An Iterative Decision-Making Scheme for Markov Decision Processes and Its Application to Self-adaptive Systems

Guoxin Su[1(✉)], Taolue Chen[2], Yuan Feng[3], David S. Rosenblum[1], and P.S. Thiagarajan[4]

[1] School of Computing, National University of Singapore, Singapore, Singapore
guoxinsu@gmail.com
[2] Department of Computer Science, Middlesex University London, London, UK
[3] Centre for Quantum Computation and Intelligent Systems, University of Technology Sydney, Sydney, Australia
[4] Laboratory of Systems Pharmacology, Harvard Medical School, Boston, USA

Abstract. Software is often governed by and thus adapts to phenomena that occur at runtime. Unlike traditional decision problems, where a decision-making model is determined for reasoning, the adaptation logic of such software is concerned with empirical data and is subject to practical constraints. We present an *Iterative Decision-Making Scheme* (IDMS) that infers both point and interval estimates for the undetermined transition probabilities in a Markov Decision Process (MDP) based on sampled data, and iteratively computes a confidently optimal scheduler from a given finite subset of schedulers. The most important feature of IDMS is the flexibility for adjusting the criterion of confident optimality and the sample size within the iteration, leading to a tradeoff between accuracy, data usage and computational overhead. We apply IDMS to an existing self-adaptation framework Rainbow and conduct a case study using a Rainbow system to demonstrate the flexibility of IDMS.

1 Introduction

Software is often governed by and thus adapts to phenomena that occur at runtime [22]. One typical example is the control software of autonomous systems, such as driverless vehicles. Because the occurrence of runtime phenomena is asynchronous with respect to the flow of the application logic, because not all information about the phenomena is available at the design time, and because the specification of the adaptive behavior may evolve over time, it is advantageous to gather the complex adaptation logic into a component separated from

This work was partially supported by the Singapore Ministry of Education (Grant Nos. R-252-000-458-133 and MOE2015-T2-1-137), the Australian Research Council (Grant Nos. DP130102764 and DP160101652), the National Natural Science Foundation of China (Grant Nos. 61428208 and 61502260), the CAS/SAFEA International Partnership Program for Creative Research Team, and an overseas grant from the State Key Laboratory of Novel Software Technology at Nanjing University.

P. Stevens and A. Wąsowski (Eds.): FASE 2016, LNCS 9633, pp. 269–286, 2016.
DOI: 10.1007/978-3-662-49665-7_16

the application logic. In contrast to traditional decision problems where a decision model, such as a Markov Decision Process (MDP) [27], is determined for reasoning, the adaptation logic of autonomous systems is governed by empirical data and is subject to practical constraints. In many situations, one has to sacrifice the optimality of an adaptive solution to a certain extent in order to satisfy various Quality-of-Service (QoS) constraints.

Consider a Web system that provides news content services. At some moment, the system may detect high latency of content delivery. Suppose that the system can lower the content fidelity (such as delivering multimedia contents in the text mode) and/or increase the server pool size, and that the benefits or costs of these operations are measured quantitatively. Further, to achieve more sophisticated effects, operations can be combined to form a strategy. For example, one simple strategy could be the following: Once "high latency" is detected, increase the number of Virtual Machine (VM) instances by one; if "high latency" persists, switch from the multimedia mode to the text mode. Because multiple strategies built into the adaptation logic may be triggered by the same condition, an additional mechanism is required to select one of them.

A key challenge of the strategy selection for the Web system is that some probability parameters, such as successful chances of operations, are not fixed. For example, if the VM number is increased by one, the probability that latency will drop below the threshold may increase, but it still has to be estimated based on runtime data. While the idealized goal is to select an optimal strategy, it is important to take into account the practical constraints. For example, obsolete data no longer reflects the current environmental situation; the time frame of data sampling may be constrained by the tolerance of adaptation delay; the sampling frequency may be restricted because of its performance overhead on the network; and last but not the least, the adaptation should not downgrade the functional performance of the system by consuming too much computational capacity (e.g., CPU and RAM). In short, besides decision accuracy, runtime decision-making has to address the limitation of data and computation resource.

The above adaptation model for the Web system can be formalized as an MDP in which actions represent operations and schedulers represent strategies. The runtime data are stored in a data structure (i.e., a set of integer matrices) for estimating the transition probabilities of the MDP. Therefore, the problem of strategy selection is an instance of the general problem of minimizing the (expected) cumulative cost for an MDP with empirically determined transition probabilities and a given subset of schedulers. Despite this problem is well understood in the theory of MDPs [27], our first contribution is an *Iterative Decision-Making Scheme* (IDMS) that supports a trade-off between three important metrics, namely, accuracy, data usage and computational overhead. The basic idea of IDMS is as follows:

1. We infer both point and interval estimates of transition probabilities for the MDP decision model based on the data structure for runtime data.
2. Next, we compute a scheduler that minimizes the cumulative cost for a given reachability problem.

3. We then determine whether this scheduler meets a criterion called *confident optimality*. If yes, or if the maximal number of iterative steps is reached, the iteration terminates; otherwise, the iteration returns to data sampling.

We formalize three metrics for IDMS: (i) the probability that a confidently optimal scheduler is truly optimal, namely accuracy; (ii) the average sample size of the iteration, which is a direct metric of data usage; and (iii) the average time of iteration, which measures computational overhead conveniently. The tradeoff among these three metrics is realized by adjusting the criterion of confident optimality and the sample size during the iteration. The core method of IDMS is a value-iteration algorithm developed from probabilistic model checking [19].

The second contribution of this paper is an application of IDMS to self-adaptive systems. Several high-level frameworks and approaches based on probabilistic model checking have been proposed to aid the design of self-adaptive systems, but with emphasis on different aspects of the adaptation [3,4,18,20,23]. However none of these works address the problem of making the aforementioned tradeoff in the adaptation. We demonstrate that IDMS can be naturally embedded into the Rainbow framework [11] which employs a standard, point-valued MDP as its decision model, and thus extends the adaptation function of the latter. We present a case study on a Rainbow system and the empirical evidence that demonstrates the flexibility of IDMS.

The remainder of the paper is organized as follows. Section 2 presents the formal models and core method. Section 3 presents the IDMS scheme. Section 4 describes the application to self-adaptive systems. Section 5 presents the case study. Section 6 reports the related work. Section 7 concludes the paper.

2 Formal Model and Value-Iteration Method

In this section, we present our formal models and value-iteration method. The position of our method in the state of the art is discussed in Sect. 6.

