Hindawi Publishing Corporation
Advances in Operations Research
Volume 2010, Article ID 659432, 19 pages
doi:10.1155/2010/659432

Review Article

Exact Decomposition Approaches for Markov Decision Processes: A Survey

Cherki Daoui,[1] Mohamed Abbad,[2] and Mohamed Tkiouat[3]

[1] *Département de Mathématiques, Faculté des Sciences et Techniques, P.O. Box 523, Béni-Mellal 23000, Morocco*
[2] *Département de Mathématiques, Faculté des Sciences, P.O. Box 1014, Rabat 10000, Morocco*
[3] *Département Génie Industriel, Ecole Mohammedia d'Ingénieurs,P.O. Box 765, Rabat 10000, Morocco*

Correspondence should be addressed to Cherki Daoui, daouic@yahoo.com

Received 27 August 2009; Revised 26 March 2010; Accepted 24 May 2010

Academic Editor: Imed Kacem

As classical methods are intractable for solving Markov decision processes (MDPs) requiring a large state space, decomposition and aggregation techniques are very useful to cope with large problems. These techniques are in general a special case of the classic Divide-and-Conquer framework to split a large, unwieldy problem into smaller components and solving the parts in order to construct the global solution. This paper reviews most of decomposition approaches encountered in the associated literature over the past two decades, weighing their pros and cons. We consider several categories of MDPs (average, discounted, and weighted MDPs), and we present briefly a variety of methodologies to find or approximate optimal strategies.

1. Introduction

This survey focuses on decomposition techniques for solving large MDPs. However, in this section we begin by discussing briefly some approaches to treat large Markov chain models because they contain basic ideas of methods to tackle large MDPs.

1.1. Large Markov Chain Models

Markov chain models are still the most general class of analytic models widely used for performance and dependability analysis. This class of problems is robust in terms of its ability to represent a very broad class of models of interest. Unfortunately, the characteristics of many models (complex interactions between components, sophisticated scheduling

strategies, synchronization among parallel tasks, etc.) preclude the possibility of closed-form solutions, in general. Numerical solution methods are the most general solution methods [1]. The most pervasive practical limitation to the use of numerical solution techniques is the size of the state space of realistic models. One natural way to deal with this problem is via state space reduction techniques, that is, a transformation of the model to one with fewer states. Another is to modify the model to one with a structure which enables efficient solution. The transformed model in general is an approximation of the original model. Several classes of such transformations are described below.

There are several methods of state space reduction that are applicable to Markov chain models. One such method is combining "lumpable states" [2]. When lumping applies, it is exact, but only partial information is retrievable from the solution except in very special circumstances. Also, lumping is only valid for models that have certain very limited types of structure.

Another method to treat large state spaces is aggregation/disaggregation [3]. This applies particularly well to cases in which the system model can be viewed as an interacting set of tightly coupled subsystems. The solution method is generally an iterative one in which submodels are solved and the obtained results are used to adjust the submodels repetitively until a convergence criterion is met. This is an efficient procedure if (a) the model is decomposable into tightly coupled submodels, (b) the state space of each submodel is practically solvable, and (c) the number of such submodels is not too large.

There is an additional property of many models that is not capitalized on by any of the above methods. In many important cases, although the state space is extremely large, the stationary state probability distribution is highly skewed, that is, only a small subset of the states account for the vast majority of the probability mass. This is easily illustrated by considering the nature of several modeling application areas. This observation indicates that most of the probability mass is concentrated in a relatively small number of states in comparison to the total number of states in the model. During its lifetime, the system spends most of its time in this relatively small subset of states and only rarely reaches other states. The above observation is used to motivate truncation of the model state space, that is, choose to only represent a subset of the more "popular" states and ignore the remainder, deleting transitions from the represented states to the "discarded states". Some form of heuristic is often used to decide which states to retain and which to discard. For dependability models a simple heuristic might be to discard all states in which the system has more than a certain number of failed components. A more sophisticated heuristic for determining which states to retain is described in [4]. The most persistent practical constraint to the use of the above state space truncation is that in certain models (health management, reliability, security problems) deleting some infrequently states may cause significant fatalities.

There is also the issue of how much error is introduced by state space truncation. For some transient measures, error bounds are easily obtained by introducing trap states [5]. For example, in dependability analysis we may be interested in interval reliability. For this purpose one can introduce a trap state and change all transitions to "discarded states" into transitions to the trap state. Then the importance of the error is a direct function of the probability that the system is in the trap state at the end of the interval. For-steady state measures, the issue of error bounds can be more difficult. There are a number of approaches available for computing bounds or for proving that one model provides a bound compared with some other model. Among these we can quote the method of Courtois and Semal [6, 7], the methods based on sample-path analysis and stochastic ordering [8, 9], and the bias terms approach of Van Dijk [10].

In [11], the authors explain with some examples where the model modifications only affect the less popular states and therefore have less effect on the performance measures. They also present some methods to compute bounds on performance measures.

1.2. Large Markov Decision Models

Over the past five decades there has been a lot of interest within planning community in using stochastic models, which provide a good framework to cope with uncertainties. Among them, Markov decision processes [12–22], either fully observable (MDP or FOMDP) or partially observable (POMDP), have been the subject of several recent studies [23–30]. The optimal strategy is computed with respect to all the uncertainties, but unfortunately, classical algorithms used to build these strategies are intractable with the twin drawbacks of large environments and lack of model information in most real-world systems [31–33]. Several recent studies aim to obtain a good approximation of the optimal strategy. Among them, aggregation and decomposition techniques have been the subject of a lot of attention. These techniques are really different flavors of the well-known framework Divide-and-Conquer: (1) partitioning the state space on regions, that is, transforming the initial MDP into small local MDPs, (2) independently solving the local MDPs, and (3) combining the local solutions to obtain an optimal (or near-optimal) strategy for the global MDP.

In preparing this survey, we first attempt to summarize some general decomposition approaches introduced in [34–38]. These works have proposed some algorithms to compute optimal strategies as opposed to nearly all methods to treat large MDPs that compute only near optimal strategies. Moreover, they are suitable to several categories of MDPs (average, discounted and weighted MDPs) and can be applicable to many practical planning problems. These algorithms are based on the graph associated to the original MDP and by introducing some hierarchical structure of this graph.

A lot of other recent works are concerned with autonomous exploration systems which require planning under uncertainty. In [39], the authors present a main decomposition technique to cope with large state spaces for practical problems encountered in autonomous exploration systems. They assume that an expert has given a partition of the state space into regions; a strategy is independently computed for each region. These local solutions are pieced together to obtain a global solution. Subproblems correspond to local MDPs over regions associated with a certain parameter that provides an abstract summary of the interactions among regions. Next, two algorithms are presented for combining solutions to subproblems. The first one, called Hierarchical Policy Construction, solves an abstract MDP (each region becomes an abstract state). This algorithm finds for each region an optimal local strategy, that is, an optimal region to reach. The second algorithm, called Iterative Improvement Approach, iteratively approximates parameters of the local problems to converge to an optimal solution. In [40], the approach is analogous to the latter one but aims to show how domain characteristics are exploited to define the way regions communicate with each other. The authors represent the MDP as a directed graph which allows to precisely define the cost of going from one region to another. The approach in [41] is quite different. The authors present a particular structured representation of MDPs, using probabilistic STRIPS rules. This representation allows to quickly generate clusters of states which are used as states of an abstract MDP. The solution to this abstract MDP can be used as an approximate solution for the global MDP.

Other authors aim to use macro-actions to efficiently solve large MDPs [42, 43]. Given a partition of the state space, a set of strategies are computed for each region. A strategy

in a region is called a macro-action. Once the transition and the reward functions have been transformed to cope with macro-actions, an abstract MDP is solved using these macro-actions. The problem with this approach is that the quality of the solution is highly dependent on the quality of the set of strategies computed for each region. To ensure high quality, a large set of macro-actions have to be computed. As the transition and reward functions have to be computed for each macro-action, the time needed to find macro-actions can outweigh the speedup provided by macro-actions during strategy computing. The approach in [44] is quite similar, but based on Reinforcement Learning (RL).

