# Model Repair for Probabilistic Systems

Ezio Bartocci[1], Radu Grosu[2], Panagiotis Katsaros[3],
C.R. Ramakrishnan[2], and Scott A. Smolka[2]

[1] Department of Applied Math and Statistics, Stony Brook University
Stony Brook, NY 11794-4400, USA
[2] Department of Computer Science, Stony Brook University
Stony Brook, NY 11794-4400, USA
[3] Department of Informatics, Aristotle University of Thessaloniki
54124 Thessaloniki, Greece

**Abstract.** We introduce the problem of Model Repair for Probabilistic Systems as follows. Given a probabilistic system $M$ and a probabilistic temporal logic formula $\phi$ such that $M$ fails to satisfy $\phi$, the Model Repair problem is to find an $M'$ that satisfies $\phi$ and differs from $M$ only in the transition flows of those states in $M$ that are deemed controllable. Moreover, the cost associated with modifying $M$'s transition flows to obtain $M'$ should be minimized. Using a new version of parametric probabilistic model checking, we show how the Model Repair problem can be reduced to a nonlinear optimization problem with a minimal-cost objective function, thereby yielding a solution technique. We demonstrate the practical utility of our approach by applying it to a number of significant case studies, including a DTMC reward model of the Zeroconf protocol for assigning IP addresses, and a CTMC model of the highly publicized Kaminsky DNS cache-poisoning attack.

**Keywords:** Model Repair, Probabilistic Model Checking, Nonlinear Programming

## 1 Introduction

Given a model $M$ and a temporal logic formula $\phi$, the *Model Checking problem* is to determine if $M \models \phi$, i.e. does $M$ satisfy $\phi$? In the case of a positive result, a model checker returns true and may also provide further diagnostic information if *vacuity checking* is enabled [9]. In the case of a negative result, a model checker returns false along with a counterexample in the form of an execution path in $M$ leading to the violation of $\phi$. One can then use the counterexample to debug the system model (assuming the problem lies within $M$ as opposed to $\phi$) and ultimately *repair* the model so that revised version satisfies $\phi$.

Even in light of model checking's widespread success in the hardware, software, and embedded systems arenas (see, e.g., [6]), one can argue that existing model checkers *do not go far enough* in assisting the user in repairing a model that fails to satisfy a formula. Automating the repair process is the aim of the *Model Repair problem*, which we consider in the context of *probabilistic systems*

such as discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs), and Markov decision processes (MDPs).

The Model Repair problem we consider can be stated as follows. Given a probabilistic system $M$ and a probabilistic temporal logic formula $\phi$ such that $M$ fails to satisfy $\phi$, the *probabilistic Model Repair problem* is to find an $M'$ that satisfies $\phi$ and differs from $M$ only in the transition flows[1] of those states in $M$ that are deemed *controllable*. Moreover, the *cost* associated with modifying $M$'s transition flows to obtain $M'$ should be minimized. A related but weaker version of the Model Repair problem was first considered in [5], in the (non-probabilistic) context of Kripke structures and the CTL temporal logic. See Section 8 for a discussion of related work.

Our main contributions to the probabilistic Model Repair problem can be summarized as follows:

- Using a new version of *parametric probabilistic model checking* [7, 13], we show how the Model Repair problem can be reduced to a *nonlinear optimization* problem with a minimal-cost objective function, thereby yielding a solution technique.
- We consider related solution feasibility and optimality conditions, and provide an implementation of our solution technique using the PARAM tool for parametric model checking [15] and the Ipopt open-source software package for large-scale nonlinear optimization [2].
- We also consider a *Max-Profit* version of the Model Repair problem for reward-based systems, where profit is defined as the difference between the expected reward and the cost of model repair.
- We also provide a control-theoretic characterization of the probabilistic Model Repair problem, and in the process establish a formal link between model repair and the controller-synthesis problem for linear systems.
- We demonstrate the practical utility of our approach by applying it to a number of significant case studies, including a DTMC reward model of the Zeroconf protocol for assigning IP addresses, and a CTMC model of the highly publicized Kaminsky DNS cache-poisoning attack [1].

The rest of the paper develops along the following lines. Section 2 provides background on parametric model checking. Section 3 contains our formulation of the probabilistic Model Repair problem, while Section 4 characterizes Model Repair as a nonlinear optimization problem. Section 5 considers related feasibility and optimality conditions. Section 6 examines the link between model repair and optimal controller synthesis. Section 7 presents our case studies, while Section 8 discusses related work. Section 9 offers our concluding remarks and directions for future work.

[1] For a DTMC, each row of the probability transition matrix represents the *transition flow* out of the corresponding state.

## 2 Parametric Probabilistic Model Checking

In this section, we show how the model-checking problem for parametric DTMCs can be reduced to the evaluation of a multivariate rational function. The definition of a parametric DTMC is from [13] and the definition of the finite state automaton derived from a parametric DTMC is from [7].

**Definition 1.** *A (labeled) Discrete-Time Markov Chain (DTMC) is a tuple* $D = (S, s_0, \mathbf{P}, L)$ *where:*

- $S$ *is a finite set of states*
- $s_0 \in S$ *is the initial state*
- $\mathbf{P} : S \times S \to [0, 1]$ *is a function such that* $\forall s \in S$, $\sum_{s' \in S} \mathbf{P}(s, s') = 1$
- $L : S \to 2^{AP}$ *is a labeling function assigning to each state a set of atomic propositions from the denumerable set of atomic propositions* $AP$.