Definition 1 (MDP). *An MDP is a tuple $\mathcal{M} = (S, Act, \mathcal{P}, \alpha, C)$ where*

- *S is a finite, non-empty state space,*
- *Act is a finite non-empty set of actions,*
- *α is the initial distribution over S,*
- *$\mathcal{P} = \{\mathcal{P}_a\}_{a \in Act}$ is a family of* transition probability matrices *indexed by $a \in Act$, and*
- *$C : S \to \mathbf{R}_{\geq 0}$ is a cost function.*

We require that, for each $a \in Act$ and $s \in S$, $\mathcal{P}_a[s,t] \geq 0$ for all $t \in S$ and $\sum_{t \in S} \mathcal{P}_a[s,t] \in \{0,1\}$. We say action a is enabled *at s if $\sum_{t \in S} \mathcal{P}_a[s,t] = 1$.*

Schedulers play a crucial role in the analysis of MDPs. For our purposes, it suffices to consider *simple* schedulers, in which for each state s, the scheduler fixes one of the enabled actions at s and selects the same action every time when the system resides in s. Formally, a simple scheduler is a function $\sigma : S \to Act$

such that $\sigma(s)$ is one of the actions enabled at state s. In our setting, instead of considering the whole set of schedulers, we work only with a (finite) subset of simple schedulers Σ specified by the user. A *path* in $\mathcal{M}$ under σ is an infinite sequence of states $\rho = s_0 s_1 \cdots$ such that, for all $i \geq 0$, $\mathcal{P}_a[s_i, s_{i+1}] > 0$ for $a = \sigma(s_i)$. Let $Path_{\mathcal{M},\sigma}$ be the set of paths in $\mathcal{M}$ under σ. Let $Path_{\mathcal{M},\sigma}(s)$ be the subset of paths that start from s. Let $Pr_{\mathcal{M},\sigma}$ be the standard *probability distribution* over $Path_{\mathcal{M},\sigma}$ as defined in the literature [1, Chap. 10].

The *expected cumulative cost*, or simply *cumulative cost*, of reaching a set $G \subseteq S$ of *goal* states (called G-states hereafter) in $\mathcal{M}$ under σ, denoted $C_{\mathcal{M},\sigma}(G)$, is defined as follows: First, let $C_{\mathcal{M},\sigma}(s, G)$ be the expected value of random variable $X : Path_{\mathcal{M},\sigma}(s) \to \mathbf{R}_{\geq 0}$ such that (i) if $s \in G$ then $X(\rho) = 0$, (ii) if $\rho[i] \notin G$ for all $i \geq 0$ then $X(\rho) = \infty$, and (iii) otherwise $X(\rho) = \sum_{i=0}^{n-1} C(s_i)$ where $s_n \in G$ and $s_j \notin G$ for all $j < n$. Then, let $C_{\mathcal{M},\sigma}(G) = \sum_{s \in S} \alpha(s) \cdot C_{\mathcal{M},\sigma}(s, G)$.

By the above definitions, for those states which do *not* reach the goal states almost surely (viz. with probability less than 1), the cumulative cost is ∞. We remark that other definitions on the costs of paths not reaching the goal states do exist and can be found in [8]. However, they are more involved and are not needed in the current setting. In order to compute the cumulative cost, we first have to identify the set of states $S_{=1}$ from which the probability to reach the goal states in G is 1. This can be done by a standard graph analysis [1, Chap. 10]. Next, we solve the following system of linear equations with variables $(x_s)_{s \in S_{=1}}$:

$$\begin{aligned} x_s &= 0 && \text{if } s \in G \\ x_s &= C(s) + \sum\nolimits_{t \in S_{=1}} \mathcal{P}_a[s,t] \cdot x_t && \text{if } s \notin G \end{aligned} \tag{1}$$

where $a = \sigma(s)$. When the scheduler is fixed, the MDP is reduced to a discrete-time Markov chain (DTMC) and hence solving (1) is straightforward. One can employ standard Jacobi or Gauss-Seidel itertaion methods to compute the least fixpoint [31]. In detail, one starts from $\boldsymbol{x}^{(0)}$ where $x_s^{(0)} = 0$ for all $s \in S_{=1}$, and computes $x_s^{(n+1)} = C(s) + \sum_{t \in S_{=1}} \mathcal{P}_a[s,t] \cdot x_t^{(n)}$ if $s \notin G$ and 0 otherwise, until $\max_{s \in S} |x_s^{(n+1)} - x_s^{(n)}| < \epsilon$ for some predetermined $\epsilon > 0$. In practice, and especially in probabilistic verification, this is usually more efficient than the Gaussian elimination [19].

Interval-valued MDPs (IMDP) are MDPs where some of the transition probabilities are specified as real intervals.

Definition 2 (IMDP). *An IMDP is a tuple* $\mathcal{M}^I = (S, Act, \mathcal{P}^+, \mathcal{P}^-, \alpha, C)$ *where*

- *S, Act, α and C are defined the same as in Definition 1,*
- *$\mathcal{P}^+ = \{\mathcal{P}_a^+\}_{a \in Act}$, $\mathcal{P}^- = \{\mathcal{P}_a^-\}_{a \in Act}$ are two families of nonnegative matrices indexed by $a \in Act$, giving the* upper *and* lower *bounds of transition probabilities respectively.* Further, for each $a \in Act$, $\mathcal{P}_a^+$ and $\mathcal{P}_a^-$ have the same corresponding 0- and 1-entries.

With $\mathcal{M}^I = (S, Act, \mathcal{P}^+, \mathcal{P}^-, \alpha, C)$ we associate a set of MDPs $[\![\mathcal{M}^I]\!]$ such that $\mathcal{M} = (S, Act, \mathcal{P}, \alpha, C) \in [\![\mathcal{M}^I]\!]$ *if and only if* for each $a \in Act$, $\mathcal{P}_a^- \leq \mathcal{P}_a \leq \mathcal{P}_a^+$. where $\leq$ is interpreted entry-wise. We call an $\mathcal{M} \in [\![\mathcal{M}^I]\!]$ an *instance* of $\mathcal{M}^I$.

Given an IMDP $\mathcal{M}^I$ and a simple scheduler σ, since the possible cumulative cost of reaching G-states is in the form of an interval, we are interested in the *bounds* of such an interval. The *minimum* cumulative cost of reaching G-states in $\mathcal{M}^I$ under σ is

$$C^{\min}_{\mathcal{M}^I,\sigma}(G) = \inf_{\mathcal{M}\in[\![\mathcal{M}^I]\!]} C_{\mathcal{M},\sigma}(G).$$

Because the *maximum* cumulative cost $C^{\max}_{\mathcal{M}^I,\sigma}(G)$ is symmetrical to the minimum case, in the remainder of this section, we mainly deal with the latter.