It is important to note that the above approaches differ essentially for the following reasons: the choice of the regions, the manner to combine the local solutions, and the quality of the final solution (optimal or only near optimal). For instance, in [35, 36] the authors use graph theory and choose the communicating classes as regions whereas in [39, 40, 45, 46] they use algorithmic geometry of field for determining the regions.There is also the remark that most of these state space decomposition methods are valid if (a) each subproblem is practically solvable, (b) the number of such subproblems is not too large, and (c) combining such subproblems is not difficult.

On the other hand, most research in Reinforcement Learning (RL) is based on the theoretical discrete-time state and action formalism of the MDP. Unfortunately, as stated before, it suffers from the curse of dimensionality [47] where the explicit state and action space enumeration grow exponentially with the number of state variables and the number of agents, respectively. To deal with this issue, RL introduces Monte Carlo methods, stochastic approximation, trajectory sampling, temporal difference backups, and function approximation. However, even these methods have reached their limits. As a result, we discuss briefly in this survey broad categorizations of factored and hierarchical approaches which break up a large problem into smaller components and solve the parts [48–58]. The main idea of these approaches is to leverage the structure present in most real-world domains. For instance, the state of the environment is much better described in terms of the values of the states variables than by a monolithic number. This fact leads to powerful concepts of state aggregation and abstraction.

In order to solve linear programs of very large sizes, decomposition principles [59] that divide a large linear program into many correlated linear programs of smaller sizes have been well studied, among which the Dantzig-Wolfe decomposition [60] may be the most well known. Further, in view of [61] the MDPs can be solved as linear programs using Dantzig-Wolfe decomposition. Thus, in this paper we present briefly the classical Dantzig-Wolfe decomposition procedure.

The paper is organized as follows Section 2 presents the problem formulation. Section 3 treats extensively some decomposition approaches introduced in [34–36, 38]. Section 4 presents the decomposition technique proposed by Dean and Lin [39]. Section 5 reviews some Reinforcement Learning methods proposed in the associated literature to alleviate the curse of dimensionality. Section 6 describes briefly the classic Dantzig-Wolfe decomposition procedure. We conclude and discuss open issues in Section 7.

2. Markov Decision Processes

We consider a stochastic dynamic system which is observed at discrete-time points $t = 1, 2, \ldots$. At each time point t, the state space of the system is denoted by X_t, where X_t is a random variable whose values are in a state space E. At each time point t, if the system is in state i, an action $a \in A(i) = \{1, 2, \ldots, m(i)\}$ has to be chosen. In this case, two things happen:

a reward $r(i,a)$ is earned immediately, and the system moves to a new state j according to the transition probability p_{iaj}. Let A_t be the random variable which represents the action chosen at time t. We denote by $H_t = (E \times A)^{t-1} \times E$ the set of all histories up to time t and by $\Psi = \{(q_1, q_2, \ldots, q_{|A|}) : \sum_{a=1}^{|A|} q_a = 1, q_a \geq 0, 1 \leq a \leq |A|\}$ the set of probability distributions over $A = \bigcup_{i \in E} A(i)$.

A strategy π is defined by a sequence $\pi = (\pi^1, \pi^2, \ldots)$, where π^t: $H_t \rightarrow \Psi$ is a decision rule. A Markov strategy is one in which π^t depends only on the current state at time t. A stationary strategy is a Markov strategy with identical decision rules. A deterministic (or pure) strategy is a stationary strategy whose single decision rule is nonrandomized. An ultimately deterministic strategy is a Markov strategy $\pi = (\pi^1, \pi^2, \ldots)$ such that there exist a deterministic strategy g and an integer t_0 such that $\pi^t = g$ for all $t \geq t_0$. Let F, F_M, F_S, F_D, and F_{UD} be the sets of all strategies: Markov strategies, stationary strategies, deterministic strategies, and ultimately deterministic strategies, respectively.

Let $P_\pi(X_t = j, A_t = a \mid X_1 = i)$ be the conditional probability that at time t the system is in state j and the action taken is a, given that the initial state is i and the decision maker uses a strategy π. Now, if R_t denotes the reward earned at time t, then, for any strategy π and initial state i, the expectation of R_t is given by $E_\pi(R_t, i) = \sum_{j \in E} \sum_{a \in A(j)} P_\pi(X_t = j, A_t = a \mid X_1 = i) r(j,a)$.

The manner in which the resulting stream of expected rewards $\{E_\pi(R_t, i) : t = 1, 2, \ldots\}$ is aggregated defines the MDPs discussed in the sequel.

In the *discounted reward MDP* , the corresponding overall reward criterion is defined by $V_i^\alpha(\pi) = \sum_{t=1}^{\infty} \alpha^{t-1} E_\pi(R_t, i), i \in E$, where $\alpha \in [0,1)$ is a fixed discount rate. A strategy f^* is called discounted optimal if, for all $i \in E, V_i^\alpha(f^*) = \max_{\pi \in F} V_i^\alpha(\pi) = V^\alpha(i)$. We will denote this MDP by $\Gamma(\alpha)$.

In the *average reward MDP*, the overall reward criterion is defined by $\Phi_i(\pi) = \lim_{T \rightarrow \infty} \inf (1/T) \sum_{t=1}^{T} E_\pi(R_t, i), i \in E$. A strategy f^* is called average optimal if for all $i \in E, \Phi_i(f^*) = \max_{\pi \in F} \Phi_i(\pi) = V(i)$. We will denote this MDP by Γ.

In the *weighted reward MDP*, the overall reward criterion is defined by $\omega_i(\pi) = \lambda(1 - \alpha) V_i^\alpha(\pi) + (1 - \lambda)\Phi_i(\pi), i \in E$, where $\lambda \in [0,1]$ is a fixed weight parameter, and α is the discount rate in the MDP $\Gamma(\alpha)$. We denote this MDP by $\Gamma(\alpha, \lambda)$. A strategy f^* is called optimal if for all $i \in E, \omega_i(f^*) = \max_{\pi \in F} \omega_i(\pi)$. Let $\epsilon > 0$; for any $i \in E$, some strategy f^* is called ϵ-i-optimal if $\omega_i(f^*) \geq \sup_{\pi \in F} \omega_i(\pi) - \epsilon$. A strategy f^* is called ϵ-optimal if f^* is ϵ-i-optimal for all $i \in E$.

Remark 2.1. (i) It is well known that each of the two first above problems possesses an optimal pure strategy, and there are a number of finite algorithms for its computation (e.g., see [18, 62–64]). (ii) In [65], the authors consider weighted MDPs and show that optimal strategies may not exist and propose an algorithm to determine an ϵ-optimal strategy.

3. Some Decomposition Techniques for MDPs

Classical algorithms for solving MDPs cannot cope with the size of the state spaces for typical problems encountered in practice [31–33]. In this section, we will summarize two decomposition techniques for tackling this complexity: the first is proposed by Ross and Varadarajan [38] and the second is introduced by Abbad and Boustique [36]. These techniques propose some algorithms to compute optimal strategies for several categories of MDPs (average, discounted, and weighted MDPs), as opposed to most solutions for large

MDPs that compute only near-optimal strategies and are suitable to only some types of planning problems. Also, we will present some related works.

3.1. Ross-Varadarajan Decomposition

Considered are discrete-time MDPs with finite state and action spaces under the average reward optimality criterion [38]. We begin by introducing some notions of communication for MDPs.

Definition 3.1. A set of states I communicate if, for any two states $x, y \in I$ with $x \neq y$, there exists a pure strategy g such that y is accessible from x under $P(g)$.An MDP is said to be communicating if the state space E communicates.

