ELSEVIER

Information and Software Technology 44 (2002) 841–855

INFORMATION AND SOFTWARE TECHNOLOGY

www.elsevier.com/locate/infsof

Optimal software testing and adaptive software testing in the context of software cybernetics☆

Kai-Yuan Cai

Department of Automatic Control, Beijing University of Aeronautics and Astronautics, Beijing 100083, People's Republic of China

Abstract

Software cybernetics explores the interplay between software theory/engineering and control theory/engineering. Following the idea of software cybernetics, the controlled Markov chains (CMC) approach to software testing treats software testing as a control problem. The software under test serves as a controlled object, and the (optimal) testing strategy determined by the theory of controlled Markov chains serves as a controller. This paper analyzes the behavior of the corresponding optimal test profile determined by the CMC approach to software testing and introduces adaptive software testing. It is shown that in some cases the optimal test profile is Markovian, whereas in some other cases the optimal test profile demonstrates a different scenario. The adaptive software testing adjusts software testing strategy on-line by using testing data collected during software testing in response to changes in our understanding of the software under test. Simulation results show that the adaptive strategy of software testing is feasible and significantly reduces the number of test cases required to detect and remove a certain number of software defects in comparison with the random strategy of software testing.

Keywords: Software cybernetics; Controlled Markov chain; Software testing; Optimal test profile; Software operational profile; Adaptive software testing

1. Introduction

The idea of software cybernetics was first proposed by the author in 1994 with an attempt to apply cybernetic or control-theoretical approaches to solve software engineering problem [3]. If we consider the potential of applying approaches in the area of software theory or theoretical computer science to solve control engineering problem, then we can define software cybernetics as the interplay between software theory/engineering and control theory/engineering. The feasibility and benefits of the idea of software cybernetics are justified when we treat software testing as a control problem and propose a controlled Markov chains (CMC) approach to software testing [6]. With the CMC approach, the software under test serves as the controlled object modeled as a controlled Markov chain, whereas the software testing strategy serves as the corresponding controller. The software under test and the corresponding testing strategy make up a closed-loop feedback control system. Unlike conventional approaches to software testing (e.g. branch testing, functional testing) [2], which are applied to the software under test without an explicit optimization goal, the CMC approach designs an optimal testing strategy to achieve an explicit optimization goal given a priori. Test cases are selected or generated by the optimal testing strategy and thus constitute an optimal test profile. Here we note that a common understanding of test profile is a probability distribution of test cases to be selected. However test profile may also refer to more general models such as Markov chains, Poisson models and the like [4]. In a broad sense, an optimal test profile defines how to apply the right (testing) techniques (or, actions) at the right times in order to achieve the best results. An interesting but unsolved question is whether the optimal test profile makes up a Markov chain, although the corresponding behavior of the software under the optimal testing strategy fits a Markov chain [6]. This question defines a topic of this paper.

Another topic of the paper is adaptive software testing.[1]

☆ Supported by the National Outstanding Youth Foundation of China, the Aviation Science Foundation of China and the '863' Program of China. This paper is a revised and extended version of the paper "Optimal test profile in the context of software cybernetics" presented at the Second Asia-Pacific Conference on Quality Software, December 10–11, 2001, Hong Kong.

E-mail address: kyc@ns.dept3.buaa.edu.cn (K.Y. Cai).

[1] In their paper [7], Chen, Tse and Yu describe the so-called adaptive random testing as a form of fault-based random testing with attempt to improve the effectiveness of simple random testing. This strategy is actually defined a priori and is not designed. No information of the software under test or testing/reliability goal is used. However our adaptive software testing is treated as the counterpart of adaptive control in software testing and thus is essentially different from the so-called adaptive random testing.

0950-5849/02/$ - see front matter
PII: S0950-5849(02)00108-8

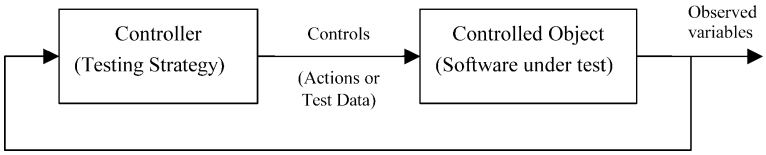

Fig. 1. Software testing as a control problem.

At the beginning of software testing our knowledge of the software under test is limited. As the software testing proceeds, more testing data are collected and our understanding of the software under test is improved. Software parameters (e.g. software defect detection rates) of concern may be estimated and updated, and the software testing strategy is accordingly adjusted on-line.

Here we should note that a controlled Markov chain is essentially different from an ordinary Markov chain in general and the CMC approach to software testing is dramatically different from existing modeling paradigms of software testing. An ordinary Markov chain is a stochastic process whose future behavior depends only on its current state and is independent of its history. If a software defect detection process can be described by a Markov chain, roughly speaking, it means that when or which defect will be detected and removed depends only on what and how many defects are remaining in the software, and is not related to how detected defects were detected and removed from the software. Avritzer and Weyuker discussed how to test software that is intended to behave as a Markov chain [1]. Whittaker and Thomason observed that software under Markov test data generates Markov failure behavior [16]. Cai developed several Markov models to describe software operational profile [4]. However no theoretical justification for Markov test data is given and how to optimally test software in an adaptive manner is still an unsolved problem. A controlled Markov chain means that although the future behavior of the chain does not depend directly on the history of the chain, it really depends on the controls that will be applied to the chain, and the controls applied may depend on the history of the chain. A controlled Markov chain reduces to an ordinary Markov chain when no control or only an identical control is applied to the chain all the times. The usage of controlled Markov chains, except the CMC approach to software testing, has not been explored in the existing software testing literature.

Section 2 treats software testing as a control problem. Section 3 reviews the CMC approach to software testing. Section 4 discusses when optimal test profile makes up a Markov chain. Section 5 introduces the adaptive strategy of software testing. Concluding remarks are contained in Section 6.

2. Software testing as a control problem

In the process of software testing, test cases are selected in accordance with a given testing strategy and applied to the software under test. Some test cases reveal failures. The corresponding failure-causing defects are then removed and thus the underlying software states undergo changes. Some test cases don't reveal failures and no state transitions happen to the software under test. So uncertainty is associated with the behavior of the software under test. If we treat the corresponding testing strategy as a control policy or controller, then we can treat the software under test as an uncertain controlled object. Further, if an optimization (testing) goal is given explicitly and a priori, then the test data selection becomes as an optimal control problem. In an optimal control problem, a dynamical system (controlled object) is given whose behavior not only follows its own 'laws of motion' (e.g. Markov state transition laws), but may be influenced or regulated by a suitable choice of some of the system's variables, which are called control or action variables. The controls that can be applied at any given time are chosen according to control policies or laws that are derived from histories of the system and the given performance criterion or performance index. The performance criterion measures or evaluates in some sense that system's response to the control policies being used. The control policies (or controller) and the controlled system constitute a closed-loop feedback system as depicted in Fig. 1. Then the optimal control problem is to determine a control policy that optimizes (i.e. either minimizes or maximizes) the performance criterion.[2] Therefore in order to solve an optimal control problem, we need to identify three components: a model for the controlled object, a set of admissible control policies, and a performance index (or objective function).

For the software testing problem of concern, let

$$Y_t = j \text{ if the software under test contains } j \text{ defects at time } t,\quad j = 0, 1, 2, \ldots, N;\quad t = 0, 1, 2, \ldots$$

$$Z_t = \begin{cases} 1 & \text{if the action taken at time } t \text{ detects a defect} \\ 0 & \text{if the action taken at time } t \text{ doesn}'\text{t detect a defect} \end{cases}$$

Here we should note that the term 'action' can widely be interpreted. An action can be selection of an input value from certain equivalence class in partition testing, it can also be selection of a test case, a software run, or an equivalence class of test cases for a specific testing technique. It can refer to selection of a software testing technique such as boundary testing, path testing and random testing. An action can even

[2] The performance criterion is not explicitly shown in Fig. 1. It is implicitly incorporated into the controller. This is standard in the control literature.

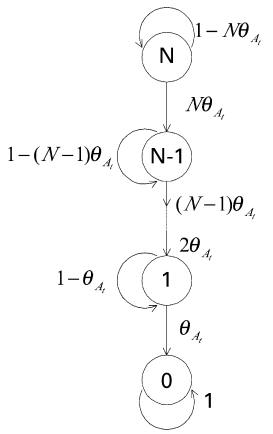

Fig. 2. Software state transition under test.

mean selection of a tester. Different actions have different defect detection ability, or we can say that the software demonstrates different testability under different actions. A control policy (testing strategy) specifies which action should be taken at every time.

We have the following assumptions.

