October 2018 **LIDS Report 3273**

Biased Aggregation, Rollout, and Enhanced Policy Improvement for Reinforcement Learning

Dimitri P. Bertsekas†

Abstract

We propose a new aggregation framework for approximate dynamic programming, which provides a connection with rollout algorithms, approximate policy iteration, and other single and multistep lookahead methods. The central novel characteristic is the use of a *bias function* V of the state, which biases the values of the aggregate cost function towards their correct levels. The classical aggregation framework is obtained when $V \equiv 0$, but our scheme works best when V is a known reasonably good approximation to the optimal cost function J^*.

When V is equal to the cost function J_μ of some known policy μ and there is only one aggregate state, our scheme is equivalent to the rollout algorithm based on μ (i.e., the result of a single policy improvement starting with the policy μ). When $V = J_\mu$ and there are multiple aggregate states, our aggregation approach can be used as a more powerful form of improvement of μ. Thus, when combined with an approximate policy evaluation scheme, our approach can form the basis for a new and enhanced form of approximate policy iteration.

When V is a generic bias function, our scheme is equivalent to approximation in value space with lookahead function equal to V plus a local correction within each aggregate state. The local correction levels are obtained by solving a low-dimensional aggregate DP problem, yielding an arbitrarily close approximation to J^*, when the number of aggregate states is sufficiently large. Except for the bias function, the aggregate DP problem is similar to the one of the classical aggregation framework, and its algorithmic solution by simulation or other methods is nearly identical to one for classical aggregation, assuming values of V are available when needed.

† The author is with the Dept. of Electr. Engineering and Comp. Science, and the Laboratory for Information and Decision Systems (LIDS), M.I.T., Cambridge, Mass., 02139. Thanks are due to John Tsitsiklis for helpful comments.

1. INTRODUCTION

We introduce an extension of the classical aggregation framework for approximate dynamic programming (DP for short). We will focus on the standard discounted infinite horizon problem with state space $\mathcal{S} = \{1, \ldots, n\}$, although the ideas apply more broadly. State transitions (i, j), $i, j \in \mathcal{S}$, under control u occur at discrete times according to transition probabilities $p_{ij}(u)$, and generate a cost $\alpha^k g(i, u, j)$ at time k, where $\alpha \in (0, 1)$ is the discount factor. We consider deterministic stationary policies μ such that for each i, $\mu(i)$ is a control that belongs to a finite constraint set $U(i)$. We denote by $J_\mu(i)$ the total discounted expected cost of μ over an infinite number of stages starting from state i, and by $J^*(i)$ the minimal value of $J_\mu(i)$ over all μ. We denote by J_μ and J^* the n-dimensional vectors that have components $J_\mu(i)$ and $J^*(i)$, $i \in \mathcal{S}$, respectively. As is well known, J_μ is the unique solution of the Bellman equation for policy μ:

$$J_\mu(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\Big(g\big(i, \mu(i), j\big) + \alpha J_\mu(j)\Big), \qquad i \in \mathcal{S}, \tag{1.1}$$

while J^* is the unique solution of the Bellman equation

$$J^*(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i, u, j) + \alpha J^*(j)\big), \qquad i \in \mathcal{S}. \tag{1.2}$$

Aggregation can be viewed as a problem approximation approach: we approximate the original problem with a related "aggregate" problem, which we solve exactly by some form of DP. We thus obtain a policy that is optimal for the aggregate problem but is suboptimal for the original. In the present paper we will extend this framework so that it can be used more flexibly and more broadly. The key idea is to enhance aggregation with prior knowledge, which may be obtained using any approximate DP technique, possibly unrelated to aggregation.

Our point of departure is the classical aggregation framework that is discussed in the author's two-volume textbook [Ber12], [Ber17] (Section 6.2.3 of Vol. I, and Sections 6.1.3 and 6.4 of Vol. II). This framework is itself a formalization of earlier aggregation ideas, which originated in scientific computation and operations research (see for example Bean, Birge, and Smith [BBS87], Chatelin and Miranker [ChM82], Douglas and Douglas [DoD93], Mendelssohn [Men82], Rogers et. al. [RPW91], and Vakhutinsky, Dudkin, and Ryvkin [VDR79]). Common examples of aggregation within this context arise in discretization of continuous spaces, and approximation of fine grids by coarser grids, as in multigrid methods. Aggregation was introduced in the simulation-based approximate DP context, mostly in the form of value iteration; see Singh, Jaakkola, and Jordan [SJJ95], Gordon [Gor95], Tsitsiklis and Van Roy [TsV96] (also the book by Bertsekas and Tsitsiklis [BeT96], Sections 3.1.2 and 6.7). More recently, aggregation was discussed in the context of partially observed Markovian decision problems (POMDP) by Yu and Bertsekas [YuB12], and in a reinforcement learning context involving the notion of "options" by Ciosek and Silver [CiS15], and the notion of "bottleneck simulator" by Serban et. al. [SSP18]; in all these cases encouraging computational results were presented.

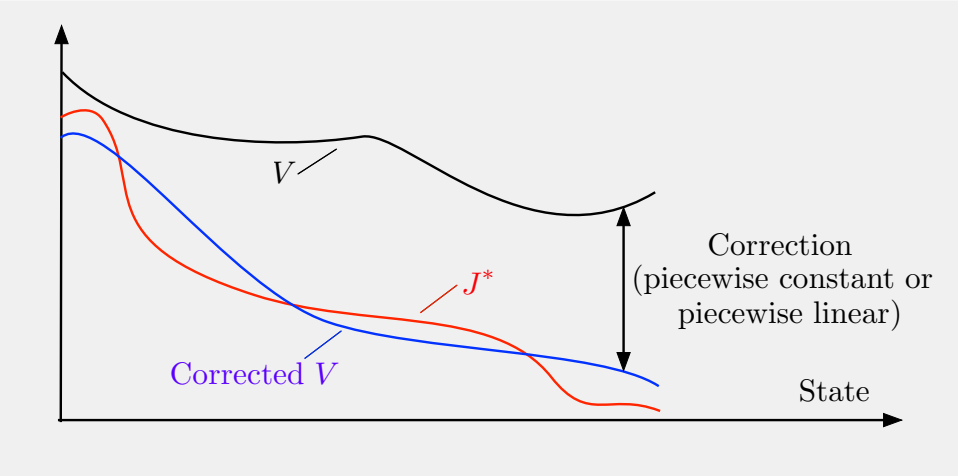

Figure 1.1 Schematic illustration of biased aggregation. It provides an approximation to J^* that is equal to the bias function V plus a local correction. When $V = 0$, we obtain the classical aggregation framework.

In a common type of classical aggregation, we group the states $1, \ldots, n$ into disjoint subsets, viewing each subset as a state of an "aggregate DP problem." We then solve this problem exactly, to obtain a piecewise constant approximation to the optimal cost function J^* (a function that is constant within each subset). This is called *hard aggregation*. The state space partition in hard aggregation is arbitrary, but is often determined by using a vector of "features" of the state (states with "similar" features are grouped together). Features can generally form the basis for specifying aggregation architectures, as has been discussed in Section 3.1.2 of the neuro-dynamic programming book [BeT96], in Section 6.5 of the author's DP book [Ber12], and in the recent survey [Ber18a]. While feature-based selection of the aggregate states can play an important role within our framework, it will not be discussed much, because it is not central to the ideas of this paper.

Our proposed extended framework, which we call *biased aggregation*, is similar in structure to classical aggregation, but involves in addition a function/vector

$$V = \big(V(1), \ldots, V(n)\big)$$

of the state, called the *bias function*, which affects the cost structure of the aggregate problem, and biases the values of its optimal cost function towards their correct levels. With $V = 0$, we obtain the classical aggregation scheme. With $V \neq 0$, biased aggregation yields an approximation to J^* that is equal to V plus a local correction. In hard aggregation the local correction is piecewise constant (it is constant within each aggregate state), while in other forms of aggregation it is piecewise linear; see Fig. 1.1.

An obvious context where biased aggregation can be used is to improve on an approximation to J^* obtained using a different method, such as for example by neural network-based approximate policy iteration, by rollout, or by exact DP applied to a simpler version of the given problem. Another context, is to

integrate the choice of V with the choice of the aggregation framework with the aim of improving on classical aggregation. Generally, if V captures a fair amount of the nonlinearlity of J^*, we speculate that a major benefit of the bias function approach will be a reduction of the number of aggregate states needed for adequate performance, and the attendant facilitation of the algorithmic solution of the aggregate problem.

The bias function idea is new to our knowledge within the context of aggregation. The closest connection to the existing aggregation literature is the adaptive aggregation framework of the paper by Bertsekas and Castanon [BeC89], which proposed to adjust an estimate of the cost function of a policy using local corrections along aggregate states that are formed adaptively using the Bellman equation residual. We will discuss later aggregate state formation based on Bellman equation residuals (see Section 3). However, the ideas of the present paper are much broader and aim to find an approximately optimal policy rather than to evaluate exactly a single given policy. Moreover, consistent with the reinforcement learning point of view, our focus in this paper is on large-scale problems with very large number of states n.

On the other hand there is a close connection between classical aggregation ($V = 0$) and biased aggregation ($V \neq 0$). Indeed, as we will discuss further in the next section, *biased aggregation can be viewed as classical aggregation applied to a modified DP problem*, which is equivalent to the original DP problem in the sense that it has the same optimal policies. The modified DP problem is obtained from the original by changing its cost per stage from $g(i,u,j)$ to

$$g(i,u,j) - V(i) + \alpha V(j)), \qquad i,j \in \mathcal{S},\ u \in U(i). \tag{1.3}$$

Thus the optimal cost function of the modified DP problem, call it $\tilde{J}$, satisfies the corresponding Bellman equation:

$$\tilde{J}(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) - V(i) + \alpha V(j) + \alpha \tilde{J}(j)\big), \qquad i \in \mathcal{S},$$

or equivalently

$$\tilde{J}(i) + V(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\Big(g(i,u,j) + \alpha\big(\tilde{J}(j) + V(j)\big)\Big), \qquad i \in \mathcal{S}. \tag{1.4}$$

By comparing this equation with the Bellman Eq. (1.2) for the original DP problem, we see that the optimal cost functions of the modified and the original problems are related by

$$J^*(i) = \tilde{J}(i) + V(i), \qquad i \in \mathcal{S}, \tag{1.5}$$

and that the the problems have the same optimal policies. This of course assumes that the original and modified problems are solved exactly. If instead they are solved approximately using aggregation or another approximation architecture, such as a neural network, the policies obtained may be substantially different. In particular, the choice of V and the approximation architecture may affect substantially the quality of suboptimal policies obtained.

Equation (1.4) has been used in various contexts, involving error bounds for value iteration, since the early days of DP theory, where it is known as the *variational form of Bellman's equation*, see e.g., [Ber12], Section 2.1.1. The equation can be used to find $\tilde{J}$, which is the variation of J^* from any guess V; cf. Eq. (1.5). The variational form of Bellman's equation is also implicit in the adaptive aggregation framework of [BeC89]. In reinforcement learning, the variational equation (1.4) has been used in various algorithmic contexts under the name *reward shaping* or *potential-based shaping*; see e.g., the papers by Ng, Harada, and Russell [NHR99], Wiewiora [Wie03], Azmuth, Littman, and Zinkov [ALZ08], Devlin and Kudenko [DeK11], Grzes [Grz17] for some representative works. While reward shaping does not change the optimal policies of the original DP problem, it may change significantly the suboptimal policies produced by approximate DP methods that use linear basis function approximation, such as forms of TD(λ), SARSA, and others (see [BeT96], [SuB98]). Basically, with reward shaping and a linear approximation architecture, *V is used as an extra basis function*. This is closely related with the idea of using approximate cost functions of policies as basis functions, already suggested in the neuro-dynamic programming book [BeT96] (Section 3.1.4).

