P. Bertsekas and J. Tsisiklis, "Neuro-dynamic programming", Athena Scientific, 1996.

[7] G. Bolch, S. Greiner, H. De Meer and K. S. Trivedi, Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications, John Wiley & Sons, Inc., 1998.

[8] C. Chow and J. N. Tsitsiklis, "An optimal one-way multigrid algorithm for discrete-time stochastic control", IEEE Transactions on Automatic Control, vol. 36, no. 8, pp.898-914, August 1991.

[9] F. Delebecque and J. Quadrat, "Optimal control of Markov chains admitting strong and weak interaction", Automatica, vol. 17, no. 2, pp. 281-296, 1981.

[10] M. Heger, "Consideration of risk in reinforcement learning", Machine Learning, Proceedings of the Eleventh International Conference, pp. 105-111, 1994.

[11] G. Hu and C. Wu, "Relative value iteration algorithm with soft state aggregation", Control Theory and Applications, pp. 415-418, vo. 17, no. 3, 2000.

[12] G. Jiang, C. Wu and G. Cybenko, "Minimax-based reinforcement learning with state aggregation", Proceedings of the 37th IEEE Conference on Decision and Control, pp. 1236-1241, 1998.

[13] R. Mdndelssohn, "An iterative aggregation procedure for Markov decision processes", Operations Research, vol. 30, no. 1, pp. 62-73, 1981.

[14] A. A. Pervozvanskii and I. N. Gaitsgori, "Suboptimization, decomposition and aggregation", Proceedings of Seventh IFAC World Congress, Helsinki, Finland, 1978.

[15] R. G. Phillips and P. V. Kokotovic, "A singular perturbation approach to modeling and control of Markov chains", IEEE Trans. Automatic Control, vol. AC-26, no. 5, pp. 1087-1094, 1981.

[16] J. P. Quadrat, M. Akian, J. P. Chancelier, "Dynamic programming complexity and application", Proceedings of the 27th Conference on Decision and Control, pp. 1551-1558, December 1988.

[17] L. I. Sennott, Stochastic Dynamic Programming and the Control of Queueing Systems, John Wiley & Sons, Inc, 1999.

[18] S. Singh, T. Jaakkola, M. I. Jordan, "Reinforcement learning with soft state aggregation", Advances in Neural Information Processing Systems 7", pp. 361-368, 1995.

[19] W. J. Stewart, Numerical Solution of Markov Chains, Marcel Dekker, Inc., 1991.

[20] R. S. Sutton, "Learning to predict by the methods of temporal differences", Machine Learning, pp. 9-44, 1988.

[21] W. K. Tsai, G. M. Huang and J. K. Antonio, "Fast parallel hierarchical aggregation/disaggregation algorithm for multistage optimization problems and shortest path problems", Journal of Parallel and Distributed Computing, pp. 1789-1794, 1991

[22] J. N. Tsitsiklis and B. V. Roy, "An analysis of temporal-difference learning with function approximation", technical report, MIT, 1996.

[23] J. N. Tsitsiklis and B. V. Roy, "Feature-based methods for large scale dynamic programming", Machine Learning, vol. 22, pp. 59-94, 1996.

[24] C. Zhang and J. S. Baras, "A hierarchical structure for finite horizon dynamic programming problems", Technical Report, TR 2000-53, Institute for Systems Research, University of Maryland, College Park, 2000.