Definition 3.2. A set of states I strongly communicate if there exists a stationary strategy f such that I is a subset of a recurrent class associated with $P(f)$.

Definition 3.3. A set of states C are a communicating class (a strongly communicating class) if (i) C communicates (strongly communicates), and (ii) if $x \in C$ and $y \notin C$, then $\{x, y\}$ does not communicate (strongly communicate).

Ross and Varadarajan [38] show that there is a unique natural partition of state space E into strongly communicating classes (SCC): $C_1, C_2, \ldots, C_p$ and a set T of states that are transient under any stationary strategy. This decomposition is inspired from Bather's decomposition algorithm [37]. The sets E_i in the Bather decomposition are the strongly communicating classes C_i and the set of transient states T is the union of the sets $T_1, T_2, \ldots, T_m$ in [37]. This decomposition is also later formalized by Kallenberg in [66, Algorithm 7] which studied irreducibility, communicating, weakly communicating, and unichain classification problems for MDPs. A polynomial algorithm is given to compute this partition. Further, they propose an algorithm to solve large MDPs composed of the following steps: (i) solving some small MDPs restricted to each SCC, (ii) aggregating each SCC into one state and finding an optimal strategy for the corresponding aggregated MDP, and (iii) combining solutions found in the latter steps to obtain a solution to the entire MDP.

Here, we will define correctly the latter two types of MDPs.

The Restricted MDPs

For each $i = 1, 2, \ldots, p$, the restricted MDP to the class C_i is denoted by Γ_i, and is defined by: the state space C_i. For each $x \in C_i$, the action space is $F_x = \{a \in A(x) : p_{xay} = 0 \text{ for all } y \notin C_i\}$; the transition probabilities and the rewards are still analogous to those in the original problem; however, they are restricted to the state space C_i and the action spaces $F_x, x \in C_i$.

The Aggregated MDP

The aggregated MDP is defined by:

(i) the state space $\overline{E} = \{1, \ldots, p, p+1, \ldots, p+t\}$, where $t = |T|$,

(ii) the action spaces $\overline{A}(i) = A(i)$ if $i \in \{p+1, \ldots, p+t\}$ and $\overline{A}(i) = \{\theta\} \cup \{(x, a) : x \in C_i, a \notin F_x\}$ if $i \in \{1, \ldots, p\}$,

(iii) the transition probabilities

$$\begin{aligned}
\overline{p}_{i\theta i} &= 1 \quad \text{if } i \in \{1,\dots,p\},\\
\overline{p}_{i(x,a)j} &= \sum_{y\in C_j} p_{xay} \quad \text{if } i,\ j \in \{1,\dots,p\},\\
\overline{p}_{iaj} &= \sum_{y\in C_j} p_{iay} \quad \text{if } i \in \{p+1,\dots,p+t\}, j \in \{1,\dots,p\},\ a \in A(i),\\
\overline{p}_{iaj} &= p_{iaj} \quad \text{if } i,\ j \in \{p+1,\dots,p+t\},
\end{aligned} \tag{3.1}$$

(iv) the rewards $\overline{r}(i,a) = r(i,a)$ if $i \in \{p+1,\dots,p+t\}$ and $a \in A(i)$, $\overline{r}(i,\theta) = \gamma_i$ if $i \in \{1,\dots,p\}$, $\overline{r}(i,(x,a)) = r(x,a)$ if $i \in \{1,\dots,p\}$.

Remark 3.4. For each $i = 1,\dots,p$, the restricted MDP Γ_i is communicating, and then it can be solved by the simpler linear programming in [51]. As a consequence, all the states in Γ_i have a similar optimal value γ_i.

The most important step in the algorithm above consists in solving the aggregated MDP which is also an MDP, and then it can be solved by using the classical algorithms [34]. Ross and Varadarajan [38] did not give any new method for solving the aggregated MDP. As a result, the aim of [34] is to provide some algorithms which exploit the particular structure of the aggregated MDP and improve the classical ones. The authors consider deterministic MDPs and aggregated MDPs without cycles. In the sequel of this subsection, we will present briefly these algorithms.

3.1.1. Deterministic MDPs

The work in [34] considers firstly deterministic MDPs and shows that the singletons $\{i\}, i \in \{1,2,\dots,p\}$, are the only strongly communicating classes for the aggregated MDP. This result permits to prove the correctness of the following simple algorithm which constructs $\overline{g}$: an optimal strategy in the aggregated MDP.

Algorithm 3.5.

Step 1. One has $\overline{E} := \{1,\dots,p,p+1,\dots,p+t\}$, $v(i) := \gamma_i$ for $i \in \{1,\dots,p\}$ and $v(i) := (\min_{k\in\{1,\dots,p\}}\gamma_k) - 1$ for $i \in \{p+1,\dots,p+t\}$.

Step 2. Compute i^* such that $\gamma_{i^*} = (\max_{j\in\{1,\dots,p\}}\gamma_j)$ and determine $W = \{i \in \{1,\dots,p\} : \gamma_i = \gamma_{i^*}\}$. For $j \in W$, set $\overline{g}(j) := \theta$ and $v(j) = \gamma_{i^*}$.

Step 3. While $i \in \overline{E} - W$, $\overline{a}_i \in \overline{A}(i)$, $j \in W$: $\overline{p}_{i\overline{a}_i j} > 0$ do $\overline{g}(i) = \overline{a}_i$, $W := W \cup \{i\}$, $v(i) := \gamma_{i^*}$.

Step 4. (i) If $\overline{E} = W$, stop: $\overline{g}$ is an aggregated optimal strategy. (ii) If $\overline{E} \neq W$, find i' such that $v(i') = \max_{k\in\overline{E}-W} v(k)$ and determine $W = \{i \in \{1,\dots,p\} : v(i) = v(i')\}$. For $j \in W$, set $\overline{g}(j) := \theta$, $v(j) := v(i')$, and $\overline{E} := \overline{E} - W$.

3.1.2. The Aggregated MDP and Cycles

Let $\overline{G} = (\overline{V}, \overline{S})$ be the graph associated with the aggregated MDP; that is, the state space represents the set of nodes and $\overline{S} := \{(i,j) \in \overline{E} \times \overline{E} : \overline{p}_{i\overline{a}j} > 0 \text{ for some } \overline{a} \in \overline{A}(i)\}$ is the set of arcs. We say that an aggregated MDP has no cycle if the associated graph has no cycle containing two or more nodes. The next algorithm extends Algorithm 3.5 in the case where the original MDP may be not deterministic and its aggregated MDP has no cycle [34].

Algorithm 3.6.

Step 1. One has $\overline{E} := \{1,\ldots,p,p+1,\ldots,p+t\}$; $v(i) := \gamma_i$ for $i \in \{1,\ldots,p\}$; $v(i) := (\min_{k\in\{1,\ldots,p\}}\gamma_k) - 1$ for $i \in \{p+1,\ldots,p+t\}$.

Step 2. Compute $\gamma^* = \max_{k\in\{1,\ldots,p\}}\gamma_k$ and determine $W = \{i \in \{1,\ldots,p\} : \gamma_i = \gamma^*\}$. For $j \in W$, set $\overline{g}(j) := \theta$.

Step 3. While there exist $i \in \overline{E} - W$, $\overline{a}_i \in \overline{A}(i)$ such that $\sum_{j\in W}\overline{p}_{i\overline{a}_i j} = 1$, do: $\overline{g}(i) = \overline{a}_i$, $W := W \cup \{i\}$, $v(i) := \gamma^*$.

Step 4. While there exists $i \in \{1,\ldots,p\} - W$ such that $\overline{p}_{i\overline{a}j} = 0$ for all $\overline{a} \in \overline{A}(i) - \theta$, $j \neq i$, do: $\overline{g}(i) := \theta$, $W := W \cup \{i\}$.