In the next section we describe the architecture of our aggregation scheme and the corresponding aggregate problem. In Section 3, we discuss Bellman's equation for the aggregate problem, and the manner in which its solution is affected by the choice of the bias function. We also discuss some of the algorithmic methodology based on the use of the aggregate problem.

2. AGGREGATION FRAMEWORK WITH A BIAS FUNCTION

The aggregate problem is an infinite horizon Markovian decision problem that involves three sets of states: two copies of the original state space, denoted I_0 and I_1, as well as a finite set $\mathcal{A}$ of aggregate states, as depicted in Fig. 2.1. The state transitions in the aggregate problem go from a state in $\mathcal{A}$ to a state in I_0, then to a state in I_1, and then back to a state in $\mathcal{A}$, and the process is repeated. At state $i \in I_0$ we choose a control $u \in U(i)$, and then transition to a state $j \in I_1$ at a cost $g(i,u,j)$ according to the original system transition probabilities $p_{ij}(u)$. This is the only type of control in the aggregate problem; the transitions from $\mathcal{A}$ to I_0 and the transitions from I_1 to $\mathcal{A}$ involve no control. Thus policies in the context of the aggregate problem map states $i \in I_0$ to controls $u \in U(i)$, and can be viewed as policies of the original problem.

The transitions from $\mathcal{A}$ to I_0 and the transitions from I_1 to $\mathcal{A}$ involve two (somewhat arbitrary) choices of transition probabilities, called the *disaggregation and aggregation probabilities*, respectively; cf. Fig. 2.1. In particular:

(1) For each aggregate state x and original system state i, we specify the *disaggregation probability* d_{xi}, where

$$\sum_{i=1}^{n} d_{xi} = 1, \qquad \forall\ x \in \mathcal{A}.$$

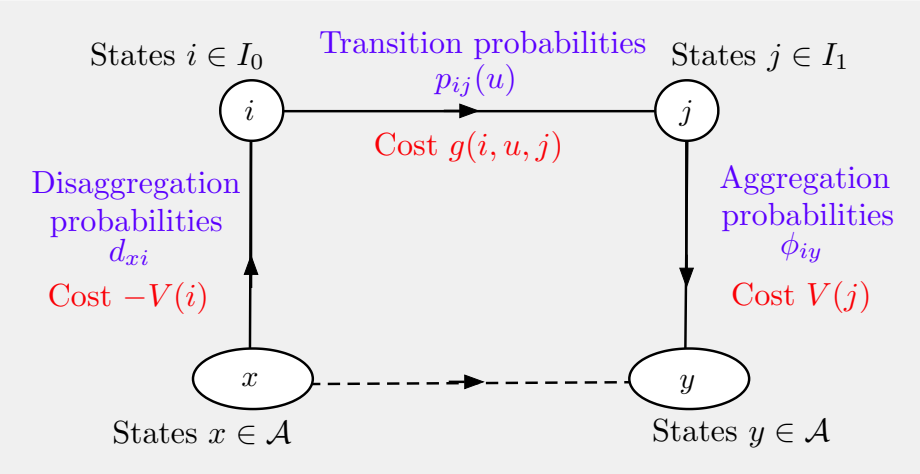

Figure 2.1 Illustration of the transition mechanism and the costs per stage of the aggregate problem in the biased aggregation framework. When the bias function V is identically zero, we obtain the classical aggregation framework.

(2) For each aggregate state y and original system state j, we specify the *aggregation probability* ϕ_{jy}, where

$$\sum_{y \in \mathcal{A}} \phi_{jy} = 1, \qquad \forall\, j \in \mathcal{S}.$$

Several examples of methods to choose the disaggregation and aggregation probabilities are given in Section 6.4 of [Ber12], such as hard and soft aggregation, aggregation with representative states, and feature-based aggregation. The latter type of aggregation is explored in detail in [Ber18a], including its use in conjunction with feature construction schemes that involve neural networks and other architectures.

The definition of the aggregate problem will be complete once we specify the cost for the transitions from $\mathcal{A}$ to I_0 and from I_1 to $\mathcal{A}$. In the classical aggregation scheme, as described in Section 6.4 of [Ber12], these costs are all zero. The salient new characteristic of the scheme proposed in this paper is a (possibly nonzero) cost $-V(i)$ for transition from any aggregate state to a state $i \in I_0$, and of a cost $V(j)$ from a state $j \in I_1$ to any aggregate state; cf. Fig. 2.1. The function V is called the *bias function*, and we will argue that V *should be chosen as close as possible to* J^*. Indeed we will show in the next section that *when* $V \equiv J^*$*, then an optimal policy for the aggregate problem is also optimal for the original problem*, regardless of the choice of aggregate states, and aggregation and disaggregation probabilities. Moreover, we will discuss several schemes for choosing V, such as for example cost functions of various heuristic policies, in the spirit of the rollout algorithm. Generally, while V may be arbitrary, for practical purposes its values at various states should be easily computable.

It is important to note that the biased aggregation scheme of Fig. 2.1 is equivalent to classical aggregation applied to the modified problem where the cost per stage $g(i,u,j)$ is replaced by the cost

$g(i,u,j) - V(i) + \alpha V(j)$ of Eq. (1.3). Thus we can straightforwardly transfer results, algorithms, and intuition from classical aggregation to the biased aggregation framework of this paper. In particular, we may use simulation-based algorithms for policy evaluation, policy improvement, and Q-learning for the aggregate problem, with the only requirement that the value $V(i)$ for any state i is available when needed.

3. BELLMAN'S EQUATION AND ALGORITHMS FOR THE AGGREGATE PROBLEM

The aggregate problem is fully defined as a DP problem once the aggregate states, the aggregation and disaggregation probabilities, and the bias function are specified. Thus its optimal cost function satisfies a Bellman equation, which we will now derive. To this end, we introduce the cost functions/vectors

$$\tilde{r} = \big\{\tilde{r}(x) \mid x \in \mathcal{A}\big\}, \qquad \tilde{J}_0 = \big\{\tilde{J}_0(i) \mid i \in I_0\big\}, \qquad \tilde{J}_1 = \big\{\tilde{J}_1(j) \mid j \in I_1\big\},$$

where:

$\tilde{r}(x)$ is the optimal cost-to-go from aggregate state x.

$\tilde{J}_0(i)$ is the optimal cost-to-go from state $i \in I_0$ that has just been generated from an aggregate state (left side of Fig. 2.1).

$\tilde{J}_1(j)$ is the optimal cost-to-go from state $j \in I_1$ that has just been generated from a state $i \in I_0$ (right side of Fig. 2.1).

Note that because of the intermediate transitions to aggregate states, $\tilde{J}_0$ and $\tilde{J}_1$ are different.

These three functions satisfy the following three Bellman equations:

$$\tilde{r}(x) = \sum_{i=1}^{n} d_{xi}\big(\tilde{J}_0(i) - V(i)\big), \qquad x \in \mathcal{A}, \tag{3.1}$$

$$\tilde{J}_0(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha \tilde{J}_1(j)\big), \qquad i \in \mathcal{S}, \tag{3.2}$$

$$\tilde{J}_1(j) = V(j) + \sum_{y \in \mathcal{A}} \phi_{jy} \tilde{r}(y), \qquad j \in \mathcal{S}. \tag{3.3}$$

By combining these equations, we obtain an equation for $\tilde{r}$:

$$\tilde{r}(x) = (H\tilde{r})(x), \qquad x \in \mathcal{A}, \tag{3.4}$$

where H is the mapping defined by

$$(Hr)(x) = \sum_{i=1}^{n} d_{xi} \left(\min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha \left(V(j) + \sum_{y \in \mathcal{A}} \phi_{jy} r(y) \right) \right) - V(i) \right), \quad x \in \mathcal{A}. \tag{3.5}$$

It can be seen that H is a sup-norm contraction mapping and has $\tilde{r}$ as its unique fixed point. This follows from standard contraction arguments and the fact that d_{xi}, $p_{ij}(u)$, and ϕ_{jy} are all probabilities (see [Ber12], Section 6.5.2). It also follows from the corresponding result for classical aggregation, by replacing the cost per stage $g(i,u,j)$ with the modified cost

$$g(i,u,j) - V(i) + \alpha V(j), \qquad i,j \in \mathcal{S},\ u \in U(i),$$

of Eq. (1.3).

In typical applications of aggregation, $\tilde{r}$ has much lower dimension than $\tilde{J}_0$ and $\tilde{J}_1$, and can be found by solving the fixed point equation (3.4) using simulation-based methods, even when the number of states n is very large; this is the major attraction of the aggregation approach. Once $\tilde{r}$ is found, the optimal cost function J^* of the original problem may be approximated by the function $\tilde{J}_1$ of Eq. (3.3). Moreover, an optimal policy $\tilde{\mu}$ for the aggregate problem may be found through the minimization in Eq. (3.2) that defines $\tilde{J}_0$, i.e.,

$$\tilde{\mu}(i) \in \arg\min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha \tilde{J}_1(j)\big), \qquad i \in \mathcal{S}. \tag{3.6}$$

The policy $\tilde{\mu}$ is optimal for the original problem if and only if $\tilde{J}_1$ and J^* differ uniformly by a constant on the set of states j that are relevant to the minimization above. This is true in particular if $V = J^*$, as shown by the following proposition.

Proposition 3.1: When $V = J^*$ then $\tilde{r} = 0$, $\tilde{J}_0 = \tilde{J}_1 = J^*$, and any optimal policy for the aggregate problem is optimal for the original problem.

Proof: Since the mapping H of Eq. (3.5) is a contraction, it has a unique fixed point. Hence from Eqs. (3.2) and (3.3), it follows that the Bellman equations (3.1)-(3.3) have as unique solution, the optimal cost functions $(\tilde{r}, \tilde{J}_0, \tilde{J}_1)$ of the aggregate problem. It can be seen that when $V = J^*$, then $\tilde{r} = 0$, $\tilde{J}_0 = \tilde{J}_1 = J^*$ satisfy Bellman's equations (3.1)-(3.3), so they are equal to these optimal cost functions. **Q.E.D.**

The preceding proposition suggests that one should aim to choose V as close as possible to J^*. In particular, a good choice of V is one for which the sup-norm of the Bellman equation residual is small, as indicated by the following proposition.

Proposition 3.2: We have

$$\|\tilde{r}\| \leq \frac{\|V - TV\|}{1-\alpha},$$

where $\|\cdot\|$ denotes sup-norm of the corresponding Euclidean space, and $T : \Re^n \to \Re^n$ is the Bellman equation mapping defined by

$$(TV)(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha V(j)\big), \qquad V \in \Re^n,\ i \in \mathcal{S}.$$

Proof: From Eq. (3.5) we have for any vector $r = \big\{r(x) \mid x \in \mathcal{A}\big\}$

$$\|Hr\| \leq \|TV - V\| + \alpha\|r\|.$$

Hence by iteration, we have for every $m \geq 1$,

$$\|H^m r\| \leq (1 + \alpha + \cdots + \alpha^{m-1})\|TV - V\| + \alpha^m \|r\|.$$

By taking the limit as $m \to \infty$ and using the fact $H^m r \to \tilde{r}$, the result follows. **Q.E.D.**

If V is close to J^*, then $\|V - TV\|$ is small (since J^* is the fixed point of T), which implies that $\|\tilde{r}\|$ is small by the preceding proposition. This in turn, by Eq. (3.3), implies that $\tilde{J}_1$ is close to J^*, so that the optimal policy of the aggregate problem defined by Eq. (3.6) is near optimal for the original problem. Choosing V to be close to J^* may not be easy. A reasonable practical strategy may be to use as V the optimal cost function of a simpler but related problem. Another possibility is to select V to be the cost function of some reasonable policy, or an approximation thereof. We will discuss this possibility in some detail in what follows.

