Model Transformations Given Policy Modifications in Autonomic Management

Raphael M. Bahati and Michael A. Bauer
Department of Computer Science
The University of Western Ontario, London, ON N6A 5B7, CANADA
Email: {rbahati;bauer}@csd.uwo.ca

***Abstract*—This paper presents an approach for adapting a model learned from the use of an active set of policies to run-time policy modifications. The work leverages our most resent efforts utilizing Reinforcement Learning methodologies to facilitate dynamic use of policies within autonomic computing. The use of policies in this context offers significant benefits to autonomic systems in that it allows systems to determine how to effectively meet the desired, and at times seemingly conflicting, objectives under dynamically changing conditions. Contrary to other approaches that make use of some form of learning to model performance management, our approach enables the model learnt from the use of an active set of policies to be reused once those policies change. Since the learning mechanisms are modelled from the structure of the policies, policy modifications can be mapped onto the learnt model. Our analysis of the policy modifications suggest that most of the learned model could be reused, potentially accelerating the learning process. In this paper, we provide formal definitions on the different kinds of policy modifications that might occur as well as elaborate, with detailed examples, on how such modifications could impact the currently learned model.**

***Index Terms*—Model Transformation, Model Adaptation, Reinforcement Learning, Autonomic Management, Policy-based Management.**

I. Introduction

Policy-based management has been proposed as a way in which the behavior of systems and applications can be adjusted at run-time rather than through re-engineering [1]. The concept of policy-based management has been adopted by standard bodies such as the (Internet Engineering Task Force (IETF) [2], the Distributed Management Task Force (DMTF) [3], and the Object Management Group (OMG) [4]. Policy-based management has also attracted significant interest from a wide variety of efforts, ranging from academia [5] to industry [6], which have resulted in a great deal of diversity in terms of what policies are and how they should be used.

It is often common that policies are used to express required or desired behavior of systems and applications. In the context of autonomic computing, for example, policies can be input to or embedded within the autonomic management elements of the system to provide the kinds of directives which an autonomic manager could make use of in order to meet operational requirements. Thus, through the modifications of the policies driving autonomic management, the behavior of systems could be dynamically adjusted. Such directives could come from the users of such systems or other autonomic management elements within the computing environment.

While the use of policies has been the subject of considerable research focus, very little work has been done on evaluating the effectiveness of the policies when in use to determine whether the performance objectives are, in fact, being met. More often than not, policy use within autonomic computing has typically focused on the specification and use "as is" within systems. It is often assumed, for example, that once the policies are specified, the system would behave as expected. Consequently, many of these approaches fail to recognize the stochastic, dynamic, and heterogeneous nature of today's systems which necessitates the need for autonomic systems to adapt not only to how they use policies, but also to run-time policy modifications. This is essential in order for systems to cope with not only changes to the configuration of the managed environment, but also changes to the perceived quality of services, something that is often time-, user-dependent, and application-specific.

To this end, we have proposed an adaptive policy-driven autonomic management framework highlighting two key contributions in the use of policies within autonomic computing: (1) The ability to evaluate, through model-learning, the effectiveness of the enabled policies based on past experience from their use which, in turn, provide guidance to the autonomic system on how to effectively use the policies [7]. (2) The ability to adapt the model learned from the use of policies to run-time policy modifications, essentially allowing most of the learned model to be reused in the new environment [8]. The focus of this work is on the second of these approaches. It expands on the strategies we initially proposed in [8] by formally describing the types of changes to a set of policies that could occur and how we map such changes to model transformations. This is done within the context of the adaptive policy-driven autonomic management framework in [9].

The rest of the paper is organized as follows. We begin with an overview of how we model learning in Section II, summarizing some of the key definitions we make use of in this paper. We then present our approach to model adaptation in Section III. The section begins with formal definitions of the types of changes that might occur to a set of policies, and then describe how such modifications are mapped to specific cases of model transformations. We summarize the results of the experiments in Section IV, describe some related work in Section V, and conclude in Section VI.

II. Learning by Reinforcement

This section summarizes key definitions outlining our approach to modelling Reinforcement Learning [10] in the context of policy-driven autonomic management. A more detailed account can be found in [7].

Definition 1: An expectation policy is defined by the tuple $p_i = \langle C, A \rangle$ where:

- C is conjunctive conditions associated with policy p_i with each condition, $c_j \in C$, defined by the tuple $c_j = \langle$ `ID, metricName, operator,` $\Gamma\rangle$, where `ID` is a unique identification for the condition; `metricName` is the name of the metric associated with the condition; `operator` is the relational operator associated with the condition; and Γ is the threshold of the condition.
- A is a set of actions associated with policy p_i with each action, $a_j \in A$, defined by the tuple $a_j = \langle$ `ID, function, parameters,` $\tau\rangle$, where `ID` is a unique identification for the action; `function` is the name of the function that should be executed; `parameters` is a set of function parameters; and τ is a set of tests associated with the action.

A policy-driven autonomic management system is likely to consist of multiple expectation policies, a subset of which may be active (or enabled) at any given time, which brings us to our next definition.

Definition 2: Suppose that P^A denotes a set of all expectation policies such that $p_i \in P^A$ where $p_i = \langle C, A \rangle$. Let P be a subset of expectation policies an autonomic manager uses to make management decisions; i.e., $P \subseteq P^A$. A policy system corresponding to P is defined by the tuple $PS = \langle P, W_C \rangle$ where:

- $W_C = \langle c_i, \omega_i \rangle$ associates each policy condition, c_i, with a metric weight, ω_i, such that, for all $c_i \in p_m$ and $c_j \in p_n$, $\omega_i = \omega_j$ if c_i.`metricName` $= c_j$.`metricName`.

Essentially, a *policy system* is the set of active policies where each condition occurring in some policy has an associated weight. The metrics weights, which are specified manually in our current implementation, provide a way of distinguishing policy conditions based on the significance of violating a particular metric. In essence, W_C provides a way of biasing how the autonomic system responds to quality of service violations. These definitions provide the fundamental structure that is used to build our reinforcement learning model.

A. System States

To model system's dynamics from the use of an active set of policies, we make use of a mapping between the enabled expectation policies and the managed system's states whose structure is derived from the metrics associated with the enabled policy conditions.

Definition 3: A policy system $PS = \langle P, W_C \rangle$ derives a set of system metrics, $m_i \in M$, such that, for each $p_j = \langle C_j, A_j \rangle$ where $p_j \in P$, $M = \bigcup_{c_i \in C_j} \{c_i.\texttt{metricName}\}$.

The set M is the set of all metrics occurring in any of the active policies. For each metric in this set, there are a finite number of threshold values to which the metric is compared; there can be ordered to form regions:

Definition 4: A policy system $PS = \langle P, W_C \rangle$ with metrics set M derives a set of metric regions, $r_{m_i} \in M_R^P$, for each metric $m_i \in M$, whose structure is defined by the tuple $r_{m_i} = \langle \alpha_{m_i}, \sigma_{m_i} \rangle$, where:

- $\alpha_{m_i} = \langle$ `ID, metricName,` $\omega\rangle$ corresponds to a unique metric from among the metrics of the conditions of the policies in P; such that, `metricName` is the name of the metric and ω is the weight of the condition (see Definition 2) associated with metric m_i.
- $\sigma_{m_i} = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_k\}$ is a set of thresholds from the conditions associated with metric m_i such that, $\Gamma_i < \Gamma_j$ if $i < j$. As such, σ_{m_i} derives a set of metric regions which map the observed metric measurement onto appropriate localities (i.e., intervals) as defined by the thresholds of the policy conditions associated with metric m_i, such that $R_{m_i} = \{R_{m_i}^1, R_{m_i}^2, \ldots, R_{m_i}^{k+1}\}$, where $R_{m_i}^1 = (-\infty, \Gamma_1)$, $R_{m_i}^2 = (\Gamma_1, \Gamma_2)$, etc., and $R_{m_i}^{k+1} = (\Gamma_k, \infty)$.

To be precise, the actual boundaries of region $R_{m_i}^j$ are determined by the operators of the policy conditions associated with a given metric. For example, a system with two conditions $c_1 = \langle$ `ID,` m_i`, >,` $\Gamma_1\rangle$ and $c_2 = \langle$ `ID,` m_i`,` $\geq$`,` $\Gamma_2\rangle$ such that $\sigma_{m_i} = \langle \Gamma_1, \Gamma_2 \rangle$ would yield three regions in our approach, namely: $R_{m_i}^1 = [-\infty, \Gamma_1)$, $R_{m_i}^2 = [\Gamma_1, \Gamma_2)$, and $R_{m_i}^3 = [\Gamma_2, \infty]$. Thus, a metric measurement of, say, Γ_1 would fall into region $R_{m_i}^2$. This brings us to our next definition.

Definition 5: Given a set of metric-regions $r_{m_i} \in M_R^P$ for each metric $m_i \in M$ such that $r_{m_i} = \langle \alpha_{m_i}, \sigma_{m_i} \rangle$, where σ_{m_i} derives a set of metric regions $R_{m_i}^j \in R_{m_i}$; we define a weighting function, $f(R_{m_i}^j) \rightarrow \mathbb{R}$, which assigns a numeric value to the j-th region in R_{m_i} such that, $f(R_{m_i}^k) > f(R_{m_i}^l)$ if $k < l$.

An example of such a mapping, which we make use of in our current implementation, is defined by Equation 1:

$$f(R_{m_i}^j) \;=\; 100 - (\frac{100}{n-1})(j-1) \qquad (1)$$

where n is the total number of regions in R_{m_i}. This function assigns a numeric value between 100 and 0 for each metric's region in R_{m_i}, starting from 100 for the most desirable region and decrementing at equal intervals towards the opposite end of the spectrum, whose region is assigned a value of 0. This approach guarantees that the highest value is assigned to the most desirable region (i.e., the region corresponding to the highest quality of service), assuming, of course, that the assumptions about the conditions of the expectation policies hold (see Definition 1). That is, smaller metric values are "more desirable" though, in general, this is not a necessary requirement of the weighting function. The idea is that, regions of greater "desirability", i.e., preferred quality of service, are assigned higher values. The key role of these regions is that they partition the space of values that a metric can take on with respect to the thresholds in conditions involving that metric. We use these to define a state within our model.

Definition 6: A policy system $PS = \langle P, W_C \rangle$ with metrics M and metrics-regions M_R^P derives a set of system states

S such that each state $s_i \in S$ is defined by the tuple $s_i = \langle \mu, M(s_i), A(s_i) \rangle$, and where:

- μ is a `type` which classifies a state as either "violation" or "non-violation" depending, respectively, on whether or not there are any policy violations as a result of visiting state s_i.
- $P(s_i)$ is a set of expectation policies that are violated when the system is in state s_i such that, $P(s_i) \in P$. As noted previously, a policy is said to be violated if all its conditions evaluate to true when matched against violation notifications received within a management cycle.
- $A(s_i)$ is a set of actions advocated by the expectation policies in $P(s_i)$ plus the γ-action; i.e., a_0 which corresponds to "do-nothing".
- $M(s_i)$ is a set of state metrics for each metric $m_j \in M$, $r_{m_j} \in M_R^P$, $r_{m_j} = \langle \alpha_{m_j}, \sigma_{m_j} \rangle$, such that each state metric $s_i.m_j \in M(s_i)$ is defined as follows:

Definition 7: A state metric $s_i.m_j \in M(s_i)$ given $\alpha_{m_j} = \langle \texttt{ID}, \texttt{metricName}, \omega \rangle$ and $\sigma_{m_j} = \langle \Gamma_1, \Gamma_2, \ldots, \Gamma_k \rangle$ is defined by the tuple $s_i.m_j = \langle \texttt{ID}, \omega, \texttt{value}, R^l_{m_j} \rangle$ where:

- `ID` is an integer value that uniquely identifies each metric $m_i \in M$.
- ω is the weight associated with metric m_i.
- `value` is the observed metric measurement, or average value when state s is visited multiple times.
- $R^l_{m_j}$ is the region corresponding to a region in σ_{m_j} in which the average metric measurement (i.e., `value`) falls; i.e., if $R^l_{m_j} = (\Gamma_1, \Gamma_2)$, then $\Gamma_1 < s_i.m_j.\texttt{value} < \Gamma_2$. Thus, for each such region, $f(R^l_{m_j})$ associates a value as described by Equation 1.

Using this approach, each state can be uniquely identified by the region occupied by each state metric based on the conditions of the expectation policies and the `value` associated with each metric. That is, for a set of policies involving n metrics, each state would have n metrics $\{m_1, m_2, ..., m_n\}$ each assigned a region whose intervals are derived from the thresholds of the conditions associated with the metric. We elaborate further on this in Section II-C.

B. System Transitions

Transitions are essentially determined by the actions taken by the management system and labelled by a value determined by the learning algorithm.

Definition 8: A state transition $t_i(s_p, a_p, s_c)$ is a directed edge corresponding to a transition originating from state s_p and ending on state s_c as a result of taking action a_p while in state s_p, and is labelled by $\langle \lambda, Q_{t_i}(s_p, a_p) \rangle$, where:

- λ is the frequency (i.e., the number of times) through which the transition occurs.
- $Q_{t_i}(s_p, a_p)$ is the action-value estimate associated with taking action a_p in state s_p. In our current implementation, $Q_{t_i}(s_p, a_p)$ is computed using a one-step Q-Learning [10], [11] algorithm.

It is worth pointing out that, non-deterministic transitions are also possible. That is, taking the same action in the same state during different time-steps may result in transitions to different states. A change in the system's state may also be due to external factors other than the impact of the actions taken by the autonomic manager. In a dynamic Web server environment, for example, a transition may be a result of a request to a page with a database-intensive query, which could potentially cause a state transition. In this work, such transitions are referred to as γ-transitions; the action responsible for γ-transitions is denoted by a_0 (i.e., γ-`action`) as illustrated in Table IV.

C. State-Transition Model

A state-transition model is then defined for a set of active expectation policies:

Definition 9: A state-transition model derived from the policy system $PS = \langle P, W_C \rangle$ is defined by the graph $G^P = \langle S, T \rangle$ where:

- S is a set of system states (see Section II-A) derived from the metrics of the conditions of the enabled expectation policies and where each state, $s_i \in S$, corresponds to a vertex on the graph.
- T is a set of transitions (see Section II-B) where each transition, $t_i \in T$, corresponds to a directed edge on the graph. A transition is determined when the autonomic manager takes an action as a result of being in one state, which may, or may not, result in a transition to another state.

For illustration purposes, suppose that the policies of Figure 1 are the only enabled expectation policies. From the conditions of Table I, six metrics are then formed as illustrated in Table II. Metric m_1, for instance, would correspond to the "`CPU:utilization`" policy conditions. This metric is mapped onto three regions based on the thresholds of the conditions the metric is associated with; i.e., $\sigma_{m_1} = \{15.0, 85.0\}$. This means that, "`CPU:utilization`" could fall into three unique localities; its value could be less than 15.0% (i.e., region $R^1_{m_1}$) between 85.0% and 15.0% inclusive (i.e., region $R^2_{m_1}$), or greater than 85.0% (i.e., region $R^3_{m_1}$). Each of the regions is also assigned a numeric value between 0 and 100 as described in Definition 5. It is worth pointing out that, for the system with only the policies in Figure 1 enables, 144 (i.e., $3^2 \times 2^4$) states are possible. This is because there are two state metrics (i.e., m_1 and m_5) with three possible regions each (i.e., 3^2 possible permutations) and four state metrics (i.e., m_2, m_3, m_4, and m_6) with two possible regions each (i.e., 2^4 possible permutations).

c_i	$W_C(c_i)$	Policy Condition
1	1/8	`CPU:utilization > 85.0`
2	1/8	`CPU:utilizationTREND > 0.0`
3	1/8	`CPU:utilization < 15.0`
4	1/8	`MEMORY:utilization > 40.0`
5	1/8	`MEMORY:utilizationTREND > 0.0`
6	1/8	`APACHE:responseTime > 2000.0`
7	1/8	`APACHE:responseTimeTREND > 0.0`
8	1/8	`APACHE:responseTime < 250.0`

TABLE I
POLICY CONDITIONS SET FROM THE EXPECTATION POLICIES IN FIGURE 1.

