Reinforcement Learning for Autonomic Network Repair

Michael L. Littman and Nishkam Ravi
Department of Computer Science
Rutgers University
Piscataway, NJ
{mlittman,nravi}@cs.rutgers.edu

Eitan Fenson and Rich Howard
PnP Networks, Inc.
Los Altos, CA
{eitan,reh}@pnphome.com

Abstract

We report on our efforts to formulate autonomic network repair as a reinforcement-learning problem. Our implemented system is able to learn to efficiently restore network connectivity after a failure.

Our research explores a reinforcement-learning (Sutton & Barto 1998) formulation we call *cost-sensitive fault remediation* (CSFR), which was motivated by problems that arise in sequential decision making for diagnosis and repair. We have considered problems of web-server maintenance and disk-system replacement, and have fully implemented an experimental network-repair application.

In cost-sensitive fault remediation, a decision maker is responsible for repairing a system when it breaks down. To narrow down the source of the fault, the decision maker can perform a *test action* at some cost, and to repair the fault it can carry out a *repair action*. A repair action incurs a cost and either restores the system to proper functioning or fails. In either case, the system informs the decision maker of the outcome. The decision maker seeks a minimum cost policy for restoring the system to proper functioning.

We can find an optimal repair policy via dynamic programming. Let B be the power set of the set of fault states S, which is the set of belief states of the system. For each $b \in B$, define the expected value of action a in belief state s as the expected cost of the action plus the value of the resulting belief state:

$$Q(b,a) = \begin{cases} \Pr(b_0)/\Pr(b)(c(b_0,a) + V(b_0)) \\ \quad + \Pr(b_1)/\Pr(b)(c(b_1,a) + V(b_1)), \\ \quad \text{if } \Pr(b_0) > 0 \text{ and } \Pr(b_1) > 0, \\ \quad \text{or } \Pr(b_1) > 0 \text{ and } a \in A_R; \\ \infty, \text{ otherwise.} \end{cases}$$

Here, b_i is the belief state resulting from taking action a in belief state b and obtaining outcome $i \in \{0, 1\}$; it is the subset of b consistent with this outcome. If a is a repair action and $i = 1$, we define the future value $V(b_1) = 0$, as

	A		B	
DnsLookup	0	(2500ms)	0	(2500ms)
DefaultGateway	0	(50ms)	1	(50ms)
PingIp	0	(50ms)	1	(250ms)
RenewLease	1	(2500ms)	0	(1000ms)
UseCachedIP	0	(10000ms)	1	(25000ms)
FixIP	1	(20000ms)	1	(20000ms)

Table 1. A small table of CSFR states. Each row lists the outcome and cost of a test action (DnsLookup, DefaultGateway, and PingIP) or repair action (FixIP, UseCachedIP, RenewLease, in boldface) for each of the states. The priors are $\Pr(A) = .25$ and $\Pr(B) = .75$.

there is no additional cost incurred once a repair action is successful. In all other cases, the value of a belief state is the minimum action value taken over all available choices: $V(b) = \min_a Q(b,a)$. The quantities $\Pr(b)$ and $c(b,a)$ are the prior probability of a belief state and expected cost of an action, which can be computed easily from a CSFR specification of the problem.

Table 1 illustrates a small CSFR example with two fault states, A and B. The planning process for this example begins with the belief state $\{A, B\}$. It considers the test actions DefaultGateway and PingIP and the repair actions **FixIP**, **UseCachedIP**, and **RenewLease**. It does not consider DnsLookup since the action neither provides information (always 0), nor has a non-zero chance of repair. In evaluating the action PingIP, the algorithm finds that outcome 0 has a probability of .25 and outcome 1 has a probability of .75. Its expected cost from belief state $\{A, B\}$ is then

$$.25(50 + \text{cost}(\{A\})) + .75(250 + \text{cost}(\{B\})). \quad (1)$$

The expected cost from belief state $\{A\}$ is computed recursively. Since all test actions have an outcome with an estimated probability of 0, only repair actions are considered. Of these, **RenewLease** is chosen as the optimal action, with
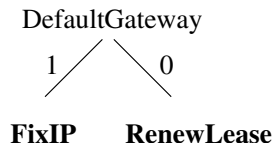
Figure 1. The optimal policy for the small example CSFR.