Probabilistic model checking is based on the definition of a probability measure over the set of paths that satisfy a given property specification [20]. In the PCTL [17] temporal logic, property specifications are of the form $\mathcal{P}_{\sim b}(\psi)$, with $\sim \in \{<, \leq, >, \geq\}$, $0 \leq b \leq 1$, and $\psi$ a path formula defined using the $X$ (next) and $\mathcal{U}^{\leq h}$ (bounded/unbounded until) operators for $h \in \mathbb{N} \cup \{\infty\}$. A state $s$ satisfies $\mathcal{P}_{\sim b}(\psi)$, denoted as $s \models \mathcal{P}_{\sim b}(\psi)$, if $\mathbb{P}(Path_s(s, \psi)) \sim b$; i.e. the probability of taking a path from $s$ that satisfies $\psi$ is $\sim b$.

**Definition 2.** *Let* $V = \{v_1, \cdots, v_r\}$ *be a set of real variables and let* $\mathbf{v} = (v_1, \ldots, v_r)$. *A* multivariate rational function $f$ *over* $V$ *is a function of the form*

$$f(\mathbf{v}) = \frac{f_1(\mathbf{v})}{f_2(\mathbf{v})}$$

*where* $f_1$, $f_2$ *are two polynomials in* $\mathbf{v}$.

Let $\mathcal{F}_V(\mathbf{v})$ be the field of real-valued rational functions. Given $f \in \mathcal{F}_V$ and an evaluation function $u : V \to \mathbb{R}$, we denote by $f[V/u]$ the value obtained by substituting each occurrence of $v \in V$ with $u(v)$.

**Definition 3.** *A* parametric DTMC *(PDTMC) is a tuple* $\mathcal{D} = (S, s_0, \mathbf{P}, V)$ *where* $S$, $s_0$ *are as in Def. 1 and* $\mathbf{P} : S \times S \to \mathcal{F}_V$, *where* $V = \{v_1, \cdots, v_r\}$ *is a finite set of parameters.*

Given a PDTMC $\mathcal{D}$ over parameters $V$, an evaluation $u$ of $V$ is said to be *valid* for $\mathcal{D}$ if the induced probability transition matrix $\mathbf{P}_u : S \times S \to [0, 1]$ is such that $\sum_{s' \in S} \mathbf{P}_u(s, s')[V/u] = 1$, $\forall s \in S$.

**Definition 4.** *For a PDTMC* $\mathcal{D}$ *and PCTL formula* $\phi = \mathcal{P}_{\sim b}(\psi)$, *the* derived finite state automaton *(dFSA)* $\mathcal{A}_{\mathcal{D},\psi}$ *is given by* $\mathcal{A}_{\mathcal{D},\psi} = \{S, \Sigma, s_0, \delta, S_f\}$, *where:*

- $S$ *is the same set of states of* $\mathcal{D}$
- $\Sigma = \{f \in \mathcal{F}_V \mid \exists i, j, \mathbf{P}(i, j) = f(\mathbf{v}) \neq 0\}$ *is the finite alphabet*

– $s_0$ *is* $\mathcal{D}$*'s initial state*
– $\delta : S \times \Sigma \mapsto 2^S$ *is the transition function derived from* $\mathbf{P}$ *such that* $\delta(s, f) = Q$ *implies* $\forall q \in Q$, $\mathbf{P}(s, q) = f(\mathbf{v})$
– $S_f \subseteq S$ *is the set of final states and depends on* $\psi$.

The set $\mathcal{R}(\Sigma)$ of regular expressions over alphabet $\Sigma$ can be translated into a multivariate rational function. The composition function $comp : \mathcal{R} \mapsto \mathcal{F}_V$ is defined inductively by the following rules:

$$comp(f) = f(\mathbf{v}) \qquad comp(x|y) = comp(x) + comp(y)$$
$$comp(x.y) = comp(x) \cdot comp(y) \qquad comp(x^*) = \frac{1}{1 - comp(x)}$$

It can be proved that $comp(\alpha)$ yields the probability measure of the set $\bigcup_{s_f \in S_f} Path_s(s_0, s_f)$ of paths in $\mathcal{A}_{\mathcal{D},\psi}$ from $s_0$ to some state $s_f$ in $S_f$. In [7], Daws characterizes the set of paths satisfying an unbounded formula $\phi = \mathcal{P}_{\sim b}(\psi)$, but without nested probabilistic quantifiers, as a dFSA $\mathcal{A}_{\mathcal{D},\psi}$, and proves:

**Proposition 1.** *For a PDTMC* $\mathcal{D}$ *and PCTL formula* $\phi = \mathcal{P}_{\sim b}(\psi)$*, with* $\psi$ *a path formula, let* $\alpha$ *be the regular expression for* $\mathcal{L}(\mathcal{A}_{\mathcal{D},\psi})$*. Then,*

$s_0 \models \phi$ iff there is an evaluation $u$ s.t. $u$ is valid for $\mathcal{D}$ and $comp(\alpha) \sim b$

Given a PDTMC $\mathcal{D}$ and bounded reachability property $\phi$, Hahn [14] presents a simple recursive algorithm for deriving the multivariate function $f$ that computes the probability by which $\mathcal{D}$ satisfies $\phi$. Since the only arithmetic operators appearing in $f$ are addition and multiplication, $f \in \mathcal{F}_V$, as $\mathcal{F}_V$ is a field.

## 3 The Model Repair Problem

Given a set of parameters $V$, we write $span(V)$ for the set of linear combinations of the elements in $V$. A $n$-state DTMC $D$ can be turned into a *controllable DTMC* $\tilde{\mathcal{D}}$ by pairing it with an $n \times n$ matrix $\mathbf{Z}$ that specifies which states of $D$ are controllable, and how the transitions out of these states are controlled using elements of $span(V)$.

**Definition 5.** *A* controllable DTMC *over a set of parameters* $V$ *is a tuple* $\tilde{\mathcal{D}} = (S, s_0, \mathbf{P}, \mathbf{Z}, L)$ *where* $(S, s_0, \mathbf{P}, L)$ *is a DTMC and* $\mathbf{Z} : S \times S \to span(V)$ *is an* $|S| \times |S|$ *matrix such that* $\forall s \in S$, $\sum_{t \in S} \mathbf{Z}(s, t) = 0$*. A state* $s \in S$ *is a* controllable state *of* $\tilde{\mathcal{D}}$ *if* $\exists t \in S$ *such that* $\mathbf{Z}(s, t) \neq 0$.