Once the metrics structure has been constructed, the next step in the creation of actual states involves mapping the Mon-

```
[1] expectation policy{CPUViolation(PDP, PEP)}
 if(CPU:utilization > 85.0) &
   (CPU:utilizationTREND > 0.0)
 then{AdjustMaxClients(-25)
        test{newMaxClients > 49} |
      AdjustMaxKeepAliveRequests(-30)
        test{newMaxKeepAliveRequests > 1} |
      AdjustMaxBandwidth(-128)
        test{newMaxBandwidth > 255}}
[2] expectation policy{RESPViolation(PDP, PEP)}
 if(APACHE:responseTime > 2000.0) &
   (APACHE:responseTimeTREND > 0.0)
 then{AdjustMaxClients(+25)
        test{newMaxClients < 151} &
        test{IdleWorkers  < 5} |
      AdjustMaxKeepAliveRequests(-30)
        test{newMaxKeepAliveRequests > 1} |
      AdjustMaxBandwidth(-128)
        test{newMaxBandwidth > 255}}
[3] expectation policy{CPUandRESPViol.(PDP, PEP)}
 if(CPU:utilization > 85.0) &
   (CPU:utilizationTREND > 0.0) &
   (APACHE:responseTime > 2000.0)
 then{AdjustMaxKeepAliveRequests(-30)
        test{newMaxKeepAliveRequests > 1} |
      AdjustMaxBandwidth(-128)
        test{newMaxBandwidth > 255}}
[4] expectation policy{MEMORYViolation(PDP, PEP)}
 if(MEMORY:utilization > 40.0) &
   (MEMORY:utilizationTREND > 0.0)
 then{AdjustMaxClients(-25)
        test{newMaxClients > 49}}
[5] expectation policy{SERVERnormal(PDP, PEP)}
 if(CPU:utilization < 15.0) &
   (APACHE:responseTime < 250.0)
 then{AdjustMaxClients(-25)
        test{newMaxClients > 49} &
        test{IdleWorkers > 25} |
      AdjustMaxKeepAliveRequests(+30)
        test{newMaxKeepAliveRequests < 95}}
```

Fig. 1. Sample policy system where $P = \{p_1, p_2, p_3, p_4, p_5\}$.

itor events collected during a single management interval onto appropriate state metrics and regions. Suppose, for example, that during a single management interval, the management system receives the events in Figure 2. By mapping the events using the metrics structure in Table II, a new state is then created as illustrated in Table III; lets call it state s_*. For a management system with only the policies of Figure 1 enabled, being in state s_* would mean the violation of policy p_1; i.e.. $P(s_*) = \{p_1\}$. Thus, in addition to the action "a_0 : γ-`action`" which corresponds to "doing nothing", $A(s_*)$ would consist of a set of unique actions from the actions of the policies in $P(s_*)$ as illustrated in Table IV.

```
CPU:utilization = 90.0
CPU:utilizationTREND = 1.0
MEMORY:utilization = 32.0
MEMORY:utilizationTREND = -2.0
APACHE:responseTime = 1500.0
APACHE:responseTimeTREND = -1.0
```

Fig. 2. A set of Monitor events collected during a single management interval.

In the sections that follow, we describe our approach for adapting the state-transition model to run-time policy modifications (denoted by $\triangle[P]$) which we formally define in Section III-A.

III. Adapting to Policy Changes

Figure 3 describes our approach to model-adaptation. The left-part of the diagram (i.e., prior to $\triangle[P]$) depicts the Reinforcement Learning mechanisms described in Section II. whereby the agent learns the model of the environment based on past experience in the use of an active set of policies, P. In this approach, the conditions of the policies define system states while the actions of the policies define possible causes of transitions between states. That is, at each time step, the agent takes action $a_i \in A(s)$ where $A(s)$ is a set of actions advocated by the policies that are violated when the system is in state s. This, in turn, causes a transition as depicted in the diagram. Thus, the model is continuously updated based on the experience garnered from the agent's interaction with its environment; i.e., based on the states and transitions encountered. The right-part of the diagram (i.e., after $\triangle[P]$ occurs) depicts the mapping (using the transformation function Ψ) from the current system's model (i.e., $G_n^P = \langle S, T \rangle$) to a new model (i.e., $G_{n+1}^{P'} = \langle S', T' \rangle$), as a result of run-time modification to the policies in P.

A. Types of Modifications

By run-time policy modifications, we mean any type of modification resulting in changes to the characteristics of the policies driving autonomic management. Since the structure specific to the model of the environment is derived from the characteristics of the enabled policies (see Section II-A), any run-time changes to these characteristics could have possible ramifications on the learning process. This would likely depend on what kind of modification is done on the policies.

1) Adding/Removing a Policy: A policy modification could involve adding a policy onto the policies in P; i.e.,

Given:

I) $PS = \langle P, W_C \rangle$

II) $p_* = \langle C, A \rangle$ where $p_* \notin P$:

- $C = \{c_1, c_2, \ldots, c_x\}$: $c_x = \langle$`ID, metricName, operator,` $\Gamma \rangle$
- $A = \{a_1, a_2, \ldots, a_y\}$: $a_y = \langle$`ID, function, parameters,` $\tau \rangle$

Then:

1) $PS' = \langle P', W_C \rangle : P' \leftarrow P \cup p_*$

A policy modification could also involve removing an existing policy from P; i.e.,

Given:

I) $PS = \langle P, W_C \rangle$

II) $p_* = \langle C, A \rangle$ where $p_* \in P$:

- $C = \{c_1, c_2, \ldots, c_x\}$: $c_x = \langle$`ID, metricName, operator,` $\Gamma \rangle$
- $A = \{a_1, a_2, \ldots, a_y\}$: $a_y = \langle$`ID, function, parameters,` $\tau \rangle$

m_i	c_k	Policy Condition	R_{m_i}	$R^j_{m_i}$	$f(R^j_{m_i})$
m_1	3	`CPU:utilization < 15.0`	m_1.value < 15.0	$R^1_{m_1}$	100
			$15.0 \leq m_1$.value ≤ 85.0	$R^2_{m_1}$	50
	1	`CPU:utilization > 85.0`	m_1.value > 85.0	$R^3_{m_1}$	0
m_2			m_2.value ≤ 0.0	$R^1_{m_2}$	100
	2	`CPU:utilizationTREND > 0.0`	m_2.value > 0.0	$R^2_{m_2}$	0
m_3			m_3.value ≤ 40.0	$R^1_{m_3}$	100
	4	`MEMORY:utilization > 40.0`	m_3.value > 40.0	$R^2_{m_3}$	0
m_4			m_4.value ≤ 0.0	$R^1_{m_4}$	100
	5	`MEMORY:utilizationTREND > 0.0`	m_4.value > 0.0	$R^2_{m_4}$	0
m_5	8	`APACHE:responseTime < 250.0`	m_5.value < 250.0	$R^1_{m_5}$	100
			$250.0 \leq m_5$.value ≤ 2000.0	$R^2_{m_5}$	50
	6	`APACHE:responseTime > 2000.0`	m_5.value > 2000.0	$R^3_{m_5}$	0
m_6			m_6.value ≤ 0.0	$R^1_{m_6}$	100
	7	`APACHE:responseTimeTREND > 0.0`	m_6.value > 0.0	$R^2_{m_6}$	0

TABLE II
METRICS STRUCTURE DERIVED FROM THE EXPECTATION POLICIES OF FIGURE 1.

m_i	Metric Value	R_{m_i}	$R^j_{m_i}$	$f(R^j_{m_i})$
m_1	`CPU:utilization = 90.0`	m_1.value > 85.0	$R^3_{m_1}$	0
m_2	`CPU:utilizationTREND = 1.0`	m_2.value > 0.0	$R^2_{m_2}$	0
m_3	`MEMORY:utilization = 32.0`	m_3.value ≤ 40.0	$R^1_{m_3}$	100
m_4	`MEMORY:utilizationTREND = -2.0`	m_4.value ≤ 0.0	$R^1_{m_4}$	100
m_5	`APACHE:responseTime = 1500.0`	$250.0 \leq m_5$.value ≤ 2000.0	$R^2_{m_5}$	50
m_6	`APACHE:responseTimeTREND = -1.0`	m_6.value ≤ 0.0	$R^1_{m_6}$	100

TABLE III
s_* - A STATE DERIVED FROM THE MONITOR EVENTS IN FIGURE 2 AND THE METRICS STRUCTURE OF TABLE II.

a_i	$Q(s_*, a_i)$	Policy Actions
a_0		γ-action
a_2		`AdjustMaxClients(-25)` **test**{`newMaxClients > 49`}
a_4		`AdjustMaxKeepAliveRequests(-30)` **test**{`newMaxKeepAliveRequests > 1`}
a_6		`AdjustMaxBandwidth(-128)` **test**{`newMaxBandwidth > 255`}

TABLE IV
$A(s_*)$ - A SET OF UNIQUE ACTIONS FROM THE ACTIONS OF THE POLICIES THAT ARE VIOLATED WHEN THE SYSTEM IS IN STATE s_*.

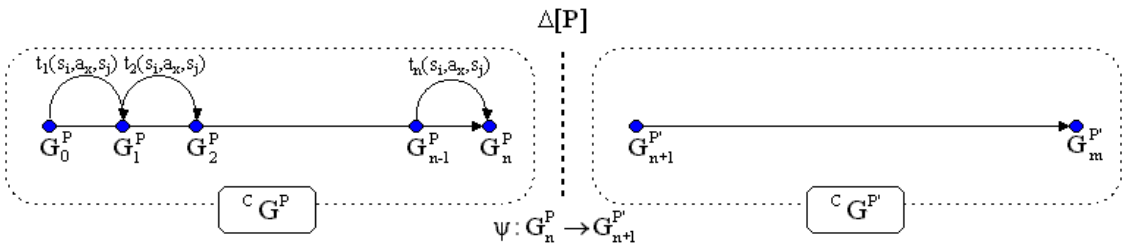


Fig. 3. Adapting to run-time policy modifications.

Then:

1) $PS' = \langle P', W_C \rangle : P' \leftarrow P | p_*$

Since the impact of such modifications on the state-transition model would depend on what conditions and/or actions are affected as a result, adding/removing an entire policy can be modelled in the context of adding/removing individual conditions and actions within policies.

On the one hand, adding policy $p_* = \langle C, A \rangle$ into the policies in P can be modelled by the following sequence of policy modifications: (i) Adding the policy without conditions or actions; i.e., $p_* = \langle \emptyset, \emptyset \rangle$ (see below). (ii) Adding one condition at a time from the conditions in C (see Section III-A2). (iii) Adding one action at a time from the actions in A (see Section III-A5).

Policy Modification - $\triangle_x$[P]: *Adding a policy without conditions or actions.*

Given:

I) $PS = \langle P, W_C \rangle$
II) $M^P_R = \{r_1, r_2, \ldots, r_l\} : r_i = \langle \alpha_i, \sigma_i \rangle$
III) $p_* = \langle \emptyset, \emptyset \rangle$ such that $p_* \notin P$

Then:

1) $PS' = \langle P', W_C \rangle : P' \leftarrow P \cup p_*$
2) $M' \leftarrow M$
3) $M_R^{P'} \leftarrow M_R^P$

On the other hand, removing policy $p_* = \langle C, A \rangle$ from the policies in P can be modelled by the following sequence of policy modifications: (i) Removing one action at a time from the actions in A (see Section III-A6). (ii) Removing one condition at a time from the conditions in C (see Section III-A3). (iii) Removing the policy without conditions or actions; i.e., $p_* = \langle \emptyset, \emptyset \rangle$ (see below).

Policy Modification - $\triangle_y$[P]: *Removing a policy without conditions or actions.*

Given:

I) $PS = \langle P, W_C \rangle$
II) $M_R^P = \{r_1, r_2, \ldots, r_l\} : r_i = \langle \alpha_i, \sigma_i \rangle$
III) $p_* = \langle \emptyset, \emptyset \rangle$ such that $p_* \in P$

Then:

1) $PS' = \langle P', W_C \rangle : P' \leftarrow P | p_*$
2) $M' \leftarrow M$
3) $M_R^{P'} \leftarrow M_R^P$

2) Adding a Policy Condition: Another type of policy modification could involve adding a condition into one of the policies in P.

Given:

- $PS = \langle P, W_C \rangle$
- $p_* = \langle C, A \rangle$ such that $p_* \in P$
- $c_* = \langle$`ID, metricName, operator,` $\Gamma \rangle$

Then:

1) $PS' = \langle P', W_C \rangle : p'_* = \langle C', A \rangle$ where $C' \leftarrow C \cup c_*$

There are several ways in which adding a policy condition could impact the system's metrics structure (see Definition 4):

In the first case, adding a policy condition may result in an increase in the number of system metrics. This may be a result of adding a policy condition whose `metricName` is not in any of the conditions of the policies in P.

Policy Modification - $\triangle_1$[P]: *Adding a policy condition resulting in a new metric.*

Given:

I) $PS = \langle P, W_C \rangle$
II) $M_R^P = \{r_1, r_2, \ldots, r_l\} : r_i = \langle \alpha_i, \sigma_i \rangle$
III) $p_* = \langle C, A \rangle$ such that $p_* \in P$ where $\forall\ p_i \in P$, $c_* \notin p_i$
IV) $c_* = \langle$`ID, metricName, operator,` $\Gamma \rangle$ such that $m = c_*$.`metricName` and where $m \notin M$
V) $r_m \notin M_R^P : r_m = \langle \alpha_m, \sigma_m \rangle$:
   - $\alpha_m = \langle$`ID, metricName,` $\omega \rangle$
   - $\sigma_m = \{\Gamma\}$

Then:

1) $PS' = \langle P', W_C \rangle : p'_* = \langle C', A \rangle$ where $C' \leftarrow C \cup c_*$
2) $M' \leftarrow M \cup m$
3) $M_R^{P'} \leftarrow M_R^P \cup r_m$

In the second case, adding a policy condition may result in changes to the regions of an existing metric; i.e., $r_m \in M_R^P$. This may be a result of adding a policy condition whose `metricName` is in at least one of the conditions of the policies in P and where there is no occurrence of the condition within the policies in P.