Figure 2. A learned policy for the network-repair domain.

a cost of 2500. Similarly, the optimal expected cost from belief state $\{B\}$ is 20000 by taking action **FixIP**. Substituting these costs into Equation 1, we find the optimal expected cost of repair from belief state $\{A, B\}$, starting with PingIP, is 15825. The minimum over all actions from belief state $\{A, B\}$ is achieved by DefaultGateway (expected cost 15675), leading to a repair policy shown in Figure 1.

To apply the planning algorithm above in the learning setting, we observe that complete information about faults is not available to the learner. Our system collects a set of partial episodes, E, each of which is considered a separate fault state for planning purposes.

Work has been done to systematically study and classify the kinds of faults that can occur in a system, including hardware and configuration faults (Li *et al.* 2002; Bohra *et al.* 2004). Our initial autonomic learning-based approach to diagnosis and repair complements existing systems research by providing a framework for flexibly introducing information gathering and repair actions to a self-healing system without the need for detailed hand tuning. In our first experiments, we focus on recovering from faults that result from a corrupted network-interface configuration. Elapsed time to repair is the cost measure the planner seeks to minimize.

Our test and repair actions were implemented in Java in a Windows XP environment. The test actions were: PluggedIn, IsWireless, PingIp, PingLhost, PingGateway, DnsLookup, DefaultIpAddr, DefaultNetmask, DefaultNameServer, DefaultGateway, DHCPEnabled, and PnPReachDns. The repair actions were: **RenewLease**, **UseCachedIP**, and **FixIP**.

We constructed a separately running piece of software we called the "breaker", programmed to introduce any of the following faults: Set bad static netmask, Set bad static gateway, Set bad static IP address, Set bad static value for DNS server, Set DNS to DHCP with no lease, Set IP address to DHCP with no lease, and Lose existing DHCP lease.

For our data collection, we ran the network-repair software concurrently with the breaker to inject faults. Faults were selected uniformly at random and executed at regular intervals of approximately two minutes, providing ample time for recovery.

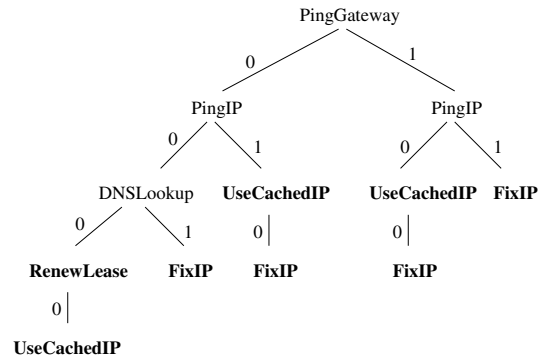The learned policy after 95 repair episodes is illustrated

in Figure 2. In this policy, **RenewLease** is attempted only when PingGateway, PingIP and DNSLookup all fail, with **UseCachedIP** as the backup repair action. **RenewLease** works differently from **UseCachedIP** and **FixIP** in that it obtains the IP parameters dynamically. In almost all the other cases, **UseCachedIP**, being cheaper than **FixIP**, is tried first with **FixIP** used as a backup action. The exceptions are the two cases where **FixIP** is tried without trying **UseCachedIP**, which stems from the lack of adequate exploration. Aside from these imperfections, the policy appears perfectly suited to our test domain.

This work demonstrates an approach to learning for autonomic network repair. Although we have explored several other scenarios to which our cost-sensitive fault remediation model can be applied in simulation, our experience implementing the learning algorithm on a live network helped illustrate the robustness of the basic idea, and also indicated that we should revisit some of the underlying assumptions in our model. More in-depth analysis and algorithmic results are presented in a longer paper (Littman *et al.* 2004).

References

[Bohra *et al.* 2004] Bohra, A.; Neamtiu, I.; Gallard, P.; Sultan, F.; and Iftode, L. 2004. Remote repair of OS state using backdoors. Technical Report DCS-TR-543, Rutgers Univ.

[Li *et al.* 2002] Li, X.; Martin, R.; Nagaraja, K.; Nguyen, T. D.; and Zhang, B. 2002. Mendosus: A SAN-based fault-injection test-bed for the construction of highly available network services. In *First Workshop on Novel Uses of System Area Networks (SAN-1)*.

[Littman *et al.* 2004] Littman, M. L.; Ravi, N.; Fenson, E.; and Howard, R. 2004. An instance-based state representation for network repair. To appear in the Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI).

[Sutton & Barto 1998] Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. The MIT Press.