Matrix $\mathbf{Z}$ can be understood as a strategy for altering or controlling the behavior of a DTMC, typically for the purpose of repair; i.e. forcing a particular property to hold for the DTMC. The constraint on $mathbfZ$ implies that the control strategy embodied in $mathbfZ$ should neither change the structure of the DTMC nor its stochasticity. Which states of the DTMC are controllable depends on the model parameters that can be tuned. In general, a model may be repaired by a number of different strategies.

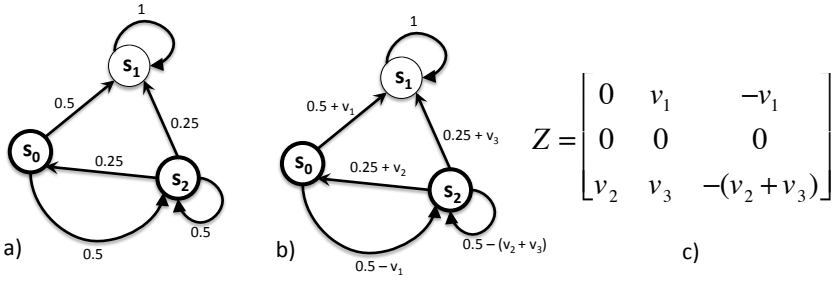


**Fig. 1.** A DTMC (a) and the controllable DTMC (b), with $s_0$, $s_2$ controllable by $\mathbf{Z}$ (c)

*Example 1.* Fig. 1 shows a DTMC in (a), the controllable DTMC in (b), with $s_0$, $s_2$ controllable, and the associated matrix $\mathbf{Z}$ in (c).

**Definition 6.** *Let $\tilde{\mathcal{D}} = (S, s_0, \mathbf{P}, \mathbf{Z}, L)$ be a controllable DTMC over parameters $V$, $D = (S, s_0, \mathbf{P}, L)$ the DTMC underlying $\tilde{\mathcal{D}}$, $\phi$ a PCTL formula for which $D, s_0 \not\models \phi$, and $g(\mathbf{v})$ a possibly nonlinear* cost function*, which is always positive, continuous, and differentiable. The* Model Repair problem *is to find a DTMC $D' = (S, s_0, \mathbf{P}' = \mathbf{P} + \mathbf{Z}[V/u], L)$, where $u : V \to \mathbb{R}$ is an evaluation function satisfying the following conditions:*

$$u = \mathbf{arg\ min}\ g \tag{1}$$

$$D', s_0 \models \phi \tag{2}$$

$$\mathbf{P}(i,j) = 0 \text{ iff } \mathbf{P}'(i,j) = 0, 1 \leq i, j \leq |S| \tag{3}$$

The repair process seeks to manipulate the parameters of $\tilde{\mathcal{D}}$ to obtain a DTMC $D'$ such that $D', s_0 \models \phi$ and the cost of deriving probability transition matrix $\mathbf{P}'$ from $\mathbf{P}$ is minimized. A typical cost function is $g(\mathbf{v}) = w_1 v_1^2 + \cdots + w_r v_r^2$, $\mathbf{w} \in \mathbb{R}_+^r$: a weighted sum of squares of the parameters with weights $w_k$, $1 \leq k \leq r$, specifying that some parameters affect the model to a greater extent than others. For $\mathbf{w} = 1_r$, $g$ is the square of the L2-norm $\|\mathbf{v}\|_2^2$. Condition 3 does not allow the insertion of new transitions nor the elimination of existing ones.

The repair process as specified by Def. 6 is robust in the following sense.

**Proposition 2.** *A controllable DTMC $\tilde{\mathcal{D}}$ and its repaired version $D'$ are $\epsilon$-bisimilar, where $\epsilon$ is the largest value in the matrix $\mathbf{Z}[V/u]$.*

*Proof.* The result immediately follows from Def. 5 and the definition of $\epsilon$-bisimulation [11].

*Example 2.* As an example of the repair process, consider the Die problem proposed by Knuth et al. in [19], where a fair coin is tossed three or more times to obtain the face of a die; see Fig. 2(a). We wish to repair the model so that the

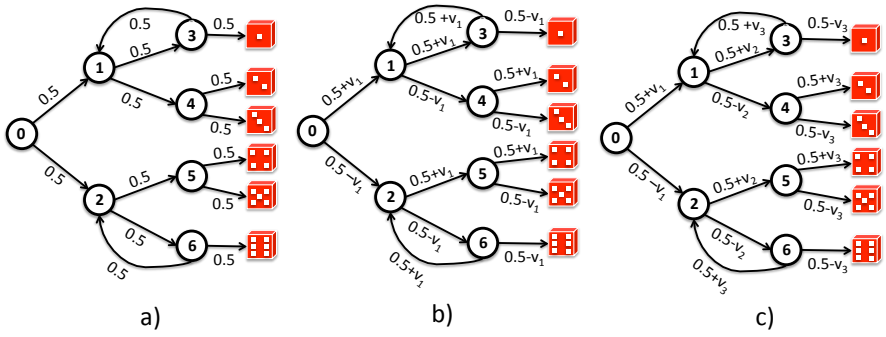


**Fig. 2.** Two different model-repair strategies for the Knuth Die problem.

formula $\mathcal{P}_{\leq 1/8} F[die = 1]$ is satisfied, thereby imposing a probability bound of $\frac{1}{8}$ to eventually obtain a 1. In the original model, the probability to eventually reach any face of the die is $\frac{1}{6}$, and the property is not satisfied.