To this end, as before we first identify states that reach the goal states G almost surely (under σ) and are denoted by $S_{=1}$. Owing to the assumption made on IMDPs in Definition 2, this can be done by graph-analysis as on MDPs $\mathcal{M}^I$. For those states not in $S_{=1}$, the minimal cost is ∞ according to our convention. We then consider the following Bellman equation over the variables $(x_s)_{s\in S_{=1}}$:

$$\begin{aligned} x_s &= 0 && \text{if } s \in G \\ x_s &= \min_{\mathcal{P}_a^- \le \mathcal{P}_a \le \mathcal{P}_a^+} \{C(s) + \textstyle\sum_{t\in S_{=1}} \mathcal{P}_a[s,t]\cdot x_t\} && \text{if } s \notin G \end{aligned} \tag{2}$$

where $a = \sigma(s)$. Note that $\mathcal{P}_a$ is required to be a transition probability matrix. Let $\boldsymbol{x} = (x_s)_{s\in S_{=1}}$ be the *least* fixpoint of (2). We easily obtain:

Proposition 1. $C^{\min}_{\mathcal{M}^I,\sigma}(G) = \sum_{s\in S} \alpha(s)x_s$.

To solve (2), there are essentially two approaches. The first one is to reduce it to linear programming (LP). However, despite theoretically elegant, this is not practical for real-life cases. Instead, we apply the second approach, i.e., the value-iteration method. For each iteration, the crucial part is to compute

$$\min_{\mathcal{P}_a^- \le \mathcal{P}_a \le \mathcal{P}_a^+} \left\{C(s) + \sum_{t\in S_{=1}} \mathcal{P}_a[s,t]\cdot x_t\right\}$$

for a given $\boldsymbol{x}$. This problem can be reduced to a standard linear program. Indeed, for each s, introduce variables $(y_t)_{t\in S}$ and consider the problem:

$$\begin{aligned} \text{minimize} \quad & C(s) + \textstyle\sum_{t\in S_{=1}} y_t x_t \\ \text{subject to} \quad & \textstyle\sum_{t\in S_{=1}} y_t = 1 \text{ and } \mathcal{P}_a^-[s,t'] \le y_{t'} \le \mathcal{P}_a^+[s,t'] \text{ for all } t' \in S_{=1}. \end{aligned}$$

This can be solved efficiently via off-shelf LP solvers (note that here x_t's and a are given). Hence each iteration takes polynomial time. We also remark that the LP here admits a very simple structure and only contains at most $|S|$ variables (and usually much less for practical examples), while the direct approach (based on LP as well) requires at least $|S|^2+|S|$ variables and is considerably more involved. Although it might take exponentially many iterations to reach the least fixpoint, in practice one usually sets a stopping criteria such as $\max_{s\in S} |x_s^{(n+1)} - x_s^{(n)}| < \epsilon$ for a fixed error bound $\epsilon > 0$.

Let $C^{\text{dif}}_{\mathcal{M}^I,\sigma}(G) = C^{\max}_{\mathcal{M}^I,\sigma}(G) - C^{\min}_{\mathcal{M}^I,\sigma}(G)$. Because $\mathcal{M}$, $\mathcal{M}^I$ and G are clear in the context, to simplify notations we make the following abbreviations:

Fully-Spelled	$C_{\mathcal{M},\sigma}(G)$	$C^{\min}_{\mathcal{M}^I,\sigma}(G)$	$C^{\max}_{\mathcal{M}^I,\sigma}(G)$	$C^{\text{dif}}_{\mathcal{M}^I,\sigma}(G)$
Abbreviated	C_σ	$C^{\min}_\sigma$	$C^{\max}_\sigma$	C^{dif}_σ

3 Iterative Decision-Making Scheme

In this section, we present main stages and techniques of IDMS and describe the realization of trade-offs between the three metrics.

3.1 IDMS Preview and Example

IDMS is an iterative process that contains one pre-stage and five runtime stages (i.e., Stage 1 to 5), as depicted in Fig. 1. The pre-stage builds up a parametric MDP with transition probability parameters in the design time. At runtime Stage 1 collects data samples and Stage 2 infers point and interval estimates based on the samples. By instantiating the parameters with the point and interval estimates, Stage 3 builds up a (concrete) MDP and an IMDP. Stage 4 attempts to compute a confidently optimal scheduler. Then the process either moves to Stage 5 where a decision is made or goes back to Stage 1. The process terminates when either a confidently optimal scheduler is returned, or the maximal time of iteration (namely the maximal number of steps within the iteration) is reached. Note as the decision making may need to be repeated periodically at runtime, Stage 5 may be followed by Stage 1.

A parametric MDP example $\mathcal{M}_{eg}(\boldsymbol{\theta})$ is described in Fig. 2. The state space of $\mathcal{M}_{eg}(\boldsymbol{\theta})$ is $\{s_0, \ldots, s_7, s_G\}$ with s_0 being the only initial state (i.e., the initial distribution assigns 1 to s_0 and 0 to other states) and s_G being the only goal state. The dashed arrows are probabilistic transitions, labeled by parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_5)$. The solid arrows are non-probabilistic transitions (or, equivalently, transitions with the fixed probability 1). The wavy arrows represent non-deterministic transitions, with a and b being two actions. For $\mathcal{M}_{eg}(\boldsymbol{\theta})$, the two actions induce two schedulers, denoted σ_a and σ_b, respectively. States of $\mathcal{M}_{eg}(\boldsymbol{\theta})$ are associated with costs ranging from 0 to 2.

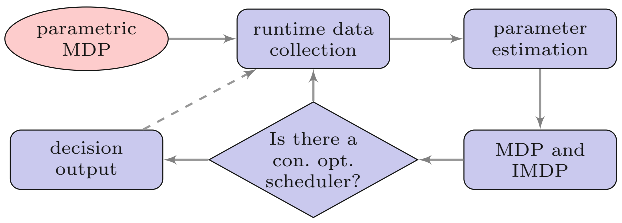


Fig. 1. Pre-stage and runtime stages of IDMS

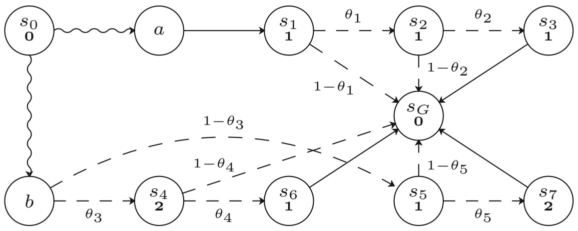


Fig. 2. A parametric MDP example $\mathcal{M}_{eg}(\boldsymbol{\theta})$

3.2 Data Structure and Parameter Estimation

IDMS does not presume a particular method for collecting runtime data but it stores them in a specific data structure, namely a set of non-negative integer matrices that are related to schedulers of the parametric MDP. The integer in each entry represents the number of times that the corresponding transition is recorded in the sampling time frame. For example, the two integer matrices related to σ_a and σ_b of $\mathcal{M}_{eg}(\boldsymbol{\theta})$ are as follows:

$$M_a : \begin{array}{c} \\ s_1 \\ s_2 \end{array} \begin{array}{c} \begin{array}{cccc} s_1 & s_2 & s_3 & s_G \end{array} \\ \begin{bmatrix} 0 & N_{1,2} & 0 & N_{1,G} \\ 0 & 0 & N_{2,3} & N_{2,G} \end{bmatrix} \end{array} \qquad M_b : \begin{array}{c} \\ s_0 \\ s_4 \\ s_5 \end{array} \begin{array}{c} \begin{array}{ccccc} s_4 & s_5 & s_6 & s_7 & s_G \end{array} \\ \begin{bmatrix} N_{0,4} & N_{0,5} & 0 & 0 & \\ 0 & 0 & N_{4,5} & 0 & N_{4,G} \\ 0 & 0 & 0 & N_{5,7} & N_{5,G} \end{bmatrix} \end{array}$$

where $N_{\sharp,\natural} > 0$, with $\sharp$ and $\natural$ denoting (some) elements in $\{0,\ldots,7,G\}$, are integer variables. $N_{\sharp,\natural}$ is increased by 1 (i.e., $N_{\sharp,\natural} \leftarrow N_{\sharp,\natural}+1$) if a transition from $s_\sharp$ to $s_\natural$ is newly observed. Note that zero entries in M_a and M_b remain unchanged for all time, because according to the structural specification of $\mathcal{M}_{eg}(\boldsymbol{\theta})$, the correspondent transitions are impossible to occur.

The data structure is used to estimate parameters in the parametric MDP. IDMS adopts two forms of estimation, namely point estimation and interval estimation, which we illustrate using M_a. Note that M_a is used to estimate parameters θ_1 and θ_2. For point estimation, θ_1 is estimated as the numerical value $N_{1,2}/(N_{1,2}+N_{1,G})$ and θ_2 is estimated as $N_{2,3}/(N_{2,3}+N_{2,G})$. For interval estimation, IDMS assumes that θ_1 (resp., θ_2) is the mean of a Bernoulli distribution and $(N_{1,2}, N_{1,G})$ (resp., $(N_{2,3}, N_{2,G})$) forms a random sample of the distribution. In other words, $(N_{1,2}, N_{1,G})$ denote a random sample containing $N_{1,2}$ copies of 1 and $N_{1,G}$ copies of 0, and $(N_{2,3}, N_{2,G})$ has a similar meaning. Therefore, one can employ the standard statistical inference method to derive a confidence interval for θ_1 and one for θ_2. By the laws of large numbers, if $N_{1,2} + N_{1,G}$ (resp., $N_{2,3} + N_{2,G}$) increases then the width of the resulted confidence interval for θ_1 (resp., θ_2) likely decreases (when the confidence level is fixed).

3.3 Confident Optimality

By instantiating the transition probability parameters in the parametric MDP with the corresponding point estimates and interval estimates, one obtains a concrete MDP $\mathcal{M}$ and an IMDP $\mathcal{M}^I$. Note that if $[p, q] \subset [0, 1]$ instantiates a parameter θ then, equivalently, $[1 - q, 1 - p]$ instantiates $1 - \theta$. Clearly, $\mathcal{M}$ and $\mathcal{M}^I$ share the same state space S, initial distribution α and cost function C. Moreover, $\mathcal{M}$ is an instance of $\mathcal{M}^I$, namely, $\mathcal{M} \in [\![\mathcal{M}^I]\!]$. From now on, for given $\mathcal{M}$ and $\mathcal{M}^I$, we always assume $\mathcal{M} \in [\![\mathcal{M}^I]\!]$. A key decision-making criterion in IDMS is formalized as follows:

Definition 3 (Confident Optimality). *Given $\mathcal{M}$, $\mathcal{M}^I$, $G \subseteq S$ of goal states and a finite nonempty subset Σ of schedulers, $\sigma^* \in \Sigma$ is* confidently optimal *if, for all $\sigma \in \Sigma \backslash \sigma^*$, the following two conditions hold:*

$$\begin{aligned} C_{\sigma^*} &\leq C_\sigma \;\; \textit{and} \\ C^{\max}_{\sigma^*} &\leq C^{\min}_{\sigma} + \gamma \cdot C^{\mathrm{dif}}_{\sigma^*} \quad \textit{where } \gamma \geq 0. \end{aligned} \tag{3}$$

In words, a scheduler σ^* in the given scheduler subset Σ of $\mathcal{M}$ (or, equivalently, $\mathcal{M}^I$) is confidently optimal if for *all other* schedulers σ in Σ (i.e., $\sigma \neq \sigma^*$):

- The cumulative cost (of reaching G-states) in $\mathcal{M}$ under σ^* is not larger than the cumulative cost in $\mathcal{M}$ under σ;
- The $(1/\gamma)$-portion of the difference between the maximum cumulative cost in $\mathcal{M}^I$ under σ^* and the minimum cumulative cost in $\mathcal{M}^I$ under σ is not larger than the maximum-minimum difference of cumulative cost in $\mathcal{M}^I$ under σ^*.

A correct illustrative example is presented in the latter text. It is noteworthy that, different from an standard MDP problem, a subset of schedulers is explicitly given in our definition.

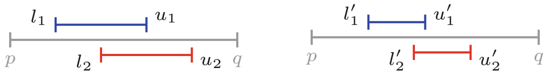
The parameter γ, which is specified by the user, has the function of adjusting the criterion of confident optimality. A confidently optimal scheduler may not exist for the given MDP and IMDP; in some rare case, there may be more than one confidently optimal schedulers. Note that if a sufficiently large value for γ is selected, then the second condition in Eq. (3) is guaranteed to be true. If so, the definition is degenerated to the standard definition of optimal cumulative costs for MDPs with point-valued transition probabilities.