Step 5. If $\overline{E} \neq W$, while there exists $i \in \overline{E} - W$ such that, for all $\overline{a} \in \overline{A}(i)$, $\sum_{j\in W}\overline{p}_{i\overline{a}j} = 1$, do: $w(i) := \max_{\overline{a}\in\overline{A}(i)}\sum_{j\in W}\overline{p}_{i\overline{a}j}v(j)$; if $v(i) > w(i)$, set $\overline{g}(i) := \theta$; if $v(i) \leq w(i)$ set $\overline{g}(i) := \operatorname{argmax}_{\overline{a}\in\overline{A}(i)}\sum_{j\in W}\overline{p}_{i\overline{a}j}v(j)$ and $v(i) := w(i)$; $W := W \cup \{i\}$. If $\overline{E} = W$, stop; $\overline{g}$ is an aggregated optimal strategy.

Remark 3.7. (i) The previous algorithm is applicable if the decomposition leads to $T = \emptyset$, because in this case the aggregated MDP has no cycle. (ii) In [1], the authors have also considered an arbitrary MDP without any condition and they have presented two algorithms for the computation of an aggregated optimal strategy. The latter comes up with some significant simplifications on the classical policy improvement algorithm and linear programming algorithm. In the construction they have exploited the fact that the recurrent classes in the aggregated MDP are singletons.

3.2. Abbad-Boustique Decomposition

Abbad and Boustique [36] propose an algorithm to compute an average optimal strategy which is based on the graph associated to the original MDP introducing some hierarchical structure of this graph. The main contribution of their approach consists in constructing by induction the levels of the graph G and solving the restricted MDPs corresponding to each level. The local solutions of the latter MDPs provide immediately an optimal strategy in the entire MDP. This state space decomposition into levels is inspired by the work in[67].

Let $G = (V, S)$ be a directed graph associated to the original MDP. A communicating class for the MDP corresponds to a connected component in the graph G. Thus, there exists a unique partition of the state space E into communicating classes $C_1, C_2, \ldots, C_q$, which can be determined via standard depth-first algorithms [68]. The level L_0 is formed by all classes C_i such that C_i is closed. The nth level L_n is formed by all classes C_i such that the end of any arc emanating from C_i is in some level $L_{n-1}, L_{n-2}, \ldots, L_0$.

Let (C_{lk}), $k \in \{1,2,\ldots,K(l)\}$, be the classes corresponding to the nodes in level l. The restricted MDPs corresponding to each level L_n, $n = 0,1,2,\ldots,L$, are constructed, by induction, as follows.

3.2.1. Construction of the Restricted MDPs in Level L_0

For each $k = 1,2,\ldots,K(0)$, we denote by Γ_{0k} the restricted MDP corresponding to the class C_{0k}, that is, the restricted MDP in which the state space is $S_{0k} = C_{0k}$. Note that any restricted MDP, Γ_{0k} is well defined since any class C_{0k} is closed and can be solved easily by a finite algorithm (see [69]).

3.2.2. Construction of the Restricted MDPs in Level L_n, $n \geq 1$

Let $E_n = \cup\{C_{mk}, m = 0,\ldots,n-1; k = 1,\ldots,K(m)\}$. Let $T_{mk}(i)$ be the optimal value in state $i \in E_n$, computed in the previous MDP: $\Gamma_{mk}(m < n)$. For each $k = 1,2,\ldots,K(n)$, we denote by Γ_{nk} the MDP defined by the following.

(i) *State space*. $S_{nk} = C_{nk} \cup \{j \in E_n : p_{iaj} > 0 \text{ for some } i \in C_{nk}, a \in A(i)\}$.

(ii) *Action spaces*. For each $i \in S_{nk}$, the associated action space is $A_{nk}(i) = A(i)$ if $i \in C_{nk}$ and $A_{nk}(i) = \{\theta\}$ if $i \notin C_{nk}$.

(iii) *Transition probabilities*. For each $i, j \in S_{nk}$, $P_{nk}(j \mid i, a) = p_{iaj}$ if $i \in C_{nk}$, $a \in A(i)$ and $P_{nk}(j \mid i, a) = 1$ if $i = j$, $i \notin C_{nk}$.

(iv) *Rewards*. Let $i \in S_{nk}$; if $i \in C_{nk}$, then $r_{nk}(i,a) := r(i,a)$.

If $i \notin C_{nk}$, then there exist $m \in \{0,1,\ldots,n-1\}$, there exist $h \in \{1,2,\ldots,K(m)\} : i \in C_{mh}$, and $r_{nk}(i,\theta) := T_{mh}(i)$.

The basic result of Abbad-Boustique approach shows that the optimal value for a fixed state in any restricted MDP is equal to the optimal value in the original MDP. Consequently, optimal actions in the restricted MDPs are still optimal in the original MDP. Such approach is advantageous because it allows that an optimal action and the optimal value for a fixed state can be computed only through some restricted MDPs before solving the entire MDP; however, there is still considerable overhead in determining the communicating classes.

The work in [35] is a main related work on Abbad-Boustique decomposition. The authors have considered the discounted and weighted optimality criterion with finite state and action spaces. With these criterions the Ross-Varadarajan decomposition is not available. That is why the authors have used the approach introduced in [36], and they have constructed the levels and the restricted MDPs in a similar way as above. In the discounted optimality criterion, they also have showed that optimal actions in the restricted MDP are still optimal in the original MDP, and they have proposed an algorithm to find an optimal strategy. Under the weighted optimality criterion, they have first proposed an algorithm which constructs an ϵ-optimal strategy corresponding to each restricted MDP; whereas, in [65], for each state $i \in E$, an ϵ-i- optimal strategy is constructed by solving the entire MDP. Finally, by coalescing the last restricted ϵ-optimal strategies they have presented an algorithm which determines an ϵ-optimal strategy in the original MDP.

Remark 3.8. (i) The decomposition approaches presented in [34, 37, 57, 66] are merely motivating if (a) the cardinalities of the Strongly Communicating Classes (SCCs): $C_1, C_2, \ldots, C_p$

and a set T are small compared to the cardinality of the state space E and (b) the number p of the SCCs is not too large. However, they are most suitable to solve constrained MDPs. (ii) the approaches proposed in [35, 36] alleviate the last inconvenient; however, there is still considerable overhead in determining the communicating classes.

4. Dean-Lin Decomposition

the study by Dean and Lin in [39] is one of the first works that introduces decomposition techniques for planning in stochastic domains. Their framework as stated before is also a special case of Divide-and-Conquer: given a partition of the state space into regions, (i) reformulate the problem in terms of smaller MDPs over the subspaces of the individual regions, (ii) solve each of these subproblems, and then (iii) combine the solutions to obtain a solution to the original problem. In this section, we discuss briefly Dean-Lin approach and some related works.

Let P be any partition of E, $P = \{R_1, \ldots, R_m\}$ such that $E = \cup_{i=1}^{m} R_i$ and $R_i \cap R_j = \emptyset$, for all $i \neq j$. We refer to a region $R \in P$ as an aggregate (or a macro) state. The periphery of an aggregate state R (denoted Periphery(R)) is the set of states not in R but reachable in a single transition from some state in R, that is, $\{\ j \mid j \notin R$ and there exist $i \in R, a \in A(i), p_{iaj} > 0\}$.