In view of Eq. (3.6), an optimal policy for the aggregate problem may be viewed as a one-step lookahead policy with lookahead function $\tilde{J}_1$. In the special case where the scalars

$$\sum_{y \in \mathcal{A}} \phi_{jy}\tilde{r}(y), \qquad j \in \mathcal{S},$$

are the same for all j, the functions $\tilde{J}_1$ and V differ uniformly by a constant, so an optimal policy for the aggregate problem may be viewed as a one-step lookahead policy with lookahead function V. Note that the functions $\tilde{J}_1$ and V differ by a constant in the extreme case where there is a single aggregate state. If in addition V is equal to the cost function J_μ of a policy μ, then an optimal policy for the aggregate problem

is a rollout policy based on μ (i.e., the result of a single policy improvement starting with the policy μ), as shown by the following proposition.

Proposition 3.3: When there is a single aggregate state and $V = J_\mu$ for some policy μ, an optimal policy for the aggregate problem is a policy produced by the rollout algorithm based on μ.

Proof: When there a single aggregate state y, we have $\phi_{jy} = 1$ for all j, so the values $\tilde{J}_1(j)$ and $V(j)$ differ by the constant $\tilde{r}(y)$ for all j. Since $V = J_\mu$, the optimal policy $\tilde{\mu}$ for the aggregate problem is determined by

$$\tilde{\mu}(i) \in \arg\min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J_\mu(j)\big), \qquad i \in \mathcal{S}, \tag{3.7}$$

[cf. Eq. (3.7)], so $\tilde{\mu}$ is a rollout policy based on μ. **Q.E.D.**

We next consider the choice of the aggregate states, and its effect on the quality of the solution of the aggregate problem. We focus on the common case of hard aggregation, but the ideas qualitatively extend to other types of aggregation, such as aggregation with representative states, or feature-based aggregation; see [Ber12], [Ber18a].

3.1. Choosing the Aggregate States - Hard Aggregation

In this section we will focus on the hard aggregation scheme, which has been discussed extensively in the literature (see, e.g., [BeT96], [TsV96], [Van06], [Ber12], [Ber18a]). The starting point is a partition of the state space $\mathcal{S}$ that consists of disjoint subsets $I_1, \ldots, I_q$ of states with $I_1 \cup \cdots \cup I_q = \mathcal{S}$. The aggregate states are identified with these subsets, so we also use the index $\ell = 1, \ldots, q$ to refer to them. The disaggregation probabilities $d_{i\ell}$ can be positive only for states $i \in I_\ell$. To define the aggregation probabilities, let us denote by $\ell(j)$ the index of the aggregate state to which j belongs. The aggregation probabilities are equal to either 0 or 1, according to aggregate state membership:

$$\phi_{j\ell} = \begin{cases} 1 & \text{if } \ell = \ell(j), \\ 0 & \text{otherwise,} \end{cases} \qquad j \in \mathcal{S},\ \ell = 1, \ldots, q. \tag{3.8}$$

In hard aggregation the correction

$$\sum_{\ell=1}^{n} \phi_{j\ell}\tilde{r}(\ell), \qquad j \in \mathcal{S},$$

that is added to $V(j)$ in order to form the aggregate optimal cost function

$$\tilde{J}_1(j) = V(j) + \sum_{\ell=1}^{n} \phi_{j\ell}\tilde{r}(\ell), \qquad j \in \mathcal{S}, \tag{3.9}$$

is piecewise constant: *it is constant within each aggregate state* ℓ, and equal to

$$\tilde{r}\big(\ell(j)\big), \qquad j \in \mathcal{S}.$$

It follows that the success of a hard aggregation scheme depends on both the choice of the bias function V (to capture the rough shape of the optimal cost function J^*) and on the choice of aggregate states (to capture the fine details of the difference $J^* - V$). This suggests that the variation of J^* over each aggregate state should be nearly equal to the variation of V over that state, as indicated by the following proposition, which was proved by Tsitsiklis and Van Roy [TsV96] for the classical aggregation case where $V = 0$. We have adapted their proof to the biased aggregation context of this paper.

Proposition 3.4: In the case of hard aggregation, where we use a partition of the state space into disjoint sets $I_1, \ldots, I_q$, we have

$$\big|J^*(i) - V(i) - \tilde{r}(\ell)\big| \leq \frac{\epsilon}{1-\alpha}, \qquad \forall\ i \text{ such that } i \in I_\ell,\ \ell = 1, \ldots, q, \tag{3.10}$$

where

$$\epsilon = \max_{\ell=1,\ldots,q} \max_{i,j \in I_\ell} \big|J^*(i) - V(i) - J^*(j) + V(j)\big|. \tag{3.11}$$

Proof: Consider the mapping $H : \Re^q \mapsto \Re^q$ defined by Eq. (3.5), and consider the vector $\overline{r}$ with components defined by

$$\overline{r}(\ell) = \min_{i \in I_\ell} \big(J^*(i) - V(i)\big) + \frac{\epsilon}{1-\alpha}, \qquad \ell \in 1, \ldots, q.$$

Using Eq. (3.5), we have for all ℓ, we have

$$\begin{aligned}
(H\overline{r})(\ell) &= \sum_{i=1}^{n} d_{\ell i} \left(\min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) \Big(g(i,u,j) + \alpha\big(V(j) + \overline{r}\big(\ell(j)\big)\big) \Big) - V(i) \right) \\
&\leq \sum_{i=1}^{n} d_{\ell i} \left(\min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha J^*(j) + \frac{\alpha\epsilon}{1-\alpha} \right) - V(i) \right) \\
&= \sum_{i=1}^{n} d_{\ell i} \left(J^*(i) - V(i) + \frac{\alpha\epsilon}{1-\alpha} \right) \\
&\leq \min_{i \in I_\ell} \big(J^*(i) - V(i) + \epsilon\big) + \frac{\alpha\epsilon}{1-\alpha} \\
&= \min_{i \in I_\ell} \big(J^*(i) - V(i)\big) + \frac{\epsilon}{1-\alpha} \\
&= \overline{r}(\ell),
\end{aligned}$$

where for the second equality we used the Bellman equation for the original system, which is satisfied by J^*, and for the second inequality we used Eq. (3.11). Thus we have $H\overline{r} \leq \overline{r}$, from which it follows that $\tilde{r} \leq \overline{r}$ (since H is monotone, which implies that the sequence $\{H^k\overline{r}\}$ is monotonically nonincreasing, and we have

$$\tilde{r} = \lim_{k\to\infty} H^k\overline{r}$$

since H is a contraction and $\tilde{r}$ is its unique fixed point). This proves one side of the desired error bound. The other side follows similarly. **Q.E.D.**

The scalar ϵ of Eq. (3.11) is the maximum variation of $J^* - V$ within the sets of the partition of the hard aggregation scheme. Thus the meaning of the preceding proposition is that if $J^* - V$ changes by at most ϵ within each set of the partition, the hard aggregation scheme provides a piecewise constant correction $\tilde{r}$ that is within $\epsilon/(1-\alpha)$ of the optimal $J^* - V$. If the number of aggregate states is sufficiently large so that the variation of $J^*(i) - V(i)$ within each one is negligible, then the optimal policy of the aggregate problem is very close to optimal for the original problem.

Finally, let us compare hard aggregation for the classical framework where $V = 0$ and for the biased aggregation framework of the present paper where $V \neq 0$. In the former case the optimal cost function $\tilde{J}_1$ of the aggregate problem is piecewise constant (it is constant over each aggregate state). In the latter case $\tilde{J}_1$ consists of V plus a piecewise constant correction (a constant correction over each aggregate state). This suggests that when the number of aggregate states is small, the corrections provided by classical aggregation may be relatively poor, but the biased aggregation framework may still perform well with a good choice for V. As an example, in the extreme case of a single aggregate state, the classical aggregation scheme results in $\tilde{J}_1$ being constant, which yields a myopic policy, while for $V = J_\mu$ the biased aggregation scheme yields a rollout policy based on μ. The combined selection of V and the aggregate states so that they work synergistically is an important issue that requires further investigation.

We will next discuss various aspects of algorithms for constructing a bias function V, and for formulating, solving, and using the aggregate problem in various contexts. There are two algorithmic possibilities that one may consider:

(a) Select V using some method that is unrelated to aggregation, and then solve the aggregate problem, obtain its optimal policy, and use it as a suboptimal policy for the original problem. In the case where V is an approximation to the cost function of some policy, we may view this process as an enhanced form of one-step lookahead.

(b) Solve the aggregate problem multiple times with different choices of V. An example of such a procedure is a policy iteration method, where a sequence of successive policies $\{\mu^k\}$ is obtained by solving the aggregate problems with $V \approx J_{\mu^k}$.

We consider these two possibilities in the next two subsections.

3.2. An Example of Obtaining a Suboptimal Policy by Biased Aggregation

In this section we will provide an example that illustrates one possible way to obtain a suboptimal policy by aggregation. We start with a bias function V. The nature and properties of V are immaterial for the purposes of this section. Here are some possibilities:

(a) V may be an approximation to J^* obtained by one or more approximate policy iterations, using a reinforcement learning algorithm such as TD(λ), LSTD(λ), or LSPE(λ), as a linear combination of basis functions (see textbooks such as [BBD10], [BeT96], [Ber12], [Gos15], [Pow11], [SuB98], [Sze10]). Alternatively, J^* may be approximated using a neural network and simulation-generated cost data, thus obviating the need for knowing suitable basis functions (see e.g., [Ber17]).

(b) V may be an approximation to J^* obtained in some way that is unrelated to reinforcement learning, such as solving exactly a simplified DP problem. For example, the original problem may be a stochastic shortest path problem, while the simpler problem may be a related deterministic shortest path problem, obtained from the original through some form of certainty equivalence approximation. The deterministic problem can be solved fast to yield V using highly efficient deterministic shortest path methods, even when the number of states n is much larger than the threshold for exact solvability of its stochastic counterpart.

Given V, we first consider the formation of the aggregate states. The issues are similar to the case of classical hard aggregation, where the main guideline is to select the aggregate states/subsets so that J^* varies little within each subset. The biased aggregation counterpart of this rule is to select the aggregate states so that $V - J^*$ varies little within each subset (cf. Prop. 3.4). This suggests that a promising guideline may be to select the aggregate states so that the variation of the s-step residual $V - T^s V$ is small within each subset, where $s \geq 1$ is some integer (based on the idea that $T^s V$ is close to J^*). A residual-based approach of this type has been used in the somewhat different context of the paper by Bertsekas and Castanon [BeC89]. Other approaches for selecting the aggregate states based on features are discussed in the survey [Ber18a]. Some further research is needed, both in general and in problem specific contexts, to provide some reliable guidelines for structuring the aggregate problem.