To repair the model, we need to decide by how much the coin should be biased each time it is tossed. Figs. 2(b) and (c) provide two different model-repair strategies. In Fig. 2(b), we use a single biased coin (controllable DTMC with one parameter), while in Fig. 2(c), we use three differently biased coins (three parameters). In the latter case, the third coin can be tossed again if state 3 or 6 is reached. The case with three parameters gives us the opportunity to prioritize the parameters according to the impact they should have on the repaired model's state space using the weighted sum-of-squares cost function described above.

### 3.1 The Max-Profit Model Repair Problem

**Definition 7.** *A* controllable Markov Reward Model *(MRM) is a tuple* $\mathcal{R} = (\tilde{\mathcal{D}}, \rho, \iota)$ *where* $\tilde{\mathcal{D}} = (S, s_0, \mathbf{P}, \mathbf{Z}, L)$ *is a controllable DTMC,* $\rho : S \to \mathbb{R}_{\geq 0}$ *a state reward function, and* $\iota : S \times S \to \mathbb{R}_{\geq 0}$ *a transition reward function.*

During the repair process, the Max-Profit Model Repair problem seeks to maximize the expected profit, as measured in terms of the difference between the expected reward and the expected cost,

**Definition 8.** *Let* $\mathcal{R} = (\tilde{\mathcal{D}}, \rho, \iota)$ *be a controllable MRM with* $\tilde{\mathcal{D}} = (S, s_0, \mathbf{P}, \mathbf{Z}, L)$ *the associated controllable DTMC and* $D = (S, S_0, \mathbf{P}, L)$ $\tilde{\mathcal{D}}$*'s underlying DTMC. Also, let* $\phi$ *be a PCTL formula such that* $D, s_0 \not\models \phi$*,* $g(\mathbf{v})$ *a possibly nonlinear* cost function*, and* $e(\mathbf{v})$ *an* expected reward function*, which, using* $\rho$ *and* $\iota$*, measures the expected reward accumulated along any path in* $\mathcal{R}$ *that eventually reaches a state in a given set of states* $B$*,* $B \subseteq S$ *[13]. We assume that* $e(\mathbf{v}) - g(\mathbf{v})$ *is always positive, continuous, and differentiable. The* Max-Profit Model Repair problem *is to find a DTMC* $D' = (S, s_0, \mathbf{P}' = \mathbf{P} + \mathbf{Z}[V/u], L)$ *with* $u : V \to \mathbb{R}$

*an evaluation function that satisfies the Conditions (2) and (3) of Def. 6, and the following condition:*

$$u = \textbf{arg max}\ e - g \tag{4}$$

## 4 Model Repair as a Nonlinear Programming Problem

The controllable DTMC $\tilde{\mathcal{D}} = (S, s_0, \mathbf{P}, \mathbf{Z}, L)$ over set of parameters $V$ corresponds to a PDTMC with probability matrix $\mathbf{P} + \mathbf{Z}$. If $\tilde{\mathcal{D}} \not\models \mathcal{P}_{\sim b}(\psi)$, by parametric model checking, we derive the nonlinear constraint $f(\mathbf{v}) \sim b$, where $f$ is a multivariate rational function. Let $Y$ be the set of linear combinations in $V$ defined as $Y = \{y(\mathbf{v}) = \mathbf{P}(i,j) + \mathbf{Z}(i,j) : \mathbf{Z}(i,j) \neq 0, 1 \leq i,j \leq |S|\}$.

**Proposition 3.** *A solution to the Model Repair problem of Def. 6 satisfies the constraints of the following nonlinear program (NLP):*

$$\textbf{min}\ g(\boldsymbol{v}) \tag{5}$$

$$f(\boldsymbol{v}) \sim b \tag{6}$$

$$\forall y \in Y : 0 < y(\mathbf{v}) < 1 \tag{7}$$

*Proof.* Cost function 5 ensures that if there is a solution to the NLP, this yields an evaluation $u$ such that the resulting DTMC is the closest to $\tilde{\mathcal{D}}$ with the desired property, as implied by Condition 1 of Def. 6. Constraint 6 enforces requirement 2 of Def. 6. Constraint 7, along with the selection of $\mathbf{Z}$ in Def. 5, assure that evaluation $u$ is valid and preserves the stochasticity of the new transition probability matrix, while the strict inequalities also enforce constraint 3 of Def. 6. If the NLP has no solution, then model repair for $\tilde{\mathcal{D}}$ is infeasible. For the solution of the Max Profit repair problem, we only have to replace *cost function* (5) with *profit function* (4).

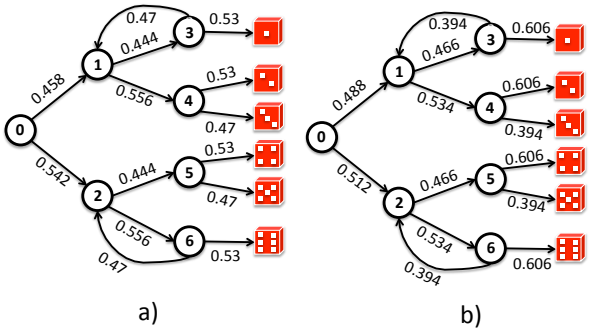


**Fig. 3.** Model Repair solutions for Knuth Die problem.

*Example 3.* To find a solution for the Model Repair problem of Example 2 with the strategy shown in Fig. 2(b), the associated NLP is formulated as follows:

$$\textbf{min } w_1 v_1^2 + w_2 v_2^2 + w_3 v_3^2$$

$$\frac{8v_1v_2v_3 - 4(v_2v_1 - v_2v_3 - v_1v_3) - 2(v_1 + v_2 - v_3) - 1}{8v_2v_3 + 4v_2 + 4v_3 - 6} - \frac{1}{8} \leq 0$$

$$\forall i \in \{1, \cdots, 3\}, -0.5 < v_i < 0.5$$

Fig. 3 shows two different solutions based on the choice of the vector $\mathbf{w}$ of weights associated with the cost function. In Fig. 3(a), we consider the same cost for changing each transition probability, i.e. $\mathbf{w} = [1\ 1\ 1]$. In Fig. 3(b), we provide a solution where the cost of $v_1$ is ten times the cost of $v_3$ and two times the cost of $v_2$, i.e. $\mathbf{w} = [10\ 5\ 1]$.