To model interactions among regions, a set of parameters are introduced. Let $U = \cup_{R \in P}$ Periphery(R), and λ_i for each $i \in U$ denote a real-valued parameter. Let $\overline{\lambda} \in \Re^{|U|}$ denote a vector of all such λ_i parameters, and let $\overline{\lambda}|_R$ denote a subvector of $\overline{\lambda}$ composed of λ_i, where i is in Periphery(R). Parameter λ_i serves as a measure of the expected cost of starting from a periphery state, and $\overline{\lambda}|_R$ provides an abstract summary of how the other regions affect R. Given a particular $\overline{\lambda}$, the original MDP is decomposed into smaller MDPs. For a region R and the subvector $\overline{\lambda}|_R$, a local MDP $M_{\overline{\lambda}|_R}$ is defined by the following:

(i) state space $R \cup$ Periphery(R),

(ii) state transition matrix (q_{ij}): $q_{ij} = p_{ij}$ for $i \in R$, and $q_{ii} = 1$ for $i \in$ Periphery(R),

(iii) cost matrix (k_{ij}): $k_{ij} = c_{ij}$ for $i, j \in R$; $k_{ij} = \lambda_j + c_{ij}$ for $i \in R$ and $j \in$ Periphery(R); $k_{ii} = 0$ for $i \in$ Periphery(R).

Let π^* denote an optimal strategy for the original MDP. If $\lambda_i = V_i^{\alpha}(\pi^*)$, the authors show that the resulting local strategies for the local MDPs define an optimal strategy on the entire state space. They further propose two methods for either guessing or successively approximating $V_i^{\alpha}(\pi^*)$ for all $i \in U$: a hierarchical construction approach and iterative improvement approach.

The former constructs an abstract MDP by considering individual regions as abstract states and their local strategies as abstract actions. The solution to this abstract MDP finds for each region an optimal region to reach, thus yielding a solution on the original MDP. Unfortunately, this approach does not guarantee to produce an optimal strategy; however, it has an intuitive interpretation that makes it particularly suitable for robot navigation domains.

The second method iteratively approximates $V_i^{\alpha}(\pi^*)$ to converge to an optimal solution. On each iteration, for each region R a specific estimate of the parameter values of region R is considered and the resulting MDP to obtain a local strategy is solved. By examining the resulting local strategies we get information to engender a new estimate of the parameter values that is guaranteed to improve the global solution. This information about local strategies also tells us when the current solution is optimal or within some specified

tolerance and it is therefore appropriate to terminate the iterative procedure. The iterative approach computes several strategies for each region, which is not good news, but it provides an optimal strategy. Also, it is important to note that this approach is based on a reduction to the methods of Kushner and Chen [61] that demonstrates how to solve MDPs as linear programs using Dantzig-Wolfe decomposition [60]. For more details, we refer the reader to the longer version of the paper in [70].

Closely related to hierarchical construction approach, two methods have been proposed in [42, 43]. They also solve an abstract MDP composed of one abstract state per region, but many strategies, called macro-actions, are computed in each region. In [43], to compensate this weakness only a small set of strategies are calculated per region, without loss of optimality.

In [40], the approach aims also to solve weakly coupled MDPs, but it is quite different to Dean-Lin approach for the following reasons: (i) only one strategy is computed in each region which reduces time consuming on each iteration and (ii) the MDP is represented as a directed graph, and so a simple heuristic valuated graph is used to estimate periphery state values. This approach constructs strategies that are only near optimal; however, they are computed quickly.

The related work on abstraction and decomposition is extensive. In the area planning and search assuming deterministic action models, there is the work on macro-operators [71] and hierarchies of state space operators [72, 73]. Closely related is the work on decomposing discrete-event systems modeled as (deterministic) finite state machines [74]. In the area of reinforcement learning, there is work on deterministic action models and continuous state spaces [75] and stochastic models and discrete state spaces [76]. Finally, the approach described in [46] represents a special case of the Dean-Lin framework, in which the partition consists of singleton sets for all of the states in the envelope and a set for all the states in the complement of the envelope.

5. Hierarchical Reinforcement Learning

Reinforcement Learning (RL) is a machine learning paradigm in which an agent learns a behavioral policy through direct interaction with an unknown, stochastic environment [77]. Most research in RL is based on the theoretical discrete-time state and action formalism of the MDP. Unfortunately, it suffers from the curse of dimensionality [47], where the explicit state and action space enumeration grow exponentially with the number of state variables and the number of agents, respectively. So, RL has introduced Monte Carlo methods, stochastic approximation, trajectory sampling, temporal difference backups, and function approximation. However, even these methods have reached their limits. As a result, we discuss briefly in this section broad categorizations of factored, and hierarchical approaches which break up a large problem into smaller components and solving the parts.

5.1. Factored Approaches

When the state space of the MDP can be specified as a cross-product of sets of state variables E_i ($E = E_0 \times E_1 \times \cdots \times E_n$), it is called a factored MDP (FMDP). The concepts of state abstraction and aggregation are strongly related to the idea of a factored state space. A factored formulation also allows for system dynamics to be specified using a more natural and intuitive representation instead of an $S \times S$ probability matrix per action. Representations that can describe such structure are 2-slice Dynamic Bayesian Networks (DBNs) [78] and probabilistic STRIPS operators, the former being more popular in the literature.

In [48], the authors exploit such a factored state space directly, and reveal reduction in the computation and memory required to compute the optimal solution. The assumption is that the MDP is specified by a set of DBNs, one for each action, although the claim made is that it is amenable to a probabilistic STRIPS specification too. In addition to using the network structure to elicit variable independence, they use decision-tree representations of the conditional probability distributions (CPDs) to further exploit propositional independence. Next, they construct the structured policy iteration (SPI) algorithm which aggregates states for two distinct reasons: either if the states are assigned the same action by the current strategy, or if states have the same current estimated value. With the aggregation in place, the learning algorithm (based on modified policy iteration) only computes at the coarser level of these state partitions instead of that of the individual states. The algorithm itself is split into two phases, structured successive approximation and structured policy improvement, mirroring the two phases of classical policy iteration. It is important to note that SPI will see fewer advantages if the optimal strategy cannot be compactly represented by a tree structure, and for the reason that there is still big overhead in finding the state partitions.

In [53], Algebraic Decision Diagrams (ADDs) replace the decision-tree learning of SPI for the value function and strategy representation. The paper deals with a very large MDP (≈63 million states) and shows that the learned ADD value function representation is considerably more compact than the corresponding learned decision tree in most cases. However, a big disadvantage of using ADDs is that the state variables must be boolean, which makes the modified state space larger than the original.

In order to solve large weakly coupled FMDPs, the state space of the original MDP is divided into regions that comprise sub-MDPs which run concurrently (the original MDP is a cross-product of the sub-MDPs) [79]. It is assumed that states variables are only associated with a particular task and the numbers of resources that can be allocated to the individual tasks are constrained; these global constraints are what cause the weak coupling between the sub-MDPs. Their approach contains two phases: an offline phase that computes the optimal solutions (value functions) for the individual sub-MDPs and an online phase that uses these local value functions to heuristically guide the search for global resource allocation to the subtasks.

One class of methods for solving weakly coupled FMDPs involves the use of linear value function approximation. In [52], the authors present two solution algorithms (based on approximate linear and dynamic programming) that approximate the value functions using a linear combination of basis functions, each basis function only depending on a small subset of the state variables. In [80], a general framework is proposed that can select a suitable basis set and modify it based on the solution quality. Further, they use piecewise linear combination of the subtask value functions to approximate the optimal value function for the original MDP. The above approaches to solving FMDPs are classified under decision-theoretic planning in that they need a perfect model (transition and reward) of the FMDP. The work in [50] proposes the SDYNA framework that can learn in large FMDPs without initial knowledge of their structure. SDYNA incrementally builds structured representations using incremental decision-tree induction algorithms that learn from the observations made by the agent.

5.2. Hierarchical Approaches

To deal with large-scale FMDPs, RL approaches aim to leverage the structure of the state space. However, they do not impart enough structure to the strategy space itself. Hierarchical reinforcement learning (HRL) is a broad and very active subfield of RL that imposes

hierarchical structure onto the state, action, and strategy spaces. To alleviate the curse of dimensionality, HRL applies principled methods of temporal abstraction to the problem; decision-making should not be required at every step but instead temporally extended activities or macro-operators or subtasks can be selected to achieve subgoals.