Let us now consider the implementation of the residual-based formation of the aggregate states. We first generate a large sample set of states

$$\hat{\mathcal{S}} = \{i_m \mid m = 1, \ldots, M\},$$

and compute the set of their corresponding s-step residuals

$$\big\{V(i) - (T^s V)(i) \mid i \in \hat{\mathcal{S}}\big\}, \tag{3.12}$$

where

$$(TV)(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big\{g(i,u,j) + \alpha V(j)\big\}.$$

We will not discuss here the method of generating the sample set $\hat{\mathcal{S}}$. Note that the calculation of $(T^s V)(i)$ requires the solution of an s-step DP problem with terminal cost function V, and starting state i. This may be a substantial calculation, which must of course be carried out off-line. However, it is facilitated when s is relatively small, the number of controls is relatively small, there are "few" nonzero probabilities $p_{ij}(u)$ for any i and $u \in U(i)$, and also when parallel computation capability is available.

We divide the range of the set of residuals (3.12) into disjoint intervals $R_1, \ldots, R_q$, and we group the set of sampled states $\hat{\mathcal{S}}$ into disjoint nonempty subsets $I_1, \ldots, I_q$, where I_ℓ is the subset of sampled states whose residuals fall within the interval R_ℓ, $\ell = 1, \ldots, q$.† The aggregate states are the subsets $I_1, \ldots, I_q$, and the disaggregation probabilities are taken to be equal over each of the aggregate states, i.e.,

$$d_{\ell i} = \begin{cases} 1/|I_\ell| & \text{if } i \in I_\ell, \\ 0 & \text{otherwise,} \end{cases} \qquad \ell = 1, \ldots, q, \ i \in \mathcal{S}, \tag{3.13}$$

where $|I_\ell|$ denotes the cardinality of the set I_ℓ (this is a default choice, there may be other problem-dependent possibilities). The aggregation probabilities are arbitrary, although it makes sense to require that for $j \in \hat{\mathcal{S}}$ we have

$$\phi_{j\ell} = \begin{cases} 1 & \text{if } j \in I_\ell, \\ 0 & \text{otherwise,} \end{cases} \qquad \ell = 1, \ldots, q, \ j \in \hat{\mathcal{S}}, \tag{3.14}$$

and also to choose $\phi_{j\ell}$ based on the degree of "similarity" of j with states in I_ℓ (using for example some notion of "distance" between states).

Having specified the aggregate problem, we may solve it by any one of the established simulation-based methods for classical aggregation (see the references noted earlier, and [Ber18a], Section 4.2; see also the next section). These methods apply because the biased aggregation problem corresponding to V is identical to the classical aggregation problem where the cost per stage $g(i,u,j)$ is replaced by $g(i,u,j) - V(i) + \alpha V(j)$, as noted earlier. There are several challenges here, including that the algorithmic solution may be very computation-intensive. On the other hand, algorithms for the aggregate problem aim to solve the fixed point equation (3.4), which is typically low-dimensional (its dimension is the number of aggregate states), even when the number of states n is very large. It is also likely that the use of a "good" bias function will

† More elaborate methods to form the aggregate states are of course possible. A straightforward extension is to partition further the subsets $I_1, \ldots, I_q$ based on some state features, or some other problem-dependent criterion, in the spirit of feature-based aggregation [Ber18a].

reduce the need for a large number of aggregate states in many problem contexts. This remains to be verified by future research. At the same time we should emphasize that as in classical aggregation, a single policy iteration can produce a policy that is arbitrarily close to optimal, provided the number of aggregate states is sufficiently large. Thus, fewer policy improvements may be needed in aggregation-based policy iteration.

As an example of a method to solve the aggregate problem, we may consider a stochastic version of the fixed point iteration $r^{t+1} = Hr^t$, where $H : \Re^q \mapsto \Re^q$ is the mapping with components given by

$$(Hr)(\ell) = \sum_{i=1}^{n} d_{\ell i} \left(\min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha \left(V(j) + \sum_{\ell=1}^{q} \phi_{j\ell} r(\ell) \right) \right) - V(i) \right), \quad \ell = 1, \ldots, q, \quad (3.15)$$

cf. Eq. (3.5).† This algorithm, due to Tsitsiklis and Van Roy [TsV96], generates a sequence of aggregate states

$$\{I_{\ell_t} \mid t = 0, 1, \ldots\}$$

by some probabilistic mechanism, which ensures that all aggregate states are generated infinitely often. Given r^t and I_{ℓ_t}, the algorithm generates an original system state $i_t \in I_{\ell_t}$ according to the uniform probabilities $d_{\ell i}$, and updates the component $r(\ell_t)$ according to

$$r^{t+1}(\ell_t) = (1 - \gamma_t) r^t(\ell_t) + \gamma_t \left(\min_{u \in U(i)} \sum_{j=1}^{n} p_{i_t j}(u) \left(g(i_t, u, j) + \alpha \left(V(j) + \sum_{\ell=1}^{q} \phi_{j\ell} r_\ell^t \right) - V(i_t) \right) \right), \quad (3.16)$$

where γ_t is a positive stepsize, and leaves all the other components unchanged:

$$r^{t+1}(\ell) = r^t(\ell), \qquad \text{if } \ell \neq \ell_t.$$

† The other major alternative approach for solving the aggregate problem is simulation-based policy iteration. This algorithm, discussed in [Ber12], Section 6.5.2, and [Ber18a], Section 4.2, generates a sequence of policies $\{\mu^k\}$ and corresponding vectors $\{r_{\mu^k}\}$ that converge to an optimal policy and cost function $\tilde{r}$ of the aggregate problem, respectively. It involves repeated policy evaluation operations that involve solution of (low-dimensional) fixed point problems of the form $r = H_{\mu^k}(r)$, where H_{μ^k} is the mapping given by

$$(H_{\mu^k} r)(\ell) = \sum_{i=1}^{n} d_{\ell i} \sum_{j=1}^{n} p_{ij}\big(\mu^k(i)\big) \left(g\big(i, \mu^k(i), j\big) + \alpha \left(V(j) + \sum_{m=1}^{q} \phi_{jm}\, r(m) \right) - V(i) \right), \qquad \ell = 1, \ldots, q,$$

[cf. Eq. (3.15)], interleaved policy improvement operations that define μ^{k+1} from the fixed point r_{μ^k} of H_{μ^k} by

$$\mu^{k+1}(i) = \arg \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha \left(V(j) + \sum_{\ell=1}^{q} \phi_{j\ell} r_{\mu^k}(\ell) \right) \right), \qquad i = 1, \ldots, n.$$

The fixed point problems involved in the policy evaluations are linear, so they can be solved using simulation-based algorithms similar to TD(λ), LSTD(λ), and LSPE(λ) (see e.g., [Ber12] and other reinforcement learning books). For detailed coverage of simulation-based methods for solving general linear systems of equations, see the papers by Bertsekas and Yu [BeY07], [BeY09], Wang and Bertsekas [WaB13a], [WaB13b], and the book [Ber12], Section 7.3.

The stepsize γ_t should be diminishing (typically at the rate of $1/t$). We refer to the paper [TsV96] for further discussion and analysis (see also [BeT96], Section 3.1.2 and 6.7).

Under appropriate mild assumptions, the iterative algorithm (3.16) is guaranteed to yield in the limit the optimal cost function $\tilde{r} = \big(\tilde{r}(1), \ldots, \tilde{r}(q)\big)$ of the aggregate problem, where $\tilde{r}(\ell)$ corresponds to aggregate state I_ℓ. Once this happens, the optimal policy of the aggregate problem is obtained from the minimization

$$\tilde{\mu}(i) \in \arg\min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha \left(V(j) + \sum_{\ell=1}^{q} \phi_{j\ell} \tilde{r}(\ell) \right) \right), \qquad i \in \mathcal{S}. \tag{3.17}$$

We note that the convergence of the iterative algorithm (3.16) can be substantially enhanced if a good initial condition r^0 is known. Such an initial condition can be obtained in the special case where $U(i)$ is the same and equal to some set U_ℓ for all $i \in I_\ell$. Then r^0 can be set to $\hat{r} = \big(\hat{r}(1), \ldots, \hat{r}(q)\big)$, the optimal cost function of the simpler aggregate problem, where the policy is restricted to use the same control for all $i \in I_\ell$ (see [Ber12], Section 6.5.1). One way to obtain $\hat{r}$ is by solving the low-dimensional linear programming problem of maximizing $\sum_{\ell=1}^{q} \hat{r}(\ell)$ subject to the constraint $\hat{r} \leq H(\hat{r})$, or

$$\begin{aligned} &\text{maximize} \quad \sum_{\ell=1}^{q} \hat{r}(\ell) \\ &\text{subject to} \quad \hat{r}(\ell) \leq \sum_{i \in I_\ell} d_{\ell i} \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha \left(V(j) + \sum_{s=1}^{q} \phi_{js} \hat{r}(s) \right) - V(i) \right), \quad \ell = 1, \ldots, q, \ u \in U_\ell; \end{aligned}$$

see [Ber12], Sections 2.4 and 6.5.1.

By its nature, the stochastic iterative algorithm (3.16) must be implemented off-line; this is also true for other solution methods for solving the aggregate problem, such as simulation-based policy iteration. On the other hand, after the limit $\tilde{r}$ is obtained, the improved policy $\tilde{\mu}$ of Eq. (3.17) cannot be computed and stored off-line when the number of states n is large, so it must be implemented on-line. This requires forming the expectation in Eq. (3.17) for each $u \in U(i)$, which can be prohibitively time-consuming. An alternative is to calculate off-line approximate Q-factors

$$\tilde{Q}(i,u) \approx \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha \left(V(j) + \sum_{\ell=1}^{q} \phi_{j\ell} \tilde{r}(\ell) \right) \right), \qquad i \in \mathcal{S},$$

using Q-factor samples and a neural network or other approximation architecture. One can then implement the policy $\tilde{\mu}$ by means of the simpler calculation

$$\tilde{\mu}(i) \in \arg\min_{u \in U(i)} \tilde{Q}(i,u), \qquad i \in \mathcal{S},$$

which does not require forming an expectation.

Let us now compare the preceding algorithm with the rollout algorithm, which is just Eq. (3.17) with V equal to the cost function of a base policy and $\tilde{r} = 0$. The main difference is that the rollout algorithm

bypasses the solution of the aggregate problem, and thus avoids the off-line calculation of $\tilde{r}$. Thus a single policy improvement using the preceding aggregation-based algorithm may be viewed as an enhanced form of rollout, where a better performing policy is obtained at the expense of substantial off-line computation.

3.3. Aggregation-Based Approximate Policy iteration

In this section we consider an approximate policy iteration-like scheme, which each iteration k starts with a policy μ^k, and produces an optimal policy μ^{k+1} of the aggregate problem corresponding to $V_k \approx J_{\mu^k}$. We discuss general aspects of this process here, and in the next subsection we provide an example implementation, where J_{μ^k} is itself approximated by aggregation. There are three main issues:

(a) How to calculate bias function values $V_k(i)$ that are good approximations to $J_{\mu^k}(i)$.

(b) How to complete the definition of the aggregate problem, i.e., select the aggregate states, and the aggregation and disaggregation probabilities.

(c) How to obtain μ^{k+1} as an approximately optimal solution of the aggregate problem corresponding to V_k, the aggregate states, and the aggregation/disaggregation probabilities defined by (a) and (b) above.

Regarding question (a), one possibility is to use Monte-Carlo simulation to approximate $J_{\mu^k}(i)$ for any state i as needed, in the spirit of the rollout algorithm. Another possibility is to introduce a simulation-based policy evaluation to approximate J_{μ^k}. For example, we may use TD(λ), LSTD(λ), or LSPE(λ), or a neural network and simulation-generated cost data. Note that such a policy evaluation phase is separate and must be conducted before starting the biased aggregation-based policy improvement that produces the policy μ^{k+1}. A related approach is to aim for a bias function that is closer to J^* than J_{μ^k} is. This may be attempted by using a neural network-based approach based on optimistic policy iteration or Q-learning method such as SARSA and its variants; see [SuB98] and other reinforcement learning textbooks cited earlier.