(1) The software contains N defects at the beginning ($t = 0$).
(2) An action taken at one time detects at most one defect.
(3) If a defect is detected, then it is removed immediately and no new defects are introduced; that is, $Y_t = j$ and $Z_t = 1$ mean $Y_{t+1} = j - 1$.
(4) If no defect is detected, then the number of remaining software defects remains unchanged; that is, $Y_t = j$ and $Z_t = 0$ mean $Y_{t+1} = j$.
(5) $Y_t = 0$ is an absorbing state; it is the target state.
(6) At every time there are always m admissible actions; the action set is $A = \{1, 2, \ldots, m\}$.
(7) Z_t depends only on the software state Y_t and the action A_t taken at time t;

$$\Pr\{Z_t = 1 | Y_t = j, A_t = i\} = j\theta_i$$

$$\Pr\{Z_t = 0 | Y_t = j, A_t = i\} = 1 - j\theta_i;\ \ j = 1, 2, \ldots, N$$

$$\Pr\{Z_t = 1 | Y_t = 0, A_t = i\} = 0$$

$$\Pr\{Z_t = 0 | Y_t = 0, A_t = i\} = 1;\ \ t = 0, 1, 2, \ldots$$

where θ_i can be interpreted as the probability of each defect being detected by action i.
(8) Action A_t taken at time t incurs a cost of $W_{Y_t}(A_t)$, no matter whether or not it detects a defect;

$$W_{Y_t}(A_t = i) = \begin{cases} w_j(i) & \text{if } Y_t = j \neq 0 \\ 0 & \text{if } Y_t = 0 \end{cases}$$

(9) The cost of removing a detected defect is ignored.

Here we note that assumption (9) is not necessary for the subsequent mathematical treatment. However it makes the set of assumptions look complete for realistic software testing processes since removing a detected defect should really cost something. In software reliability modeling, one needs to distinguish between continuous-time domain and discrete-time domain [5], and we formulate the control (testing) problem in the context of discrete-time domain. Let τ be the first-passage time to state 0, that is, τ is the minimal number of actions that are required to move the software under test to state 0. Denote

$$J_\omega(N) = E_\omega \sum_{t=0}^{\tau} W_{Y_t}(A_t) \tag{2.1}$$

where ω denotes a control policy (testing strategy) and E_ω refers to the expected cost required by the control policy ω to detect and remove all the N defects. Our problem is to find the control policy (testing strategy) that minimizes $J_\omega(N)$. Such a policy removes all the N defects at the least expected cost.

Following the above assumptions, we can use Fig. 2 to depict the software state transition behavior under test. State j means that there are still j defects remaining in the software under test. If action A_t is applied to the software in state j ($j \neq 0$) at time t, then the software remains in state j with probability $1 - j\theta_{A_t}$ and moves to state $j - 1$ with probability $j\theta_{A_t}$. Upon entering into the target state 0, the software stays there forever.

Here we see that a control policy or testing strategy needs to decide which test case or action A_t should be taken with respect to state Y_t at time t. A software testing approach should be able to design or determine the required testing strategy according to the given optimization (testing) goal. The resulting behavior of $\{A_t, t = 0, 1, 2, \ldots\}$ specifies a test profile that is optimal with respect to the given testing goal for the software under test.

3. The controlled Markov chains (CMC) approach to software testing

3.1. Controlled Markov chains

In the setting of controlled Markov chains, the controlled object is given as a discrete-time Markov control model, which is five-tuple

$$\langle S, A, \{A(y) | y \in S\}, Q, W \rangle \tag{3.1}$$

where S and A are given sets, called the state space and the control (action) set, respectively. $\{A(y) | y \in S\}$ is a family of nonempty subsets $A(y)$ of A, with $A(y)$ being the set of feasible controls or actions in the state $y \in S$. Q is a transition law such that

$$Q(B | y, a) = \Pr\{Y_{t+1} \in B | Y_t = y, A_t = a\}, B \subset S$$

where Y_t denotes the system state at time t, and A_t the action

(control) applied at time t. Finally, W is a cost-per-stage (or one-stage cost) function. When S is discrete, model (3.1) is the so-called controlled Markov chain.

Control model (3.1) represents a controlled stochastic system that is observed at times $t = 0, 1, 2, \ldots$. If the system is in the state $Y_t = y \in S$ at time t and the control $A_t = a \in A(y)$ is applied, then two things happen: (1) a cost $W_{Y_t}(a)$ is incurred, and (2) the system moves to the next state Y_{t+1} according to the transition law Q. Once the transition into the new state has occurred, a new control is chosen and the process is repeated. Obviously, a controlled Markov chain or discrete-time Markov control model is not a Markov chain in general. It reduces to an ordinary Markov chain when A consists of only one action and $A(y) \equiv A$.

Let

$$H_t = \{Y_0, A_0, Y_1, A_1, \ldots, Y_{t-1}, A_{t-1}, Y_t\}$$

i.e. H_t denotes the history of the system up to time t. Let

$$D_t(A_t; H_t) = \Pr\{A_t | H_t\}$$

i.e. $D_t(a; h_t)$ denotes the conditional probability of $A_t = a$ at time t under the condition $H_t = h_t$.[3] Obviously,

$$\sum_{a \in A(y_t)} D_t(a; h_t) = 1$$

Then a control policy is defined as

$$\omega = \{D_t(A_t; H_t);\ t = 0, 1, 2, \ldots\}$$

If D_t is always confined to 1 or 0, that is, at any time a particular action is certainly taken, then ω is deterministic.[4] ω is randomized if it is not deterministic. If $D_t(A_t; H_t)$ depends only on t, Y_t and A_t, then the corresponding control policy is Markovian. If $D_t(A_t; H_t)$ depends only on Y_t and A_t, then the corresponding control policy is stationary Markovian (stationary for brevity). In this way we have several classes of control policies: randomized policy, (randomized) Markov policy, (randomized) stationary policy, deterministic policy, deterministic Markov policy, and deterministic stationary policy. Under stationary control policies, we can easily see that the resulting $\{Y_t; t = 0, 1, 2, \ldots\}$ forms a (time-homogenous) Markov chain. But this is not true under a general control policy.

Let $W_{Y_t}(A_t)$ denote the cost incurred at time t by action A_t applied to the system being in state Y_t. Obviously $W_{Y_t}(A_t)$ is a random variable in general. In the optimal control of a controlled Markov chain we need to optimize certain performance criterion involving $W_{Y_t}(A_t)$. In Refs. [8,11], several classes of performance criteria are proposed, including average cost criteria, first-passage cost criteria, and discounted cost criteria, among others. Since we need to detect and remove all the N defects, here we are most interested in the first-passage cost criteria or the first-passage problem. Suppose the system begins with state $Y_0 = y_0$ and has a target state $y^* \neq y_0$ where the system evolution process should be stopped. Let τ denote the smallest positive integer such that $Y_\tau = y^*$. The optimal first-passage problem is concerned with minimizing the objective function

$$J_\omega(y_0) = E_\omega \sum_{t=0}^{\tau} W_{Y_t}(A_t)$$

$$= \sum_{t=0}^{\tau} \sum_{y} \sum_{a} \Pr^{(\omega)}\{Y_t = y, A_t = a\} W_y(a)$$

with respect to control policy ω, where E_ω denotes the mathematical expectation under control policy ω, $\Pr^{(\omega)}$ denotes the corresponding probability under control policy ω. Here we note that τ is a random variable and therefore $J_\omega(y_0)$ is the expected value of a random sum of random variables.

The following lemma is basic in the theory of controlled Markov chains [8]. It asserts that the solution to the optimal first-passage problem is a deterministic stationary control policy.

Lemma 3.1.

(1) *If* $\{W_{Y_t}(A_t), t = 0, 1, 2, \ldots\}$ *are nonnegative, then there exists a deterministic stationary control policy* ω^* *such that*

$$J_{\omega^*}(y_0) = \inf_{\omega} J_\omega(y_0)$$

(2) *The conclusion is still valid if for all initial state* y_0 *and control policy* ω,

$$\Pr\{Y_t = y^* \text{ for some } t \geq 1 | Y_0 = y_0\} = 1$$

Now the problem is how to compute the optimal control policy. The following so-called method of successive approximation can be used to do this. Let $\{v_0(y), y \in S - \{y^*\}\}$ be arbitrary, and define

$$v_{n+1}(y) = \min_a \times \left\{ W_y(a) + \sum_{x \in \{S - \{y^*\}\}} q_{yx}(a) v_n(x) \right\},$$

$$y \in S - \{y^*\} \tag{3.2}$$

where $q_{yx}(a)$ denotes the probability that the system is moved from state y to state x by action a. Then we have the following lemma [8].

[3] In the context of software testing we can treat D_t as a moving or time-varying test profile at time t.

[4] Inconsistent uses of terminology can be observed in Refs. [8,11]. In Ref. [8] 'deterministic' actually means 'stationary deterministic' here.