The work in [49] proposes a new approach which relies on a programmer to design a hierarchy of abstract machines that limit the possible strategies to be considered. In this hierarchy, each subtask is defined in terms of goal states or termination conditions. Each subtask in the hierarchy corresponds to its own MDP, and the methods seek to compute a strategy that is locally optimal for each subtask.

Many researches in RL allow the learner to work not just with primitive actions, but with higher-level, temporally-extended actions, called options [55, 57, 58, 81]. An option is a closed-loop policy that operates over a period of time, and is defined by the tuple (I, π, β), where π is its strategy, $I \subset S$ is the initiation set of states, and $\beta(s)$ is the probability of termination in state s. The theory of options is based on the theories of MDPs and semi-Markov decision processes (SMDPs), but extends these in significant ways. Options and models of options can be learned for a wide variety of different subtasks, and then rapidly combined to solve new tasks. Using options enables planning and learning simultaneously, at a wide variety of times scales, and toward a wide variety of subtasks. However, the agent's action set is augmented rather than simplified by options which intensify the dimensionality of action spaces.

A Hierarchical Abstract Machine (HAM) is a program that diminishes the number of decisions by partially representing the strategy in the form of finite state machines (FSMs) with a few nondeterministic choice points [56]. HAMs also exploit the theory of SMDPs, but the emphasis is on restricting the policy space rather than augmenting the action space. A HAM is a collection of three tuples $H_i = (\mu, I, \delta)$, where μ is a finite set of machine states, I is the initial state, and δ is the transition function determining the next state using either deterministic or stochastic transitions. The main types of machine states are: start (execute the current machine), action (execute an action), call (execute another machine), choice (select the next machine state), and stop (halt execution and return control). Further, for any MDP M and any HAM H, there exists an induced MDP $M_0 = H \circ M$ [56] that works with a reduced search space using single-step and multistep (or high-level) actions. As a consequence, the induced MDP is in fact a SMDP, because actions can take more that one timestep to complete. A learning algorithm for the induced SMDP is a variation of Q-learning called SMDP Q-learning. This algorithm can be applied to the HAMs framework using an extended Q-table $Q([s, m], a)$, which is indexed by an environment state s, machine state m, and action a taken at a choice state m. Just like options, these HAMs have to be expertly designed because they place strict restrictions on the final strategy possible for the original MDP.

In the MAXQ framework [51], the temporally extended actions or subtasks are organized hierarchically. Faster learning is facilitated by constraining and structuring the space of strategies, encouraging the reuse of subtasks, and enabling effective task-specific state abstraction and aggregation. Unlike options and HAMs, the MAXQ framework does not reduce the original MDP into one SMDP. Instead, the original MDP M is split into sub-SMDPs $M_0, M_1, \ldots, M_n$, where each sub-SMDP represents a subtask.

The big contribution of [82–85] consists in extending the MAXQ framework to the average reward setting with promising results. The work in [86] is a simple extension of the MAXQ framework to the multiagent setting, and it leverages the structure of the task hierarchy to communicate high-level coordination information among the agents. Learning the structure of the task hierarchy is a very promising area of HRL research. The work in

[87] introduces HEXQ, an algorithm that uses frequency of change in the state variables to partition the state space into subtasks—the faster a variable changes, the more likely it is part of the state abstraction of a lower-level subtask. Empirical results pit HEXQ against MAXQ (with a predefined hierarchy) and show that, though there is initial overhead in discovering the hierarchy, HEXQ ends up performing comparably. The work in [88] uses planning to automatically construct task hierarchies based on abstract models of the behaviors' purpose. It then applies RL to flesh out these abstractly defined behaviors, and to learn the choices for ambiguous plans.

6. Dantzig-Wolfe Decomposition

Kushner and Chen [61] investigate the use of the Dantzig-Wolfe decomposition in solving large MDPs as linear programs. Thus, in this section we describe briefly this classic decomposition procedure. A reference for a more complete description is [89].

We consider linear programming problems with the following a block angular structure:

$$\begin{aligned}
&\operatorname{Min} \sum_{l=1}^{h} c_l x_l, \\
&\text{s.t.} \begin{bmatrix} A_1 & A_2 & \cdots & A_h \\ B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_h \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_h \end{bmatrix} = \begin{bmatrix} a \\ b_1 \\ b_2 \\ \vdots \\ b_h \end{bmatrix}, \\
&x_l \geq 0, \quad l = 1, 2, \ldots, h,
\end{aligned} \tag{6.1}$$

where the A_l are $m_o \times n_l$ matrices and the B_l are $m_l \times n_l$ matrices with full row rank. Problems of this type can be decomposed by using the Dantzig-Wolfe decomposition. We get h subproblems, corresponding to the constraints $B_l x_l = b_l$. Let p_j^l, $j \in P_l$, denote the extreme points of the subproblem related to B_l. Problem (6.1) can be reformulated as:

$$\begin{aligned}
&\min \sum_{l=1}^{h} \sum_{j \in P_l} \left(c_l p_j^l\right) \overline{x}_j^l, \\
&\text{s.t.} \sum_{i=1}^{h} \sum_{j \in P_l} \left(A_l p_j^l\right) \overline{x}_j^l = a, \\
&\sum_{j \in P_l} \overline{x}_j^l = 1, \quad l = 1, \ldots, h, \quad \overline{x}_j^l \geq 0.
\end{aligned} \tag{6.2}$$

Problem (6.2) is called the full master problem. Instead of looking at the full master problem we only consider a subset of the columns and then generate new columns when they are needed. The reduced or the restricted master problem has $m_0 + h$ rows, where the last h corresponds to the convexity constraints. We assign the dual variables y to the first m_0 constraints of the restricted master problem and λ to the h convexity constraints. In each master iteration, the restricted master problem is solved. Each of the subproblems is then solved with the cost vector $(c_l - yA_l)$. Hence, the lth subproblem to be solved is

$$\begin{aligned} &\min(c_l - yA_l)\tilde{x}_l, \\ &\text{s.t. } B_l\tilde{x}_l = b_l, \\ &\tilde{x}_l \geq 0. \end{aligned} \tag{6.3}$$

If there exists a solution $\tilde{x}_l$ with $(c_l - yA_l)\tilde{x}_l - \lambda_l < 0$, then a column with negative reduced cost has been found, and it is introduced into the restricted master. The new column is given as $\begin{bmatrix} A_l\tilde{x}_l \\ e_l \end{bmatrix}$, where $e_l \in \Re^h$ is the lth unit-vector. It has the cost $c_l\tilde{x}_l$.

Both the master problem and the subproblems axis on the use of simplex method in solving the linear program in (6.2). We refer the readers to [90] for more details about (i) cycling prevention and (ii) the initialization of the simplex method using big-M method. Using the Dantzig-Wolfe decomposition, the solution is improved iteratively and converges to an optimum solution in a finite number of iterations.

7. Conclusion

The benefit of decomposition techniques is that we are able to deal with subproblems of smaller size; the tradeoff is that often extra effort is required to combine the solutions to these subproblems into a solution to the original problem. Thus, some new methods are expected to cope with the difficulty of combining the subproblems.

Many of the approaches discussed in this survey are collection of separate mature fields coming together to deal with the twin drawbacks of curse of dimensionality and lack of model information in most real-world systems. For instance, many Hierarchical Reinforcement Learning concepts utilize some notions of programming languages such as subroutines, task stacks, and control threads. We speculate if any other domains are forthcoming to be imported.

References

[1] W. J. Stewart and A. Goyal, "Matrix methods in large dependability models," Tech. Rep. RC 11485, IBM T.J. Watson Research Center, 1985.

[2] J. G. Kemeny and J. L. Snell, *Finite Markov Chains*, The University Series in Undergraduate Mathematics, D. Van Nostrand, Princeton, NJ, USA, 1960.