Policy Modification - $\triangle_2$[P]: *Adding a policy condition resulting in changes to the regions of existing metrics.*

Given:

I) $PS = \langle P, W_C \rangle$
II) $M_R^P = \{r_1, r_2, \ldots, r_l\} : r_i = \langle \alpha_i, \sigma_i \rangle$
III) $p_* = \langle C, A \rangle$ such that $p_* \in P$ where $\forall\ p_i \in P$, $c_* \notin p_i$
IV) $c_* = \langle$`ID, metricName, operator,` $\Gamma \rangle$ such that $m = c_*$.`metricName` and where $m \in M$
V) $r_m \in M_R^P : r_m = \langle \alpha_m, \sigma_m \rangle$:
   - $\alpha_m = \langle$`ID, metricName,` $\omega \rangle$
   - $\sigma_m = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_j, \Gamma_{j+1}, \ldots, \Gamma_k\}$

Then:

1) $PS' = \langle P', W_C \rangle : p'_* = \langle C', A \rangle$ where $C' \leftarrow C \cup c_*$
2) $M' \leftarrow M$
3) $M_R^{P'} \leftarrow M_R^P : \exists\ r_m \in M_R^P$: $r'_m = \langle \alpha'_m, \sigma'_m \rangle$:
   a) $\alpha'_m = \langle$`ID, metricName,` $\omega \rangle$
   b) $\sigma'_m = \sigma_m \cup c_*.\Gamma : \sigma'_m = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_j, \mathbf{\Gamma}, \Gamma_{j+1}, \ldots, \Gamma_k\}$ where $\Gamma_j < \mathbf{\Gamma} < \Gamma_{j+1}$ for some j.

In the third case, adding a policy condition may result in no changes to the number of metrics and their regions. This may be a result of adding a policy condition that is identical (i.e., share the same `metricName`, `operator`, and Γ) to at least one of the conditions of the policies in P.

Policy Modification - $\triangle_3$[P]: *Adding a policy condition resulting in no changes to the metrics' regions.*

Given:

I) $PS = \langle P, W_C \rangle$
II) $M_R^P = \{r_1, r_2, \ldots, r_l\} : r_i = \langle \alpha_i, \sigma_i \rangle$
III) $p_* = \langle C, A \rangle$ such that $p_* \in P$ and where $\exists\ p_i \in P$ such that $c_* \in p_i$
IV) $c_* = \langle$`ID, metricName, operator,` $\Gamma \rangle$ such that $m = c_*$.`metricName` and where $m \in M$
V) $r_m \in M_R^P : r_m = \langle \alpha_m, \sigma_m \rangle$:
   - $\alpha_m = \langle$`ID, metricName,` $\omega \rangle$
   - $\sigma_m = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_{j-1}, \Gamma_j, \Gamma_{j+1} \ldots \Gamma_k\}$: $c_*.\Gamma = \Gamma_j$

Then:

1) $PS' = \langle P', W_C \rangle : p'_* = \langle C', A \rangle$ where $C' \leftarrow C \cup c_*$
2) $M' \leftarrow M$
3) $M_R^{P'} \leftarrow M_R^P$

3) Removing a Policy Condition: Another type of policy modification could involve removing a condition from one of the policies in P.

Given:

- $PS = \langle P, W_C \rangle$
- $p_* = \langle C, A \rangle$ such that $p_* \in P$ and $c_* \in C$
- $c_* = \langle$`ID, metricName, operator,` $\Gamma \rangle$

Then:

1) $PS' = \langle P', W_C \rangle : p'_* = \langle C', A \rangle$ where $C' \leftarrow C|c_*$

There are several ways in which removing a policy condition could impact the system's metrics structure:

In the first case, removing a policy condition may result in a decrease in the number of system metrics. This may be a result of removing a policy condition whose `metricName` is not in any of the remaining conditions of the policies in P; i.e., there is only a single occurrence of the condition with the given `metricName` within the policies in P.

Policy Modification - $\triangle_4$[P]: *Removing a policy condition resulting in a decrease in the number of metrics.*

Given:

I) $PS = \langle P, W_C \rangle$
II) $M_R^P = \{r_1, r_2, \ldots, r_l\} : r_i = \langle \alpha_i, \sigma_i \rangle$
III) $p_* = \langle C, A \rangle$ such that $p_* \in P$ and $c_* \in C$ where $\forall\ p_i \in P|p_*$, $c_* \notin p_i$
IV) $c_* = \langle$`ID, metricName, operator,` $\Gamma \rangle$ such that $m = c_*$.`metricName` and where $m \in M$, such that, $\forall\ m_i \in M|m$, $m_i \neq m$
V) $r_m \in M_R^P : r_m = \langle \alpha_m, \sigma_m \rangle$:
- $\alpha_m = \langle \mathtt{ID}, \mathtt{metricName}, \omega \rangle$
- $\sigma_m = \{\Gamma\}$

Then:

1) $PS' = \langle P', W_C \rangle : p'_* = \langle C', A \rangle$ where $C' \leftarrow C|c_*$
2) $M' \leftarrow M|m$
3) $M_R^{P'} \leftarrow M_R^P|r_m$

In the second case, removing a policy condition may result in changes to the regions of existing metrics. This may be a result of removing a policy condition whose `metricName` is in at least one of the conditions of the policies in P and where there is only a single occurrence of the condition within the policies in P.

Policy Modification - $\triangle_5$[P]: *Removing a policy condition resulting in changes to the regions of existing metrics.*

Given:

I) $PS = \langle P, W_C \rangle$
II) $M_R^P = \{r_1, r_2, \ldots, r_l\} : r_i = \langle \alpha_i, \sigma_i \rangle$
III) $p_* = \langle C, A \rangle$ such that $p_* \in P$ and $c_* \in C$ where $\forall\ p_i \in P|p_*$, $c_* \notin p_i$
IV) $c_* = \langle$`ID, metricName, operator,` $\Gamma \rangle$ such that $m = c_*$.`metricName` where $m \in M$, and where $\exists\ m_i \in M|m$ such that $m = m_i$
V) $r_m \in M_R^P : r_m = \langle \alpha_m, \sigma_m \rangle$:
- $\alpha_m = \langle \mathtt{ID}, \mathtt{metricName}, \omega \rangle$
- $\sigma_m = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_{j-1}, \mathbf{\Gamma_j}, \Gamma_{j+1}, \ldots, \Gamma_k\}$: $c_*.\Gamma = \Gamma_j$

Then:

1) $PS' = \langle P', W_C \rangle : p'_* = \langle C', A \rangle$ where $C' \leftarrow C|c_*$
2) $M' \leftarrow M$
3) $M_R^{P'} \leftarrow M_R^P : \exists\ r_m \in M_R^P : r'_m = \langle \alpha'_m, \sigma'_m \rangle$:
   a) $\alpha'_m = \langle \mathtt{ID}, \mathtt{metricName}, \omega \rangle$
   b) $\sigma'_m = \sigma_m|c_*.\Gamma : \sigma'_m = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_{j-1}, \Gamma_{j+1}, \ldots, \Gamma_k\}$

In the third case, removing a policy condition may result in no changes to the number of metrics and their regions. This may be a result of removing a policy condition that is identical (i.e., share the same `metricName`, `operator`, and Γ) to at least one of the conditions of the policies in P; i.e., there are multiple occurrences of the condition within the policies in P.

Policy Modification - $\triangle_6$[P]: *Removing a policy condition resulting in no changes to the number of metrics and their regions.*

Given:

I) $PS = \langle P, W_C \rangle$
II) $M_R^P = \{r_1, r_2, \ldots, r_l\} : r_i = \langle \alpha_i, \sigma_i \rangle$
III) $p_* = \langle C, A \rangle$ such that $p_* \in P$ and $c_* \in C$ where $\exists\ p_i \in P|p_*$ such that $c_* \in p_i$
IV) $c_* = \langle$`ID, metricName, operator,` $\Gamma \rangle$ such that $m = c_*$.`metricName` and where $m \in M$
V) $r_m \in M_R^P : r_m = \langle \alpha_m, \sigma_m \rangle$:
- $\alpha_m = \langle \mathtt{ID}, \mathtt{metricName}, \omega \rangle$
- $\sigma_m = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_{j-1}, \mathbf{\Gamma_j}, \Gamma_{j+1}, \ldots, \Gamma_k\}$: $c_*.\Gamma = \Gamma_j$

Then:

1) $PS' = \langle P', W_C \rangle : p'_* = \langle C', A \rangle$ where $C' \leftarrow C|c_*$
2) $M' \leftarrow M$
3) $M_R^{P'} \leftarrow M_R^P$

4) Modifying a Policy Condition: Another type of policy modification could involve modifying the threshold of a policy condition within the conditions of the policies in P.

Policy Modification - $\triangle_7$[P]: *Modifying the threshold of a policy condition.*

Given:

I) $PS = \langle P, W_C \rangle$
II) $M_R^P = \{r_1, r_2, \ldots, r_l\} : r_i = \langle \alpha_i, \sigma_i \rangle$
III) $c_* = \langle$`ID, metricName, operator,` $\Gamma \rangle$ such that $m = c_*$.`metricName` and where $m \in M$
IV) $r_m \in M_R^P : r_m = \langle \alpha_m, \sigma_m \rangle$:
- $\alpha_m = \langle \mathtt{ID}, \mathtt{metricName}, \omega \rangle$
- $\sigma_m = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_{j-1}, \Gamma_j, \Gamma_{j+1}, \ldots, \Gamma_k\}$: $c_*.\Gamma = \Gamma_j$

Then:

1) $PS' = \langle P', W_C \rangle : c'_* = \langle$`ID, metricName, operator,` $\Gamma' \rangle\ \forall\ p_* \in P : c_* \in p_*$
2) $M' \leftarrow M$
3) $M_R^{P'} \leftarrow M_R^P : \exists\ r_m \in M_R^P : r'_m = \langle \alpha'_m, \sigma'_m \rangle$:
   a) $\alpha'_m = \langle \mathtt{ID}, \mathtt{metricName}, \omega \rangle$

b) $\sigma'_m = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_l, \Gamma', \Gamma_{l+1}, \ldots, \Gamma_k\}$ where $\Gamma_l < \Gamma' < \Gamma_{l+1}$ for some l

5) Adding a Policy Action: Another type of policy modification could involve adding a policy action onto one of the policies in P.

Policy Modification - $\triangle_8$[P]: *Adding a policy action onto a policy in P.*

Given:

I) $PS = \langle P, W_C \rangle$
II) $M_R^P = \{r_1, r_2, \ldots, r_l\} : r_i = \langle \alpha_i, \sigma_i \rangle$
III) $p_* = \langle C, A \rangle$ such that $p_* \in P$
IV) $a_* = \langle$`ID, function, parameters,` $\tau \rangle$
V) $r_m \in M_R^P : r_m = \langle \alpha_m, \sigma_m \rangle$:
- $\alpha_m = \langle$`ID,metricName,` $\omega \rangle$
- $\sigma_m = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_k\}$

Then:

1) $PS' = \langle P', W_C \rangle : p'_* = \langle C, A' \rangle$ where $A' \leftarrow A \cup a_*$
2) $M' \leftarrow M$
3) $M_R^{P'} \leftarrow M_R^P$

6) Removing a Policy Action: Another type of policy modification could involve removing a policy action from one of the policies in P.

Policy Modification - $\triangle_9$[P]: *Removing an action from a policy in P.*

Given:

I) $PS = \langle P, W_C \rangle$
II) $M_R^P = \{r_1, r_2, \ldots, r_l\} : r_i = \langle \alpha_i, \sigma_i \rangle$
III) $p_* = \langle C, A \rangle$ such that $p_* \in P$ and $a_* \in A$
IV) $a_* = \langle$`ID, function, parameters,` $\tau \rangle$
V) $r_m \in M_R^P : r_m = \langle \alpha_m, \sigma_m \rangle$:
- $\alpha_m = \langle$`ID,metricName,` $\omega \rangle$
- $\sigma_m = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_k\}$

Then:

1) $PS' = \langle P', W_C \rangle : p'_* = \langle C, A' \rangle$ where $A' \leftarrow A | a_*$
2) $M' \leftarrow M$
3) $M_R^{P'} \leftarrow M_R^P$

7) Modifying a Policy Action: Another type of policy modification could involve changing the attributes (i.e., `parameters`, or τ) of a policy action within the actions of the policies in P.

Policy Modification - $\triangle_{10}$[P]: *Modifying the attributes of a policy action.*

Given:

I) $PS = \langle P, W_C \rangle$
II) $M_R^P = \{r_1, r_2, \ldots, r_l\} : r_i = \langle \alpha_i, \sigma_i \rangle$
III) $a_* = \langle$`ID, function, parameters,` $\tau \rangle$
IV) $r_m \in M_R^P : r_m = \langle \alpha_m, \sigma_m \rangle$:
- $\alpha_m = \langle$`ID,metricName,` $\omega \rangle$
- $\sigma_m = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_k\}$

Then:

1) $PS' = \langle P', W_C \rangle : a'_* = \langle$`ID, function, parameters',` $\tau' \rangle \; \forall \; p_* \in P : a_* \in p_*$
2) $M' \leftarrow M$
3) $M_R^{P'} \leftarrow M_R^P$

B. Adaptation Transformations

This section presents our approach for adapting the model learned from the use of an active set of policies to the policy modifications described in Section III-A. We have identified several cases relating to the types of changes to an active set of policies, P, in the context of how they might impact the model (i.e., the state-transition graph) leaned from using P. For illustration purposes, this section assumes that initially, set P consists of the expectation policies depicted in Figure 1. Suppose also that the current model of the system (i.e., $G_n^P = \langle S, T \rangle$), as depicted in Figure 4, includes the seven states shown in Table V. Note that, states s_2 and s_6 are considered "non-violation" whereas the other states are considered "violation" states (see μ in Definition 6). The focus of our discussion centers on how the various changes to the policies in P might affect the original model.

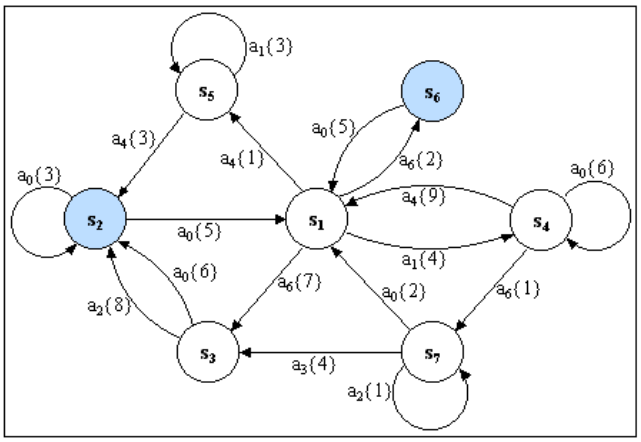


Fig. 4. A sample state transition model.

1) Adapting to $\triangle[P]$ Affecting States Actions: This describes our approach for dealing with policy modifications that only impact the actions within states and not states transitions. From the definitions of Section III-A, this may be in response to policy modification $\triangle_8[P]$ or $\triangle_{10}[P]$.