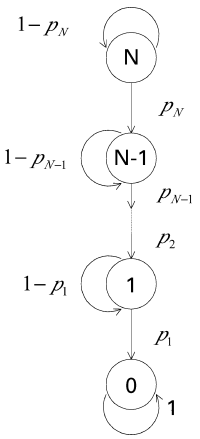

Fig. 3. Software state transitions under optimal testing strategy ($p_j; j = 1, 2, \ldots, N$, is the probability that a defect when the software under test contains j remaining defects will be detected and removed by the action determined by the optimal testing strategy).

Lemma 3.2. *The optimal control policy* ω^* *for the first-passage problem is determined by*

$$J_{\omega^*}(y_0) = \lim_{n\infty} v_{n+1}(y_0) \text{ for arbitrary } \{v_0(y), y \in S - \{y^*\}\}$$

3.2. The CMC approach

Up to this point we can see that Fig. 2 actually defines a special controlled Markov chain and the theory of controlled Markov chains can be used to determine the required optimal testing strategy. Following Lemma 3.1, we immediately conclude that there exists a deterministic stationary control policy (testing strategy) that minimizes $J_\omega(N)$ (Eq. (2.1)). Such a policy removes all the N defects in an optimal way through a series of appropriate actions. To compute the deterministic stationary control policy (testing strategy), we follow the method of successive approximation as indicated in Lemma 3.2.

Suppose the software under test is in state $Y_t = j$, $j \in \{N, N-1, \ldots, 2\}$. The optimal action A_t that should be taken at time t can be determined by the following recursive formula

$$v_{n+1}(j) = \min_{1\le i\le m}\left\{w_{Y_t}(i) + \sum_{k\neq 0} q_{jk}(i) v_n(k)\right\};\ \ n = 0, 1, 2, \ldots$$

where $q_{jk}(i)$ denotes the probability that action i transitions the software from state j to state k, and $\{v_0(k); k = 1, 2, \ldots, N\}$ are arbitrary.

Let

$$\lim_{n\infty} v_{n+1}(j) = v(j)$$

We have the following conclusion [6]

$$v(1) = \min_{1\le i\le m}\left\{\frac{w_1(i)}{\theta_i}\right\} \tag{3.3}$$

$$v(j) = \min_{1\le i\le m}\left\{\frac{w_j(i)}{j\theta_i} + v(j-1)\right\};\ j = 2, 3, \ldots, N \tag{3.4}$$

The above equations imply that in state $Y_t \neq 0$, the optimal action that should be taken is the one that minimizes $w_{Y_t}(i)/\theta_i$ with respect to i.

Example 3.1. Let $N = 3$. Suppose the action set consists of four actions, $A = \{1, 2, 3, 4\}$, and the corresponding defect detection probabilities are

$$\theta_1 = 0.5,\ \ \theta_2 = 0.3,\ \ \theta_3 = 0.2,\ \ \theta_4 = 0.1$$

Further,

$$w_3(1) = 9,\ \ w_3(2) = 7,\ \ w_3(3) = 4,\ \ w_3(4) = 3$$

$$w_2(1) = 25,\ \ w_2(2) = 10,\ \ w_2(3) = 8,\ \ w_2(4) = 5$$

$$w_1(1) = 35,\ \ w_1(2) = 17,\ \ w_1(3) = 10,\ \ w_1(4) = 8$$

$$w_0(1) = 0,\ \ w_0(2) = 0,\ \ w_0(3) = 0,\ \ w_0(4) = 0$$

Then by using Eqs. (3.3) and (3.4), we conclude that in state $Y_t = 3$, action 3 should be taken. In state $Y_t = 2$, action 2 should be taken. In state $Y_t = 1$, action 1 should be taken.

A special case is that $w_{Y_t}(i)$ does not depend on Y_t, or $w_{Y_t}(i) \equiv w(i)$ for $Y_t \neq 0$, then in all the non-target states, the optimal action is always the one that minimizes $w(i)/\theta_i$ with respect to i. We should devote all the testing resources to a single action.

In general in different states different actions might be taken since $w_{Y_t}(i)$ depends on Y_t. A_t thus switches from one action to another, and $\{A_t, t = 0, 1, 2, \ldots\}$ defines a test profile of software. For the optimal control policy (testing strategy), since it is deterministic stationary, the resulting $\{Y_t, t = 0, 1, 2, \ldots\}$ definitely makes up a Markov chain. However things are different for $\{A_t, t = 0, 1, 2, \ldots\}$ which is not necessarily Markovian.

4. Optimal test profile

Under optimal testing strategy, Fig. 2 is converted into Fig. 3. The software begins with state N (i.e. $Y_0 = N$), and the state transition behavior defines a homogeneous Markov chain. The question of interest here is whether the corresponding $\{A_t, t = 0, 1, 2, \ldots\}$ is a Markov chain too. This question (or Questions 4.1 and 4.2) is interesting for twofold reason. First, $\{A_t, t = 0, 1, 2, \ldots\}$ defines an optimal test profile and it is natural to analyze its behavior from a theoretical perspective. Second, Markovian test profile has been used for statistical software testing and its rationale is justified in engineering [12]. If $\{A_t, t = 0, 1, 2, \ldots\}$ is Markovian, then we provide a theoretical justification for Markovian statistical software testing. If not, Markovian statistical software testing is not optimal and should be improved. Theoretical analysis of $\{A_t, t = 0, 1, 2, \ldots\}$ may

shed some light on how to improve Markovian statistical software testing.

We ignore the trivial case that only one identical action is taken for all the software states, or in another word, A_t has only one state. Another trivial case is that A_t has one-to-one correspondence with Y_t, that is, no identical action is taken for two different software states. Then $\{A_t, t = 0, 1, 2, \ldots\}$ is certainly Markovian. In general cases, some different software states correspond to an identical action, some different software states require different actions. A_t is a deterministic function of Y_t. In order to examine the behavior of $\{A_t, t = 0, 1, 2, \ldots\}$, we distinguish several cases in the rest of this section. Note that Fig. 3 represents a reducible and homogenous Markov chain. This is different from the often encountered cases that $\{Y_t, t = 0, 1, 2, \ldots\}$ is an irreducible and homogenous and A_t is a deterministic or uncertain function of Y_t, for which the behavior of $\{A_t, t = 0, 1, 2, \ldots\}$ has been studied in the mathematics literature [14,15]. Roughly speaking, $\{Y_t, t = 0, 1, 2, \ldots\}$ is irreducible if each of its states can recur with non-zero probability. It is homogenous if its probabilistic behavior does not change over time.

4.1. Case 1

In this case software states N and $N - 1$ correspond to an identical action, and all the other software states have each one-to-one correspondence with actions. Simply, we can say that the original states N and $N - 1$ are aggregated into a hyperstate $B = \{N, N - 1\}$ and the other states remain unchanged. In the mathematical language, our question is as follows.

Question 4.1. Suppose $\{X_t, t = 0, 1, 2, \ldots; X_0 = N\}$ is represented by Fig. 3 with state space $\{N, N - 1, N - 2, \ldots, 1, 0\}$, is $\{X_t, t = 0, 1, 2, \ldots; X_0 = N\}$ still Markovian when the state space is transformed into $\{B, N - 2, \ldots, 1, 0\}$ with $B = \{N, N - 1\}$?

Proposition 4.1. *The answer to Question* 4.1 *is affirmative for* $N - 2$.

Proof. See Appendix A.

However we should note that $\{X_t, t = 0, 1, 2, \ldots; X_0 = N\}$ is no longer homogenous. In fact,

$$\Pr\{X_{n+1} = 0 | X_n \in B\} = \frac{\Pr\{X_{n+1} = 0, X_n = 1\}}{\Pr\{X_n = 1\} + \Pr\{X_n = 2\}} = \frac{p_1 \Pr\{X_n = 1\}}{\Pr\{X_n = 1\} + \Pr\{X_n = 2\}}$$

where

$$\Pr\{X_n = 1\} = \frac{p_2[(1 - p_1)^n - (1 - p_2)^n]}{p_2 - p_1},$$

$$\Pr\{X_n = 2\} = (1 - p_2)^n$$

Proposition 4.2. *The answer to Question* 4.1 *is affirmative for* $N = 3$.

Proof. See Appendix B.

Proposition 4.3. *The answer to* Question 4.1 *is affirmative for* $N > 3$.

Proof. See Appendix C.

Proposition 4.4. *Proposition* 4.3 *is still valid if the state space is replaced by* $\{B, j - 1, j - 2, \ldots, 1, 0\}$ *with* $B = \{N, N - 1, \ldots, j\}$.

Proof. Trivial by using Proposition 4.3 recursively.

4.2. Case 2

We consider the following question that can be viewed as dual to Question 4.1.

Question 4.2. Suppose $\{X_t, t = 0, 1, 2, \ldots; X_0 = N\}$ is represented by Fig. 3 with state space $\{N, N - 1, N - 2, \ldots, 1, 0\}$, is $\{X_t, t = 0, 1, 2, \ldots; X_0 = N\}$ still Markovian when the state space is transformed into $\{N, N - 1, \ldots, 3, 2, B_0\}$ with $B_0 = \{1, 0\}$?