Regarding question (b), the issues are similar to the situation discussed in the preceding section. As noted there, a promising guideline is to select the aggregate states so that the variation of the s-step residual $V_k - T^s V_k$ is small within each, where $s \geq 1$ is some integer. Note that the calculation of the residuals is simpler than in the case of the preceding section because in the context of the present section only one policy is involved. Similarly, regarding question (c), one may use any of the established simulation-based methods for classical aggregation, as noted in the preceding section.

In the special case where we start with some policy and perform just a single policy iteration, the method may be viewed as an enhanced version of the rollout algorithm. Such a rudimentary form of policy iteration could be the method of choice in a given problem context, because the "improved" policy may be implemented on-line by simple Monte Carlo simulation, similar to the rollout algorithm. This is particularly

so in deterministic problems, such as scheduling, routing, and other combinatorial optimization settings, where the rollout algorithm has been used with success.

Error Bound for Approximate Policy Improvement

Let us now provide an error bound for the difference $J_{\tilde{\mu}} - J_\mu$, where μ is a given policy, and $\tilde{\mu}$ is an optimal policy for the aggregate problem with $V = J_\mu$. By Prop. 3.1, the Bellman equations for μ have the form

$$r_\mu(x) = \sum_{i=1}^{n} d_{xi}\big(J_{0,\mu}(i) - J_\mu(i)\big), \qquad x \in \mathcal{A}, \tag{3.18}$$

$$J_{0,\mu}(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\Big(g\big(i,\mu(i),j\big) + \alpha J_{1,\mu}(j)\Big), \qquad i \in \mathcal{S}, \tag{3.19}$$

$$J_{1,\mu}(j) = J_\mu(j) + \sum_{y\in\mathcal{A}} \phi_{jy} r_\mu(y), \qquad j \in \mathcal{S}. \tag{3.20}$$

and their unique solution, the cost function of μ in the aggregate problem, is $(r_\mu, J_{0,\mu}, J_{1,\mu})$ given by

$$r_\mu = 0, \qquad J_{0,\mu} = J_{1,\mu} = J_\mu.$$

It follows that the optimal cost function $(\tilde{r}, \tilde{J}_0, \tilde{J}_1)$ of the aggregate problem satisfies

$$\tilde{r} \le r_\mu = 0, \qquad \tilde{J}_0 \le J_{0,\mu} = J_\mu, \qquad \tilde{J}_1 \le J_{1,\mu} = J_\mu. \tag{3.21}$$

Moreover optimal policy $\tilde{\mu}$ for the aggregate problem satisfies

$$\tilde{\mu}(i) \in \arg\min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha \tilde{J}_1(j)\big), \qquad i \in \mathcal{S}. \tag{3.22}$$

From Eq. (3.3), we have

$$\tilde{J}_1(j) = J_\mu(j) + \sum_{y\in\mathcal{A}} \phi_{jy}\tilde{r}(y)), \qquad i \in \mathcal{S}.$$

Using the fact $\tilde{r} \le 0$ in the preceding equation, we obtain $\tilde{J}_1 \le J_\mu$, so that

$$T_{\tilde{\mu}}\tilde{J}_1 = T\tilde{J}_1 \le TJ_\mu \le T_\mu J_\mu = J_\mu,$$

where we have introduced the Bellman equation mappings $T_\mu : \Re^n \to \Re^n$, given by

$$(T_\mu J)(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\big(g\big(i,\mu(i),j\big) + \alpha J(j)\big), \qquad J \in \Re^n,\ i \in \mathcal{S},$$

and $T_{\tilde{\mu}} : \Re^n \to \Re^n$, given by

$$(T_{\tilde{\mu}} J)(i) = \sum_{j=1}^{n} p_{ij}\big(\tilde{\mu}(i)\big)\big(g\big(i,\tilde{\mu}(i),j\big) + \alpha J(j)\big), \qquad J \in \Re^n,\ i \in \mathcal{S}.$$

By combining the preceding two relations, we obtain

$$(T_{\tilde{\mu}} J_\mu)(i) + \gamma(i) \leq J_\mu(i), \qquad i \in \mathcal{S}, \tag{3.23}$$

where $\gamma(i)$ is the number

$$\gamma(i) = \alpha \sum_{j=1}^{n} p_{ij}\big(\tilde{\mu}(i)\big) \sum_{y \in \mathcal{A}} \phi_{jy} \tilde{r}(y),$$

which is nonpositive since $\tilde{r} \leq 0$ [cf. Eq. (3.21)]. From Eq. (3.23) it follows, by repeatedly applying $T_{\tilde{\mu}}$ to both sides, that for all $m \geq 1$, we have

$$(T_{\tilde{\mu}}^m J_\mu)(i) + (\gamma + \alpha\gamma + \cdots + \alpha^{m-1}\gamma) \leq J_\mu(i), \qquad i \in \mathcal{S},$$

where

$$\gamma = \min_{i \in \mathcal{S}} \gamma(i),$$

so that by taking the limit as $m \to \infty$ and using the fact $T_{\tilde{\mu}}^m J_\mu \to J_{\tilde{\mu}}$, we obtain

$$J_{\tilde{\mu}}(i) \leq J_\mu(i) - \frac{\gamma}{1-\alpha}, \qquad i \in \mathcal{S}.$$

Thus solving the aggregate problem with V equal to the cost function of a policy yields only an approximate policy improvement for the original problem, with an approximation error that is bounded by $-\frac{\gamma}{1-\alpha}$.

Let us return now to the policy iteration-like scheme that produces at iteration k an optimal policy μ^{k+1} of the aggregate problem corresponding to $V_k = J_{\mu^k}$. From the preceding analysis it follows that even if J_{μ^k} is computed exactly, this scheme will not produce an optimal policy of the original problem. Instead, similar to typical approximate policy iteration schemes, the policy cost functions J_{μ^k} will oscillate within an error bound that depends on the aggregation structure (the aggregate states, and the aggregation and aggregation probabilities); see [BeT96], Section 6.2.2. This suggests that there is potential benefit for changing the aggregation structure with each iteration, as well as for not solving each aggregate problem to completion (as in "optimistic" policy iteration; see [BeT96]). Further research may shed some light into these issues.

3.4. An Example of Policy Evaluation by Aggregation

We will now consider the use of biased aggregation to evaluate approximately the cost function of a fixed given policy μ, thus providing an alternative to Monte Carlo simulation as in rollout, or basis function approximation methods such as TD(λ), LSTD(λ), or LSPE(λ), or neural network-based policy evaluation. We start with some initial function $\hat{J}_0$, and we generate a sequence of functions $\{\hat{J}_k\}$ that converges to an approximation of J_μ. The idea is to mix approximate value iterations for the original problem, with low-dimensional aggregate value iterations, with the aim of approximating J_μ, while bypassing high-dimensional

calculations of order n. This is inspired by the iterative aggregation ideas of Chatelin and Miranker [ChM82] for solving linear systems of equations, and the adaptive aggregation ideas of Bertsekas and Castanon [BeC89], but differs in one important respect: in both papers [ChM82] and [BEC89], the aim is to compute J_μ exactly, so the value iterations should be exact and should involve all states $1,\ldots,n$; this restricts applicability to problems where the value of n is modest. By contrast, in our framework the value iterations involve only a subset of the states, which makes our approach applicable to large-scale problems. The price for this is that we cannot hope to obtain J_μ exactly. We can only aspire to a "good" approximation of J_μ.

The methodology of the paper [ChM82] is also different in that the aggregation framework is static and does not change from one iteration to the next. By contrast, similar to the paper [BeC89], the aggregation framework of this section is adaptive and is changed at the start of each aggregation iteration, with the aggregate states formed based on magnitude of Bellman equation residuals.

Since we will be dealing with a single policy μ, to simplify notation, we will abbreviate $p_{ij}\big(\mu(i)\big)$, $g\big(i,\mu(i),j\big)$, and T_μ, with p_{ij}, $g(i,j)$, and T, respectively. Thus for the purposes of this section, we use the notation

$$(TJ)(i)=\sum_{j=1}^{n}p_{ij}\big(g(i,j)+\alpha J(j)\big),\qquad i\in\mathcal{S},\ J\in\Re^n. \tag{3.24}$$

We introduce a large sample set of states

$$\hat{\mathcal{S}}=\{i_m\mid m=1,\ldots,M\},$$

which may remain fixed through the algorithm, or may be redefined at the beginning of each iteration.

At the beginning of iteration k, we have a function $\hat{J}_k=\big(\hat{J}_k(1),\ldots,\hat{J}_k(n)\big)$. We do not assume that this function is stored in memory, since we do not want to preclude situations where n is very large. Instead we assume that $\hat{J}_k(i)$ can be calculated for any given state i, when needed. Similar to Subsection 3.2, we generate and compute the multistep residuals for the sample states

$$\big\{\hat{J}_k(i)-(T^{s_k}\hat{J}_k)(i)\mid i\in\hat{\mathcal{S}}\big\}, \tag{3.25}$$

where $s_k\geq 1$ is an integer. As noted earlier, this is a feasible calculation using the expression (3.24), even for a large-scale problem, provided the transition probability matrix of μ is sparse (there are "few" nonzero probabilities p_{ij} for any i). The reason is that $(T^{s_k}\hat{J}_k)(i)$ can be calculated with an s_k-step DP calculation as the cost accumulated by the policy μ over s_k steps with terminal cost function $\hat{J}_k$ and starting from i. In particular, to calculate $(T^{s_k}\hat{J}_k)(i)$ we need to generate the tree of s_k-step transition paths starting from i, and accumulate the costs along these paths [including $\hat{J}_k(j)$ at the leaf states j of the tree], weighted by the corresponding probabilities (this is why we need the values of $\hat{J}_k$ at all states j, not just on the states in $\hat{\mathcal{S}}$).

Next, as earlier, we divide the range of the residuals (3.25) into disjoint intervals $R_1,\ldots,R_q$, and we group the set of sampled states $\hat{\mathcal{S}}$ into disjoint nonempty subsets $I_1,\ldots,I_q$, where I_ℓ is the subset of states

whose residuals fall within the interval R_ℓ, $\ell = 1, \ldots, q$. The aggregate states are the subsets $I_1, \ldots, I_q$, and the disaggregation and aggregation probabilities are formed similar to the preceding subsection [cf. Eqs. (3.13) and (3.14)].†

To complete the aggregation framework, we specify the bias function to be

$$V_k(i) = (T^{s_k-1}\hat{J}_k)(i), \qquad i \in \hat{\mathcal{S}},$$

so that

$$TV_k(i) = (T^{s_k}\hat{J}_k)(i), \qquad i \in \hat{\mathcal{S}}.$$

We set $\hat{J}_{k+1}$ to be the aggregate cost function obtained from the aggregation framework just specified. In particular, we first solve the aggregate problem, and obtain the vector $\hat{r}_k$, the fixed point of the corresponding mapping H, cf. Eq. (3.5). This equation in the context of the present section takes the form

$$\begin{aligned}(Hr)(\ell) &= \sum_{i=1}^{n} d_{\ell i}\left((TV_k)(i) - V_k(i) + \alpha\sum_{j=1}^{n} p_{ij}\sum_{\ell=1}^{q}\phi_{j\ell}r(\ell)\right)\\ &= \sum_{i=1}^{n} d_{\ell i}\left((T^{s_k}\hat{J}_k)(i) - (T^{s_k-1}\hat{J}_k)(i) + \alpha\sum_{j=1}^{n} p_{ij}\sum_{\ell=1}^{q}\phi_{j\ell}r(\ell)\right), \qquad \ell = 1, \ldots, q,\end{aligned}$$

where the first equality follows from Eq. Eq. (3.5), and the fact that we are dealing with a single policy, so there is no minimization over u. We do this with either the iterative method (3.16) (without the minimization over u, since we are dealing with a single policy), or by matrix inversion that computes the fixed point of the mapping H (which is linear and low-dimensional). We then define the function $\hat{J}_{k+1}$ on the sample set of states by

$$\hat{J}_{k+1}(i) = (T^{s_k}\hat{J}_k)(i) + \alpha\sum_{j=1}^{n} p_{ij}\sum_{\ell=1}^{q}\phi_{j\ell}\hat{r}_k(\ell), \qquad i \in \hat{\mathcal{S}}.$$

Note that $\hat{J}_{k+1}$ is the result of s_k value iterations applied to $\hat{J}_k$, followed by a correction determined from the solution to the aggregate problem. Proposition 3.2 suggests that it is desirable that $\|V_k - TV_k\|$ is small.