Model Transformation - $\Psi_1[G_n^P]$: *Adapting to policy modification $\triangle_8[P]$ or $\triangle_{10}[P]$.*

Given:

I) $G_n^P = \langle S, T \rangle$:
- $\forall \; s_i \in S, \; s_i = \langle \mu, M(s_i), P(s_i), A(s_i) \rangle : s_i.m_k = \langle$`ID,` ω`,value,` $R^l_{m_k} \rangle$ where $s_i.m_k \in M(s_i)$
- $\forall \; t(s_i, a, s_j) \in T, \; t(s_i, a, s_j) = \langle \lambda, Q_t(s_i, a) \rangle$

II) Policy modification $\triangle_8[P]$ or $\triangle_{10}[P]$

Then: $G_{n+1}^{P'} = \langle S', T' \rangle$ where

1) $S' \leftarrow S$:

	$f(R^k_{m_j})$						$A(s_i)$		$t(s_i, a_l, s_*)$	
s_i	$f(R^k_{m_1})$	$f(R^k_{m_2})$	$f(R^k_{m_3})$	$f(R^k_{m_4})$	$f(R^k_{m_5})$	$f(R^k_{m_6})$	a_l	State action	λ	s_*
s_1	0	0	0	0	0	0	a_0	γ-action		
							a_1	AdjustMaxClients(+25)	4	s_4
							a_2	AdjustMaxClients(-25)		
							a_4	AdjustMaxKeepAliveRequests(-30)	1	s_5
							a_6	AdjustMaxBandwidth(-128)	7;2	s_3;s_6
s_2	0	100	0	100	0	100	a_0	γ-action	5;3	s_1;s_2
s_3	50	100	0	0	100	100	a_0	γ-action	6	s_2
							a_2	AdjustMaxClients(-25)	8	s_2
s_4	0	100	0	100	0	0	a_0	γ-action	6	s_4
							a_1	AdjustMaxClients(+25)		
							a_4	AdjustMaxKeepAliveRequests(-30)	9	s_1
							a_6	AdjustMaxBandwidth(-128)	1	s_7
s_5	0	0	100	100	0	0	a_0	γ-action		
							a_1	AdjustMaxClients(+25)	3	s_5
							a_2	AdjustMaxClients(-25)		
							a_4	AdjustMaxKeepAliveRequests(-30)	3	s_2
							a_6	AdjustMaxBandwidth(-128)		
s_6	50	100	0	100	100	100	a_0	γ-action	5	s_1
s_7	100	100	100	100	100	100	a_0	γ-action	2	s_1
							a_2	AdjustMaxClients(-25)	1	s_7
							a_3	AdjustMaxKeepAliveRequests(+30)	4	s_3

TABLE V
SAMPLE STATES BASED ON THE METRICS STRUCTURE OF TABLE II AND THE STATE-TRANSITION MODEL OF FIGURE 4.

a) $\forall\ s_i \in S : p_* \in P'(s_i);\ A'(s_i) \leftarrow A(s_i) \cup a_*\ (\triangle_8[P])$
b) $\forall\ s_i \in S;\ a_* \leftarrow a'_* \iff \exists\ a \in A(s_i) : a_* = a\ (\triangle_{10}[P])$

2) $T' \leftarrow T$

Suppose, for example, that policy modification $\triangle_{10}[P]$ involves modifying policy action "AdjustMaxBandwidth(-128)" (which corresponds to action a_6 in Table V) to "AdjustMaxBandwidth(-64)" in the policies of Figure 1. The following describes the steps involved in transforming the state-transition model of Figure 5(a), whose states are shown in Table V and correspond to the steps of model-transformation $\Psi_1[G^P_n]$:

1) For each state that has been encountered to date (see Table V), the agent must update the actions within the states to ensure that the appropriate action is modified (see Step 1(b)). Since action a_6 was initially in actions sets $A(s_1)$, $A(s_4)$, and $A(s_5)$, each occurrence of the action would be modified to "AdjustMaxBandwidth(-64)" as illustrated in Figure 5(c).
2) This change, however, would not affect the transitions associated with the modified action. Consider, for example, transitions $t(s_1, a_6\{7\}, s_3)$ and $t(s_1, a_6\{2\}, s_6)$, both involving action a_6 (see Figure 5(a)). The fact that the action is still part of the actions within the states means that the agent may still take such transitions. Hence, no changes would be made to the transitions unless they are encountered in the future, in which case action-value estimates and frequencies will be updates accordingly.

2) Adapting to $\triangle[P]$ Affecting States Transitions: This describes our approach for dealing with policy modifications that only affect the state transitions as a result of modifications to the composition of policy actions within states. From the definitions of Section III-A, this may be in response to policy modification $\triangle_3[P]$, $\triangle_6[P]$, or $\triangle_9[P]$. Our approach essentially involves ensuring that the state transitions set $T(s_i)$ is consistent with the state actions set $A(s_i)$.

Model Transformation - $\Psi_2[G^P_n]$: *Adapting to policy modification $\triangle_3[P]$, $\triangle_6[P]$, or $\triangle_9[P]$.*

Given:

I) $G^P_n = \langle S, T\rangle$:
- $\forall\ s_i \in S,\ s_i = \langle \mu, M(s_i), P(s_i), A(s_i)\rangle$: $s_i.m_k = \langle \texttt{ID}, \omega, \texttt{value}, R^l_{m_k}\rangle$ where $s_i.m_k \in M(s_i)$
- $\forall\ t(s_i, a, s_j) \in T,\ t(s_i, a, s_j) = \langle \lambda, Q_t(s_i, a)\rangle$

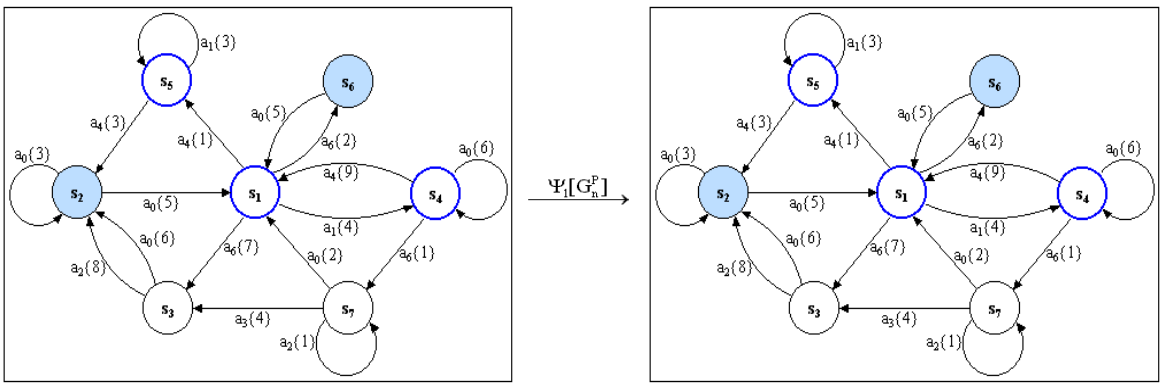
II) Policy modification $\triangle_3[P]$, $\triangle_6[P]$, or $\triangle_9[P]$

Then: $G^{P'}_{n+1} = \langle S', T'\rangle$ where

1) $S' \leftarrow S : \forall\ s_i \in S\ ;\ a \in A'(s_i) \iff a \in \{P'(s_i) \cup a_0\}$
2) $\forall\ t(s_i, a_*, s_j) \in T(s_i) : a_* \notin A'(s_i);\ T' \leftarrow T|t(s_i, a_*, s_j)$
3) $\forall\ a_i \in A(s_i : \exists\ t(s_i, a, s_*) : s_i, s_* \in S'$, compute $Q(s_i, a)$ (see Equation 2).

$$Q(s,a) = \sum_{t_i(s,a,s'_j)\in T(s)} Pr[t_i(s,a,s'_j)] \times Q_{t_i}(s,a) \quad (2)$$

Suppose, for example, that policy modification $\triangle_9[P]$ involves deleting policy action "AdjustMaxBandwidth(-128)" (which corresponds to action a_6 in Table V) from policy p_2 in Figure 1. The following describes the steps involved in transforming the state-transition model of Figure 6(a), whose states are shown in Table V and correspond to the steps of model-transformation $\Psi_2[G^P_n]$:



(a) Original model: $G_n^P = \langle S, T \rangle$

(b) New model: $G_{n+1}^{P'} = \langle S', T' \rangle$

	$f(R_{m_j}^k)$						$A(s_i)$		$t(s_i, a_l, s_*)$	
s_i	m_1	m_2	m_3	m_4	m_5	m_6	a_l	State action	λ	s_*
s_1	0	0	0	0	0	0	a_0	γ-action		
							a_1	AdjustMaxClients(+25)	4	s_4
							a_2	AdjustMaxClients(-25)		
							a_4	AdjustMaxKeepAliveRequests(-30)	1	s_5
							a_6	AdjustMaxBandwidth(-64)	7;2	s_3;s_6
s_2	0	100	0	100	0	100	a_0	γ-action	5;3	s_1;s_2
s_3	50	100	0	0	100	100	a_0	γ-action	6	s_2
							a_2	AdjustMaxClients(-25)	8	s_2
s_4	0	100	0	100	0	0	a_0	γ-action	6	s_4
							a_1	AdjustMaxClients(+25)		
							a_4	AdjustMaxKeepAliveRequests(-30)	9	s_1
							a_6	AdjustMaxBandwidth(-64)	1	s_7
s_5	0	0	100	100	0	0	a_0	γ-action		
							a_1	AdjustMaxClients(+25)	3	s_5
							a_2	AdjustMaxClients(-25)		
							a_4	AdjustMaxKeepAliveRequests(-30)	3	s_2
							a_6	AdjustMaxBandwidth(-64)		
s_6	50	100	0	100	100	100	a_0	γ-action	5	s_1
s_7	100	100	100	100	100	100	a_0	γ-action	2	s_1
							a_2	AdjustMaxClients(-25)	1	s_7
							a_3	AdjustMaxKeepAliveRequests(+30)	4	s_3

(c) System states of the new model.

Fig. 5. State-transition model after $\triangle_{10}[P]$; i.e., modifying action a_6 ="AdjustMaxBandwidth(-128)" to "AdjustMaxBandwidth(-64)" in the policies of Figure 1.

1) Since policy p_2 is the only violated policy in state s_4, $\triangle_9[P]$ would result in the removal of action a_6 from the actions set $A(s_4)$ as illustrated in Figure 6(c). While policy p_2 is also violated in states s_1 and s_5, $\triangle_9[P]$ would not affect such states. This is because action a_6 is also part of other violated policies in those states, such as policy p_1 (i.e., $a_6 \in p_1$) where $p_1 \in P'(s_1)$ and $p_1 \in P'(s_5)$.
2) The removal of action a_6 from state s_4 in the above step would mean that transition $t(s_4, a_6\{1\}, s_7)$ (see Figure 6(a)) is no longer valid. As such, the transition will be removed as illustrated in Figure 6(b).
3) The final step of the adaptation is for the agent to perform backup updates so that any impact on the model as a result of removing transitions is propagated to the action-value estimates of the states actions.

3) Adapting to $\triangle[P]$ Affecting Metrics Regions: This describes our approach for dealing with policy modifications that affects the regions within state metrics. From the definitions of Section III-A, this may be in response to policy modification $\triangle_2[P]$, $\triangle_5[P]$, or $\triangle_7[P]$.

Model Transformation - $\Psi_3[G_n^P]$: *Adapting to policy modification $\triangle_2[P]$, $\triangle_5[P]$, or $\triangle_7[P]$.*

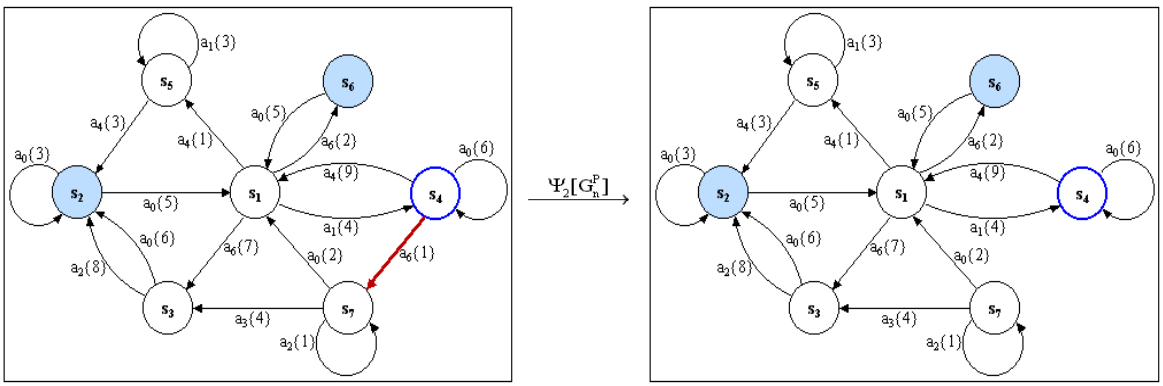
Given:

I) $G_n^P = \langle S, T \rangle$:
- $\forall\ s_i \in S,\ s_i = \langle \mu, M(s_i), P(s_i), A(s_i) \rangle$: $s_i.m_k = \langle \texttt{ID}, \omega, \texttt{value}, R_{m_k}^l \rangle$ where $s_i.m_k \in M(s_i)$
- $\forall\ t(s_i, a, s_j) \in T,\ t(s_i, a, s_j) = \langle \lambda, Q_t(s_i, a) \rangle$

II) Policy modification $\triangle_4[P]$ or $\triangle_5[P]$ or $\triangle_7[P]$

Then: $G_{n+1}^{P'} = \langle S', T' \rangle$ where

1) $S' \leftarrow S : \forall\ s_i \in S$;
   a) $s_i.m.R_m^l \leftarrow (\Gamma_l, \Gamma_{l+1}) : \sigma'_m.\Gamma_l < s_i.m.\texttt{value} < \sigma'_m.\Gamma_{l+1}$
   b) $a \in A'(s_i) \iff a \in \{P'(s_i) \cup a_0\}$



(a) Original model: $G_n^P = \langle S, T \rangle$

(b) New model: $G_{n+1}^{P'} = \langle S', T' \rangle$

	$f(R^k_{m_j})$						$A(s_i)$		$t(s_i, a_l, s_*)$	
s_i	m_1	m_2	m_3	m_4	m_5	m_6	a_l	State action	λ	s_*
s_1	0	0	0	0	0	0	a_0	γ-action		
							a_1	AdjustMaxClients(+25)	4	s_4
							a_2	AdjustMaxClients(-25)		
							a_4	AdjustMaxKeepAliveRequests(-30)	1	s_5
							a_6	AdjustMaxBandwidth(-128)	7;2	s_3;s_6
s_2	0	100	0	100	0	100	a_0	γ-action	5;3	s_1;s_2
s_3	50	100	0	0	100	100	a_0	γ-action	6	s_2
							a_2	AdjustMaxClients(-25)	8	s_2
s_4	0	100	0	100	0	0	a_0	γ-action	6	s_4
							a_1	AdjustMaxClients(+25)		
							a_4	AdjustMaxKeepAliveRequests(-30)	9	s_1
s_5	0	0	100	100	0	0	a_0	γ-action		
							a_1	AdjustMaxClients(+25)	3	s_5
							a_2	AdjustMaxClients(-25)		
							a_4	AdjustMaxKeepAliveRequests(-30)	3	s_2
							a_6	AdjustMaxBandwidth(-128)		
s_6	50	100	0	100	100	100	a_0	γ-action	5	s_1
s_7	100	100	100	100	100	100	a_0	γ-action	2	s_1
							a_2	AdjustMaxClients(-25)	1	s_7
							a_3	AdjustMaxKeepAliveRequests(+30)	4	s_3

(c) System states of the new model.