Proposition 4.5. *The answer to* Question 4.2 *is affirmative for* $N = 2$.

Proof. See Appendix D.

Proposition 4.6. *The answer to* Question 4.2 *is affirmative for* $N = 3$.

Proof. See Appendix E.

Proposition 4.7. *The answer to* Question 4.2 *is affirmative for* $N > 3$.

Proof. For $N > 3$ the proof is similar to that of Proposition 4.6.

Proposition 4.8. *Proposition* 4.7 *is still valid if the state space is replaced by* $\{N, N - 1, \ldots, j, B_0\}$ *with* $B_0 = \{j - 1, j - 2, \ldots, 1, 0\}$.

Proof. Trivial by using Proposition 4.7 recursively.

4.3. Other cases

Consider Fig. 3 again. Let $N = 3$ and $B = \{1, 2\}$. That is, the state space is transformed into $\{3, B, 0\}$. Note

$$\Pr\{X_{n+1} = 0 | X_n \in B, X_{n-1} = 3, \ldots, X_0 = 3\} = \frac{\Pr\{X_{n+1} = 0, X_n \in B, X_{n-1} = 3, \ldots, X_0 = 3\}}{\Pr\{X_n \in B, X_{n-1} = 3, \ldots, X_0 = 3\}}$$

$$= \frac{\Pr\{X_{n+1} = 0, X_n = 2, X_{n-1} = 3, \ldots, X_0 = 3\} + \Pr\{X_{n+1} = 0, X_n = 1, X_{n-1} = 3, \ldots, X_0 = 3\}}{\Pr\{X_n = 2, X_{n-1} = 3, \ldots, X_0 = 3\} + \Pr\{X_n = 1, X_{n-1} = 3, \ldots, X_0 = 3\}} = 0$$

However

$$\Pr\{X_{n+1} = 0 | X_n \in B\} = \frac{\Pr\{X_{n+1} = 0, X_n \in B\}}{\Pr\{X_n \in B\}} = \frac{\Pr\{X_{n+1} = 0, X_n = 1\}}{\Pr\{X_n = 1\} + \Pr\{X_n = 2\}} = \frac{p_1 \times \Pr\{X_n = 1\}}{\Pr\{X_n = 1\} + \Pr\{X_n = 2\}} \neq 0$$

So $\{X_t, t = 0, 1, 2, \ldots; X_0 = 3\}$ is no longer Markovian with respect to the state space $\{3, B, 0\}$.

For $N = 3$, suppose $B = \{1, 3\}$. We have

$$\Pr\{X_{n+1} = 2 | X_n \in B, X_{n-1} \in B\} = \frac{\Pr\{X_{n+1} = 2, X_n \in B, X_{n-1} \in B\}}{\Pr\{X_n \in B, X_{n-1} \in B\}} = \frac{\Pr\{X_{n+1} = 2, X_n = 3, X_{n-1} = 3\}}{\Pr\{X_n = 3, X_{n-1} = 3\} + \Pr\{X_n = 1, X_{n-1} = 1\}}$$

$$= \frac{(1 - p_3)p_3 \Pr\{X_{n-1} = 3\}}{(1 - p_3)\Pr\{X_{n-1} = 3\} + (1 - p_1)\Pr\{X_{n-1} = 1\}} \neq 0$$

However

$$\Pr\{X_{n+1} = 2 | X_n \in B, X_{n-1} = 2\} = \frac{\Pr\{X_{n+1} = 2, X_n \in B, X_{n-1} = 2\}}{\Pr\{X_n \in B, X_{n-1} = 2\}}$$

$$= \frac{\Pr\{X_{n+1} = 2, X_n = 1, X_{n-1} = 2\} + \Pr\{X_{n+1} = 2, X_n = 3, X_{n-1} = 2\}}{\Pr\{X_n = 1, X_{n-1} = 3\}} = 0$$

So $\{X_t, t = 0, 1, 2, \ldots; X_0 = 3\}$ is no longer Markovian with respect to the state space $\{B, 2, 0\}$.

Again for $N = 3$, let $B = \{0, 2\}$. We have

$$\Pr\{X_{n+1} \in B | X_n \in B, X_{n-1} = 3\} = \frac{\Pr\{X_{n+1} \in B, X_n \in B, X_{n-1} = 3\}}{\Pr\{X_n \in B, X_{n-1} = 3\}} = \frac{\Pr\{X_{n+1} = 2, X_n = 2, X_{n-1} = 3\}}{\Pr\{X_n = 2, X_{n-1} = 3\}} = 1 - p_2$$

However

$$\Pr\{X_{n+1} \in B | X_n \in B, X_{n-1} \in B\} = \frac{\Pr\{X_{n+1} \in B, X_n \in B, X_{n-1} \in B\}}{\Pr\{X_n \in B, X_{n-1} \in B\}}$$

$$= \frac{\Pr\{X_{n+1} = 0, X_n = 0, X_{n-1} = 0\} + \Pr\{X_{n+1} = 2, X_n = 2, X_{n-1} = 2\}}{\Pr\{X_n = 0, X_{n-1} = 0\} + \Pr\{X_n = 2, X_{n-1} = 2\}} = \frac{\Pr\{X_{n-1} = 0\} + (1 - p_2)^2 \Pr\{X_{n-1} = 2\}}{\Pr\{X_{n-1} = 0\} + (1 - p_2)\Pr\{X_{n-1} = 2\}}$$

$$\neq 1 - p_2$$

Suppose $p_2 \neq 0$. So $\{X_t, t = 0, 1, 2, \ldots; X_0 = 3\}$ is no longer Markovian with respect to the state space $\{3, 1, B\}$.

Let $N = 4$ and $B = \{1, 3\}$. That is, we consider the state space {4,*B*,2,0}. Note

$$\Pr\{X_{n+1} = 2 | X_n \in B, X_{n-1} = 4\} = \frac{\Pr\{X_{n+1} = 2, X_n \in B, X_{n-1} = 4\}}{\Pr\{X_n \in B, X_{n-1} = 4\}} = \frac{\Pr\{X_{n+1} = 2, X_n = 3, X_{n-1} = 4\}}{\Pr\{X_n = 3, X_{n-1} = 4\}} = p_3$$

However

$$\Pr\{X_{n+1} = 2 | X_n \in B, X_{n-1} \in B\} = \frac{\Pr\{X_{n+1} = 2, X_n \in B, X_{n-1} \in B\}}{\Pr\{X_n \in B, X_{n-1} \in B\}} = \frac{\Pr\{X_{n+1} = 2, X_n = 3, X_{n-1} = 3\}}{\Pr\{X_n = 1, X_{n-1} = 1\} + \Pr\{X_n = 3, X_{n-1} = 3\}}$$

$$= \frac{p_3(1 - p_3)\Pr\{X_{n-1} = 3\}}{(1 - p_1)\Pr\{X_{n-1} = 1\} + (1 - p_3)\Pr\{X_{n-1} = 3\}} \neq p_3$$

So $\{X_t, t = 0, 1, 2, \ldots; X_0 = 4\}$ is no longer Markovian with respect to the state space {4,*B*,2,0}.

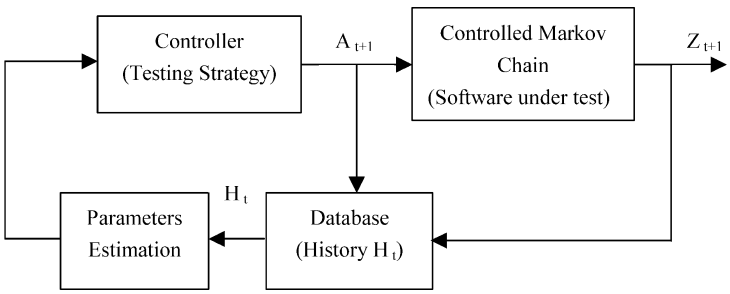

Fig. 4. Diagram of adaptive software testing

4.4. Conclusion

Up to this point we can conclude that the optimal test profile specifies a nonhomogenous Markov chain if two conditions are satisfied:

(1) A number of (possibly one) consecutive software states beginning with the initial software state or ending with the absorbing software state correspond to an identical action; and
(2) For all other software states, each corresponds to an action not taken in other states.

If these conditions are not satisfied, evidence shows that the corresponding optimal test profile is likely to be non-Markovian, although this observation needs further mathematical justification. Here we note that the absorbing software may correspond to an action not taken in any other state. Alternatively, we may say that the action taken in the absorbing state is the same as that taken in its neighbor state. This interpretation will not affect the validity of the above conclusion we draw.