† There are a few questions left unanswered here, such as the method to generate the sample states, the selection of the number of value iterations s_k, the selection of aggregation/disaggregation probabilities, etc. Also there are several variants of the method for forming the aggregate states. For example, the aggregate states may be grouped based on the values of the single step residuals

$$\big\{(T^{s_k-1}\hat{J}_k)(i) - (T^{s_k}\hat{J}_k)(i) \mid i \in \hat{\mathcal{S}}\big\},$$

rather than the multistep residuals (3.25), and they may be further subdivided based on some state features or some problem-dependent criterion. We leave these questions aside for the moment, recognizing that to address them requires experimentation in a variety of problem-dependent contexts.

This in turns indicates that s_k should be chosen sufficiently large, to the point where the value iterations are converging slowly.

The final step before proceeding to the next iteration is to extend the definition of $\hat{J}_{k+1}$ from $\hat{\mathcal{S}}$ to the entire state space $\mathcal{S}$. One possibility for doing this is through a form of interpolation using some nonnegative weights

$$\xi_{ji}, \qquad j \in \mathcal{S},\ i \in \hat{\mathcal{S}},$$

with

$$\sum_{i \in \hat{\mathcal{S}}} \xi_{ji} = 1, \qquad j \in \mathcal{S}, \tag{3.26}$$

and to define

$$\hat{J}_{k+1}(j) = \sum_{i \in \hat{\mathcal{S}}} \xi_{ji} \hat{J}_{k+1}(i), \qquad j \in \mathcal{S}.$$

A possible choice is to use the weights

$$\xi_{ji} = \sum_{\ell=1}^{q} \phi_{j\ell} d_{\ell i}, \qquad j \in \mathcal{S},\ i \in \hat{\mathcal{S}}, \tag{3.27}$$

which intuitively makes sense and satisfies the normalization condition (3.26) since

$$\sum_{i \in \hat{\mathcal{S}}} \xi_{ji} = \sum_{i \in \hat{\mathcal{S}}} \sum_{\ell=1}^{q} \phi_{j\ell} d_{\ell i} = \sum_{\ell=1}^{q} \phi_{j\ell} \sum_{i \in \hat{\mathcal{S}}} d_{\ell i} = \sum_{\ell=1}^{q} \phi_{j\ell} = 1, \qquad j \in \mathcal{S}.$$

With this last step the definition $\hat{J}_{k+1}(i)$ for all states i is complete, and we can proceed to the next iteration. In the case where the sampled set $\hat{\mathcal{S}}$ is equal to the entire state space $\mathcal{S}$, this final step is unnecessary. Then the algorithm becomes very similar to the one of the paper [BeC89]. As discussed in that paper, with appropriate safeguards, the sequence $\{\hat{J}_k\}$ is guaranteed to converge to J_μ thanks to the convergence property of the value iteration algorithm.

Let us finally note that the ideas of this section are applicable and can be extended to the approximate solution of general contractive linear systems of equations, possibly involving infinite dimensional continuous-space operators. The aggregation-based algorithm can be viewed as a hierarchical up-and-down sampling process. Starting with an approximate solution $\hat{J}_k$ defined on the original state space $\mathcal{S}$, we downsample to a lower-dimensional state space defined by the sampled set of states $\hat{\mathcal{S}}$. We divide this set into aggregate states/subsets $I_1, \ldots, I_q$, and formulate an aggregate problem whose solution

$$\hat{r}_k = \big(\hat{r}_1(1), \ldots, \hat{r}_k(q)\big),$$

defines a function $\hat{J}_{k+1}$ on the set $\hat{\mathcal{S}}$. Finally, $\hat{J}_{k+1}$ is upsampled to the original state space using linear interpolation weights such as those of Eq. (3.27).

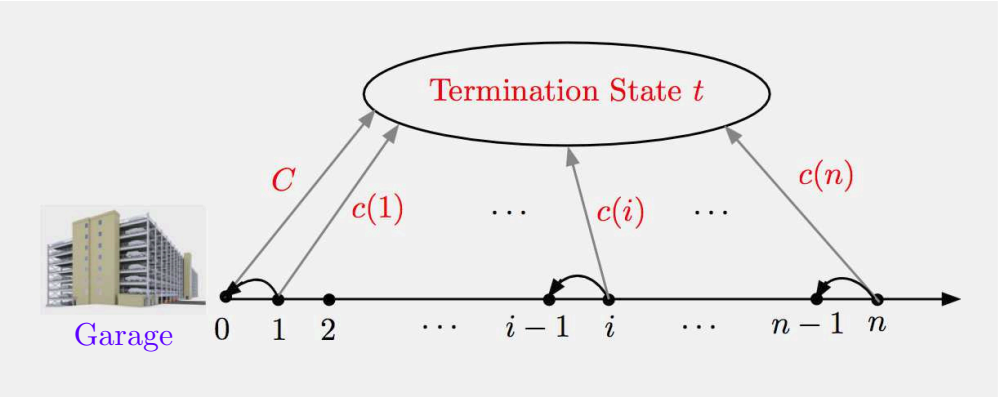

Figure 4.1 Costs and state transitions for the parking problem. At space i we may either park at cost $c(i)$, if the space is free, or continue to the next space $i-1$ at no cost. At space 0 (the garage) we must park at cost C.

4. AN EXAMPLE: INTELLIGENT PARKING

In this section we discuss the aggregation-based solution of a simple type of stopping problem, which was used to test approximate policy iteration with parametric cost function approximation in Section 8.1 of the book [BeT96]. The problem is typical of many problems involving a terminal/stopping cost. When there is no cost incurred prior to stopping, as we will assume in this section, the essence of the problem is to search for a good state to stop. The problem is simple and intuitive, but several more complicated variants, including some involving imperfect state information, are possible. They can be used to test a broad range of reinforcement learning methodologies in an intuitive setting. Here we will focus on the implementation of the biased aggregation methodology for the simplest version of the problem, postponing the discussion of more complex versions of the problem for the future.

Our problem formulation is the same as the one tested in [BeT96]. In particular, a driver is looking for inexpensive parking on the way to his destination. The parking area contains n spaces. The driver starts at space n and traverses the parking spaces sequentially; that is, from space i he goes next to space $i-1$, etc. The destination corresponds to parking space 0. Each parking space is free with probability p independently of whether other parking spaces are free or not. The driver can observe whether a parking space is free only when he reaches it, and then, if it is free, he makes a decision to park in that space or not to park and check the next space. If he parks in space i, he incurs a cost $c(i) > 0$. If he reaches the destination without having parked, he must park in the destination's garage, which is expensive and costs $C > 0$; see Fig. 4.1. We will aim to characterize the optimal parking policy and to approximate it computationally.

We formulate the problem as a stochastic shortest path problem (SSP for short). This is the type of problem where there is no discounting and there is an additional cost-free and absorbing termination state,

denoted t, which in our context corresponds to having parked. In addition we have state 0 that corresponds to reaching the garage, and states (i, F) and $(i, \overline{F})$, $i = 1, \ldots, n$, where (i, F) [or $(i, \overline{F})$] corresponds to space i being free (or not free, respectively). Here the second component of the states (i, F) and $(i, \overline{F})$ is uncontrollable (see e.g., [BeT96], Example 2.2, or [Ber17], Section 1.4), and we can write Bellman's equation in terms of the reduced state i:

$$J^*(i) = p \min\big\{c(i),\, J^*(i-1)\big\} + (1-p)J^*(i-1), \qquad i = 1, \ldots, n, \tag{4.1}$$

$$J^*(0) = C. \tag{4.2}$$

(For convenience we do not include $J^*(t) = 0$ in Bellman's equation.) The optimal costs $J^*(i)$ can be easily calculated recursively from these equations: starting from the known value $J^*(0) = C$, we compute $J^*(1)$ using Eq. (4.1), then $J^*(2)$, etc. Furthermore, an optimal policy has the form

$$\mu^*(i) = \begin{cases} \text{park}, & \text{if space } i \text{ is free and } c(i) \leq J^*(i-1), \\ \text{do not park}, & \text{otherwise.} \end{cases}$$

From Eq. (4.1), we have for all i,

$$\begin{aligned} J^*(i) &= p \min\big\{c(i),\, J^*(i-1)\big\} + (1-p)J^*(i-1) \\ &\leq pJ^*(i-1) + (1-p)J^*(i-1) \\ &= J^*(i-1). \end{aligned}$$

Thus $J^*(i)$ is monotonically nonincreasing in i. From this it follows that if $c(i)$ is monotonically nondecreasing in i, there exists a nonnegative integer i^* such that it is optimal to park at space i if and only if i is free and $i \leq i^*$, as illustrated in Fig. 4.2 for the case where

$$p = 0.05, \quad c(i) = i, \quad C = 100, \quad n = 200. \tag{4.3}$$

The integer i^* is called the *threshold of the optimal policy*. More generally, a policy μ that is characterized by a single integer i_μ such that $\mu(i)$ is to park at space i if an only if i is free and $i \leq i_\mu$, is called a *threshold policy* and i_μ is called the *threshold of* μ.

In Section 8.1 of the book [BeT96], we considered the use of approximate policy iteration with two parametric approximation architectures for J^*: a quadratic polynomial architecture and a more powerful piecewise linear/quadratic architecture. We trained these architectures using approximate policy iteration and Monte-Carlo simulation with each policy evaluated using cost samples from 1000 trajectories. The results illustrated the benefit for using the more powerful linear/quadratic architecture, which ultimately generated policies with thresholds that oscillated between 34 and 37 (the optimal threshold is 35 as indicated in Fig. 4.2). With the less powerful quadratic architecture, policies with thresholds that oscillated in the limit between 43 and 44.