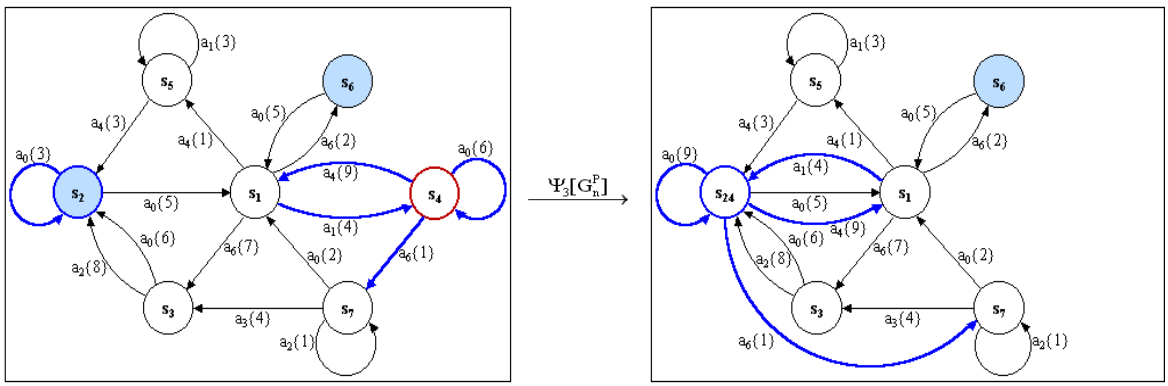
Fig. 6. State-transition model after $\triangle_9[P]$; i.e., removing action a_6 ="AdjustMaxBandwidth(-128)" from policy p_2 in Figure 1.

m_i	c_k	Policy Condition	R_{m_i}	$R^j_{m_i}$	$f(R^j_{m_i})$
m_6	7	APACHE:responseTimeTREND > -1.0	m_6.value ≤ -1.0	$R^1_{m_6}$	100
			m_6.value > -1.0	$R^2_{m_6}$	0

TABLE VI
METRIC m_6 STRUCTURE AFTER $\triangle_7[P]$; I.E., MODIFYING THE THRESHOLD OF CONDITION c_7 ="APACHE:responseTimeTREND > 0.0" TO "APACHE:responseTimeTREND > -1.0" IN THE POLICIES OF FIGURE 1.

2) $\forall\ t(s_i, a_*, s_j) \in T(s_i) : a_* \notin A'(s_i)$; $T' \leftarrow T|t(s_i, a_*, s_j)$
3) $\forall\ s_i, s_j \in S' : s_i = s_j$ where $i \neq j$;
   a) $T'(s_i) \leftarrow T'(s_i) \cup T'(s_j)$
      $\forall\ t(s_i, a, s) \in T'(s_i)$ and $\forall\ t'(s_j, a', s') \in T'(s_j) : a = a'$ and $s = s'$;
      i) $Q_t(s_i, a) \leftarrow \frac{t.\lambda \times Q_t(s_i, a) + t'.\lambda \times Q_{t'}(s_j, a')}{t.\lambda + t'.\lambda}$
      ii) $t.\lambda \leftarrow t.\lambda + t'.\lambda$
      iii) $T' \leftarrow T|t'(s_j, a', s')$
   b) $S' \leftarrow S'|s_j$
4) $\forall\ a_i \in A(s_i : \exists\ t(s_i, a, s_*) : s_i, s_* \in S'$, compute $Q(s_i, a)$ (see Equation 2).

Suppose, for example, that policy modification $\triangle_7[P]$ involves modifying the threshold of policy condition c_7 ="APACHE: responseTimeTREND > 0.0" (which corresponds to metric m_6 in Table II) to "APACHE: responseTimeTREND > -1.0" in the policies of Figure 1. Thus, instead of $\sigma_{m_6} = \{0.0\}$, as was the case prior to $\triangle_7[P]$ (see Table II), the new regions will be defined by $\sigma_{m_6} = \{-1.0\}$, as is illustrated in Table VI. Suppose also that the following measurements correspond to the average value of metric m_6; "$s_2.m_6$.value $= -0.5$", "$s_3.m_6$.value $= -2.0$", "$s_6.m_6$.value $= -1.5$", and



(a) Original model: $G_n^P = \langle S, T \rangle$

(b) New model: $G_{n+1}^{P'} = \langle S', T' \rangle$

	$f(R^k_{m_j})$						$A(s_i)$		$t(s_i, a_l, s_*)$	
s_i	m_1	m_2	m_3	m_4	m_5	m_6	a_l	State action	λ	s_*
s_1	0	0	0	0	0	0	a_0	γ-action		
							a_1	AdjustMaxClients(+25)	4	s_4
							a_2	AdjustMaxClients(-25)		
							a_4	AdjustMaxKeepAliveRequests(-30)	1	s_5
							a_6	AdjustMaxBandwidth(-128)	7;2	s_3;s_6
s_{24}	0	100	0	100	0	0	a_0	γ-action	5;9	s_1;s_{24}
							a_1	AdjustMaxClients(+25)		
							a_4	AdjustMaxKeepAliveRequests(-30)	9	s_1
							a_6	AdjustMaxBandwidth(-128)	1	s_7
s_3	50	100	0	0	100	100	a_0	γ-action	6	s_2
							a_2	AdjustMaxClients(-25)	8	s_2
s_5	0	0	100	100	0	0	a_0	γ-action		
							a_1	AdjustMaxClients(+25)	3	s_5
							a_2	AdjustMaxClients(-25)		
							a_4	AdjustMaxKeepAliveRequests(-30)	3	s_2
							a_6	AdjustMaxBandwidth(-128)		
s_6	50	100	0	100	100	100	a_0	γ-action	5	s_1
s_7	100	100	100	100	100	100	a_0	γ-action	2	s_1
							a_2	AdjustMaxClients(-25)	1	s_7
							a_3	AdjustMaxKeepAliveRequests(+30)	4	s_3

(c) System states of the new model.

Fig. 7. State-transition model after $\triangle_7[P]$; i.e., modifying the threshold of condition c_7 ="APACHE:responseTimeTREND > 0.0" to "APACHE:responseTimeTREND > -1.0" in the policies of Figure 1.

"$s_7.m_6.\texttt{value} = -2.5$". The following describes the steps involved in transforming the state-transition model of Figure 7(a), whose states are shown in Table V and correspond to the steps of model-transformation $\Psi_3[G_n^P]$:

1) Since "$m_6.\texttt{value} < -1.0$" in states s_3, s_6, and s_7, $f(R^l_{m_6})$ would not be affected by the policy modification in those states since the measurements would remain in region $R^1_{m_6}$. However, since "$s_2.m_6.\texttt{value} = -0.5$", $f(R^k_{m_6})$ would recompute the region value by changing it from 100 (see Table V) to 0 since the measurement would now fall in region $R^2_{m_6}$ (see Table VI) instead of the previous region $R^1_{m_6}$ (see Table II). The agent must also update the states actions sets of each state to ensure that actions set $A'(s_i)$ contains only the actions of the policies that are violated when the system is in state s_i (see Step 1(b)). Note that $\triangle_7[P]$ would result in the violation of policy p_2 when the agent is in state s_2. This was not the case prior to the policy modification since only one of the policy conditions was violated. As such, actions a_1, a_4, and a_6 would be added onto the actions set $A(s_2)$; i.e., $A'(s_2) = \{a_0, a_1, a_4, a_6\}$.
2) Since the previous stage did not result in the deletion of states actions, no transition will be affected.
3) Changing the region from $R^1_{m_6}$ to $R^2_{m_6}$ in state s_2 would mean states s_2 and s_4 are identical; i.e., they have the same region value assignment (i.e., $f(R^l_{m_i})$) for each identical metric. Thus, the two states would be merged together onto state s_{24}, as illustrated in Figure 7. This would also involve merging the states transitions such that transition $t(s_4, a_6\{1\}, s_7)$ becomes transition $t(s_{24}, a_6\{1\}, s_7)$, while $t(s_4, a_4\{9\}, s_1)$ becomes transition $t(s_{24}, a_4\{9\}, s_1)$ (see Step 3(a)). In the case of two identical transitions such as $t(s_4, a_0\{6\}, s_4)$ and $t(s_2, a_0\{3\}, s_2)$, a new transition $t(s_{24}, a_0\{9\}, s_{24})$ would be formed as illustrated in Figure 7(b) such that the frequency of the new transition is the sum of the frequencies of the two merged transitions. Once the transitions have been updated, the duplicate state is removed (see Step 3(b)).

4) The final step of the adaptation is for the agent to perform backup updates so that any impact on the model as a result of merging states and transitions is propagated to the action-value estimates of the states actions.

4) Adapting to $\triangle[P]$ Resulting in a Decrease in State Metrics: This describes our approach for dealing with policy modifications that reduces the size of the metrics set M, possibly resulting in two or more identical states. As mentioned previously, two states are said to be identical if they share the same metrics region values as determined by the mapping $f(R^j_{m_i})$ (see Equation 1) based on each metric's measurement (i.e., m_i.value). From the definitions of Section III-A, this may be in response to policy modification $\triangle_4[P]$.

Model Transformation - $\Psi_4[G^P_n]$: *Adapting to policy modification $\triangle_4[P]$.*

Given:

I) $G^P_n = \langle S, T\rangle$:
- $\forall\ s_i \in S,\ s_i = \langle \mu, M(s_i), P(s_i), A(s_i)\rangle$: $s_i.m_k = \langle \texttt{ID}, \omega, \texttt{value}, R^l_{m_k}\rangle$ where $s_i.m_k \in M(s_i)$
- $\forall\ t(s_i, a, s_j) \in T,\ t(s_i, a, s_j) = \langle \lambda, Q_t(s_i, a)\rangle$

II) Policy modification $\triangle_4[P]$

Then: $G^{P'}_{n+1} = \langle S', T'\rangle$ where

1) $S' \leftarrow S : \forall\ s_i \in S$;
   a) $M'(s_j) \leftarrow M(s_j)|m : m$ is the deleted metric
   b) $a \in A'(s_i) \iff a \in \{P'(s_i) \cup a_0\}$
2) $\forall\ t(s_i, a_*, s_j) \in T(s_i) : a_* \notin A'(s_i)$; $T' \leftarrow T|t(s_i, a_*, s_j)$
3) $\forall\ s_i, s_j \in S' : s_i = s_j$ where $i \neq j$;
   a) $T'(s_i) \leftarrow T'(s_i) \cup T'(s_j)$ $\forall\ t(s_i, a, s) \in T'(s_i)$ and $\forall\ t'(s_j, a', s') \in T'(s_j) : a = a'$ and $s = s'$;
      i) $Q_t(s_i, a) \leftarrow \dfrac{t.\lambda \times Q_t(s_i, a) + t'.\lambda \times Q_{t'}(s_j, a')}{t.\lambda + t'.\lambda}$
      ii) $t.\lambda \leftarrow t.\lambda + t'.\lambda$
      iii) $T' \leftarrow T|t'(s_j, a', s')$
   b) $S' \leftarrow S'|s_j$
4) $\forall\ a_i \in A(s_i : \exists\ t(s_i, a, s_*) : s_i, s_* \in S'$, compute $Q(s_i, a)$ (see Equation 2).

Suppose, for example, that policy modification $\triangle_4[P]$ involves removing policy condition c_5 ="MEMORY:utilizationTREND > 0.0" (which corresponds to metric m_4 in Table II) from policy p_4 in Figure 1. The following describes the steps involved in transforming the state-transition model of Figure 8(a), whose states are shown in Table V and correspond to the steps of model-transformation $\Psi_4[G^P_n]$:

1) For each state that has been encountered to date (see Table V), the agent must first remove all the information associated with metric m_4 (see step 1(a)). This would result in a system's model with seven states, each with five metrics; i.e., $M(s_i) = \{m_1, m_2, m_3, m_5, m_6\}$. The agent must also update the states actions sets of each state to ensure that actions set $A'(s_i)$ contains only the actions of the policies that are violated when the system is in state s_i (see Step 1(b)). Note that $\triangle_4[P]$ would result in the violation of policy p_4 when the agent is in states s_2, s_4, or s_6. This was not the case prior to the policy modification since only one of the policy conditions was violated; i.e., $f(R^l_{m_1}) = 0$ while $f(R^l_{m_4}) = 100$. Thus, removing m_4 from state the above states would mean that p_4 is now violated. As such, action a_2 would be added onto actions sets $A(s_2)$, $A(s_4)$, and $A(s_6)$, making states s_2 and s_6 violation states.
2) The state-transition model must also be updated to ensure that the actions within states transitions are consistent with the updated states actions sets. Since the previous stage did not result in the deletion of states actions, no transition will be affected.
3) The agent must then determine whether any states should be merged together if, in fact, the modifications above may have resulted in two or more identical states. Note that the deletion of metric m_4 would mean states s_3 and s_6 are identical; i.e., they have the same region value assignment (i.e., $f(R^l_{m_i})$) for each identical metric. Thus, the two states would be merged together onto state s_{36}, as illustrated in Figure 8. This would also involve merging the states transitions such that transition $t(s_6, a_0\{5\}, s_1)$ becomes transition $t(s_{36}, a_0\{5\}, s_1)$ (see Step 3(a)). In the case of two identical transitions such as $t(s_1, a_6\{2\}, s_6)$ and $t(s_1, a_6\{7\}, s_3)$, a new transition $t(s_1, a_6\{9\}, s_{36})$ would be formed, as illustrated in Figure 8(b), such that the frequency of the new transition is the sum of the frequencies of the two merged transitions. Once the transitions have been updated, the duplicate state is removed (see Step 3(b)).
4) The final step of the adaptation is for the agent to perform backup updates so that any impact on the model as a result of removing transitions and/or merging states is propagated to the action-value estimates of the states actions.

5) Adapting to $\triangle[P]$ Resulting in an Increase in State Metrics: This describes our approach for dealing with policy modifications that increase the size of the metrics set M. From the definitions of Section III-A, this may be in response to policy modification $\triangle_1[P]$.