5. Adaptive software testing

Adaptive control means that controller should be changed on-line during the period that the control system operates in accordance with the changes taking place in the controlled object. Accordingly, adaptive software testing means that software testing strategy should be adjusted on-line by using the testing data collected during software testing as our understanding of the software under test is improved. Specifically, in previous sections we implicitly assume that the parameters of concern such as N and $\theta_1, \theta_2, \ldots, \theta_m$ are known. In practice they are unknown and need to be estimated in the course of software testing. What we can observe directly are the actions taken $\{A_t, t = 0, 1, 2, \ldots\}$ and the corresponding outputs $\{Z_t, t = 0, 1, 2, \ldots\}$. From Section 3 we can see that in order to determine the optimal actions under certain states, we need to estimate or update the related parameters as new observables come up. In other words, the software parameters are estimated on-line and the corresponding optimal actions are determined based on the estimates of these parameters. This leads to an adaptive software testing strategy. A non-adaptive software testing strategy specifies what test suite or what next test case should be generated,[5] whereas an adaptive software testing strategy specified what next testing policy should be employed and thus in turn what test suite or next test case should be generated in accordance with the new testing policy.

In this section we reconsider the problem of Section 3 by including an on-line parameter estimation scheme in the decision-making process of selecting optimal actions. Fig. 4 depicts the closed-loop of adaptive software testing. We follow the assumptions and notation of Sections 3 and 4. In addition, we assume that the random variables $\{Z_t, t = 0, 1, 2, \ldots\}$ are independent.

Denote $z_{-1} = 0$. Let $\{z_t, t = 0, 1, 2, \ldots\}$ be realization of $\{Z_t, t = 0, 1, 2, \ldots\}$ and

$$r_t = \left(N - \sum_{j=-1}^{t-1} z_j\right)\theta_{a_t} \tag{5.1}$$

where a_t is realization of A_t. In this way r_t represents the probability that action a_t detects a defect. We have

$$\Pr\{Z_t = z_t\} = (r_t)^{z_t}(1 - r_t)^{1-z_t}; \quad t = 0, 1, 2, \ldots \tag{5.2}$$

Now given $z_0, z_1, \ldots, z_t$ and $a_0, a_1, \ldots, a_t$, following the least squares method of parameter estimation, we let

$$L(z_0, z_1, \ldots, z_t; a_0, a_1, \ldots, a_t) = \sum_{j=0}^{t} (z_j - r_j)^2 \tag{5.3}$$

Here r_j is determined by Eq. (5.1) and represents the expected value of Z_j, whereas z_j is the realization of Z_j in accordance with the statistical law of Eq. (5.4).

$$\Pr\{Z_t = z_t\} = \left[\left(N - \sum_{k=1}^{j-1} z_k\right)\theta_{a_j}\right]^{z_j}\left[1 - \left(N - \sum_{k=1}^{j-1} z_k\right)\theta_{a_j}\right]^{1-z_j};$$

$$z_t \in \{0, 1\}, \quad t = 0, 1, 2, \ldots \tag{5.4}$$

Suppose each of the m actions has been selected at least once up to time t, then the $m + 1$ parameters $N, \theta_1, \theta_2, \ldots, \theta_m$

[5] Ordinary testing strategies normally specify what test suites should be used. In random testing [4] or adaptive random testing [7] the next test case that should be generated is specified.

Table 1
Adaptive testing versus random testing: simulation results

n	$n_a(1)$	$n_r(1)$	$n_a(2)$	$n_r(2)$	$n_a(3)$	$n_r(3)$	$n_a(4)$	$n_r(4)$	$n_a(5)$	$n_r(5)$	$n_a(6)$	$n_r(6)$	$n_a(7)$	$n_r(7)$	$n_a(8)$	$n_r(8)$
10	21	18	37	23	18	19	13	21	16	18	12	15	15	25	14	19
11	22	19	38	24	20	24	14	23	19	20	14	16	16	28	17	24
12	23	20	42	29	21	25	17	33	21	31	17	23	19	29	18	26
13	24	28	43	31	26	26	24	34	26	32	18	26	20	31	24	28
14	26	33	44	43	27	32	26	41	28	34	21	33	22	32	28	41
15	27	37	48	48	28	35	30	42	31	35	22	35	23	33	29	46
16	29	41	52	59	32	37	40	50	33	39	24	36	27	34	38	58
17	38	47	59	64	34	40	41	59	37	43	25	37	28	39	39	59
18	44	49	61	68	39	42	45	62	43	48	26	39	31	41	41	63
19	45	52	67	69	42	44	46	63	44	49	30	42	40	48	43	66
20	46	60	71	71	53	57	52	72	45	56	32	51	45	50	45	69
21	47	61	76	79	57	59	61	83	49	65	35	55	52	52	49	79
22	53	64	80	80	61	63	67	90	50	67	40	57	64	62	50	88
23	58	69	88	86	63	64	69	101	52	76	46	61	79	67	52	92
24	59	76	95	98	65	84	70	104	54	77	50	68	91	68	57	95
25	60	77	97	101	68	100	75	122	56	105	54	79	95	75	61	125
26	64	90	105	106	77	109	79	127	77	110	55	87	98	82	67	132
27	67	94	123	125	82	114	91	131	93	125	60	106	103	107	92	158
28	78	97	135	152	87	115	116	161	101	187	80	118	111	118	96	174
29	91	140	153	182	93	127	127	214	105	188	139	163	127	166	114	192
30	104	232	171	252	96	329	147	280	175	219	205	237	163	331	131	250

n: number of software defects that are detected and removed; $n_a(i)$: number of test cases that are used by adaptive testing to detect and remove the first n software defects in the ith simulation run; $n_r(i)$: number of test cases that are used by random testing to detect and remove the first n software defects in the ith simulation run.

can be estimated by minimizing the function $L(z_0, z_1, \ldots, z_t; a_0, a_1, \ldots, a_t)$ (at least in theory). Denote the resulting estimates as $N^{(t+1)}, \theta_1^{(t+1)}, \theta_2^{(t+1)}, \ldots, \theta_m^{(t+1)}$. Or[6]

$$N^{(t+1)} = N(z_0, z_1, \ldots, z_t; a_0, a_1, \ldots, a_t),$$
$$\theta_i^{(t+1)} = \theta_i(z_0, z_1, \ldots, z_t; a_0, a_1, \ldots, a_t); \quad i = 1, 2, \ldots, m \tag{5.5}$$

By using the so-called certainty-equivalence principle or the method of substituting the estimates into optimal stationary controls [10, p. 38], we treat $N^{(t+1)}, \theta_1^{(t+1)}, \theta_2^{(t+1)}, \ldots, \theta_m^{(t+1)}$ as the true values of the corresponding parameters at time $t+1$ and take the optimal action based on these values. Consequently, we obtain the following adaptive control policy (adaptive software testing strategy).

Step 1 Initialize parameters. Set

$$z_{-1} = 0, \quad N^{(0)} = N_0, \quad \theta_i^{(0)} = \theta_{0i}; \qquad i = 1, 2, \ldots, m$$

$$Y_0 = N^{(0)}, \quad A_0 = \arg \min_{1 \le k \le m} \left\{ \frac{w_{Y_0}(k)}{\theta_k^{(0)}} \right\}, \quad t = 0$$

Step 2 Estimate parameters by minimizing $L(z_0, z_1, \ldots, z_t; a_0, a_1, \ldots, a_t)$ (Eq. (5.3)) and obtain

$$N^{(t+1)} = N(z_0, z_1, \ldots, z_t; a_0, a_1, \ldots, a_t)$$
$$\theta_i^{(t+1)} = \theta_i(z_0, z_1, \ldots, z_t; a_0, a_1, \ldots, a_t); \quad i = 1, 2, \ldots, m$$

If θ_i doesn't appear in $L(z_0, z_1, \ldots, z_t; a_0, a_1, \ldots, a_t)$, or action i has not been taken in the previous actions up to time t, then we let $\theta_i^{(t+1)} = \theta_i^{(t)}$.

Step 3 Decide the optimal action

$$A_{t+1} = \arg \min_{1 \le k \le m} \left\{ \frac{w_{Y_{t+1}}(k)}{\theta_k^{(t+1)}} \right\}$$

Step 4 Observe the testing result $Z_{t+1} = z_{t+1}$ activated by the action A_{t+1}.

Step 5 Update the current software state

$$Y_{t+1} = N^{(t+1)} - \sum_{j=-1}^{t} z_j$$

Step 6 Set $t = t + 1$.

Step 7 If a given testing stopping criterion is satisfied, then stop testing;[7] otherwise go to Step 2.

In order to verify the effectiveness of the adaptive

[6] A commonly used way of parameter estimation is to differentiate $L(z_0, z_1, \ldots, z_t; a_0, a_1, \ldots, a_t)$ with respect to each parameter of concern and obtain a system of nonlinear equations. However sometimes the system of nonlinear equations may not have a solution. So, a more general way of doing parameter estimation is to minimize $L(z_0, z_1, \ldots, z_t; a_0, a_1, \ldots, a_t)$ directly.