Given $\mathcal{M}, \mathcal{M}^I, G, \Sigma, \gamma$, the following procedure decides whether a confidently optimal scheduler σ^* exists and returns σ^* if it exists:

1. Compute C_σ for all $\sigma \in \Sigma$, and compute $\Sigma_1 \subseteq \Sigma$ such that $C_{\sigma_1} = \min_{\sigma \in \Sigma} C_\sigma$ if and only if $\sigma_1 \in \Sigma_1$.
2. Compute $C^{\max}_{\sigma_1}$ for all $\sigma_1 \in \Sigma_1$, and compute $C^{\min}_{\sigma}$ for all $\sigma \in \Sigma$.
3. If there is $\sigma^* \in \Sigma_1$ such that $C^{\max}_{\sigma^*} \leq C^{\min}_{\sigma} + \gamma \cdot C^{\mathrm{dif}}_{\sigma^*}$ where $\sigma \neq \sigma^*$, then return σ^*; otherwise, return "no confidently optimal scheduler".

The procedure relies on the core method of value-iteration presented in Sect. 2. The computational complexity of is dependent on the core value-iteration method and the size of Σ. Note that although the number of *all* schedulers in an MDP increases exponentially as the size of the MDP increases, in our case a *specific* subset of schedulers Σ is predefined by the model builder. If we suppose the value-iteration takes constant time (e.g., the model is fixed), then the time complexity of the procedure is linear in the size of Σ.

We present an example to explain how IDMS is affected by γ and the sample size. Suppose after instantiating $\boldsymbol{\theta}$ of $\mathcal{M}_{eg}(\boldsymbol{\theta})$ with point estimates and interval estimates, the cumulative cost intervals for schedulers σ_a and σ_b are $[l_1, u_1]$ and $[l_2, u_2]$, respectively. The positions of l_1, u_1, l_2 and u_2 are illustrated on the left side of the following drawing (where $0 \leq p < q$).



If $u_1 \leq l_2 + \gamma(u_1 - l_1)$, the above procedure returns σ_a. But if $u_1 > l_2 + \gamma(u_1 - l_1)$, neither σ_a nor σ_b is confidently optimal and so the procedure returns "no confidently optimal scheduler". If one lowers the value γ and/or increases the sample size, the computed cost intervals usually shrink, as depicted on the right side of the above drawing. Then there is a higher probability that a confidently optimal scheduler (namely σ_a) is returned from the procedure and the iteration of IDMS terminates.

3.4 Metrics and Tradeoff

One main advantage of IDMS is the flexibility that enables a tradeoff between the three important metrics for practical, especially runtime, decision-making. The three metrics are accuracy of the decision, data usage for making the decision and computational overhead on the runtime system. Because random sampling is involved in IDMS, under a specific scheduler of an MDP and an IMDP, the cumulative cost and the minimum/maximum cumulative costs (of reaching the goal states) are uncertain. Therefore, a confidently optimal scheduler may be decided at each iterative step with a certain probability. Further, a confidently optimal scheduler may not be the truly optimal one, which is defined based on the unknown real values of the transition probability parameters in the abstract MDP. In view of this, we define the three metrics as follows:

- Accuracy is the probability that a confidently optimal scheduler is optimal.
- Data usage is the average size of sampled data used in the iteration.
- Computational overhead is measured by the average iteration time (namely, the average number of iterative steps).

Ideally, one wants to maximize the first one while minimize the latter two. However, according to laws of statistics this is impossible. To obtain high accuracy in a statistical process (including IDMS), a large-sized sample has to be used; although it is possible to set a high accuracy threshold and then try to infer the result using a sample whose size is as small as possible, this usually leads to a costly iterative process. Therefore, a practical solution is to achieve a suitable tradeoff between the three metrics. In IDMS, to realize this tradeoff, one can adjust the constant γ and the sample size within the iteration.

4 Application to Self-adaptive System

In this section, we describe an application of IDMS to self-adaptive systems. A variety of frameworks are proposed to aid the design of self-adaptive systems [12,13,25] and we focus on the Rainbow framework.

4.1 Rainbow Framework

We illustrate Rainbow with the example Z.com [11] which is a fictional news website providing multi-media and textual news service while keeping the cost of maintaining its server pool within its operational budget. Z.com has a client-server architecture with three additional adaptation-relevant components, as shown in Fig. 3: The Sensor collects runtime data; the Manager controls the adaptation, such as switching the news content mode from multi-media to text and *vice versa*; and the Effector executes the adaptation to affect the system.

In Rainbow, the adaptation is specified as *strategies* in its customized language Stitch [9]. A strategy is a tree structure consisting of *tactics*, which in turn contain operations. Figure 4 specifies two strategies a and b, guarded by a common condition cond where SNo and MaxSNo refer to the current server number and the maximal server number, respectively.[1] If strategy a is selected, operation enlistSever[1] in tactic s1 is first executed. Next, if the variable hiLatency is true then enlistSever[1] in tactic s2 is executed; otherwise strategy a terminates. Last,

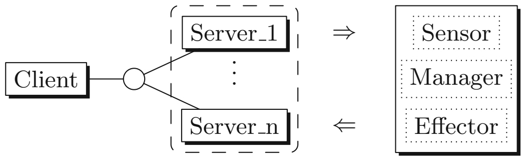


Fig. 3. Software architecture of Z.com

```
define cond:=hiLatency&!TextMode&(SNo<=MaxSNo-2);
strategy a[cond]{
 tactic s1:enlistServer[1]{
   tactic s2:hiLatency->enlistServer[1]{
     tactic s3:hiLatency->switchToTextMode;}}}
strategy b[cond]{
 tactic s4:hiLoad->enlistServer[2]{
   tactic s6:hiLatency->switchToTextMode;}
 tactic s5:!hiLoad->switchToTextMode{
   tactic s7:hiLatency->enlistServer[2];}}
... % other strategy specification
```

Fig. 4. Strategy specification for Z.com in Stitch

[1] For simplicity, the specification does not strictly follow the syntax of Stitch.

Table 1. Costs of operations in strategies a and b

Utility Dimension	Operation						
	op(s1)	op(s2)	op(s3)	op(s4)	op(s5)	op(s6)	op(s7)
Content	0	0	1	0	1	1	0
Budget	1	1	0	2	0	0	2

if hiLatency persists to be true then switchToTextMode in tactic s3 is executed; otherwise strategy a terminates. Strategy b is specified in a similar style.

To evaluate strategies, Rainbow uses *utilities* to describe the costs and benefits of operations. The quantities of utilities are provided by human experts or stakeholders. Table 1 describes two utilities called *content* and *budget* and the costs of the operations in terms of the two. Note that because there is only one operation in each tactic of the adaptation specification in Fig. 4, we use tactic names to label operations—the correspondent operation to an tactic s is denoted op(s). For example, if switchToTextMode is executed, then the content cost is, say, 1; if enlistServer[i] with $i \in \{1, 2\}$ is executed, then the budget cost is, say, i. Then, the overall cost of an operation is the weighted sum of utilities. For simplicity, we let the weights of all utilities equal to 1.