Model Transformation - $\Psi_5[G^P_n]$: *Adapting to policy modification $\triangle_1[P]$.*

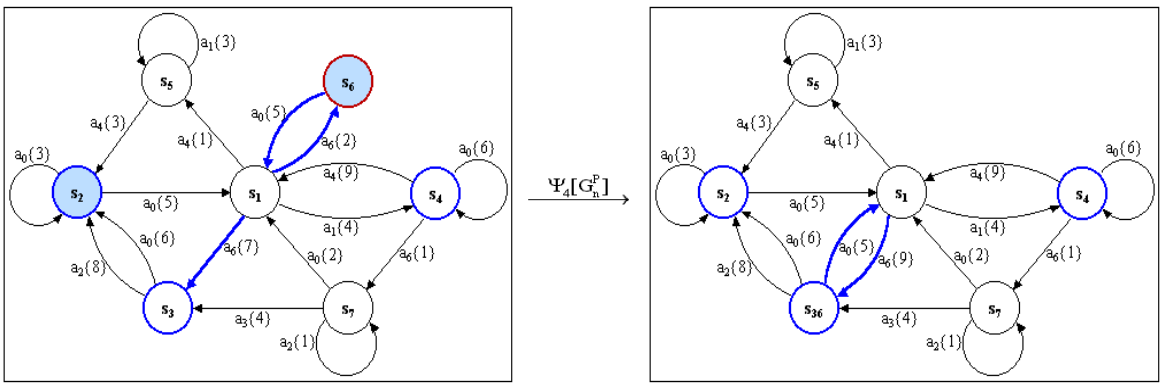
Given:

I) $G^P_n = \langle S, T\rangle$:
- $\forall\ s_i \in S,\ s_i = \langle \mu, M(s_i), P(s_i), A(s_i)\rangle$: $s_i.m_k = \langle \texttt{ID}, \omega, \texttt{value}, R^l_{m_k}\rangle$ where $s_i.m_k \in M(s_i)$
- $\forall\ t(s_i, a, s_j) \in T,\ t(s_i, a, s_j) = \langle \lambda, Q_t(s_i, a)\rangle$

II) Policy modification $\triangle_1[P]$

Then: $G^{P'}_{n+1} = \langle S', T'\rangle$ where

1) $S' \leftarrow \{\emptyset\}$



(a) Original model: $G_n^P = \langle S, T \rangle$ (b) New model: $G_{n+1}^{P'} = \langle S', T' \rangle$

	$f(R^k_{m_j})$					$A(s_i)$		$t(s_i, a_l, s_*)$	
s_i	m_1	m_2	m_3	m_5	m_6	a_l	State action	λ	s_*
s_1	0	0	0	0	0	a_0	γ-action		
						a_1	AdjustMaxClients(+25)	4	s_4
						a_2	AdjustMaxClients(-25)		
						a_4	AdjustMaxKeepAliveRequests(-30)	1	s_5
						a_6	AdjustMaxBandwidth(-128)	7;2	s_3;s_6
s_2	0	100	0	0	100	a_0	γ-action	5;3	s_1;s_2
						a_2	AdjustMaxClients(-25)		
s_{36}	50	100	0	100	100	a_0	γ-action	5;6	s_1;s_2
						a_2	AdjustMaxClients(-25)	8	s_2
s_4	0	100	0	0	0	a_0	γ-action	6	s_4
						a_1	AdjustMaxClients(+25)		
						a_2	AdjustMaxClients(-25)		
						a_4	AdjustMaxKeepAliveRequests(-30)	9	s_1
						a_6	AdjustMaxBandwidth(-128)	1	s_7
s_5	0	0	100	0	0	a_0	γ-action		
						a_1	AdjustMaxClients(+25)	3	s_5
						a_2	AdjustMaxClients(-25)		
						a_4	AdjustMaxKeepAliveRequests(-30)	3	s_2
						a_6	AdjustMaxBandwidth(-128)		
s_7	100	100	100	100	100	a_0	γ-action	2	s_1
						a_2	AdjustMaxClients(-25)	1	s_7
						a_3	AdjustMaxKeepAliveRequests(+30)	4	s_3

(c) System states of the new model.

Fig. 8. State-transition model after $\triangle_4[P]$; i.e., deleting condition c_5 ="MEMORY:utilizationTREND > 0.0" from policy p_4 in Figure 1.

m_i	c_k	Policy Condition	R_{m_i}	$R^j_{m_i}$	$f(R^j_{m_i})$
m_7	10	APACHE:refusedRequests < 10.0	m_7.value $\leq$ 10.0	$R^1_{m_7}$	100
			m_7.value > 10.0	$R^2_{m_7}$	0

TABLE VII
METRICS m_7 STRUCTURE AFTER $\triangle_1[P]$; I.E., ADDING CONDITION c_{10} ="APACHE:refusedRequests < 10.0" ONTO POLICY p_5 IN FIGURE 1.

2) $T' \leftarrow \{\emptyset\}$

Suppose, for example, that policy modification $\triangle_1[P]$ involves adding the condition c_{10} ="APACHE:refusedRequests < 10.0" onto policy p_5 in Figure 1. This would result in a new state metric (i.e., m_7 ="APACHE:refusedRequests") with two regions associated with it, whose structure is shown in Table VII. For a system whose model is derived from visiting the states in Table V, such a change to the policies set P would mean a binary split of each of the seven states to account for the two regions of the new metric. However, since no measurements associated with metrics m_7 would have been collected as was pointed out in Section III-B5, there would be no way of knowing how to map the old transitions onto the new states set. As such, $\triangle_1[P]$ is the only policy modification requiring learning a new model from scratch.

C. A Sequence of Transformations

Consider a more complex example where $\triangle[P]$ involves removing policy p_4 in Figure 1; i.e.,

Given:

I) $PS = \langle P, W_C \rangle$

II) $p_4 = \langle C, A \rangle$ such that $p_4 \in P$:

- $C = \{c_4, c_5\}$:
  a) c_4 = "`MEMORY:utilization > 40.0`" : m_3 = "`MEMORY:utilization`" and where $m_3 \in M : \forall\ m_i \in M|m_3,\ m_i \neq m_3$ and where $\forall\ p_i \in P|p_4,\ c_4 \notin p_i$
  b) c_5 = "`MEMORY:utilizationTREND > 0.0`" : m_4 = "`MEMORY:utilizationTREND`" and where $m_4 \in M : \forall\ m_i \in M|m_4$, $m_i \neq m_4$ and where $\forall\ p_i \in P|p_4$, $c_5 \notin p_i$
- $A = \{a_2\}$: a_2 = `AdjustMaxClients(-25)`

Then:

1) $PS' = \langle P', W_C \rangle : P' \leftarrow P|p_4$

Note that since the policy consists of two conditions and one action, such a policy modification can be modelled as a series of transformations involving the following:

1) Modifying p_4 to create p_4^1 by removing condition c_5 from policy $p_4 = \langle \{c_4, c_5\}, \{a_2\} \rangle$ such that $\triangle_4[P] = P_1$.
2) Modifying p_4^1 to create p_4^2 by removing action a_2 from policy $p_4^1 = \langle \{c_4\}, \{a_2\} \rangle$ such that $\triangle_9[P_1] = P_2$.
3) Modifying p_4^2 to create p_4^3 by removing condition c_4 from policy $p_4^2 = \langle \{c_5\}, \{\emptyset\} \rangle$ such that $\triangle_4[P_2] = P_3$.
4) Removing policy $p_4^3 = \langle \{\emptyset\}, \{\emptyset\} \rangle$ from P_3 such that $\triangle_y[P_3] = P'$.

In the following, we expand on each of these transformations, elaborating on the steps involved in transforming the state-transition model whose states are shown in Table V.

The first transformation involves adapting the state-transition model as a result of removing policy condition c_5 ="`MEMORY:utilizationTREND > 0.0`" from policy p_4, which relates to policy modification $\triangle_4[P]$; i.e.,

Policy Modification - $\triangle_4[P]$: *Removing a policy condition resulting in a decrease in the number of metrics.*

Given:

I) $PS = \langle P, W_C \rangle$
II) $M_R^P = \{r_1, r_2, \ldots, r_l\} : r_i = \langle \alpha_i, \sigma_i \rangle$
III) $p_4 = \langle \{c_4, c_5\}, \{a_2\} \rangle$ such that $p_4 \in P$ where $\forall\ p_i \in P|p_4,\ c_5 \notin p_i$
IV) c_5 = "`MEMORY:utilizationTREND > 0.0`" : m_4 = "`MEMORY:utilizationTREND`" and where $m_4 \in M : \forall\ m_i \in M|m_4,\ m_i \neq m_4$
V) $r_{m_4} \in M_R^P : r_{m_4} = \langle \alpha_{m_4}, \sigma_{m_4} \rangle$:
- $\alpha_{m_4} = \langle 4, m_4, \omega \rangle$
- $\sigma_{m_4} = \{0.0\}$

Then:

1) $PS_1 = \langle P_1, W_C \rangle : p_4^1 = \langle \{c_4\}, \{a_2\} \rangle$
2) $M_1 \leftarrow M|m_4$
3) $M_R^{P_1} \leftarrow M_R^P|r_{m_4}$

Such a modification can be modelled using model-transformation $\Psi_4[G_n^P]$, as was illustrated in Section III-B4 (see Figure 8).

Suppose that we apply the second transformation (i.e., in response to removing policy action a_2 = "`AdjustMaxClients(-25)`" from policy p_4^1) onto the state-transition model of Figure 8(b) whose states are shown in Figure 8(c). Note that such a change relates to policy modification $\triangle_9[P_1]$; i.e.,

Policy Modification - $\triangle_9[P_1]$: *Removing an action from a policy in P_1.*

Given:

I) $PS_1 = \langle P_1, W_C \rangle$
II) $M_R^{P_1} = \{r_1, r_2, \ldots, r_l\} : r_i = \langle \alpha_i, \sigma_i \rangle$
III) $p_4^1 = \langle \{c_4\}, \{a_2\} \rangle$ such that $p_4^1 \in P_1$
IV) a_2 = `AdjustMaxClients(-25)`
V) $r_m \in M_R^P : r_m = \langle \alpha_m, \sigma_m \rangle$:
- $\alpha_m = \langle$ `ID`, `metricName`, $\omega \rangle$
- $\sigma_m = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_k\}$

Then:

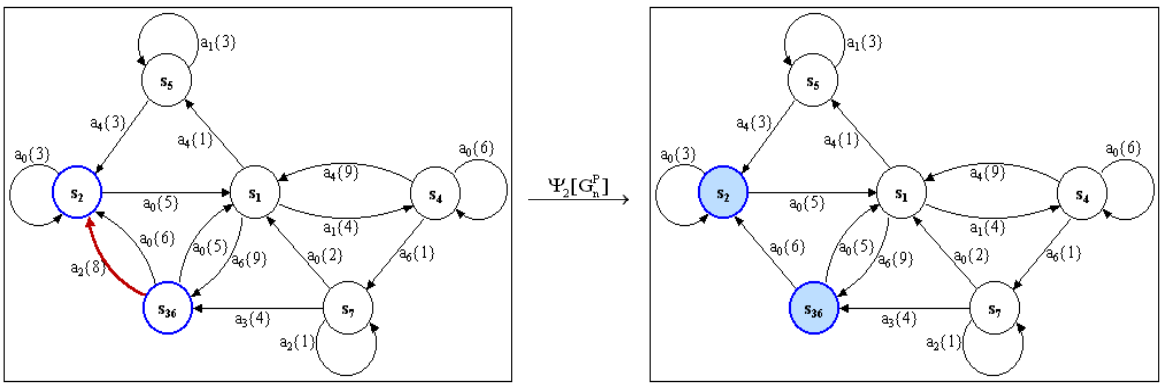
1) $PS_2 = \langle P_2, W_C \rangle : p_4^2 = \langle \{c_4\}, \{\emptyset\} \rangle$
2) $M_2 \leftarrow M_1$
3) $M_R^{P_2} \leftarrow M_R^{P_1}$

Such a modification can be modelled using model-transformation $\Psi_2[G_n^P]$. The following describes the steps involved in transforming the model and corresponds to the steps of model-transformation $\Psi_2[G_1^{P_1}]$:

1) For each state that has been encountered to date (see figure 8(c)), the agent must update the actions within the states to ensure that actions set $A'(s_i)$ contains only the actions of the policies that are violated when the system is in state s_i. Note that since policy p_4^1 is the only violated policy in states s_2 and s_{36}, $\triangle_9[P^1]$ would result in the removal of action a_2 from the actions sets $A(s_2)$ and $A(s_{36})$. While policy p_4^1 is also violated in states s_1 and s_4, $\triangle_9[P^1]$ would not affect such states. This is because action a_2 is also part of other violated policies in those states, such as policy p_1 (i.e., $a_2 \in p_1$) and policy p_2 (i.e., $a_2 \in p_2$) where $p_1 \in P'(s_1)$ and $p_2 \in P'(s_4)$. Consequently, both states s_2 and s_{36} will change from "violation" to "non-violation" states.
2) The state-transition model must also be updated to ensure that transitions within states are consistent with the updated states actions sets. Note that the removal of action a_2 from state s_{36} in the above stage would mean that transition $t(s_{36}, a_2\{8\}, s_2)$ (see Figure 9(a)) is no longer valid. As such, the transition will be removed as illustrated in Figure 9(b).
3) The final step of the adaptation is for the agent to perform backup updates so that any impact on the model as a result of removing transitions is propagated to the action-value estimates of the states actions.

Suppose that, we apply the third transformation (i.e., in response to removing policy condition c_4 ="`MEMORY:utilization > 40.0`" from policy p_4^2) onto the state-transition model of Figure 9(b), whose states are shown in Figure 9(c). Note that such a change relates to policy modification $\triangle_4[P_2]$; i.e.,

Policy Modification - $\triangle_4[P]$: *Removing a policy condition resulting in a decrease in the number of metrics.*



(a) Original model: $G_n^P = \langle S, T \rangle$

(b) New model: $G_{n+1}^{P'} = \langle S', T' \rangle$

	$f(R_{m_j}^k)$					$A(s_i)$		$t(s_i, a_l, s_*)$	
s_i	m_1	m_2	m_3	m_5	m_6	a_l	State action	λ	s_*
s_1	0	0	0	0	0	a_0	γ-action		
						a_1	`AdjustMaxClients(+25)`	4	s_4
						a_2	`AdjustMaxClients(-25)`		
						a_4	`AdjustMaxKeepAliveRequests(-30)`	1	s_5
						a_6	`AdjustMaxBandwidth(-128)`	7;2	s_3;s_6
s_2	0	100	0	0	100	a_0	γ-action	5;3	s_1;s_2
s_{36}	50	100	0	100	100	a_0	γ-action	5;6	s_1;s_2
s_4	0	100	0	0	0	a_0	γ-action	6	s_4
						a_1	`AdjustMaxClients(+25)`		
						a_2	`AdjustMaxClients(-25)`		
						a_4	`AdjustMaxKeepAliveRequests(-30)`	9	s_1
						a_6	`AdjustMaxBandwidth(-128)`	1	s_7
s_5	0	0	100	0	0	a_0	γ-action		
						a_1	`AdjustMaxClients(+25)`	3	s_5
						a_2	`AdjustMaxClients(-25)`		
						a_4	`AdjustMaxKeepAliveRequests(-30)`	3	s_2
						a_6	`AdjustMaxBandwidth(-128)`		
s_7	100	100	100	100	100	a_0	γ-action	2	s_1
						a_2	`AdjustMaxClients(-25)`	1	s_7
						a_3	`AdjustMaxKeepAliveRequests(+30)`	4	s_3

(c) System states of the new model.

Fig. 9. Adapting state-transition model of Figure 8 to $\triangle_9[P_1]$; i.e., removing action a_2 ="`AdjustMaxClients(-25)`" from policy p_4^1.