[7] A theoretical stopping criterion is $Y_{t+1} = 0$ since the goal behind our optimal control policy (Step 3) is to remove all the defects at least expected cost. However in practice $Y_{t+1} = 0$ can seldom be satisfied since $N^{(t+1)}$ is just an estimate of N. For example, a more practical stopping criterion is that Y_{t+1} is below some given threshold.

strategy of software testing, we compare it with random testing by simulation. In random testing, since no information of software defect detection rates is used, we just follow a uniform distribution of all admissible test cases (actions) to select a test case each time. In adaptive testing, testing data are used to do on-line estimations of software defect detection rates that are required to select an optimal test case (action) each time. This should reduce the number of test cases (actions) required to detect and remove a certain number of software defects. To justify this argument, suppose the action set consists of two test cases (actions), $A = \{1, 2\}$. Further, $\theta_1 = 0.025$, $\theta_2 = 0.01$. The optimal action should be 1 in theory. However the true values of software defect detection rates are unknown for software tester and thus on-line parameter estimations are necessary for choosing an optimal action. Let $N = 30$, that is, the software under test contains 30 defects at the beginning. The uniform distribution used in the random testing is $\langle 0.5, 0.5 \rangle$. Table 1 tabulates the simulation results for the numbers of test cases (actions) required to detect and remove the defects remaining in the software under test. The least squares method of parameter estimation is used in the adaptive testing, where the genetic algorithm provided in the Matlab environment is used to minimize $L(z_0, z_1, \ldots, z_t; a_0, a_1, \ldots, a_t)$. The true values of $\theta_1 = 0.025$, $\theta_2 = 0.01$ are used as their initial values in the simulation. The average number of test cases for the random testing to detect and remove all the 30 defects is 266.25, while the average number of test cases for the adaptive testing to detect and remove all the 30 defects is 149. Adaptive testing significantly reduces the number of test cases (actions) required to detect and remove a certain number of software defects.

6. Concluding Remarks

In the preceding sections we have discussed how to treat software testing as a control problem and how the controlled Markov chains (CMC) approach to software testing determines an optimal testing strategy. We also introduce an adaptive software testing strategy that is treated as the counterpart of adaptive control in software testing. Under the optimal testing strategy, the software state transitions behave as a Markov chain. However the corresponding optimal test profile is not necessarily Markovian. If a number of consecutive software states beginning with the initial software state or ending with the absorbing software state correspond to an identical action, and for all other states each corresponds to an action not taken in other states, then the corresponding optimal test profile is Markovian. In other cases, evidence shows that the corresponding optimal test profile is not Markovian. The analyses presented so far deepen our understanding of the CMC approach to software testing. In the meantime, they also provide theoretical justification that under certain circumstance software test profile or operational profile fits a Markov model. We note that homogenous Markov models have been extensively used to characterize software operational profile [4]. They are also used in the so-called model-based software testing [12]. The analyses of this paper suggest that nonhomogenous Markov models should also be considered for software operational profile modeling and statistical software testing. The optimal test profile also provides a potential solution to the test case prioritization problem by treating the problem as a control problem in the context of software cybernetics. The test case prioritization problem is concerned with how to select a permutation of a given test suite such that a given quantitative function (e.g. rate of defect detection) is optimized [9,13]. On the other hand, simulation results show that the adaptive software testing strategy is superior to random software testing strategy. It can significantly reduce the number of test cases required to detect and remove a certain number of software defects. The adaptive software testing strategy can widely be applied since the term 'action' can widely be interpreted.

Acknowledgments

The simulation results presented in Section 5 were obtained with the help of Yongchao Li. The comments of T Y Chen and anonymous reviewers on earlier versions of this paper greatly help the author to improve the readability of the paper.

Appendix A. Proof of Proposition 4.1

We can consider different cases one by one to reach the required conclusion.

(1)

$$\Pr\{X_{n+1} = 0 | X_n \in B, X_{n-1} \in B, \ldots, X_0 \in B\}$$

$$= \frac{\Pr\{X_{n+1} = 0, X_n \in B, X_{n-1} \in B, \ldots, X_0 \in B\}}{\Pr\{X_n \in B, X_{n-1} \in B, \ldots, X_0 \in B\}}$$

$$= \frac{\Pr\{X_{n-1} \in B, \ldots, X_0 \in B | X_{n+1} = 0, X_n \in B\} \Pr\{X_{n+1} = 0, X_n \in B\}}{\Pr\{X_{n-1} \in B, \ldots, X_0 \in B | X_n \in B\} \Pr\{X_n \in B\}}$$

Note

$$\Pr\{X_{n-1} \in B, \ldots, X_0 \in B | X_{n+1} = 0, X_n \in B\} = 1$$

$$\Pr\{X_{n-1} \in B, \ldots, X_0 \in B | X_n \in B\} = 1$$

Thus

$$\Pr\{X_{n+1} = 0 | X_n \in B, X_{n-1} \in B, \ldots, X_0 \in B\}$$

$$= \Pr\{X_{n+1} = 0 | X_n \in B\}$$

(2)

$$\Pr\{X_{n+1} \in B | X_n \in B, X_{n-1} \in B, \ldots, X_0 \in B\} = 1 - \Pr\{X_{n+1} = 0 | X_n \in B, X_{n-1} \in B, \ldots, X_0 \in B\} = \Pr\{X_{n+1} \in B | X_n \in B\}$$

(3)

$$\Pr\{X_{n+1} = 0 | X_n = 0, \ldots, X_{n-k} = 0, X_{n-k-1} \in B, \ldots, X_0 \in B\}$$

$$= \frac{\Pr\{X_{n-k-2} \in B, \ldots, X_0 \in B | X_{n+1} = 0, X_n = 0, \ldots, X_n = 0, X_{n-k-1} \in B\} \times \Pr\{X_{n+1} = 0, X_n = 0, \ldots, X_n = 0, X_{n-k-1} \in B\}}{\Pr\{X_{n-k-2} \in B, \ldots, X_0 \in B | X_n = 0, \ldots, X_n = 0, X_{n-k-1} \in B\} \times \Pr\{X_n = 0, \ldots, X_n = 0, X_{n-k-1} \in B\}}$$

$$= \frac{\Pr\{X_{n+1} = 0, X_n = 0, \ldots, X_n = 0, X_{n-k-1} \in B\}}{\Pr\{X_n = 0, \ldots, X_n = 0, X_{n-k-1} \in B\}}$$

$$= \frac{\Pr\{X_{n+1} = 0, X_n = 0, \ldots, X_n = 0, X_{n-k-1} = 1\} + \Pr\{X_{n+1} = 0, X_n = 0, \ldots, X_n = 0, X_{n-k-1} = 2\}}{\Pr\{X_n = 0, \ldots, X_n = 0, X_{n-k-1} = 1\} + \Pr\{X_n = 0, \ldots, X_n = 0, X_{n-k-1} = 2\}}$$

$$= \frac{\Pr\{X_{n+1} = 0, X_n = 0, \ldots, X_n = 0, X_{n-k-1} = 1\}}{\Pr\{X_n = 0, \ldots, X_n = 0, X_{n-k-1} = 1\}} = \Pr\{X_{n+1} = 0 | X_n = 0, \ldots, X_{n-k} = 0, X_{n-k-1} \in B\} = \Pr\{X_{n+1} = 0 | X_n = 0\}$$

(4)

$$\Pr\{X_{n+1} \in B | X_n = 0, \ldots, X_{n-k} = 0, X_{n-k-1} \in B, X_0 \in B\} = 0 = \Pr\{X_{n+1} \in B | X_n = 0\}$$

We conclude that $\{X_t, t = 0, 1, 2, \ldots; X_0 = N\}$ is Markovian with respect to the sate space $\{B, 0\}$. □

Appendix B. Proof of Proposition 4.2

(1)

$$\Pr\{X_{n+1} \in B | X_n \in B, \ldots, X_0 \in B\} = \Pr\{X_{n+1} \in B | X_n \in B\}$$

(2)

$$\Pr\{X_{n+1} = 1 | X_n \in B, \ldots, X_0 \in B\} = \frac{\Pr\{X_{n-1} \in B, \ldots, X_0 \in B | X_{n+1} = 1, X_n \in B\} \Pr\{X_{n+1} = 1, X_n \in B\}}{\Pr\{X_{n-1} \in B, \ldots, X_0 \in B | X_n \in B\} \Pr\{X_n \in B\}}$$

$$= \frac{\Pr\{X_{n+1} = 1, X_n \in B\}}{\Pr\{X_n \in B\}} = \Pr\{X_{n+1} = 1 | X_n \in B\}$$