Rainbow characterizes uncertainty in the detection of guarding conditions (such as hiLantency in tactic s2) as probabilities called *likelihoods*. The likelihoods in strategies a and b are specified in Table 2. Note that because there is one likelihood parameter in each tactic (except s1) of in Fig. 4, like for operations, we also use tactic names to label likelihoods—the correspondent operation to an tactic s is denoted lk(s). We explain how these likelihoods are elicited in Rainbow later; for now, they are viewed as undetermined parameters.

It is not hard to observe a correspondence between the adaptation specification of Z.com and an MPD model, where operations are represented by actions and strategies are represented by schedulers. Indeed, the Stitch specification under consideration can be translated into $\mathcal{M}_{eg}(\boldsymbol{\theta})$. Therefore, the adaptation problem in Rainbow is an instance of the problem of selecting a strategy that minimizes the cumulative cost (of reaching the goal states in the MDP).

4.2 Embedding IDMS into Rainbow

At least two methods to elicit likelihoods are supported in Rainbow. First, like utilities and their weights, concrete values of likelihoods can be explicitly given by human experts or stakeholders [9]. Second, sampling methods for estimating likelihoods are also implemented in Rainbow [7,10]. For example, the Manager can check the values of Boolean variables hiLatency and hiLoad as the system operates and record the result. Then, with respect to the condition probabilities described in Table 2, one easily obtains a sample for each parameter θ_i. Therefore, we can embed IDMS into Rainbow economically, just by enhancing the reasoning mechanism of strategy selection in the Manager with IDMS, but with little change made to the Sensor and the Effector

Table 2. Likelihood parameters in strategies a and b

Likelihood	Interpretation as a conditional probability
lk(s2)	Pr(hiLatency=true | SNo=MaxSNo - 1 & textMode=true)
lk(s3)	Pr(hiLatency=true | SNo=MaxSNo & textMode=true)
lk(s4)	Pr(hiLoad=true | hiLatency=true & SNo=MaxSNo - 2 & textMode=true)
lk(s4)	Pr(hiLoad=false | hiLatency=true & SNo=MaxSNo - 2 & textMode=true)
lk(s6)	Pr(hiLatency=true | hiLoad=true & SNo=MaxSNo & textMode=true)
lk(s7)	Pr(hiLatency=true | hiLoad=true & SNo=MaxSNo - 2 & textMode=false)

Rainbow exploits point estimates for likelihoods, as its decision model is a standard MDP. Because the runtime data set cannot be arbitrarily large, point estimates may be error-prone. Poor strategy selection often causes some extra cost and reduced benefit. Even worse, the extra cost and reduced benefit may accumulate if the non-optimal strategy is selected repeatedly. In view of this, the interval estimation method in IDMS can complement to the point estimation method in Rainbow, and leads to more stable decision-making outputs. By applying IDMS to Rainbow, another and more important benefit is the possibility of making a tradeoff between accuracy, data usage and computational overhead, thus improving the adaptation function of Rainbow.

5 Simulation-Based Experiment

5.1 Methodology and Setting

The general experimental methodology we adopt is simulation. Recall that IDMS assumes that likelihood parameters in Z.com are means of Bernoulli distributions. We use Matlab to simulate the generation and collection of runtime data. To this end, we need to fix the expected values of the Bernoulli random variables, namely the true values of $\boldsymbol{\theta}$ of $\mathcal{M}_{eg}(\boldsymbol{\theta})$. We let $\theta_1 = \frac{2}{3}$, $\theta_2 = \frac{4}{7}$, $\theta_3 = \frac{1}{3}$, $\theta_4 = \frac{4}{9}$ and $\theta_5 = \frac{4}{9}$. As the true values of $\boldsymbol{\theta}$ are given, we also know which scheduler is optimal. Indeed, by computation, the overall cost of strategy a is 2.0476 and that of strategy b is 2.0741. Thus, strategy a is optimal. It is noteworthy that the difference between the above two overall costs may seem small, but it is non-negligible because they are proportional to the weights of utility dimensions, which may be large in some case, and also because the extra cost may accumulate if the adaptation is triggered repeatedly.

To evaluate the flexibility of IDMS for making the intended tradeoff, we implement the computing procedure presented in Sect. 3.3 in Matlab. Given a sample of specific size for estimating each parameter θ_i of $\boldsymbol{\theta}$, and given a specific value of γ, IDMS terminates with a certain probability, called *termination probability* in the experiment. Based on the termination probability, we can immediately calculate the data usage and the computational overhead. Upon termination, with a certain probability, the selected scheduler is strategy a. This probability, called *correctness rate* in the experiment, is equal to the metric of accuracy. Since we can simulate IDMS (applied to $\mathcal{M}_{eg}(\boldsymbol{\theta})$), we can estimate the correctness rate and termination probability using the standard Monte Carlo estimation. In this experiment, we estimate the two for *different* sample sizes and values of γ. Note that the confidence level of interval estimation is fixed in IDMS and we set it to be 95 % in the experiment. The Matlab source code and data are available on http://www.comp.nus.edu.sg/~sugx/fase16/.

5.2 Experimental Data and Concrete Tradeoffs

The experimental data, summarized in Fig. 5, is generated from samples of n-size with n ranging from 200 to 5,000 in an increment of 200, and with a selection of values for γ as specified in the legends of the figures (where "large" refers to a sufficiently large value of γ such that the computing procedure is degenerated to a point estimation). For each n, the number of generalized samples is 10,000, based on which we calculate the correctness rate and termination probability.

Figure 5 demonstrates the dependence of the correctness rate and termination probability on γ and the sample size. Figure 5(a) shows that as γ decreases or as the sample size increases, the correctness rate increases. In particular, except for samples of small size (less than 1,000), IDMS provides a higher correctness rate than the point estimation method. Figure 5(b) shows that as γ increases or as the sample size increases, the termination probability increases. Note that if a sufficiently large value for γ is selected, the termination probability is 1 for samples of all selected sizes (and thus L8 is not depicted in Fig. 5(b).)