We denote by $D_{NLP} = Dom(g) \cap Dom(f) \cap \bigcap_{y \in Y} Dom(y)$ the domain of the NLP problem. A solution $\mathbf{v}^* \in D_{NLP}$ is a *local minimizer* if there is $\epsilon > 0$ such that $g(\mathbf{v}^*) \leq g(\mathbf{v})$ for all $|\mathbf{v} - \mathbf{v}^*| < \epsilon$ with $\mathbf{v} \in D_{NLP}$. If $g(\mathbf{v}^*) < g(\mathbf{v})$, $\mathbf{v}^*$ is a *strict* local minimizer. A local minimizer $\mathbf{v}^* \in D_{NLP}$ is a *globally optimal solution* if $g(\mathbf{v}^*) \leq g(\mathbf{v})$ for all $\mathbf{v} \in D_{NLP}$.

All nonlinear optimization algorithms search for a locally feasible solution to the problem. Such a solution can be found by initiating the search from the point $\mathbf{0}^{1 \times r}$, representing the no-change scenario. If no solution is found, the problem is locally infeasible and the analyst has to initiate a new search from another point or else to prove that the problem is infeasible.

## 5 Model Repair Feasibility and Optimality

For a Model Repair problem with property constraint $f(\mathbf{v}) \sim b$, we denote:

$$h(\mathbf{v}) = \begin{cases} f(\mathbf{v}) - b, & \text{if } \sim \in \{<, \leq\} \\ b - f(\mathbf{v}), & \text{if } \sim \in \{>, \geq\} \end{cases}$$

such that the constraint is written as $h(\mathbf{v}) \leq 0$, with the inequality becoming strict if $\sim \in \{<, >\}$.

**Definition 9.** *For the NLP problem, the Lagrangian function* $\mathcal{X} : D_{NLP} \times \mathbb{R}^{2|Y|+1} \to \mathbb{R}$ *is defined as:*

$$\mathcal{X}(\mathbf{v}, \lambda) = g(\mathbf{v}) + \lambda_0 \cdot h(\mathbf{v}) + \sum_{k=1}^{|Y|} \lambda_k \cdot (y_k(\mathbf{v}) - 1) - \sum_{k=|Y|+1}^{2|Y|} \lambda_k \cdot y_{k-|Y|}(\mathbf{v})$$

*The vector* $\lambda$ *is the Lagrange multiplier vector. The Lagrange dual function*

$$w(\lambda) = \inf_{\mathbf{v} \in D_{NLP}} (\mathcal{X}(\mathbf{v}, \lambda))$$

*yields the minimum of the Lagrangian function over* $\boldsymbol{v}$.

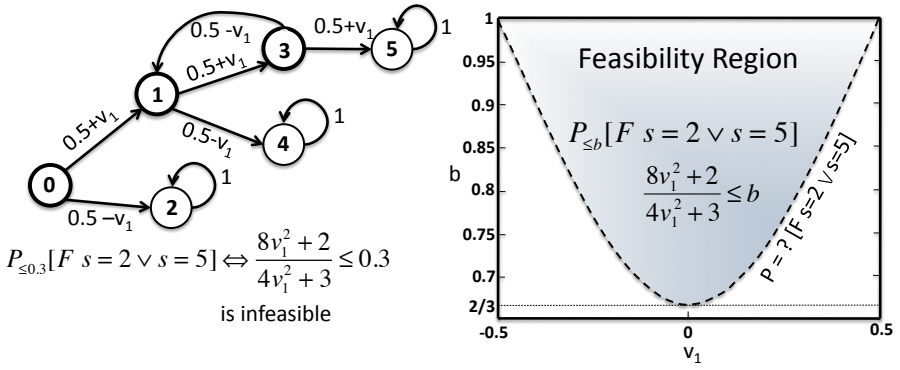


**Fig. 4.** Feasibility analysis for a probabilistic Model Repair problem.

It is easy to verify that for $\lambda \geq 0$, the Lagrangian dual function $w$ yields lower bounds on the optimal cost $g(\mathbf{v}^*)$ of the NLP problem, since $h(\mathbf{v}^*) \leq 0$, $y(\mathbf{v}^*) - 1 < 0$, and $y(\mathbf{v}^*) > 0$, for all $y \in Y$.

**Proposition 4.** *The best lower bound on the the NLP cost function is the cost for the solution of the following Lagrangian dual problem:*

$$\textbf{max} \quad w(\lambda)$$

$$\lambda \geq 0$$

$$\lambda_k \neq 0,\ k = 1, \ldots, 2 \cdot |Y|$$

*The constraints for $\lambda_k$ correspond to the strict inequalities in constraint 7, and if $\sim \in \{<, >\}$, we eventually have $\lambda > 0$.*

In [4], the authors developed a criterion for analyzing the feasibility of non-linear optimization programs, based on the theory of Lagrange duality. More precisely, the system of inequalities in 6 (Prop. 3) is feasible when the $NLP_f$

$$\textbf{min}\ 0 \tag{8}$$

$$h(\mathbf{v}) \sim 0 \tag{9}$$

$$\forall y \in Y : y(\mathbf{v}) - 1 < 0 \tag{10}$$

$$\forall y \in Y : -y(\mathbf{v}) < 0 \tag{11}$$

is also feasible. The optimal cost for the $NLP_f$ is:

$$p^* = \begin{cases} 0 & \text{if } NLP_f \text{ is feasible} \\ \infty & \text{if } NLP_f \text{ is infeasible} \end{cases} \tag{12}$$