(3)

$$\Pr\{X_{n+1} = 1 | X_n = 1, \ldots, X_{n-k} = 1, X_{n-k-1} \in B, \ldots, X_0 \in B\}$$

$$= \frac{\Pr\{X_{n+1} = 1, X_n = 1, \ldots, X_{n-k} = 1, X_{n-k-1} \in B, \ldots, X_0 \in B\}}{\Pr\{X_n = 1, \ldots, X_{n-k} = 1, X_{n-k-1} \in B, \ldots, X_0 \in B\}}$$

$$= \frac{\Pr\{X_{n+1} = 1, X_n = 1, \ldots, X_{n-k} = 1, X_{n-k-1} \in B\}}{\Pr\{X_n = 1, \ldots, X_{n-k} = 1, X_{n-k-1} \in B\}} = \Pr\{X_{n+1} = 1 | X_n = 1, \ldots, X_{n-k} = 1, X_{n-k-1} = 2\}$$

$$= \Pr\{X_{n+1} = 1 | X_n = 1\}$$

(4)

$$\Pr\{X_{n+1} = 0 | X_n \in B, \ldots, X_0 \in B\} = 0 = \Pr\{X_{n+1} = 0 | X_n \in B\}$$

(5)

$$\Pr\{X_{n+1}=0|X_n=1,\ldots,X_{n-k}=1,X_{n-k-1}\in B,\ldots,X_0\in B\}$$

$$=\frac{\Pr\{X_{n+1}=0,X_n=1,\ldots,X_{n-k}=1,X_{n-k-1}\in B,\ldots,X_0\in B\}}{\Pr\{X_n=1,\ldots,X_{n-k}=1,X_{n-k-1}\in B,\ldots,X_0\in B\}}$$

$$=\frac{\Pr\{X_{n+1}=0,X_n=1,\ldots,X_{n-k}=1,X_{n-k-1}=2\}}{\Pr\{X_n=1,\ldots,X_{n-k}=1,X_{n-k-1}=2\}}=\Pr\{X_{n+1}=0|X_n=1\}$$

(6)

$$\Pr\{X_{n+1}=0|X_n=0,\ldots,X_{n-k}=0,X_{n-k-1}=1,\ldots,X_{n-j}=1,X_{n-j-1}\in B,\ldots,X_0\in B\}$$

$$=\frac{\Pr\{X_{n+1}=0,X_n=0,\ldots,X_{n-k}=0,X_{n-k-1}=1,\ldots,X_{n-j}=1,X_{n-j-1}\in B\}}{\Pr\{X_n=0,\ldots,X_{n-k}=0,X_{n-k-1}=1,\ldots,X_{n-j}=1,X_{n-j-1}\in B\}}=\Pr\{X_{n+1}=0|X_n=0\}$$

□

Appendix C. Proof of Proposition 4.3

We only consider one general case here. Other simple cases can be handled similarly as those in the proofs of Propositions 4.1 and 4.2. For $j=N-2,N-3,\ldots,1,0$, we have

$$\Pr\{X_{n+1}=j|X_n=j,\ldots,X_{n_{j-1}+1}=j,X_{n_{j-1}}=j+1,\ldots,X_{n_{j-2}+1}=j+1,X_{n_{j-2}}=j+2,\ldots,X_{n_{N-2}+1}=N-3,\ldots,X_{n_{N-2}}$$

$$=N-2,X_{n_{N-2}-1}\in B,\ldots,X_0\in B\}$$

$$=\frac{\Pr\{X_{n+1}=j,X_n=j,\ldots,X_{n_{j-1}+1}=j,X_{n_{j-1}}=j+1,\ldots,X_{n_{j-2}+1}=j+1,X_{n_{j-2}}=j+2,\ldots,X_{n_{N-2}+1}=N-3,\ldots,X_{n_{N-2}}=N-2,X_{n_{N-2}-1}\in B\}}{\Pr\{X_n=j,\ldots,X_{n_{j-1}+1}=j,X_{n_{j-1}}=j+1,\ldots,X_{n_{j-2}+1}=j+1,X_{n_{j-2}}=j+2,\ldots,X_{n_{N-2}+1}=N-3,\ldots,X_{n_{N-2}}=N-2,X_{n_{N-2}-1}\in B\}}$$

$$=\frac{\Pr\{X_{n+1}=j,X_n=j,\ldots,X_{n_{j-1}+1}=j,X_{n_{j-1}}=j+1,\ldots,X_{n_{j-2}+1}=j+1,X_{n_{j-2}}=j+2,\ldots,X_{n_{N-2}+1}=N-3,\ldots,X_{n_{N-2}}=N-2,X_{n_{N-2}-1}=N-1\}}{\Pr\{X_n=j,\ldots,X_{n_{j-1}+1}=j,X_{n_{j-1}}=j+1,\ldots,X_{n_{j-2}+1}=j+1,X_{n_{j-2}}=j+2,\ldots,X_{n_{N-2}+1}=N-3,\ldots,X_{n_{N-2}}=N-2,X_{n_{N-2}-1}=N-1\}}$$

$$=\Pr\{X_{n+1}=j|X_n=j\}$$

□

Appendix D. Proof of Proposition 4.5

(1)

$$\Pr\{X_{n+1}=2|X_n=2,\ldots,X_0=2\}=\Pr\{X_{n+1}=2|X_n=2\}$$

(2)

$$\Pr\{X_{n+1}\in B_0|X_n=2,\ldots,X_0=2\}=\Pr\{X_{n+1}=1|X_n=2,\ldots,X_0=2\}+\Pr\{X_{n+1}=0|X_n=2,\ldots,X_0=2\}=\Pr\{X_{n+1}$$

$$\in B_0|X_n=2\}$$

(3)

$$\Pr\{X_{n+1} \in B_0 | X_n \in B_0, \ldots, X_{n-k} \in B_0, X_{n-k-1} = 2, \ldots, X_0 = 2\}$$

$$= \frac{\Pr\{X_{n+1} \in B_0, X_n \in B_0, \ldots, X_{n-k} \in B_0, X_{n-k-1} = 2, \ldots, X_0 = 2\}}{\Pr\{X_n \in B_0, \ldots, X_{n-k} \in B_0, X_{n-k-1} = 2, \ldots, X_0 = 2\}}$$

$$= \frac{\Pr\{X_{n-k-2} = 2, \ldots, X_0 = 2 | X_{n+1} \in B_0, X_n \in B_0, \ldots, X_{n-k} \in B_0, X_{n-k-1} = 2\} \times \Pr\{X_{n+1} \in B_0, X_n \in B_0, \ldots, X_{n-k} \in B_0, X_{n-k-1} = 2\}}{\Pr\{X_{n-k-2} = 2, \ldots, X_0 = 2 | X_n \in B_0, \ldots, X_{n-k} \in B_0, X_{n-k-1} = 2\} \times \Pr\{X_n \in B_0, \ldots, X_{n-k} \in B_0, X_{n-k-1} = 2\}}$$

$$= \frac{\Pr\{X_{n+1} \in B_0, X_n \in B_0, \ldots, X_{n-k} \in B_0, X_{n-k-1} = 2\}}{\Pr\{X_n \in B_0, \ldots, X_{n-k} \in B_0, X_{n-k-1} = 2\}} = \frac{\Pr\{X_{n+1} \in B_0, X_n \in B_0, \ldots, X_{n-k+1} \in B_0, X_{n-k} = 1, X_{n-k-1} = 2\}}{\Pr\{X_n \in B_0, \ldots, X_{n-k} \in B_0, X_{n-k} = 1, X_{n-k-1} = 2\}}$$

$$= \frac{\Pr\{X_{n+1} \in B_0, X_n \in B_0, \ldots, X_{n-k+1} \in B_0 | X_{n-k} = 1, X_{n-k-1} = 2\}}{\Pr\{X_n \in B_0, \ldots, X_{n-k+1} \in B_0 | X_{n-k} = 1, X_{n-k-1} = 2\}} = \frac{\Pr\{X_{n+1} \in B_0, X_n \in B_0, \ldots, X_{n-k+1} \in B_0 | X_{n-k} = 1\}}{\Pr\{X_n \in B_0, \ldots, X_{n-k+1} \in B_0 | X_{n-k} = 1\}}$$

$$= \frac{\Pr\{X_{n+1} \in B_0, X_n \in B_0, \ldots, X_{n-k+1} \in B_0, X_{n-k} = 1\}}{\Pr\{X_n \in B_0, \ldots, X_{n-k+1} \in B_0, X_{n-k} = 1\}}$$