An important implication of Fig. 5 is that, by adjusting the value of γ and the sample size in different ways, one is able to achieve different tradeoffs between accuracy, data usage and computational overhead. To illustrate this flexibility, Table 3 describes three cases where the three metrics have different priorities. Based on Fig. 5, by selecting different pairs of γ and sample size, we obtain three

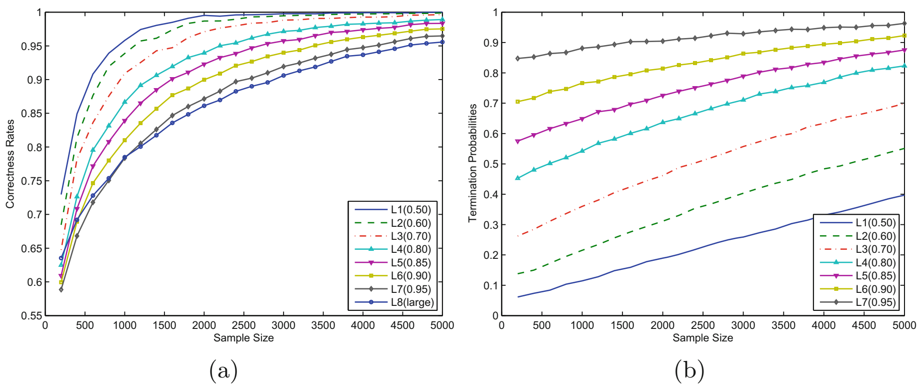
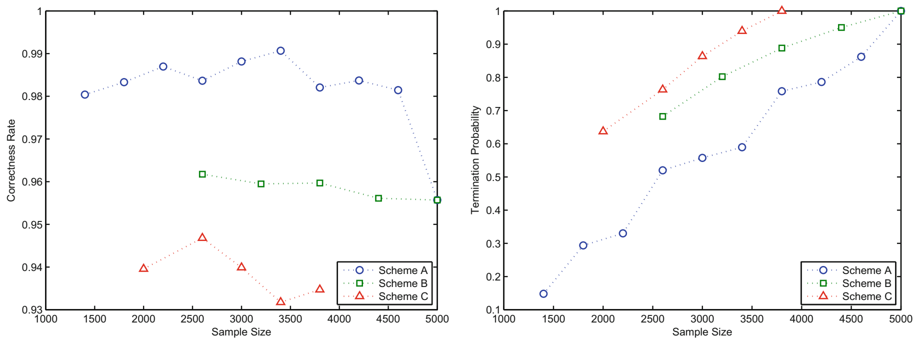


Fig. 5. (a) Correctness rates and (b) termination probabilities with different sample sizes and γ values

Table 3. Priorities of metrics in three different cases

Metric	Priority		
	A	B	C
Accuracy	high	medium	low
Data usage	low	medium	high
Computational overhead	low	high	high



(a) (b)

Fig. 6. Three iteration schemes in items of (a) correctness rates and (b) termination probabilities

examples of iteration schemes depicted in Fig. 6. Each marker in Fig. 6 refers to an iterative step with a specific value of γ and a specific sample size. For example, setting $\gamma = 0.5$ and the sample size as 1,400, according to Fig. 5, we obtain the leftmost marker of Scheme A in Fig. 6. The other markers in Fig. 6 are identified in the same way. All three schemes terminate with probability 1 before or when the sample size reaches 5,000. It is easy to observe that the schemes reflect the metric priorities in the corresponding cases in Table 3. For example, Scheme A has a high correctness rate compared with the other two schemes, because the priority of accuracy is high in Case A; it has a low average termination probability and a high number of markers, because the priorities of both data usage and computational overhead are low in Case A.

6 Related Work

Probabilistic model checking is a relatively mature technique that has been successfully applied to a wide range of domains, and we refer the readers to Forejt *et al.* [19] for a survey. The IMDP model considered in this paper falls into the class of probabilistic models with uncertainty, which have received substantial attention. For instance, in AI research, IMDPs were considered with different objectives such as discounted sum and limiting average [21,30]. The motivation of those works is to come up with an abstract framework, which is different from our motivation of runtime decision-making. In robust control theory, IMDPs or MDPs with more general forms of uncertainty are advocated to address the robustness of the controller under potential perturbation of the system [24,32]. In this paper, we consider expected cumulative costs of reachability properties. On the computational aspect, some of the mentioned approaches [21,24,30] also employed a value-iteration method. However, they mostly rely on ordering between intervals which is not needed in our case. Furthermore, Puggelli *et al.* [26] proposed polynomial algorithms for Markov chains with uncertainty based on optimization techniques, but only for reachability and PCTL properties.

Several high-level frameworks and approaches based on probabilistic model checking have been proposed for self-adaptive systems recently, but with emphasis on different aspects of the adaptation, such as QoS management and optimization [4], adaptation decisions [20], verification with information of confidence intervals [3], runtime verification efficiency and sensitivity analysis [18], and proactive verification and adaptation latency [23]. None of those works addressed the problem of making a practical tradeoff similar to the one supported by IDMS. Rainbow [9] supports the computation of cumulative costs and/or rewards when the likelihood parameters in the adaptation strategies are explicitly specified. Subsequent work [5,6] employs a combination of a simulation method and probabilistic model checking to evaluate properties such as resilience and adaptation latency. As mentioned, our IDMS can be economically embedded into Rainbow and extend the adaptation function of the latter.

We mention some other existing approaches to the design of self-adaptive systems, which rely on mathematical methods related to probability theory and statistics. Esfahani *et al.* [15,16] presented a general definition of adaptation optimality using fuzzy mathematics, which accounts for not only the current utility but also the optimal consequence of future operations. But IDMS estimates the probability parameters based on runtime data. Epifani *et al.* [14] presented the KAMI framework to deal with the inaccuracy of parameters related to the non-functional aspect of the system (such as reliability and performance), and Bencomo *et al.* [2] presented a Bayesian network for modeling self-adaptive systems. These two approaches rely on the Bayesian (point) estimation method while IDMS exploits both point and interval estimates from the frequentist statistics theory. Finally, Filieri *et al.* [17] constructed approximate dynamic models of a self-adaptive system and for synthesizing, from those models, a suitable controller that guarantees prescribed multiple non-functional system requirements. The method they used is from control theory, which is quite different from the theory of MDPs.

7 Conclusions

We have presented IDMS, an iterative framework that supports a tradeoff among three important metrics in practical runtime decision-making problems: accuracy, data usage and computational overhead. We have also instantiated IDMS on the Rainbow framework and presented a simulation-based evaluation.

For future work, we plan to enhance IDMS with a mechanism for automatically adjusting the confident optimality and the sample size based on the given priorities of the three metrics. Another interesting topic is a generalization of the value-iteration method in IMDP to synthesize a scheduler that minimizes the cumulative cost, without prescribing a subset of schedulers.