Given:

I) $PS_2 = \langle P_2, W_C \rangle$

II) $M_R^{P_2} = \{r_1, r_2, \ldots, r_l\} : r_i = \langle \alpha_i, \sigma_i \rangle$

III) $p_4^2 = \langle \{c_4\}, \{\emptyset\} \rangle$ such that $p_4^2 \in P^2$ where $\forall\ p_i \in P_2 | p_4^2,\ c_4 \notin p_i$

IV) $c_4 =$ "`MEMORY:utilization > 40.0`" : $m_3 =$ "`MEMORY:utilization`" and where $m_3 \in M : \forall\ m_i \in M_2 | m_3,\ m_i \neq m_3$

V) $r_{m_3} \in M_R^{P_2} : r_{m_3} = \langle \alpha_{m_3}, \sigma_{m_3} \rangle$:

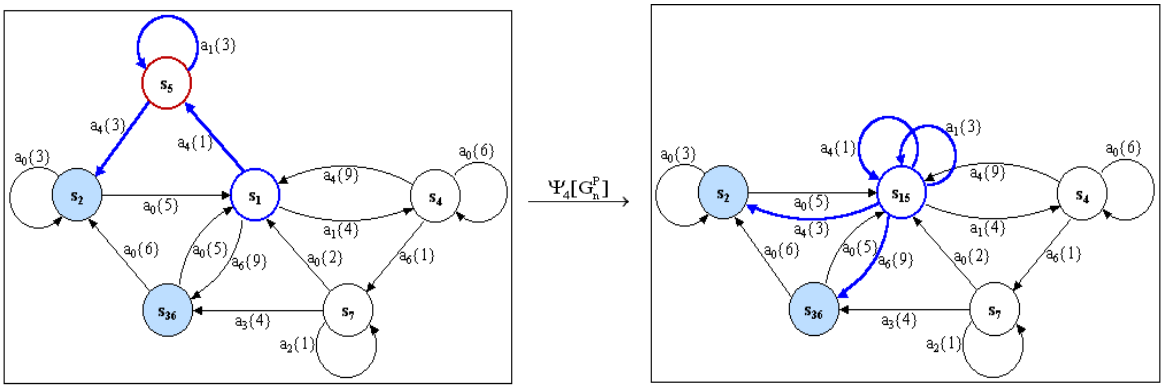
- $\alpha_{m_3} = \langle 3, m_3, \omega \rangle$
- $\sigma_{m_3} = \{40.0\}$

Then:

1) $PS_3 = \langle P_3, W_C \rangle : p_4^3 = \langle \{\emptyset\}, \{\emptyset\} \rangle$
2) $M_3 \leftarrow M_2 | m_3$
3) $M_R^{P_3} \leftarrow M_R^{P_2} | r_{m_3}$

Such a modification can be modelled using model-transformation $\Psi_4[G_n^P]$. The following describes the steps involved in transforming the model and corresponds to the steps of model-transformation $\Psi_4[G_2^P]$:

1) For each state that has been encountered to date (see Table 9(c)), the agent must first remove all the information associated with metric m_3 (see step 1(a)). This would result in a system's model with six states each with four metrics; i.e., $M(s_i) = \{m_1, m_2, m_5, m_6\}$. The agent must also update the states actions sets of each state to ensure that actions set $A'(s_i)$ contains only the actions of the policies that are violated when the system is in state s_i (see Step 1(b)). Since $\triangle_4[P_2]$ is the third step in the sequence of changes to the policy p_4 (which, at this point, would have no actions associated with it), such a policy modification would result in no changes to the composition of the states actions.
2) The state-transition model must also be updated to ensure that the actions within states transitions are consistent with the updated states actions sets. Since the previous stage did not result in the deletion of states actions, no transition will be affected.
3) The agent must then determine whether any states should be merged together if, in fact, the modifications above may have resulted in two or more identical



(a) Original model: $G_n^P = \langle S, T \rangle$

(b) New model: $G_{n+1}^{P'} = \langle S', T' \rangle$

	$f(R_{m_j}^k)$				$A(s_i)$		$t(s_i, a_l, s_*)$	
s_i	m_1	m_2	m_5	m_6	a_l	State action	λ	s_*
s_{15}	0	0	0	0	a_0	γ-action		
					a_1	AdjustMaxClients(+25)	3;4	s_{15};s_4
					a_2	AdjustMaxClients(-25)		
					a_4	AdjustMaxKeepAliveRequests(-30)	1;3	s_{15}; s_2
					a_6	AdjustMaxBandwidth(-128)	9	s_{36}
s_2	0	100	0	100	a_0	γ-action	5;3	s_{15};s_2
s_{36}	0	100	100	100	a_0	γ-action	5;6	s_{15};s_2
s_4	0	100	0	0	a_0	γ-action	6	s_4
					a_1	AdjustMaxClients(+25)		
					a_4	AdjustMaxKeepAliveRequests(-30)	9	s_{15}
					a_6	AdjustMaxBandwidth(-128)	1	s_7
s_7	100	100	100	100	a_0	γ-action	2	s_{15}
					a_2	AdjustMaxClients(-25)	1	s_7
					a_3	AdjustMaxKeepAliveRequests(+30)	4	s_{36}

(c) System states of the new model.

Fig. 10. Adapting state-transition model of Figure 9 to $\triangle_4[P_2]$; i.e., removing condition c_4 ="MEMORY:utilization > 40.0" from policy p_4^2.

states. Note that the deletion of metric m_3 would mean states s_1 and s_5 are identical; i.e., they have the same region value assignment (i.e., $f(R_{m_i}^l)$) for each identical metric. Thus, the two states would be merged together into state s_{15} as illustrated in Figure 10. This would also involve merging the states transitions (see Step 3(a)) such that, transitions $t(s_1, a_4\{1\}, s_5)$, $t(s_5, a_1\{3\}, s_5)$, and $t(s_5, a_4\{3\}, s_2)$ (see Figure 10(a)) become transitions $t(s_{15}, a_4\{1\}, s_{15})$, $t(s_{15}, a_1\{3\}, s_{15})$, and $t(s_{15}, a_4\{3\}, s_2)$, respectively (see Figure 10(b)).

4) The final step of the adaptation is for the agent to perform backup updates so that any impact on the model as a result of removing transitions and/or merging states is propagated to the action-value estimates of the states actions.

Finally, we apply the fourth transformation (i.e., in response to removing policy p_4^3) onto the state-transition model of Figure 10(b), whose states are shown in Figure 10(c). Note that such a change relates to policy modification $\triangle_y[P_3]$; i.e.,

Policy Modification - $\triangle_y[P_3]$: *Removing a policy without conditions or actions.*

Given:

I) $PS_3 = \langle P_3, W_C \rangle$

II) $M_R^{P_3} = \{r_1, r_2, \ldots, r_l\} : r_i = \langle \alpha_i, \sigma_i \rangle$

III) $p_4^3 = \langle \emptyset, \emptyset \rangle$ such that $p_4^3 \in P_3$

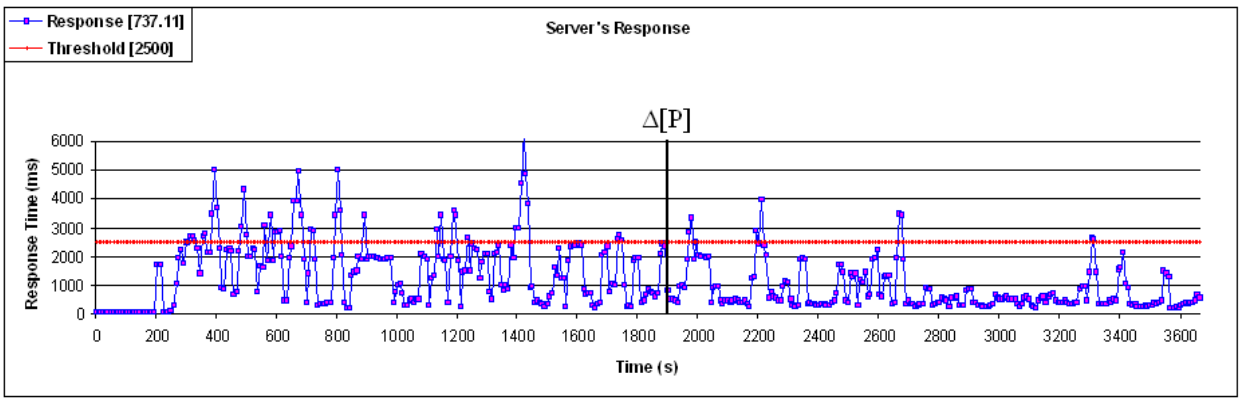
Then:

1) $PS' = \langle P', W_C \rangle : P' \leftarrow P_3 | p_4^3$
2) $M' \leftarrow M_3$
3) $M_R^{P'} \leftarrow M_R^{P_3}$

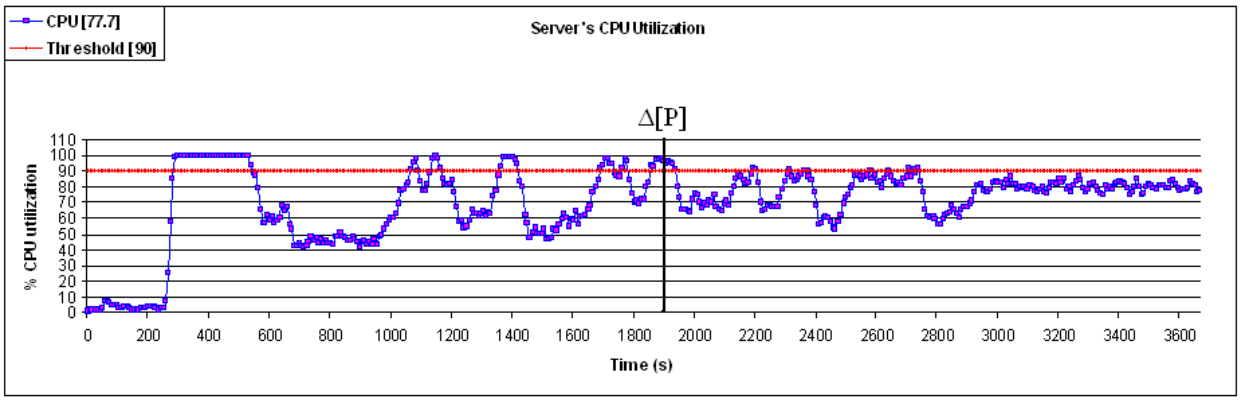
Since $\triangle_y[P_3]$ has no impact on the state-transition model, removing policy p_4^3 from Figure 1 would result in no changes to the state-transition model of Figure 10. As such, the removal of policy p_4 can be modelled as a sequence of transformations on the original state-transition model involving: $\triangle_4[P] = P_1$, $\triangle_9[P_1] = P_2$, $\triangle_4[P_2] = P_3$, and $\triangle_y[P_3] = P'$.

IV. Experience

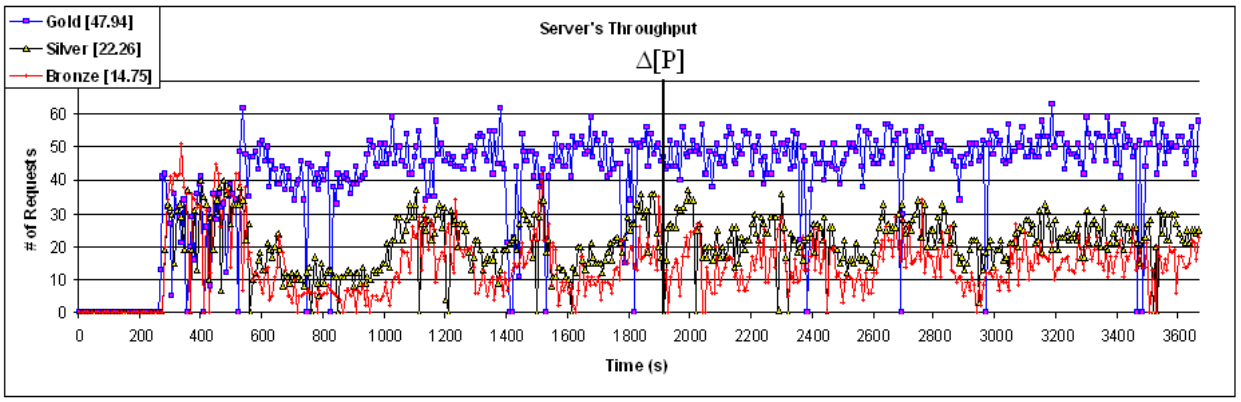
In this section, we present experiments illustrating the effectiveness of the adaptation mechanisms on the behavior of the server in response to run-time policy modifications. In particular, we compare results from two experimental settings based on how the agent responded to run-time policy modifications. The first experiment (*EXP-1*) investigated the behavior of the server where the learning agent discards everything it may have learned prior to $\triangle[P]$ and begins learning from scratch; i.e., $G_{n+1}^{P'} = \{\emptyset, \emptyset\}$. The second experiment (*EXP-2*) looked at the behavior of the server where the learning agent adapts the model of the environment dynamics to take into account the changes to an active set of policies, essentially



(a) Response Time



(b) CPU Utilization



(c) Throughput

Fig. 11. Adapting to run-time policy modification: $\triangle[P]$ involves replacing "mixed" expectation policies with "simple" expectation policies

reusing some of the learned information; i.e., $\Psi : G_n^P \rightarrow G_{n+1}^{P'}$ (see Figure 3).

A. Testbed Environment

Our testbed consisted of a collection of networked workstations: an administrative workstation to run the experiments; a Linux workstation with a 2.0 GHz processor and 2.0 Gb of memory, which hosted the Apache Web Server along with the PHP module and the MySQL database server; and three workstations used for generating the server's workload, which consisted of clients requests associated with gold, silver, and bronze service classes.

To emulate the stochastic behavior of users, an Apache benchmark tool called JMeter [12] was used. The tool provides support for concurrent sampling of requests through a multi-threaded framework. It provides the ability to specify the client's think time as well as the number of concurrent - independent - *keep-alive* connections. In the experiments reported here, these values were set to ensure that the server was under overload conditions (i.e., saturated) for the entire duration of the experiments. In our current implementation, the tool has been configured to traverse a Web graph of an actual Web site: in our case, phpBB [13]. In the experiments reported in this section, only *read-only* database requests were

considered.

In order to support service differentiation, a Linux Traffic Controller (TC) was used to configure the bandwidth associated with the gold, silver, and bronze service classes. The service classes bandwidth was assigned proportionately according to the ratio 85:10:5. Bandwidth sharing was also permitted. The tuning parameter `MaxBandwidth` is what determines how much bandwidth is assigned to each service class. It corresponds to the physical capacity (in kbps) of the network connection to the workstation hosting the servers. Thus, given that the first policy of Figure 1 has been violated and that it is no longer possible, for example, to adjust the parameters `MaxClients` and `MaxKeepAliveRequests`, then the last policy action (i.e., `AdjustMaxBandwidth(-128)`) would be executed. This action essentially reduces the total bandwidth by 128 kbps. The percentage of the new bandwidth is what is eventually assigned to the different service classes.

B. Results

The results presented in this section aim at demonstrating the effectiveness of the adaptation mechanisms on the policy modifications involving changing policy sets. Two sets of policies were considered and included "simple" and "mixed" expectation policies. Simple expectation policies included policies describing possible sets of actions to be taken whenever a single objective is violated. Mixed expectation policies include policies consisting of both "simple" and "complex" policies, with complex expectation policies describing the actions of the system whenever several objectives are violated. $\triangle[P]$, in this case, involved replacing "mixed" expectation policies with "simple" expectation policies; i.e., disabling all the complex policies within the mixed expectation policies set. The performance comparisons in terms of the averages and magnitude of violations reported in this section focused specifically on the behavior of the server after $\triangle[P]$ occurred.

Figure 11 shows the behavior of the server when dealing with the above policy modification, which occurred at about 1,900 seconds after the start of the experiment. In terms of response time measurements (see Figure 11(a)), the quality of service specific to the violations as well as the stability of the measurements greatly improved after $\triangle[P]$ occurred. This is clearly visible in the graph where there are very few instances where the measurements exceeded the threshold. The same is also true when CPU utilization and throughput measurements are considered, as Figures 11(b) and 11(c), respectively, illustrate. Thus, the overall quality of service greatly improved as a result of the adaptation mechanisms.