[3] P.-J. Courtois, *Decomposability: Queueing and Computer System Applications*, ACM Monograph Serie, Academic Press, New York, NY, USA, 1977.

[4] D. D. Dimitrijevic and M. Chen, "An integrated algorithm for probabilistic protocol verification and evaluation," Tech. Rep. RC 13901, IBM, 1988.

[5] E. de Souza e Silva and H. R. Gail, "Calculating availability and performability measures of repairable computer systems using randomization," *Journal of the ACM*, vol. 36, no. 1, pp. 171–193, 1989.

[6] P.-J. Courtois and P. Semal, "Bounds for the positive eigenvectors of nonnegative matrices and for their approximations by decomposition," *Journal of the Association for Computing Machinery*, vol. 31, no. 4, pp. 804–825, 1984.
[7] P. Courtois and P. Semal, "Computable bounds for conditional steady-state probabilities in large Markov chains and queueing models," *IEEE Journal on Selected Areas in Communications*, vol. 4, no. 6, pp. 926–937, 1986.
[8] S. M. Ross, *Stochastic Processes: Lectures in Mathematics*, Wiley Series in Probability and Mathematical Statistics: Probability and Mathematical Statistics, John Wiley & Sons, New York, NY, USA, 1983.
[9] D. Stoyan, *Comparison Methods for Queues and other Stochastic Models*, Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics, John Wiley & Sons, Chichester, UK, 1983.
[10] N. M. Van Dijk, "The importance of bias-terms for error bounds and comparison results," in *Proceedings of the 1st International Conference on Numerical Solutions of the Markov Chains*, North Carolina State University, January, 1989.
[11] R. R. Muntz and J. C. S. Lui, "Eclectric survey of bounding methods for Markov chain models," in *Proceedings of the 3rd International Workshop on Modeling, Analysis, and Simulation*, Durham, NC, USA, January 1995.
[12] A. Arapostathis, V. Borkar, E. Fernandez-Gaucherand, M. K. Ghosh, and S. I. Marcus, "Discrete-time controlled Markov processes with average cost criterion: a survey," *SIAM Journal on Control and Optimization*, vol. 31, no. 2, pp. 282–344, 1993.
[13] C. Boutilier, R. I. Brafman, and C. Geib, "Structured readability analysis for Markov decision processes," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI '98)*, 1998.
[14] Y. Carmon and A. Shwartz, "Markov decision processes with exponentially representable discounting," *Operations Research Letters*, vol. 37, no. 1, pp. 51–55, 2009.
[15] C. Daoui and M. Abbad, "On some algorithms for limiting average Markov decision processes," *Operations Research Letters*, vol. 35, no. 2, pp. 261–266, 2007.
[16] E. V. Denardo, *Dynamic Programming: Models and Applications*, Dover Publications, Mineola, NY, USA, 2003.
[17] M. Ghavamzadeh and S. Mahadevan, "Hierarchically optimal average reward reinforcement learning," in *Proceedings of the 19th International Conference on Machine Learning*, 2002.
[18] D. J. White, *Markov Decision Processes*, John Wiley & Sons, Chichester, UK, 1993.
[19] W. Wongthatsanekorn, M. J. Realff, and J. C. Ammons, "Multi-time scale Markov decision process approach to strategic network growth of reverse supply chains," *Omega*, vol. 38, no. 1-2, pp. 20–32, 2010.
[20] Q. X. Zhu, "Sample-path optimality and variance-maximization for Markov decision processes," *Mathematical Methods of Operations Research*, vol. 65, no. 3, pp. 519–538, 2007.
[21] Q. X. Zhu, "Average optimality for continuous-time Markov decision processes with a policy iteration approach," *Journal of Mathematical Analysis and Applications*, vol. 339, no. 1, pp. 691–704, 2008.
[22] Q. Zhu and X. Guo, "Markov decision processes with variance minimization: a new condition and approach," *Stochastic Analysis and Applications*, vol. 25, no. 3, pp. 577–592, 2007.
[23] C. Boutilier and D. Poole, "Computing optimal policies for partially observable decision processes using compact representations," in *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI '96)*, Portland, Ore, USA, 1996.
[24] R. I. Brafman, "A heuristic variable grid solution method for POMDPs," in *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*, pp. 727–733, July 1997.
[25] A. R. Cassandra, L. P. Kaelbling, and J. A. Kurien, "Acting under uncertainty: discrete Bayesian models for mobile-robot navigation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '96)*, pp. 963–972, November 1996.
[26] A. R. Cassandra, "Optimal policies for partially observable Markov decision processes," Tech. Rep. CS-94-14, Brown University, 1994.
[27] A. R. Cassandra, *Exact and approximate algorithms for partially observable Markov decision processes*, Ph.D. thesis, Brown University, 1998.
[28] M. Hauskrecht, N. Meuleau, L. P. Kaelbling, T. Dean, and G. Boutilier, "Hierarchical solution of Markov decision processes using macro-actions," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 1998.
[29] M. Haviv and M. L. Puterman, "An improved algorithm for solving communicating average reward Markov decision processes," *Annals of Operations Research*, vol. 28, no. 1, pp. 229–242, 1991.
[30] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[31] M. Gondran and M. Minoux, *Graphes et Algorithmes*, vol. 37 of *Collection de la Direction des Études et Recherches d'Électricité de France*, Éditions Eyrolles, Paris, France, 1979.
[32] M. Mundhenk, J. Goldsmith, C. Lusena, and E. Allender, "Complexity of finite-horizon Markov decision process problems," Tech. Rep. 273-97, 1997.
[33] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of Markov decision processes," *Mathematics of Operations Research*, vol. 12, no. 3, pp. 441–450, 1987.
[34] M. Abbad and C. Daoui, "Algorithms for aggregated limiting average Markov decision problems," *Mathematical Methods of Operations Research*, vol. 53, no. 3, pp. 451–463, 2001.
[35] M. Abbad and C. Daoui, "Hierarchical algorithms for discounted and weighted Markov decision processes," *Mathematical Methods of Operations Research*, vol. 58, no. 2, pp. 237–245, 2003.
[36] M. Abbad and H. Boustique, "A decomposition algorithm for limiting average Markov decision problems," *Operations Research Letters*, vol. 31, no. 6, pp. 473–476, 2003.
[37] J. Bather, "Optimal decision procedures for finite Markov chains. III. General convex systems," *Advances in Applied Probability*, vol. 5, pp. 541–553, 1973.
[38] K. W. Ross and R. Varadarajan, "Multichain Markov decision processes with a sample path constraint: a decomposition approach," *Mathematics of Operations Research*, vol. 16, no. 1, pp. 195–207, 1991.
[39] T. Dean and S.-H. Lin, "Decomposition techniques for planning in stochastic domains," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Montreal, Canada, 1995.
[40] P. Laroche, Y. Boniface, and R. Schott, "A new decomposition technique for solving markov decision processes," in *Proceedings of ACM Symposium on Applied Computing*, pp. 12–16, 2001.
[41] R. Dearden and C. Boutilier, "Abstraction and approximate decision-theoretic planning," *Artificial Intelligence*, vol. 89, no. 1-2, pp. 219–283, 1997.
[42] M. Hauskrecht, "Value-function approximations for partially observable Markov decision processes," *Journal of Artificial Intelligence Research*, vol. 13, pp. 33–94, 2000.
[43] R. Parr, "Flexible decomposition algorithms for weakly coupled Markov decision problems," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 1998.
[44] D. Precup and R. S. Sutton, "Multi time models for temporally abstract planning," in *Advances in Neural Information Processing Systems*, MIT Press, 1997.
[45] T. Dean, L. P. Kaelbling, J. Kirman, and A. Nicholson, "Planning under time constraints in stochastic domains," *Artificial Intelligence*, vol. 76, no. 1-2, pp. 35–74, 1995.
[46] T. Dean, L. P. Kaelbling, J. Kirman, and A. Nicholson, "Planning with deadlines in stochastic domains," in *Proceedings of the 11th National Conference on Artificial Intelligence*, pp. 574–579, July 1993.
[47] R. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ, USA, 1957.
[48] C. Boutilier, R. Dearden, and M. Goldszmidt, "Exploiting structure in policy construction," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1995.
[49] P. Dayan and G. Hinton, "Feudal reinforcement learning," in *Advances in Neural Information Processing Systems 5*, pp. 271–278, Morgan Kaufmann, 1993.
[50] T. Degris, O. Sigaud, and P.-H. Wuillemin, "Learning the structure of factored markov decision processes in reinforcement learning problems," in *Proceedings of the 23rd International Conference on Machine Learning (ICML '06)*, pp. 257–264, June 2006.
[51] T. G. Dietterich, "Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition," *Journal of Artificial Intelligence Research*, vol. 13, pp. 227–303, 2000.
[52] E. Hansen and Z. Feng, "Dynamic programming for POMDPs using a factored state representation," in *Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling (AIPS '00)*, Breckenridge, Colo, USA, 2000.
[53] R. A. Howard, *Dynamic Programming and Markov Processes*, The Technology Press of M.I.T., Cambridge, Mass, USA, 1960.
[54] L. P. Kaelbling, "Hierarchical reinforcement learning: preliminary results," in *Proceedings of the 10th International Conference Machine Learning (ICML '93)*, pp. 167–173, Amherst, Mass, USA, June 1993.
[55] A. McGovern, R. S. Sutton, and A. H. Fagg, "Roles of macro-actions in accelerating reinforcement learning," in *Grace Hopper Celebration of Women in Computing*, pp. 13–17, 1997.
[56] R. Parr and S. Russell, "Reinforcement learning with hierarchies of machines," in *Advances in Neural Information Processing Systems*, 1998.
[57] D. Precup, R. S. Sutton, and S. Singh, "Multi-time models for temporally abstract planning," in *Advances in Neural Information Processing Systems 10*, MIT Press, 1998.
[58] D. Precup, R. S. Sutton, and S. Singh, "Theoretical results on reinforcement learning with temporally abstract options," in *Proceedings of the 10th European Conference on Machine Learning*, pp. 382–393, 1998.
[59] L. S. Lasdon, *Optimization Theory for Large Systems*, The Macmillian, New York, NY, USA, 1970.