References

1. Baier, C., Katoen, J.-P.: Principles of Model Checking. The MIT Press, Cambridge (2008)
2. Bencomo, N., Belaggoun, A., Issarny, V.: Dynamic decision networks for decision-making in self-adaptive systems: A case study. In: Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2013, pp. 113–122. IEEE Press, Piscataway, NJ, USA (2013)
3. Calinescu, R., Ghezzi, C., Johnson, K., Pezzé, M., Rafiq, Y., Tamburrelli, G.: Formal verification with confidence intervals: A new approach to establishing the quality-of-service properties of software systems. IEEE Trans. Reliab. **99**, 1–19 (2015)
4. Calinescu, R., Grunske, L., Kwiatkowska, M., Mirandola, R., Tamburrelli, G.: Dynamic QoS management and optimization in service-based systems. IEEE Trans. Softw. Eng. **37**(3), 387–409 (2011)
5. Camára, J., de Lemos, R.: Evaluation of resilience in self-adaptive systems using probabilistic model-checking. In: 2012 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pp. 53–62, June 2012
6. Cámara, J., Moreno, G.A., Garlan, D.: Stochastic game analysis and latency awareness for proactive self-adaptation. In: Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS, pp. 155–164. ACM, New York, NY, USA (2014)
7. Celiku, O., Garlan, D., Schmerl, B.: Augmenting architectural modeling to cope with uncertainty. In: Proceedings of the International Workshop on Living with Uncertainty (IWLU 2007), Atlanta, Georgia, USA (2007)
8. Chen, T., Forejt, V., Kwiatkowska, M.Z., Parker, D., Simaitis, A.: Automatic verification of competitive stochastic systems. Formal Method Syst. Des. **43**(1), 61–92 (2013)
9. Cheng, S.-W.: Rainbow: Cost-Effective Software Architecture-based Self Adaptation. Ph.D. thesis, Carnegie Mellon University (2008)
10. Cheng, S.-W., Garlan, D.: Handling uncertainty in autonomic systems. In: Proceedings of the International Workshop on Living with Uncertainty (IWLU 2007), Atlanta, Georgia, USA (2007)
11. Cheng, S.-W., Garlan, D., Schmerl, B.: Architecture-based self-adaptation in the presence of multiple objectives. In: ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2006), Shanghai, China (2006)
12. Cooray, D., Malek, S., Roshandel, R., Kilgore, D.: RESISTing reliability degradation through proactive reconfiguration. In: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, ASE 2010, pp. 83–92. ACM, New York, NY, USA (2010)
13. Elkhodary, A., Esfahani, N., Malek, S.: FUSION: A framework for engineering self-tuning self-adaptive software systems. In: Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2010, pp. 7–16. ACM, New York, NY, USA (2010)

14. Epifani, I., Ghezzi, C., Mirandola, R., Tamburrelli, G.: Model evolution by run-time parameter adaptation. In: Proceedings of the 31st International Conference on Software Engineering, ICSE 2009, pp. 111–121. IEEE Computer Society, Washington, DC, USA (2009)
15. Esfahani, N., Kouroshfar, E., Malek, S.: Taming uncertainty in self-adaptive software. In: Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, ESEC/FSE 2011, pp. 234–244. ACM, New York, NY, USA (2011)
16. Esfahani, N., Malek, S.: Uncertainty in self-adaptive software systems. In: de Lemos, R., Giese, H., Müller, H.A., Shaw, M. (eds.) Software Engineering for Self-Adaptive Systems. LNCS, vol. 7475, pp. 214–238. Springer, Heidelberg (2013)
17. Filieri, A., Hoffmann, H., Maggio, M.: Automated multi-objective control for self-adaptive software design. In: Proceedings of the 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, pp. 13–24 (2015)
18. Filieri, A., Tamburrelli, G., Ghezzi, C.: Supporting self-adaptation via quantitative verification and sensitivity analysis at run time. IEEE Trans. Softw. Eng. **42**, 75–99 (2015)
19. Forejt, V., Kwiatkowska, M., Norman, G., Parker, D.: Automated verification techniques for probabilistic systems. In: Bernardo, M., Issarny, V. (eds.) SFM 2011. LNCS, vol. 6659, pp. 53–113. Springer, Heidelberg (2011)
20. Ghezzi, C., Pinto, L.S., Spoletini, P., Tamburrelli, G.: Managing non-functional uncertainty via model-driven adaptivity. In: Proceedings of the International Conference on Software Engineering, ICSE 2013, pp. 33–42. IEEE Press (2013)
21. Givan, R., Leach, S.M., Dean, T.L.: Bounded-parameter Markov Decision Processes. J. Artif. Intell. **122**(1–2), 71–109 (2000)
22. Huebscher, M.C., McCann, J.A.: A survey of autonomic computing-degrees, models, and applications. ACM Comput. Surv. **40**(3), 7: 1–7: 28 (2008)
23. Moreno, G.A., Cámara, J., Garlan, D., Schmerl, B.: Proactive self-adaptation under uncertainty: A probabilistic model checking approach. In: Proceedings of the 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, pp. 1–12. ACM, New York, NY, USA (2015)
24. Nilim, A., Ghaoui, L.E.: Robust control of Markov Decision Processes with uncertain transition matrices. Oper. Res. **53**(5), 780–798 (2005)
25. Poladian, V., Garlan, D., Shaw, M., Satyanarayanan, M., Schmerl, B., Sousat, J.: Leveraging resource prediction for anticipatory dynamic configuration. In: First International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2007, pp. 214–223 (2007)
26. Puggelli, A., Li, W., Sangiovanni-Vincentelli, A.L., Seshia, S.A.: Polynomial-time verification of PCTL properties of MDPs with convex uncertainties. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 527–542. Springer, Heidelberg (2013)
27. Puterman, M.L.: Markov decision processes. Handb. Oper. Res. Manage. Sci. **2**, 331–434 (1990)
28. Su, G., Feng, Y., Chen, T., Rosenblum, D.S.: Asymptotic perturbation bounds for probabilistic model checking with empirically determined probability parameters. IEEE Trans. Softw. Eng. **99**, 1–19 (2015)
29. Su, G., Rosenblum, D.S., Tamburrelli, G.: Reliability of run-time quality-of-service evaluation using parametirc model checking. In: Proceedings of the 38th International Conference on Software Engineering, ICSE 2016. ACM, New York, NY, USA (2016)

30. Tewari, A., Bartlett, P.L.: Bounded parameter markov decision processes with average reward criterion. In: Bshouty, N.H., Gentile, C. (eds.) COLT. LNCS (LNAI), vol. 4539, pp. 263–277. Springer, Heidelberg (2007)
31. Varga, R.S.: Matrix Iterative Analysis. Springer Series in Computational Mathematics. Springer, Heidelberg (2009)
32. Wiesemann, W., Kuhn, D., Rustem, B.: Robust markov decision processes. Math. Oper. Res. **38**(1), 153–183 (2013)