Note

$$\Pr\{X_{n+1} \in B_0, X_n \in B_0, \ldots, X_{n-k+1} \in B_0, X_{n-k} = 1\} = \Pr\{X_{n+1} = 1, X_n \in B_0, \ldots, X_{n-k+1} \in B_0, X_{n-k} = 1\}$$

$$+ \Pr\{X_{n+1} = 0, X_n \in B_0, \ldots, X_{n-k+1} \in B_0, X_{n-k} = 1\} = \Pr\{X_{n+1} = 1, X_n = 1, \ldots, X_{n-k+1} = 1, X_{n-k} = 1\}$$

$$+ \Pr\{X_{n+1} = 0, X_n = 1, X_{n-1} \in B_0, \ldots, X_{n-k+1} \in B_0, X_{n-k} = 1\}$$

$$+ \Pr\{X_{n+1} = 0, X_n = 0, X_{n-1} \in B_0, \ldots, X_{n-k+1} \in B_0, X_{n-k} = 1\} = \Pr\{X_{n+1} = 1, X_n = 1, \ldots, X_{n-k+1} = 1, X_{n-k} = 1\}$$

$$+ \Pr\{X_{n+1} = 0, X_n = 1, \ldots, X_{n-k+1} = 1, X_{n-k} = 1$$

$$+ \sum_{x_{n-1}, \ldots, x_{n-k+1} \in B_0} \Pr\{X_{n+1} = 0, X_n = 0, X_{n-1} = x_{n-1} \ldots, X_{n-k+1} = x_{n-k+1}, X_{n-k} = 1\}$$

$$= \Pr\{X_n = 1, \ldots, X_{n-k+1} = 1, X_{n-k} = 1\} + \sum_{x_{n-1}, \ldots, x_{n-k+1} \in B_0} \Pr\{X_{n+1} = 0 | X_n = 0, X_{n-1} = x_{n-1} \ldots, X_{n-k+1} = x_{n-k+1}, X_{n-k} = 1\}$$

$$\times \Pr\{X_n = 0, X_{n-1} = x_{n-1} \ldots, X_{n-k+1} = x_{n-k+1}, X_{n-k} = 1\} = \Pr\{X_n = 1, X_{n-1} \in B_0, \ldots, X_{n-k+1} \in B_0, X_{n-k} = 1\}$$

$$+ \sum_{x_{n-1}, \ldots, x_{n-k+1} \in B_0} \Pr\{X_n = 0, X_{n-1} = x_{n-1} \ldots, X_{n-k+1} = x_{n-k+1}, X_{n-k} = 1\}$$

$$= \Pr\{X_n = 1, X_{n-1} \in B_0, \ldots, X_{n-k+1} \in B_0, X_{n-k} = 1\} + \Pr\{X_n = 0, X_{n-1} \in B_0, \ldots, X_{n-k+1} \in B_0, X_{n-k} = 1\}$$

$$= \Pr\{X_n \in B_0, X_{n-1} \in B_0, \ldots, X_{n-k+1} \in B_0, X_{n-k} = 1\}$$

Therefore

$$\Pr\{X_{n+1} \in B_0 | X_n \in B_0, \ldots, X_{n-k} \in B_0, X_{n-k-1} = 2, \ldots, X_0 = 2\} = 1 = \Pr\{X_{n+1} \in B_0 | X_n \in B_0\}$$

□

Appendix E. Proof of Proposition 4.6

We only consider one case. The other cases are easy to verify. There holds

$$\Pr\{X_{n+1} \in B_0 | X_n \in B_0, \ldots, X_{n-k} \in B_0, X_{n-k-1} = 2, \ldots, X_{n-j} = 2, X_{n-j-1} = 3, \ldots, X_0 = 3\}$$

$$= \frac{\Pr\{X_{n+1} \in B_0, X_n \in B_0, \ldots, X_{n-k} \in B_0, X_{n-k-1} = 2, \ldots, X_{n-j} = 2, X_{n-j-1} = 3\}}{\Pr\{X_n \in B_0, \ldots, X_{n-k} \in B_0, X_{n-k-1} = 2, \ldots, X_{n-j} = 2, X_{n-j-1} = 3\}}$$

Note

$$\Pr\{X_{n+1} \in B_0, X_n \in B_0, \ldots, X_{n-k} \in B_0, X_{n-k-1} = 2, \ldots, X_{n-j} = 2, X_{n-j-1} = 3\}$$

$$= \Pr\{X_{n+1} \in B_0, X_n \in B_0, \ldots, X_{n-k+1} \in B_0, X_{n-k} = 1, X_{n-k-1} = 2, \ldots, X_{n-j} = 2, X_{n-j-1} = 3\}$$

$$= \Pr\{X_{n+1} = 1, X_n \in B_0, \ldots, X_{n-k+1} \in B_0, X_{n-k} = 1, X_{n-k-1} = 2, \ldots, X_{n-j} = 2, X_{n-j-1} = 3\}$$

$$+ \Pr\{X_{n+1} = 0, X_n \in B_0, \ldots, X_{n-k+1} \in B_0, X_{n-k} = 1, X_{n-k-1} = 2, \ldots, X_{n-j} = 2, X_{n-j-1} = 3\}$$

$$= \Pr\{X_{n+1} = 1, X_n = 1, \ldots, X_{n-k+1} = 1, X_{n-k} = 1, X_{n-k-1} = 2, \ldots, X_{n-j} = 2, X_{n-j-1} = 3\}$$

$$+ \Pr\{X_{n+1} = 0, X_n = 1, \ldots, X_{n-k+1} = 1, X_{n-k} = 1, X_{n-k-1} = 2, \ldots, X_{n-j} = 2, X_{n-j-1} = 3\}$$

$$+ \Pr\{X_{n+1} = 0, X_n = 0, X_{n-1} \in B_0, \ldots, X_{n-k+1} \in B_0, X_{n-k} = 1, X_{n-k-1} = 2, \ldots, X_{n-j} = 2, X_{n-j-1} = 3\}$$

$$= \Pr\{X_n = 1, \ldots, X_{n-k+1} = 1, X_{n-k} = 1, X_{n-k-1} = 2, \ldots, X_{n-j} = 2, X_{n-j-1} = 3\}$$

$$+ \Pr\{X_n = 0, X_{n-1} \in B_0, \ldots, X_{n-k+1} \in B_0, X_{n-k} = 1, X_{n-k-1} = 2, \ldots, X_{n-j} = 2, X_{n-j-1} = 3\}$$

$$= \Pr\{X_n \in B_0, X_{n-1} \in B_0, \ldots, X_{n-k+1} \in B_0, X_{n-k} = 1, X_{n-k-1} = 2, \ldots, X_{n-j} = 2, X_{n-j-1} = 3\}$$

Therefore

$$\Pr\{X_{n+1} \in B_0 | X_n \in B_0, \ldots, X_{n-k} \in B_0, X_{n-k-1} = 2, \ldots, X_{n-j} = 2, X_{n-j-1} = 3, \ldots, X_0 = 3\} = 1 = \Pr\{X_{n+1} \in B_0 | X_n \in B_0\}$$

□

References

[1] A. Avritzer, E.J. Weyuker, The automatic generation of load test suites and the assessment of the resulting software, IEEE Transactions on Software Engineering 21 (9) (1995) 705–716.

[2] B. Beizer, Software Testing Techniques, second ed., Van Nostrand Reinhold, New York, 1990.

[3] K.Y. Cai, On the concepts of total systems, total dependability and software cybernetics (unpublished manuscript), Centre for Software Reliability, City University, London, Draft version, October 1994; revised version, 1995.

[4] K.Y. Cai, Software Defect and Operational Profile Modeling, Kluwer Academic Publishers, Dordrecht, 1998.

[5] K.Y. Cai, Towards a conceptual framework of software run reliability modeling, Information Sciences 126 (2000) 137–163.

[6] K.Y. Cai, A controlled Markov chains approach to software testing, IEEE Transactions on Software Engineering, submitted for publication, 2002.

[7] T.Y. Chen, T.H. Tse, Y.T. Yu, Proportional sampling strategy: a compendium and some insights, Journal of Systems and Software 58 (2001) 65–81.

[8] C. Derman, Finite State Markovian Decision Processes, Academic Press, London, 1970.

[9] S. Elbaum, A.G. Malishevsky, G. Rothermel, Test case prioritization: a family of empirical studies, IEEE Transactions on Software Engineering 28 (2) (2002) 159–182.

[10] O. Hernandez-Lerma, Adaptive Markov Control Processes, Springer, Berlin, 1989.

[11] O. Hernandez-Lerma, J.B. Lasserre, Discrete-Time Markov Control Processes: Basic Optimality Criteria, Springer, Berlin, 1996.

[12] J.H. Poore, Introduction to the special issue on: model-based statistical testing of software intensive systems, Information and Software Technology 42 (2000) 797–799.

[13] G. Rothermel, R.H. Untch, C. Chu, M.J. Harrold, Prioritizing test cases for regression testing, IEEE Transactions on Software Engineering 27 (10) (2001) 929–948.

[14] G. Rubino, B. Sericola, A finite characterization of weak lumpable Markov processes. Part I: the discrete time case, Stochastic Processes and Their Applications 38 (1991) 195–204.

[15] P. Spreij, On the Markov property of a finite hidden Markov chain, Statistics and Probability Letters 52 (2001) 279–288.

[16] J.A. Whittaker, M.G. Thomason, A. Markov, Chain model for statistical software testing, IEEE Transactions on Software Engineering 20 (10) (1994) 812–824.