The top graph of Figure 12 compares the average throughput measurements of identical service classes. *EXP-1* corresponded to the case where the learning agent discarded everything it had learned prior to $\triangle[P]$ and began learning from scratch, while *EXP-2* corresponded to the experiment where the agent adapted the model in response to the policy modification. Thus, results reported for *EXP-2* are essentially those of Figure 11, but whose averages only include the measurements after $\triangle[P]$ occurred. From these results, the server performed slightly better in *EXP-2* when compared to the results in *EXP-1*, particularly for the gold service class where more requests were served. For the other classes (i.e., silver and bronze), however, the performance gains were statistically insignificant since the measurements fell within the standard error, as is illustrated by the error bars associated with the mean throughput.

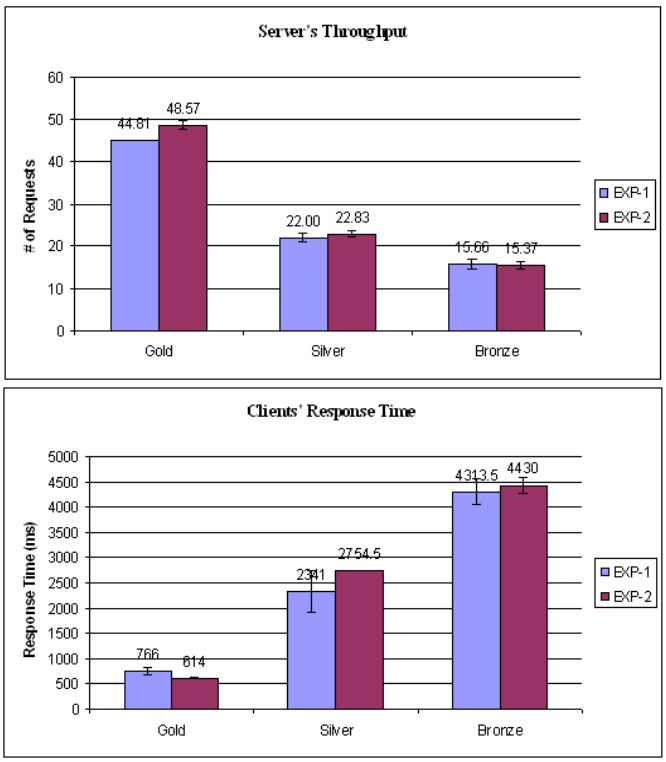
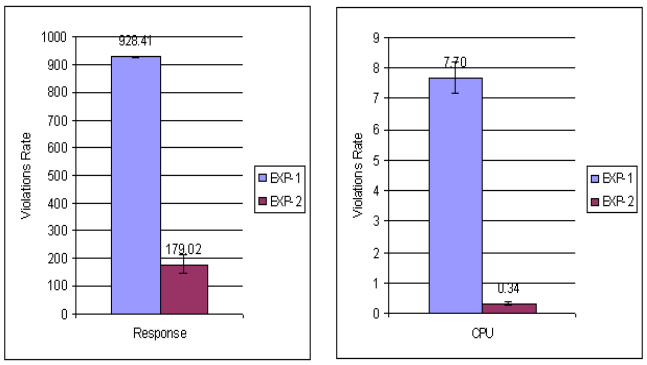


Fig. 12. Mean performance comparisons.

The same was also true when clients response time measurements across the service classes were compared as shown in the bottom graph of Figure 12. Thus, we can conclude that, model adaptation had a positive impact on the overall performance of the server when averages associated with the server's throughput and clients response times were considered. That is, the server performed slightly better in *EXP-2* when compared to the results in *EXP-1*, particularly for the gold service class where more requests were served. For the other classes (i.e., silver and bronze), however, the performance differences were statistically insignificant since the measurements fell within the standard error.

While the results above show slight improvement in the overall quality as a result of the adaptation mechanisms, mean performance is essentially an approximation of the quality of service performance of the system. The more accurate measure is the severity and the frequency with which performance objectives are violated, as summarized in Figure 13. Note that, *EXP-2* recorded at least an 80% reduction in the magnitude of response time violations (see Figure 13(a)) and a 96% reduction in the magnitude of CPU utilization violations (see Figure 13(b)). Thus, reducing the magnitude of violations such as instances in which clients requests wait on a server's queue for longer than 2500 ms, for example, is likely to



(a) Response Time (b) CPU Utilization

Fig. 13. Magnitude of violations performance comparisons.

improve the quality of service from a user's prospective - a measure mean performance tend to obscure. Furthermore, performance objective under these kinds of environments often need to account for not only customers' needs, but also system's constraints. For example, in order to control power consumption, an autonomic system may need to limit CPU utilization. Thus, the ability to achieve system-wide performance objectives, which may seem conflicting at times, under dynamically changing configuration conditions is what the experiments presented in this paper have demonstrated.

Furthermore, the results summarized in this section highlight one of the key benefits of utilizing reinforcement learning methodologies in policy-driven autonomic management, particularly when it comes to policy specification. That is, very "simple" expectation policies are sufficient to deliver significant performance gains. With learning enabled, more complex policies, particularly those spanning multiple objectives, could be learned instead of requiring systems administrators to manually encode into a single policy how systems should achieve multiple objectives. Thus, a general knowledge of the performance impact due to a change in the value of a parameter is sufficient to define simple but effective policies. This can significantly reduce the burden faced by human administrators, by having them specify simple policies, leaving it to autonomic systems to figure out how to achieve more complex objectives.

V. Related Work

Policy-based management approaches have attracted significant interest within autonomic computing. The work in [14], for example, proposes a flexible policy-expression language called AGILE [15], which facilitates run-time policy configuration and adaptation of autonomic systems. AGILE in itself is both a policy expression language and a framework that facilitates dynamic composition of autonomic techniques including signal processing, trend analysis, and utility functions. IBM [6] has also been at the forefront in policy-driven autonomic management research. Their work in [16] proposes an Agent Building and Learning Environment (ABLE) capable of configuring new behaviors and capabilities to intelligent autonomic systems. In this approach, policies are specified using rule beans which make use of simple scripting text or Extensible Markup Language (XML) [17] to define simple assignments including if-then, if-then-else rules, when-do pattern match rules, etc.

Policy-based management has recently evolved to recognize the need to evaluate the correctness and effectiveness of policies specified by system administrators. For example, there has been some interest in policy conflicts detection and resolution (see, for example, [18]). Others have proposed frameworks that make use of *event calculus* (EC) [19] for policy verifications which adds the notion of time for first-order predicate logic. It provides a formalism for representation and reasoning about actions and their effects. This approach, however, requires that rules are specified to identify all possible inconsistencies from a set of policies, which is intractable for complex systems. This is particularly the case because, in most situations, some of the characteristics for identifying conflicts can only be detected at run-time.

The need to dynamically adapt the use of policies in autonomic management has also gained some traction. The work in [20], for example, proposes an adaptive policy-based framework that support dynamic policy configuration in response to changes within the managed environment. In their approach, policy adaptation describes the ability to modify network behavior by dynamically changing the policy parameters as well as selecting, enabling, or disabling policies at run-time. Reconfiguration events are used to trigger high-level control policies, which then determines which lower-level policies must be adapted to reconfigure the managed system. They use the Ponder deployment framework [21] to distribute policies to the different management components. They describe several ways in which the system's behavior could be adapted by changing the way policies are used [20]:

- *Dynamic Modification of Policy Parameters:* In this approach, the system can adapt policies by making changes to the policy parameters. This may involve computing new attribute values based, for example, on Service Level Agreements (SLAs), resource availability, etc. Related work on this type of adaptation in the use of policies can be found in [20], [22].
- *Enabling/Disabling Policies From a Set of Active Policies:* In this approach, the system can use run-time context to dynamically use the information provided by the policies, such as enabling/disabling a policy under certain circumstances or selecting actions based on context. Related work on adaptation through policy selection can be found in [20], [23].
- *Adaptation By Learning:* In this approach, the system can determine policy use through some learning mechanisms based on past experience in the use of policies. This type of adaptation is still in its very early stages, particularly in the filed of autonomic management, with the majority of the research work focusing on learning high-level policies. The most recent work can be found in [24], [25], [26], [27].

Our interest is on how learning approaches could facilitate dynamic use of policies within autonomic computing.

In our most recent work [7], we have explored the use of Reinforcement Learning methodologies in providing guidance to the autonomic system in terms of how to effectively use existing policies. One drawback to this and other approaches was that when policies change, the existing model is discarded and the system must learn a new model of the environment from scratch. Our experience, however, suggest that policy modifications are frequently "incremental" - e.g., a change to a threshold, the disabling of a single policy, etc. The ability to reuse a learned model, or part there of, has the potential benefit of significantly accelerating the learning process, as this paper has demonstrated.

VI. Conclusion

The approach taken in this paper is only the first step in the broader goal of developing adaptive policy driven autonomic solutions involving multiple agents working together towards a set of common, and at times seemingly competing, objectives. This is often the case since performance objectives may need to account for both customers needs and systems constraints. The complexity of today's IT infrastructure, however, means that we can no longer depend on centralized approaches to performance management. It is becoming increasingly common to find multiple agents coexisting within a single computing environment. The work in [28], for example, proposes an approach for coordinating two independent autonomic managers: one designed to address power consumption objectives while another deals with performance objectives such as response time based on some learning mechanisms. As such, policy modifications directives could come not only from the users of the systems, but also from other autonomic managers. The fact that our approach to modelling the learning process is dependent only on the structure of the policies means that such changes to policies could be traced back to the model derived from the use of those policies. In this paper, we have elaborated on how a model learned from the use of an active set of policies could be adapted (i.e., reused) to cope with run-time policy modifications.

While the initial results are encouraging, we intend to validate these approaches with a comprehensive prototype evaluating the impact of the different kinds of policy modifications on the effectiveness of the adaptation transformations. We are also working on formulating formal proofs that show that, indeed if P is changed to P' and G_n^P is transformed to $G_{n+1}^{P'}$, then $G_{n+1}^{P'}$ is a model of P'.

References

[1] N. Damianou, N. Dulay, E. C. Lupu, and M. S. Sloman, "Ponder: A Language for Specifying Security and Management Policies for Distributed Systems: The Language Specification," Imperial College, London, England, Version 2.2, 2000.

[2] (2009, December) The Internet Engineering Task Force (IETF). [Online]. Available: http://www.ietf.org/

[3] (2009, December) The Distributed Management Task Force (DMTF). [Online]. Available: http://www.dmtf.org/

[4] (2009, December) The Object Management Group (OMG). [Online]. Available: http://www.omg.org/

[5] (2009, December) Policy Research Group. Imperial College. [Online]. Available: http://www-dse.doc.ic.ac.uk/Research/policies/ponder.shtml

[6] (2009, December) Autonomic Computing Research. IBM. [Online]. Available: http://www.research.ibm.com/autonomic/

[7] R. M. Bahati and M. A. Bauer, "Modelling Reinforcement Learning in Policy-driven Autonomic Management," in *IARIA International Journal On Advances in Intelligent Systems*, vol. 1, no. 1, December 2008.

[8] R. M. Bahati and M. A. Bauer, "Adapting to Run-time Changes in Policies Driving Autonomic Management," in *International Conference on Autonomic and Autonomous Systems (ICAS'08)*, Guadeloupe, March 2008, pp. 88–93.

[9] R. M. Bahati, M. A. Bauer, C. Ahn, O. K. Baek, and E. M. Vieira, "Using Policies to Drive Autonomic Management," in *WoWMoM-2006 Workshop on Autonomic Communications and Computing (ACC'06)*, Buffalo, NY, USA, June 2006, pp. 475–479.

[10] R. S. Sutton and A. G. Barto, *Reinforcement Learning: an Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[11] C. Watkins, "Learning from Delayed Rewards," Ph.D. dissertation, Cambridge University, Cambridge, UK, 1989.

[12] (2009, December) Apache JMeter. The Apache Software Foundation. [Online]. Available: http://jakarta.apache.org/jmeter/

[13] (2009, December) PHP Bulletin Board. [Online]. Available: http://www.phpbb.com/

[14] R. J. Anthony, "Policy-centric Integration and Dynamic Composition of Autonomic Computing Techniques," in *International Conference on Autonomic Computing (ICAC'07)*, Jacksonville, FL, USA, June 2007.

[15] (2009, December) AGILE. [Online]. Available: http://www.policyautonomics.net/

[16] J. P. Bigus, D. A. Schlosnagle, J. R. Pilgrim, W. N. M. III, and Y. Diao, "ABLE: A toolkit for Building Multiagent Autonomic Systems," in *IBM Systems Journal*, vol. 41, no. 3, September 2002.

[17] (2009, December) Extensible Markup Language (XML). [Online]. Available: http://www.w3.org/XML/

[18] N. Dunlop, J. Indulska, and K. Raymond, "Dynamic Conflict Detection in Policy-Based Management Systems," in *International Enterprise Distributed Object Computing Conference (EDOC'02)*, Washington, DC, USA, 2002, pp. 15–26.

[19] R. Kowalski and M. Sergot, "A Logic-based Calculus of Events," in *New Generation Computing*, vol. 4, no. 1, Tokyo, Japan, 1986, pp. 67–95.

[20] L. Lymberopoulos, E. C. Lupu, and M. S. Sloman, "An Adaptive Policy-Based Framework for Network Services Management," in *Journal of Networks and Systems Management*, vol. 11, no. 3, 2003, pp. 277–303.

[21] N. Dulay, E. C. Lupu, M. S. Sloman, and N. Damianou, "A Policy Deployment Model for the Ponder Language," in *IEEE/IFIP International Symposium on Integrated Network*, Seattle, CA, USA, May 2001, pp. 529–544.

[22] K. Yoshihara, M. Isomura, and H. Horiuchi, "Distributed Policy-based Management Enabling Policy Adaptation on Monitoring using Active Network Technology," in *IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM'01)*, Nancy, France, October 2001.

[23] R. M. Bahati, M. A. Bauer, and E. M. Vieira, "Adaptation Strategies in Policy-Driven Autonomic Management," in *International Conference on Autonomic and Autonomous Systems (ICAS'07)*, July 2007.

[24] G. Tesauro, N. K. Jong, R. Das, and M. N. Bennani, "A Hybrid Reinforcement Learning Approach to Autonomic Resource Allocation," in *International Conference on Autonomic Computing (ICAC'06)*, Dublin, Ireland, June 2006, pp. 65–73.

[25] G. Tesauro, "Online Resource Allocation Using Decompositional Reinforcement Learning," in *National Conference on Artificial Intelligence (AAAI'05)*, Pittsburgh, PA, USA, 2005, pp. 886–891.

[26] D. Vengerov and N. Iakovlev, "A Reinforcement Learning Framework for Dynamic Resource Allocation: First Results," in *International Conference on Autonomic Computing (ICAC'05)*, Seattle, WA, USA, January 2005, pp. 339–340.

[27] P. Vienne and J.-L. Sourrouille, "A Middleware for Autonomic QoS Management based on Learning," in *International Conference on Sofware Engineering and Middleware*, Lisbon, Portugal, September 2005, pp. 1–8.

[28] J. O. Kephart, H. Chan, R. Das, D. W. Levine, G. Tesauro, F. Rawson, and C. Lefurgy, "Coordinating Multiple Autonomic Managers to Achieve Specified Power-performance Tradeoffs," in *International Conference on Autonomic Computing (ICAC'07)*, Jacksonville, FL, USA, June 2007.