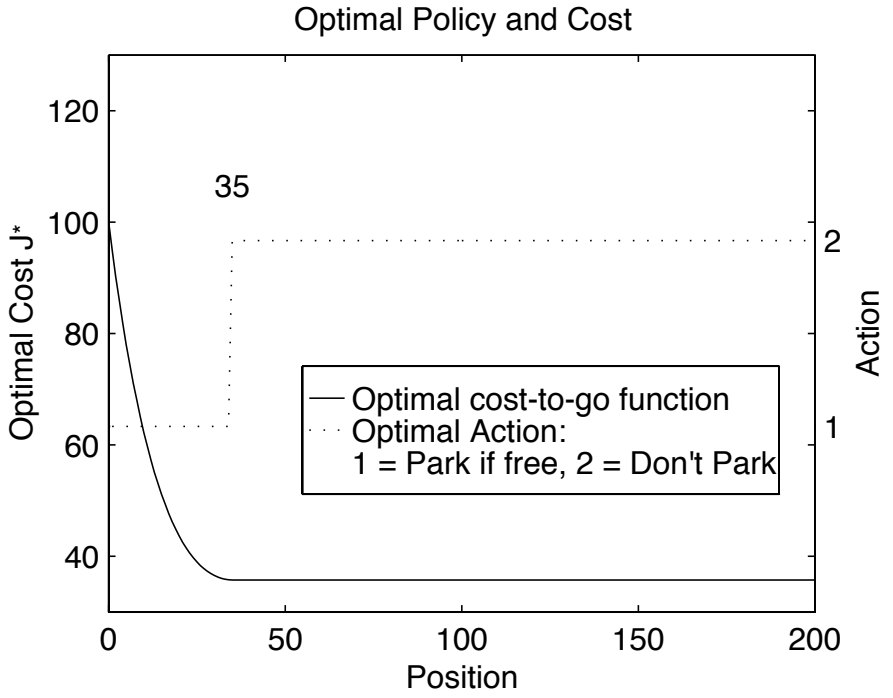

Figure 4.2 Optimal cost-to-go and optimal policy for the parking problem with the data in Eq. (4.3). The optimal policy is to park at the first available space after the threshold space 35 is reached.

Biased Aggregation

We will now discuss the application of biased aggregation to the parking problem. We first note that while we have focused on discounted problems in this paper, our framework extends straightforwardly to SSP problems. The principal change needed is to introduce an additional aggregate termination state $\{t\}$, so that the aggregate problem becomes an SSP problem whose termination state is the aggregate state $\{t\}$.

As before there are also other aggregate states, to which we refer by their index $\ell = 0, 1, \ldots, q$ (the aggregate state 0 is associated with reaching the parking garage, and then going to the termination state t at the next step). The Bellman equation of the aggregate problem has the general form

$$r(\ell) = \sum_{i=1}^{n} d_{\ell i} \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + V(j) - V(i) + \sum_{m=1}^{q} \phi_{jm}\, r(m) \right), \qquad \ell = 1, \ldots, q,\ i = 1, \ldots, n, \tag{4.4}$$

$$r(0) = C - V(0);$$

[cf. Eqs. (3.4) and (3.5)], where the minimum is taken over the two options of parking at state i (when i is free to park) and continuing to $i-1$. We will specialize this equation to the parking context shortly. The equation has a unique solution $\tilde{r}$ under some well-known conditions that date to the paper by Bertsekas and Tsitsiklis [BeT91] (there exists at least one proper policy, i.e., a stationary policy that guarantees eventual

termination from each initial state with probability 1; moreover all stationary policies that are not proper have infinite cost starting from some initial state). When designing the aggregation framework, we must make sure that these conditions are satisfied for the aggregate problem. The optimal policy for the aggregate problem is defined by V and $\tilde{r}$ according to

$$\tilde{\mu}(i) \in \arg\min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha \left(V(j) + \sum_{\ell=1}^{q} \phi_{j\ell} \tilde{r}(\ell) \right) \right), \qquad i = 1, \ldots, n.$$

To apply biased aggregation to the problem, we need to make two choices:

(a) The bias function V. We require here that $V(t) = 0$.

(b) The aggregation framework, i.e., the aggregate states, and the disaggregation and aggregation probabilities.

There are several possibilities for bias function:

(1) $V(i) = 0$ for all $i = 0, 1, \ldots, n$. This corresponds to classical aggregation.

(2) The parking/stopping cost: $V(i) = c(i)$ for $i = 1, \ldots, n$, and $V(0) = C$.

(3) The cost function of the policy that never stops, except at the parking garage: $V(i) = C$ for all i.

(4) The cost function of some threshold policy; this can be calculated by value iteration using the Bellman equation of the policy.

(5) The minimum of the cost functions of several threshold policies.

Regarding the aggregation framework, the simplest possibility is based on a form of uniform "discretization." In particular, we divide the states $1, \ldots, n$ into q equal intervals of some length N (assuming that $n = qN$ for convenience). These intervals are

$$[1, N],\ [N+1, 2N], \ldots, \big[(q-1)N + 1, qN\big]. \tag{4.5}$$

Hard Aggregation

Let us consider hard aggregation with the subset of states $\{1, \ldots, n\}$ partitioned into the above intervals. Then the aggregate states are the intervals $\big[(\ell-1)N+1, \ell N\big]$, $\ell = 1, \ldots, q$, the parking garage state $\{0\}$, and the (aggregate) termination state $\{t\}$. For any $i \in \{1, \ldots, n\}$, let us denote by $\ell(i)$ the interval/aggregate state that contains i, so that

$$\big(\ell(i) - 1\big)N + 1 \leq i \leq \ell(i)N, \qquad i = 1, \ldots, n,\ \ell = 1, \ldots, q.$$

The aggregation probabilities for hard aggregation map original system states j to the aggregate state/interval $\ell(j)$ to which they belong, i.e.,

$$\phi_{j\ell} = \begin{cases} 1 & \text{if } \ell = \ell(j), \\ 0 & \text{otherwise,} \end{cases} \qquad j = 1, \ldots, n,\ \ell = 1, \ldots, q,$$

cf. Eq. (3.8), and $\phi_{00} = \phi_{tt} = 1$.

Regarding the disaggregation probabilities $d_{\ell i}$, $i = 1, \ldots, n$, there are several possible choices, subject to the restriction that

$$d_{\ell i} = 0 \quad \text{if} \quad \ell(i) \neq \ell,$$

which is generic for hard aggregation. We also require that the disaggregation probability is positive for the left endpoints of each aggregate state/interval,

$$d_{\ell i} > 0 \quad \text{if} \quad \ell(i) = \ell \text{ and } i = \big(\ell(i) - 1\big)N + 1.$$

This guarantees that in the aggregate problem, the termination state t will be reached under all policies. Without this restriction, the aggregation framework breaks down because there is no mathematical incentive to park in the aggregate DP problem (we can stay at any one of the aggregate states $\ell = 1, \ldots, q$ at no cost). One possible choice for disaggregation probabilities is the uniform distribution

$$d_{\ell i} = \frac{1}{N} \quad \text{if} \quad \ell(i) = \ell, \qquad i = 1, \ldots, n,\ \ell = 1, \ldots, q. \tag{4.6}$$

Another possibility is to assign probability $1/2$ to the two endpoints of the interval $[(\ell - 1)N + 1, \ell N]$, i.e.,

$$d_{\ell i} = \begin{cases} \frac{1}{2} & \text{if } i = \big(\ell(i) - 1\big)N + 1, \\ \frac{1}{2} & \text{if } i = \ell(i)N, \\ 0 & \text{otherwise,} \end{cases} \qquad i = 1, \ldots, n,\ \ell = 1, \ldots, q. \tag{4.7}$$

Under the choice (4.6) for the disaggregation probabilities, we can now write in more detail Bellman's equation for the aggregate problem cf. Eq. (4.4). This equation, in the form of the three equations (3.1), (3.2), and (3.3), has the form

$$\tilde{r}(\ell) = \frac{1}{N} \sum_{m=1}^{N} \Big(\tilde{J}_0\big((\ell - 1)N + m\big) - V\big((\ell - 1)N + m\big) \Big), \quad \ell = 1, \ldots, q, \qquad \tilde{r}(0) = \tilde{J}_0(0) - V(0), \tag{4.8}$$

$$\tilde{J}_0(i) = p \min\big\{c(i),\, \tilde{J}_1(i-1)\big\} + (1-p)\tilde{J}_1(i-1), \quad i = 1, \ldots, n, \qquad \tilde{J}_0(0) = C, \tag{4.9}$$

$$\tilde{J}_1(j) = V(j) + \tilde{r}\big(\ell(j)\big), \quad j = 1, \ldots, n, \qquad \tilde{J}_1(0) = V(0) + \tilde{r}(0). \tag{4.10}$$

Under the choice (4.7) for the disaggregation probabilities, the corresponding equations are

$$\tilde{r}(\ell) = \frac{1}{2}\Big(\tilde{J}_0\big((\ell-1)N+1\big) - V\big((\ell-1)N+1\big) \Big) + \frac{1}{2}\big(\tilde{J}_0(\ell N) - V(\ell N) \big), \quad \ell = 1, \ldots, q, \qquad \tilde{r}(0) = \tilde{J}_0(0) - V(0),$$

$$\tilde{J}_0(i) = p\min\big\{c(i),\, \tilde{J}_1(i-1)\big\} + (1-p)\tilde{J}_1(i-1), \quad i = 1,\ldots,n, \qquad \tilde{J}_0(0) = C,$$

$$\tilde{J}_1(j) = V(j) + \tilde{r}\big(\ell(j)\big), \quad j = 1,\ldots,n, \qquad \tilde{J}_1(0) = V(0) + \tilde{r}(0).$$

As a check for the above equations, note that for $V = J^*$, the solution is $\tilde{r} = 0$, $\tilde{J}_0 = \tilde{J}_1 = J^*$, while the optimal policy for the aggregate problem is also optimal for the original problem. Note also that for the classical aggregation framework, i.e., when $V = 0$, $\tilde{J}_1$ is piecewise constant (it is constant over each aggregate state). However, the constant levels depend on the aggregation probabilities. When $V \neq 0$, the levels $\tilde{r}(\ell)$ provide a correction on V, and the correction is piecewise constant over each aggregate state.

The algorithmic solution of the preceding Bellman equations is based on solving the fixed point equation $r = H(r)$, where H is defined by the mapping on the right side of Eq. (4.4). This can be done using the stochastic form of value iteration (3.16), which involves generating a sequence of sample aggregate states $\{\ell_t\}$, a sequence of states $\{i_t\}$ according to the disaggregation probabilities $d_{\ell_t i}$, and corresponding updates of the components $\tilde{r}(\ell_t)$; cf. Eq. (3.16). Alternatively, we may solve the aggregate problem using a simulation-based policy iteration method, as discussed in Section 3.2. However, there is a simpler nonstochastic iterative method here, which exploits the special structure of the parking problem, and can be used when n has relatively modest size. In particular, looking at the Markov chain of the aggregate problem, we see that after landing at aggregate state $\ell = 1,\ldots,q$, we transition to a state $i \in [(\ell-1)N+1, \ell N]$ according to the disaggregation probabilities $d_{\ell i}$, and then one of the following happens:

(a) Move to t if i is free and the parking decision is made.

(b) Stay in aggregate state ℓ if $i > (\ell-1)N+1$ and the parking decision is not made.

(c) Move to aggregate state $\ell-1$ if $i = (\ell-1)N+1$ and the parking decision is not made.

A consequence of this is that the fixed point equation $\tilde{r} = H(\tilde{r})$ can be written in the form

$$\tilde{r}(\ell) = H_\ell\big(\tilde{r}(\ell-1), \tilde{r}(\ell)\big), \qquad \ell = 1,\ldots,q,$$

i.e., the ℓth component of the mapping H depends only on the variables $\tilde{r}(\ell-1)$ and $\tilde{r}(\ell)$. This allows the sequential calculation of the components of $\tilde{r}$: first compute $\tilde{r}(1)$, using the known value $\tilde{r}(0) = \tilde{J}_0(0) - V(0)$, as the limit of the sequence $\big\{\tilde{r}^m(1)\big\}$ generated by the one-dimensional fixed point iteration

$$\tilde{r}^{m+1}(1) = H_1\big(\tilde{r}(0), \tilde{r}^m(1)\big), \qquad m = 0, 1, \ldots,$$

where the initial condition $\tilde{r}^0(1)$ can be any scalar.† Then similarly compute $\tilde{r}(2)$ [using $\tilde{r}(1)$ and the corresponding one-dimensional fixed point iteration], and so on up to $\tilde{r}(q)$. The optimal cost function $\tilde{J}_1$

† Let us illustrate the computations involved in this iteration. Suppose we have computed $\tilde{r}^m(1)$. We use this value in Eq. (4.10) to calculate $\tilde{J}_1^m(j)$ for $j = 0, 1, \ldots, N-1$. Then we use these values in Eq. (4.9) to calculate $\tilde{J}_0^m(j)$ for $j = 1,\ldots,N$. Finally, we use these values in Eq. (4.8) to calculate $\tilde{r}^{m+1}(1)$.

and corresponding optimal policy of the aggregate problem can now be computed, and be compared to the optimal cost function J^* and corresponding optimal policy of the original problem.