The Lagrangian dual function $w_f$ for this program with zero cost is:

$$w_f(\lambda) = \inf_{\mathbf{v} \in D_{NLP}} (\mathcal{X}(\mathbf{v}, \lambda) - g(\mathbf{v})) \tag{13}$$

We note that $w_f$ is positive homogeneous in $\lambda$, i.e. $\forall \alpha > 0,\ w_f(\alpha \cdot \lambda) = \alpha \cdot w_f(\lambda)$. The associated Lagrange dual problem is to maximize $w_f(\lambda)$ subject to the constraints $\lambda \geq 0$ and $\lambda_k \neq 0,\ k = 1, \ldots, 2 \cdot |Y|$. The optimal cost for the homogeneous $w_f$ in the dual problem is:

$$d^* = \begin{cases} \infty & \text{if } \lambda \geq 0,\ \lambda_k \neq 0,\ w_f(\lambda) > 0 \text{ is feasible} \\ 0 & \text{if } \lambda \geq 0,\ \lambda_k \neq 0,\ w_f(\lambda) > 0 \text{ is infeasible} \end{cases} \tag{14}$$

for $k = 1, \ldots, 2 \cdot |Y|$. By taking into account the property

$$d^* \leq p^*$$

which is known as *weak duality* [22], we conclude:

**Proposition 5.** *If the Lagrange dual problem of $NLP_f$, with the NLP constraints and cost is feasible, then the NLP for model repair is infeasible. Conversely, if NLP is feasible, then the Lagrange dual problem of $NLP_f$ is infeasible.*

*Example 4.* For the controllable DTMC $\tilde{\mathcal{D}}$ of Fig. 4, we show that Model Repair is not always feasible. For path formula $\psi = F[s = 2 \vee s = 5]$, from Prop. 1 we have:

$$\mathcal{D}, s_0 \models \mathcal{P}_{\leq b}\psi \quad \textbf{iff} \quad \frac{8v_1^2 + 2}{4v_1^2 + 3} \leq b$$

The Lagrangian dual function for the $NLP_f$ program is:

$$w_f(\lambda) = \inf_{v_1 \in ]-0.5, 0.5[} \left( \lambda_0 \left( \frac{8v_1^2 + 2}{4v_1^2 + 3} - b \right) + \lambda_1(v_1 - 1) + \lambda_2(-v_1) \right)$$

where $\lambda_0 \geq 0$ and $\lambda_1, \lambda_2 > 0$. The rational function for $\psi$ is minimized in $v_1 = 0$ (Fig. 4) and therefore

$$w_f(\lambda) = \lambda_0 \left( \frac{2}{3} - b \right) - \lambda_1$$

The nonlinear program in 14 becomes feasible when $b < 2/3$ and from Prop. 5 the NLP for repairing $\tilde{\mathcal{D}}$ is infeasible for these values of $b$.

From [4], if $\mathbf{v}^*$ is a local minimizer that also fulfills certain constraint qualifications, then the Karush-Kuhn-Tucker (KKT) conditions are satisfied. On the other hand, if for some feasible point $\mathbf{v}'$ there is a Lagrange multiplier vector $\lambda^*$ such that the KKT conditions are satisfied, sufficient conditions are provided which guarantee that $\mathbf{v}'$ is a strict local minimizer. Since all the parameters are bounded it is possible to check global optimality by using an appropriate constraint solver, such as RealPaver [12].

## 6 Model Repair and Optimal Control

We show how the Model Repair problem is related to the optimal-controller synthesis problem for linear systems. Given a (right-)linear system

$$x(n+1) = x(n)A + u(n)B, \quad x(0) = x_0$$

where $x$ is the state vector, $u$ is a vector of inputs, and $A$ and $B$ are matrices representing the system dynamics, the synthesis of an optimal controller is typically performed in a compositional way as follows: (1) Synthesize a linear quadratic regulator; (2) Add a Kalman filter state estimator; (3) Add the reference input.

For (1), the input can be written as $u(n) = x(n)K$, where $K$ is the controller matrix to be synthesized. Then:

$$x(n+1) = x(n)(A + KB)$$

An optimal $K$ is then obtained by minimizing the cost function

$$\mathcal{J} = 1/2 \sum_{k=0}^{\infty} (x^T(k)Qx(k) + u^T(k)Ru(k))$$

where $Q$ and $R$ are nonnegative definite symmetric weighting matrices to be selected by the designer, based on the relative importance of the various states $x_i$ and controls $u_j$.

The purpose of (2) is to add an optimal state estimator, which estimates the current state based on the previous input and the previously measured output. This is typically done in a recursive fashion with the help of a Kalman filter.

Finally, in (3), the reference input is added, which drives the overall behavior of the controlled system. Typically, the addition of the reference input is complemented with an integral control.

Although not immediately apparent, all the above steps are related to the Model Repair problem (Definition 6). First, matrix $\mathbf{Z}$ contains in each entry a linear combination of the parameters in $V$. This linear combination can be obtained by decomposing $\mathbf{Z}$ as $\mathbf{Z} = KB$, where $K$ is a matrix of parameters (to be synthesized) and $B$ is a matrix defining the contribution of each parameter to the next state of the system.

Note however, that the above optimization problem for $\mathcal{J}$ does not encode the stochasticity requirement of $\mathbf{Z}$. Hence, using the available tools for optimal-controller synthesis, one cannot impose this requirement.

Second, the reference input can be related to the PCTL formula $\phi$. Adding the reference input to the system is, in many ways, the same as imposing the satisfaction of $\phi$. Note, however, that the reference input is added *after* $K$ is synthesized. In contrast, we synthesize the parameters in $\mathbf{Z}$ with $\phi$ included as a constraint of the nonlinear optimization problem.