[60] G. Dantzig and P. Wolfe, "Decomposition principle for dynamic programs," *Operations Research*, vol. 8, no. 1, pp. 101–111, 1960.
[61] H. J. Kushner and H. H. Chen, "Decomposition of systems governed by Markov chains," *IEEE Transactions on Automatic Control*, vol. 19, pp. 501–507, 1974.
[62] C. Derman, *Finite State Markovian Decision Processes*, Mathematics in Science and Engineering, Vol. 67, Academic Press, New York, NY, USA, 1970.
[63] L. C. M. Kallenberg, *Linear Programming and Finite Markovian Control Problems*, Mathematical Center Tracts, Amsterdam, The Netherlands, 1983.
[64] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics, John Wiley & Sons, New York, NY, USA, 1994.
[65] J. A. Filar and O. J. Vrieze, "Weighted reward criteria in Competitive Markov Decision Processes," *ZOR Zeitschrift für Operations Research Methods and Models of Operations Research*, vol. 36, no. 4, pp. 343–358, 1992.
[66] L. C. M. Kallenberg, "Classification problems in MDPs," in *Markov Processes and Controlled Markov Chains*, pp. 151–165, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.
[67] Z. M. Avsar and M. Baykal-Gürsoy, "A decomposition approach for undiscounted two-person zero-sum stochastic games," *Mathematical Methods of Operations Research*, vol. 49, no. 3, pp. 483–500, 1999.
[68] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman, "Efficient solution algorithms for factored MDPs," *Journal of Artificial Intelligence Research*, vol. 19, pp. 399–468, 2003.
[69] B. Hengst, "Safe state abstraction and discounting in hierarchical reinforcement learning," Tech. Rep. CSE TR 0308, UNSW, 2003.
[70] T. Dean and S.-H. Lin, "Decomposition techniques for planning in stochastic domains," Tech. Rep. CS-95-10, Brown University, 1995.
[71] R. E. Korf, "Macro-operators: a weak method for learning," *Artificial Intelligence*, vol. 26, no. 1, pp. 35–77, 1985.
[72] C. A. Knoblock, "Search reduction in hierarchical problem solving," in *Proceedings of the 9th National Conference on Artificial Intelligence (AAAI '91)*, pp. 686–691, 1991.
[73] E. D. Sacerdoti, "Planning in a hierarchy of abstraction spaces," *Artificial Intelligence*, vol. 5, no. 2, pp. 115–135, 1974.
[74] P. E. Caines and S. Wang, "COCOLOG: a conditional observer and controller logic for finite machines," in *Proceedings of the 29th IEEE Conference on Decision and Control*, pp. 2845–2850, December 1990.
[75] A. W. Moore and C. G. Atkeson, "Parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces," *Machine Learning*, vol. 21, no. 3, pp. 199–233, 1995.
[76] L. P. Kaelbling, "Hierarchical learning in stochastic domains: a preliminary report," in *Proceedings of the Tenth International Conference on Machine Learning*, 1993.
[77] R. Sutton and A. Barto, *Reinforcement Learning*, MIT Press, Cambridge, Mass, USA, 1998.
[78] T. Dean and K. Kanazawa, "A model for reasoning about persistence and causation," *Computational Intelligence*, vol. 5, no. 3, pp. 142–150, 1989.
[79] N. Meuleau, M. Hauskrecht, K. Kim et al., "Solving very large weakly coupled Markov decision processes," in *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI '98)*, pp. 165–172, July 1998.
[80] P. Poupart, C. Boutilier, R. Patrascu, and D. Schuurmans, "Piecewise linear value function approximation for factored MDPs," in *18th National Conference on Artificial Intelligence (AAAI-02), 14th Innovative Applications of Artificial Intelligence Conference (IAAI-02)*, pp. 292–299, can, August 2002.
[81] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, no. 1, pp. 181–211, 1999.
[82] D. Andre and S. J. Russell, "State abstraction for programmable reinforcement learning agents," in *Proceeding of the 18th National Conference on Artificial Intelligence (AAAI '02)*, pp. 119–125, August 2002.
[83] J. Goldsmith and M. Mundhenk, "Complexity issues in Markov decision processes," in *Proceedings of the 13th Annual IEEE Conference on Computational Complexity*, pp. 272–280, Buffalo, NY, USA, 1998.
[84] B. Marthi, S. Russell, and D. Andre, "A compact, hierarchically optimal Q-function decomposition," in *22nd Conference on Uncertainty in Artificial Intelligence*, 2006.
[85] S. Seri and P. Tadepalli, "Model-based hierarchical average reward reinforcement learning," in *Proceedings of the 19th International Conference on Machine Learning*, 2002.
[86] R. Makar, S. Mahadevan, and M. Ghavamzadeh, "Hierarchical multi-agent reinforcement learning," in *Proceeding of the 5th International Conference on Autonomous Agents*, pp. 246–253, June 2001.

[87] J. Hoey, R. Aubin, A. Hu, and C. Boutilier, "SPUDD: stochastic planning using decision diagrams," in *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, 1999.

[88] M. Ryan, "Using abstract models of behaviors to automatically generate reinforcement learning hierarchies," in *Proceedings of the 19th International Conference on Machine Learning*, 2002.

[89] Y. M. I. Dirickx and L. P. Jennergren, *Systems Analysis by Multilevel Methods: With Applications to Economics and Management*, vol. 6 of *International Series on Applied Systems Analysis*, John Wiley & Sons, Chichester, UK, 1979.

[90] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows*, John Wiley & Sons, New York, NY, USA, 2nd edition, 1990.