Alternative Aggregation Schemes: Representative States

Let us also discuss briefly an alternative to hard aggregation, which is *aggregation with representative states* (see [Ber12, Section 6.5, and [Ber18a], Section 4). We consider again the intervals (4.5), but we use as aggregate states the midpoints of these intervals, together with the parking garage state $\{0\}$, and the (aggregate) termination state $\{t\}$ (we assume that the interval length N is an odd number).

Here the disaggregation probabilities $d_{\ell i}$ assign probability 1 to the representative state/midpoint of interval ℓ; this is generic for aggregation with representative states. The aggregation probabilities $\phi_{j\ell}$ for the states $j = 1, \ldots, n$ are proportional to the distance of j from the two interval midpoints that bracket j on the left and the right. Thus when representative states are used in biased aggregation, the corrections applied to the bias function V are linear between the interval midpoints. The algorithmic solution of the aggregate problem is similar to the case of hard aggregation.

We finally note that one may consider more sophisticated selections of the aggregate states. For example one may choose the width of the intervals to be variable, and to be wider or narrower in parts of the state space where J^* is known to vary little or a lot, respectively. For example near the parking garage it makes sense to make the intervals smaller, in order to try to capture the fine structure of J^* (cf. Fig. 4.2). Note also that in the extreme case where $J^* - V$ is constant over each interval, the aggregate problem gives an exactly optimal solution (cf. Prop. 3.4).

Variations: Parking with Memory and a Return Option

The parking problem admits several variations, which can render the exact solution impossible, and the approximate solution quite challenging, while still maintaining the problem's intuitive character. In particular, we may consider a more complicated parking lot topology, the option to continue the search even after reaching the parking garage, and the possibility of previously full or empty parking spaces becoming empty or full, respectively, over time, as the driver is searching.

For an example of such a more complicated problem, consider a variant where the driver upon arrival at the parking garage, can choose between parking at a cost C or restarting the search for a free parking space from space n. This corresponds to a parking lot that is arranged in a one-way loop with the destination's garage (state 0) connecting to space n (the loop is $n \to n-1 \to \cdots \to 0 \to n \to n-1 \to \cdots$), so that the driver can attempt to revisit parking spaces that he has already visited.

We assume that in the first driver's pass, each parking space is free with probability p independently of whether other parking spaces are free or not. In subsequent driver passes, there is conditional probability

v that a space is free given that the space was free in the preceding pass, and $\overline{v}$ given that the space was not free in the preceding pass. The driver is assumed to remember perfectly the location of all the free spaces he has seen in the past.

We can formulate this problem as an SSP problem with imperfect state information. The state here includes the current position i, the free/nonfree status of i, and also the entire driver's memory (because of the Markov nature of the evolution of the status of each parking space, it is sufficient to remember the status of just the preceding n spaces if the driver has traversed more than n spaces). In a belief state reformulation of the problem, a sufficient statistic for the driver's memory is the posterior probability of each of the parking spaces being empty. Even for moderate values of n, the number of states is very large, so the problem cannot be solved exactly by DP. It is thus a good candidate for approximate solution with several of the reinforcement learning approaches, including biased aggregation.

5. CONCLUDING REMARKS

In this paper we have proposed a new aggregation framework, which provides a connection with several successful reinforcement learning approaches, such as rollout algorithms, approximate policy iteration, and other single and multistep lookahead methods. The key is the use of a *bias function*, which biases the values of the aggregate cost function towards their correct levels.

An important issue within our aggregation context is the choice of the bias function V. In this paper, we have paid some attention to the choice $V = J_\mu$ for some base policy μ, which highlighted the connection with rollout and approximate policy iteration algorithms. On the other hand, V can be any reasonable approximation to J^*, however obtained, including through the use of simulation-based approximation in value space, and neural networks or other approximation architectures. Another interesting related issue is the use of multiple bias functions that may be linearly combined with tunable weights to form a single bias function V. Generally, the choice of V, the formation of the corresponding biased aggregation framework, and attendant computational experimentation are subjects that require further research.

In this paper, we have focused on discounted problems, but our approach applies to all the major types of DP problems, including finite horizon, discounted, and stochastic shortest path problems. Of special interest are deterministic discrete-state problems, which arise in combinatorial optimization. For such problems, rollout algorithms have been used with success, and have provided substantial improvements over the heuristics on which they are based. One may try to improve the rollout algorithms for these problems with the use of biased aggregation.

We finally note that aggregation can be implemented in several different contexts, such as multistage or distributed aggregation (see Sections 6.5.3, 6.5.4, and [Ber18b], Section 1.2). The idea of introducing a bias function within these contexts in ways similar to the one of the present paper is possible, and is an

interesting subject for further investigation.

6. REFERENCES

[BBD10] Busoniu, L., Babuska, R., De Schutter, B., and Ernst, D., 2010. Reinforcement Learning and Dynamic Programming Using Function Approximators, CRC Press, N. Y.

[BBS87] Bean, J. C., Birge, J. R., and Smith, R. L., 1987. "Aggregation in Dynamic Programming," Operations Research, Vol. 35, pp. 215-220.

[BeC89] Bertsekas, D. P., and Castanon, D. A., 1989. "Adaptive Aggregation Methods for Infinite Horizon Dynamic Programming," IEEE Trans. on Aut. Control, Vol. AC-34, pp. 589-598.

[BeT91] Bertsekas, D. P., and Tsitsiklis, J. N., 1991. "An Analysis of Stochastic Shortest Path Problems," Math. Operations Research, Vol. 16, pp. 580-595.

[BeT96] Bertsekas, D. P., and Tsitsiklis, J. N., 1996. Neuro-Dynamic Programming, Athena Scientific, Belmont, MA.

[BeY07] Bertsekas, D. P., and Yu, H., 2007. "Solution of Large Systems of Equations Using Approximate Dynamic Programming Methods," Lab. for Information and Decision Systems Report LIDS-P-2754, MIT.

[BeY09] Bertsekas, D. P., and Yu, H., 2009. "Projected Equation Methods for Approximate Solution of Large Linear Systems," J. of Computational and Applied Mathematics, Vol. 227, pp. 27-50.

[Ber12] Bertsekas, D. P., 2012. Dynamic Programming and Optimal Control, Vol. II, 4th edition, Athena Scientific, Belmont, MA.

[Ber17] Bertsekas, D. P., 2017. Dynamic Programming and Optimal Control, Vol. I, 4th edition, Athena Scientific, Belmont, MA.

[Ber18a] Bertsekas, D. P., 2018. "Feature-Based Aggregation and Deep Reinforcement Learning: A Survey and Some New Implementations," Lab. for Information and Decision Systems Report, MIT, April 2018 (revised August 2018); arXiv preprint arXiv:1804.04577; will appear in IEEE/CAA Journal of Automatica Sinica.

[Ber18b] Bertsekas, D. P., 2018. Abstract Dynamic Programming, Athena Scientific, Belmont, MA.

[ChM82] Chatelin, F., and Miranker, W. L., 1982. "Acceleration by Aggregation of Successive Approximation Methods," Linear Algebra and its Applications, Vol. 43, pp. 17-47.

[CiS15] Ciosek, K., and Silver, D., 2015. "Value Iteration with Options and State Aggregation," Report, Centre for Computational Statistics and Machine Learning University College London.

[DeK11] Devlin, S., and Kudenko, D., 2011. "Theoretical Considerations of Potential-Based Reward Shaping for Multi-Agent Systems," In Proceedings of AAMAS.

[Gor95] Gordon, G. J., 1995. "Stable Function Approximation in Dynamic Programming," in Machine Learning: Proceedings of the 12th International Conference, Morgan Kaufmann, San Francisco, CA.

[Gos15] Gosavi, A., 2015. Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning, 2nd Edition, Springer, N. Y.

[Grz17] Grzes, M., 2017. "Reward Shaping in Episodic Reinforcement Learning," in Proc. of the 16th Conference on Autonomous Agents and MultiAgent Systems, pp. 565-573.

[NHR99] Ng, A. Y., Harada, D., and Russell, S. J., 1999. "Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping," in Proc. of the 16th International Conference on Machine Learning, pp. 278-287.

[Pow11] Powell, W. B., 2011. Approximate Dynamic Programming: Solving the Curses of Dimensionality, 2nd Edition, J. Wiley and Sons, Hoboken, N. J.

[RPW91] Rogers, D. F., Plante, R. D., Wong, R. T., and Evans, J. R., 1991. "Aggregation and Disaggregation Techniques and Methodology in Optimization," Operations Research, Vol. 39, pp. 553-582.

[SJJ95] Singh, S. P., Jaakkola, T., and Jordan, M. I., 1995. "Reinforcement Learning with Soft State Aggregation," in Advances in Neural Information Processing Systems 7, MIT Press, Cambridge, MA.

[SSP18] Serban, I. V., Sankar, C., Pieper, M., Pineau, J., Bengio, J., 2018. "The Bottleneck Simulator: A Model-Based Deep Reinforcement Learning Approach," arXiv preprint arXiv:1807.04723.v1.

[SuB98] Sutton, R. S., and Barto, A. G., 1998. Reinforcement Learning, MIT Press, Cambridge, MA. (A draft 2nd edition is available on-line.)

[Sze10] Szepesvari, C., 2010. Algorithms for Reinforcement Learning, Morgan and Claypool Publishers, San Franscisco, CA.

[TsV96] Tsitsiklis, J. N., and Van Roy, B., 1996. "Feature-Based Methods for Large-Scale Dynamic Programming," Machine Learning, Vol. 22, pp. 59-94.

[VDR79] Vakhutinsky, I. Y., Dudkin, L. M., and Ryvkin, A. A., 1979. "Iterative Aggregation - A New Approach to the Solution of Large Scale Problems," Econometrica, Vol. 47, pp. 821-841.

[Van06] Van Roy, B., 2006. "Performance Loss Bounds for Approximate Value Iteration with State Aggregation," Mathematics of Operations Research, Vol. 31, pp. 234-244.

[WaB13a] Wang, M., and Bertsekas, D. P., 2013. "Stabilization of Stochastic Iterative Methods for Singular and Nearly Singular Linear Systems," Mathematics of Operations Research, Vol. 39, pp. 1-30.

[WaB13b] Wang, M., and Bertsekas, D. P., 2013. "Convergence of Iterative Simulation-Based Methods for Singular Linear Systems," Stochastic Systems, Vol. 3, pp. 39-96.

[Wie03] Wiewiora, E., 2003. Potential-Based Shaping and Q-Value Initialization are Equivalent," J. of Arti-

ficial Intelligence Research, Vol. 19, pp. 205-208.

[YuB04] Yu, H., and Bertsekas, D. P., 2004. "Discretized Approximations for POMDP with Average Cost," Proc. of the 20th Conference on Uncertainty in Artificial Intelligence, Banff, Canada.