Finally, the variables occurring in the PCTL formula can be seen as the observables of the system, i.e., as the outputs of the system. In the model repair case, we assume that we have full access to the state variables and the model precisely captures the output.

Overall, our approach seems to be a generalization of the optimal-controller synthesis problem for the class of linear systems represented by DTMCs. We also perform a nonlinear multivariate optimization which is by necessity more powerful than the one used for linear systems.

## 7 Applications

We present two Model Repair applications: (i) the Kaminsky DNS cache-poisoning attack, along with the proposed fix (repair), and (ii) the Zeroconf protocol as a Max-Profit problem. Model Repair was performed using the PARAM tool for parametric model checking [15] and the Ipopt software package for large-scale nonlinear optimization [2].

*Example 5.* The Kaminsky DNS attack makes clever use of cache poisoning, so that when a victim DNS server is asked to resolve URLs within a non-malicious domain, it replies with the IP address of a malicious web server. The proposed fix is to randomize the UDP port used in name-resolution requests. As such, an intruder can corrupt the cache of a DNS server with a falsified IP address for a URL, only if it manages to guess a 16-bit source-port id, in addition to the 16-bit query id assigned to the name-resolution request.

Our CTMC model for the Kaminsky attack [1] implements a victim DNS server that generates `times_to_request_url` queries to resolve one or more resource names within some domain. While the victim waits for a legitimate response to its query, the intruder tries with rate `guess` to provide a fake response that, if correctly matching the query id, will be accepted by the victim, thus corrupting its cache.

The only parameter the victim can control is the range of port-id values used by the proposed fix, which affects the rate at which correct guesses arrive at the victim. Other parameters that affect the rate of correct guesses, but are not controlled by the victim are the `popularity` of the requested names, and the rate at which `other_legitimate_requests` arrive at the victim. If the fix is disabled, the number of port ids is one, and experiments show that for `guess` $\geq 200$, the attack probability is greater than 0.9 if `times_to_request_url` $\geq 6$.

By applying model repair on the controllable embedded DTMC, we determined the minimum required range of port ids such that $\mathcal{P}_{\leq 0.05} F$ cache_poisoned. While the value of `times_to_request_url` determines the size of the state space, we observed that nonlinear optimization with Ipopt is not affected by state-space growth. This is not the case, however, for the parametric model-checking times given in Table 1 (`popularity`=3,`guess`=150,`other_legitimate_requests`=150). The model was successfully repaired for all values of `times_to_request_url` from 1 to 10.

*Example 6.* According to the Zeroconf protocol for assigning IP addresses in a network, when a new host joins the network it randomly selects an IP address among $K$ possible ones. With $m$ hosts in the network, the collision probability is $q = m/K$. A new host asks the other hosts whether the randomly selected IP address is already used and waits for an answer. The probability that the new host does not get any answer is $p$, in which case it repeats the query. If after $n$ tries there is still no answer, the host will erroneously consider the chosen address as valid.

**Table 1.** Model Repair of the Kaminsky CTMC.

| times_to_request | States | Transitions | CPU_PARAM | PORT_ID | P=? [F cache_poisoned ] |
|---|---|---|---|---|---|
| 1 | 10 | 13 | 0m0.390s | 5 | 0.04498 |
| 2 | 60 | 118 | 0m0.430s | 10 | 0.04593 |
| 3 | 215 | 561 | 0m0.490s | 14 | 0.04943 |
| 4 | 567 | 1759 | 0m1.750s | 19 | 0.04878 |
| 5 | 1237 | 4272 | 0m15.820s | 24 | 0.04840 |
| 6 | 2350 | 8796 | 1m56.650s | 28 | 0.0498 |
| 7 | 4085 | 16163 | 10m55.150s | 33 | 0.0494 |
| 8 | 6625 | 27341 | 47m21.220s | 38 | 0.0491 |
| 9 | 10182 | 43434 | 167m58.470s | 42 | 0.0499 |
| 10 | 14992 | 65682 | 528m32.720s | 47 | 0.0496 |

We used Max Profit model repair on the DTMC model of [7] to determine the collision probability $q$ that optimizes the trade-off between (a) the expected number of tries until the algorithm terminates, and (b) the cost of changing $q$ from its default value. The change applied to $q$ is the only parameter used in our Max Profit model; all other transition probabilities were maintained as constants as in the original model. For $n = 3$, $p = 0.1$, and initial $q = 0.6$, we determined the optimal $q$ to be 0.5002, which reduced the expected number of steps to termination from 6.15 to 5.1.

## 8 Related Work

Prior work has addressed a related version of the Model Repair problem in the nonprobabilistic setting. In [5], abductive reasoning is used to determine a suitable modification of a Kripke model that fails to satisfy a CTL formula. Addition and deletion of state transitions are considered, without taking into account the cost of the repair process. The problem of automatically revising untimed and real-time programs with respect to UNITY properties is investigated in [3], such that the revised program satisfies a previously failed property, while preserving the other properties. A game-based approach to the problem of automatically fixing faults in a finite-state program is considered in [18]. The game consists of the product of a modified version of the program and an automaton representing an LTL specification, such that every winning finite-state strategy corresponds to a repair. In [23], the authors introduce an algorithm for solving the parametric real-time model-checking problem: given a real-time system and temporal formula, both of which may contain parameters, and a constraint over the parameters, does every allowed parameter assignment ensure that the real-time system satisfies the formula?

In related work for probabilistic models, a Bayesian estimator based on runtime data is used in [10] to address the problem of model evolution, where model parameters may change over time. The authors of [21] consider paramet-

ric models for which they show that finding parameter values for a property to be satisfied is in general undecidable. In [8], a model checker together with a genetic algorithm drive the parameter-estimation process by reducing the distance between the desired behavior and the actual behavior. The work of [16] addresses the parameter-synthesis problem for parametric CTMCs and time-bounded properties. The problem is undecidable and the authors provide an approximation method that yields a solution in most cases.

## 9 Conclusions

We have defined, investigated, implemented, and benchmarked the Model Repair problem for probabilistic systems. Ultimately, we show how Model Repair can be seen as both a nontrivial extension of the parametric model-checking problem for probabilistic systems and a nontrivial generalization of the controller-synthesis problem for linear systems. In both cases, its solution requires one to solve a non-linear optimization problem with a minimal-cost (or maximal-profit) objective function.

The problem we considered is one of *offline* model repair. As future work, we would like to investigate the *online* version of the problem, where an online controller runs concurrently with the system in question, appropriately adjusting its parameters whenever a property violation is detected. Meeting this objective will likely require a better understanding of the similarities between the model repair and controller synthesis problems.

*Acknowledgement:* We thank the anonymous referees for their valuable comments. This work was supported in part by NSF grants CCF 1018459, CNS 0831298, and CNS 0721665, and ONR grant N00014-07-1-0928.

## References

1. N. Alexiou, T. Deshpande, S. Basagiannis, S. A. Smolka, and P. Katsaros. Formal analysis of the kaminsky DNS cache-poisoning attack using probabilistic model checking. In *Proceedings of the 12th IEEE International High Assurance Systems Engineering Symposium*, pages 94–103. IEEE Computer Society, 2010.
2. L. T. Biegler and V. M. Zavala. Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization. *Computers & Chemical Engineering*, 33(3):575 – 582, 2009.
3. B. Bonakdarpour, A. Ebnenasir, and S. S. Kulkarni. Complexity results in revising UNITY programs. *ACM Trans. Auton. Adapt. Syst.*, 4(1):1–28, 2009.
4. S. Boyd and L. Vandenberghe. *Convex Optimization*. Camb. Univ. Press, 2004.
5. F. Buccafurri, T. Eiter, G. Gottlob, and N. Leone. Enhancing model checking in verification by AI techniques. *Artif. Intell.*, 112(1-2):57–104, 1999.
6. E. M. Clarke, E. A. Emerson, and J. Sifakis. Model checking: Algorithmic verification and debugging. *Communications of the ACM*, 52(11):74–84, 2009.
7. C. Daws. Symbolic and parametric model checking of discrete-time Markov chains. In Z. Liu and K. Araki, editors, *Theoretical Aspects of Computing - ICTAC 2004*, volume 3407 of *Lecture Notes in Computer Science*, pages 280–294, 2005.

8. R. Donaldson and D. Gilbert. A model checking approach to the parameter estimation of biochemical pathways. In *Proceedings of CMSB '08*, volume 5307 of *Lecture Notes in Computer Science*, pages 269–287, 2008.
9. Y. Dong, B. Sarna-Starosta, C. R. Ramakrishnan, and S. A. Smolka. Vacuity checking in the modal mu-calculus. In *Proceedings of AMAST '02*, volume 2422 of *LNCS*, pages 147–162, 2002.
10. I. Epifani, C. Ghezzi, R. Mirandola, and G. Tamburrelli. Model evolution by run-time parameter adaptation. In *ICSE '09: Proceedings of the 31st International Conference on Software Engineering*, pages 111–121, Washington, DC, USA, 2009. IEEE Computer Society.
11. A. Giacalone, C. chang Jou, and S. A. Smolka. Algebraic reasoning for probabilistic concurrent systems. In *Proc. of the IFIP TC2 Working Conference on Programming Concepts and Methods*, pages 443–458. North-Holland, 1990.
12. L. Granvilliers and F. Benhamou. RealPaver: an interval solver using constraint satisfaction techniques. *ACM Trans. Math. Softw.*, 32:138–156, 2006.
13. E. Hahn, H. Hermanns, and L. Zhang. Probabilistic reachability for parametric Markov models. *International Journal on Software Tools for Technology Transfer*, pages 1–17, Apr. 2010.
14. E. M. Hahn. Parametric Markov model analysis. Master's thesis, Saarland University, 2008.
15. E. M. Hahn, H. Hermanns, B. Wachter, and L. Zhang. PARAM: A model checker for parametric Markov models. In *Proceedings of CAV '10*, volume 6174 of *LNCS*, pages 660–664, 2010.
16. T. Han, J.-P. Katoen, and A. Mereacre. Approximate parameter synthesis for probabilistic time-bounded reachability. *Real-Time Systems Symposium, IEEE International*, pages 173–182, 2008.
17. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6:102–111, 1994.
18. B. Jobstmann, A. Griesmayer, and R. Bloem. Program repair as a game. In *Proceedings of CAV '05*, volume 3576 of *LNCS*, pages 226–238, 2005.
19. D. Knuth and A. Yao. *Algorithms and Complexity: New Directions and Recent Results*, chapter The complexity of nonuniform random number generation. Academic Press, 1976.
20. M. Z. Kwiatkowska, G. Norman, and D. Parker. Stochastic model checking. In *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation*, volume 4486 of *Lecture Notes in Computer Science*, pages 220–270, 2007.
21. R. Lanotte, A. Maggiolo-Schettini, and A. Troina. Parametric probabilistic transition systems for system sesign and analysis. *Formal Aspects of Computing*, 19(1):93–109, 2007.
22. S. M. Sinha. Duality in nonlinear programming. In *Mathematical Programming*, pages 423 – 430. Elsevier Science, Burlington, 2006.
23. D. Zhang and R. Cleaveland. Fast on-the-fly parametric real-time model checking. In *Proceedings of the 26th IEEE International Real-Time Systems Symposium*, pages 157–166. IEEE Computer Society, 2005.