A Probabilistic Analysis of Coverage Methods

Research Thesis

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
INFORMATION MANAGEMENT ENGINEERING

Ekaterina Kutsy

Submitted to the Senate of
the Technion - Israel Institute of Technology

SIVAN 5768 - HAIFA - JUNE 2008

Table of Contents

List of Figures

Abstract

Testing of hardware and software frequently cost more than the development of the product itself. As a result there is a constant effort, both in the industry and in the academia, to make this process as efficient as possible. Automatic test generators are tools that generate test vectors (i.e., input sequences) that are run on the tested program / hardware, and the outputs are then compared to some predefined specification (ideally this process is fully automatic). The simplest type of automatic test generation is generating random input vectors. There are various analytic results showing the effectiveness of such testing, assuming a uniform distribution of the tests. However, uniformly generated tests are not the only way tests are generated in the industry.

Hardware testing is typically done by modeling the environment of the tested component with a set of constraints. Every solution to this set of constraints is a test, and hence such constraints-based test generators produce a large number of solutions. A typical scenario is that of beginning the testing process by defining a *coverage model*, which is a partition of the sampling space to *coverage goals* according to the developer's knowledge of the expected functionality of the tested product. Then, using a constraint solver, various constraints are added in order to *direct* the constraint solver to goals that were not yet covered. Another possible scenario is that random sampling is combined with such directed sampling in various ways. Directed sampling has a cost, since a more constrained model is many times harder to solve. There is therefore a trade-off between cheaper sampling – achieved by generating random samples – and directed sampling that have a higher probability of covering

the predefined goals. In this thesis we examine four testing models, each of which makes different assumptions about the error distribution, the testing environment, the associated costs, and the possible testing strategies. In each such model we compute the probability to find an error, assuming it exists, or the expected cost until achieving full coverage of the goals.

Acknowledgements

I would like to thank Dr. Ofer Strichman, my supervisor, for his many suggestions and constant support during this research. I am also thankful to him for his guidance and *patience*.

I thank Laurent Fournier from IBM Corp. who expressed his interest in my work, supplied me with the necessary information about functional coverage model and provided useful references. I also thank Avi Ziv from IBM Corp. who helped me understand the world of verification.

The generous financial help of the Technion is gratefully acknowledged.

Of course, I am grateful to my family for its patience and *love*.

Finally, I wish to thank my friends Anna and Polyna for sharing with me their knowledge, providing useful advices and friendly encouragement.

Haifa, Israel Ekaterina Kutsy
May 26, 2008

Chapter 1

Introduction

Testing is an integral, inevitable part of any sufficiently large development process of software and hardware. Both the hardware and software industries face problems related to the expenses of the process, balancing between the quality of the product on one hand, and the time-to-market and cost on the other hand. According to [1], the budget dedicated to testing ranges from 40 to 80 percent of the overall project's cost[1].

The problem with testing, as Dijstra once said, is that it can only expose problems, not prove their absence. It is a major problem, then, to estimate the value of a given test effort. The main technique for measuring the thoroughness of testing is called *test coverage analysis*, or simply *coverage*. The main idea is to partition the set of states, lines of code, statements etc., and to check whether each one of these sets were sampled in the given set of test vectors (also called the *test suite*). A given partition is called a *coverage model*.

There are two general types of coverage: *code coverage* and *functional coverage* [1].Given a set of test vectors (sequences of inputs), the first type of coverage measures things such as 'statement coverage' (percentage of statements in the code that were visited in one or more of the tests), 'branch coverage' (percentage of branch statements

[1]In the software industry, these numbers keep going up: Bill Gates declared a few years ago that in Microsoft for every developer there is a full time testing engineer, and more recently it was published that only 1 out of 6 engineers in Microsoft do product development, whereas the rest are dedicated to testing and maintenance.

that were evaluated to both true and false) and so forth. Code coverage applies to both software and hardware, the latter through coverage of code written in a hardware-description language such as VHDL. Functional coverage is more flexible and not tied to the code structure, rather to user-defined behaviors, or *functionalities*, of the tested system. Hence, one can define test *goals*, or *subsets* of states to be covered. For example, when testing a circuit that includes a buffer, the tester can specify two goals to cover: a run of the system in which the buffer is under-full, and another run in which it overflows. Given a set of test vectors, the tester can run these tests on the checked system and determine, either manually or automatically, whether both goals were covered. If they were not covered, he/she can add more tests manually or even use a system for automatically generating tests that are directed towards the non-covered subsets (an approach called *coverage-driven test-generation*).

The main tool in the hardware domain for directing tests is to model the environment of the tested system, abstractly, with a set of constraints over the inputs of the tested system (and possibly additional variables local to the environment). Any solution to these constraints (more precisely, the projection of these solutions to the inputs of the tested system) is a sequence of inputs that reflect a 'behavior' of the real world in which the tested hardware is supposed to function. The problem of solving such a set of arbitrary constraints (over finite, discrete domains) is known as the Constraints Satisfaction Problem, or CSP. Indeed, the hardware testing industry develops and / or uses commercial CSP solvers to generate tests. The description of the environment in CSP terms is typically abstract and can be under- or over-approximating of the real environment, and typically even both at the same time (i.e., the modeled environment allows non-realistic behaviors and at the same time disallows real behaviors). The process of coverage-driven test generation corresponds to changing the CSP model. Note that a CSP model can have many solutions, and the specific solution given by the CSP solver is simply the first it finds, and hence can be considered as arbitrary. Therefore directing the solver towards a solution with

specific attributes amounts to adding constraints. If the CSP model is overly constrained, on the other hand, constraints may need to be removed in order to cover some of the goals.

The iterative process of changing the CSP model for achieving the coverage goals has an associated cost. This is in contrast to random testing in which the cost per test is negligible, but has a smaller chance of succeeding to cover the specified coverage goals. This tradeoff between cost and quality under various coverage model schemes is the subject of this research: we develop probabilistic models under which various test generation strategies can be analyzed and compared. Specifically, for a given test generation strategy, we compute its probability to expose a logical error (given a model of the error, as will be explained later) and its expected cost. For some of the testing models we show an optimal testing strategy in terms of the probability to detect the error or the overall cost, depending on the model.

We assume that the coverage model itself is given (the definition of the coverage goals), and hence concentrate on the value of various test generation strategies in terms of their ability to cover these predefined coverage goals, and the cost for executing them.

There are some previous works that tried to estimate the success of a testing effort based on various techniques which are quite far from the current work. We delay the description of these works to Chapter 7.

This research is a joint work with the IBM-HRL verification group, which develops various tools for hardware testing. The running example used in this proposal is a realistic example of testing a floating-point unit of a microprocessor [5].

Chapter 2

A cross-product functional coverage model

We begin with several general definitions.

A cross-product functional coverage model is defined by:

- A set of attributes (variables) $V = \{v_1, \ldots, v_n\}$.
- A set of domains $D = \{D_1, \ldots, D_n\}$ corresponding to the attributes.
- A set of partitions $P = \{P_1, \ldots, P_n\}$ of the attributes' domains.
- A set of constraints/restrictions $C = \{C_1, \ldots, C_k\}$. These constraints define (only approximately, in practice) which of the assignments can occur in the real environment of the tested system.

Example 1. *In a cross-product coverage model for a floating-point unit, the attributes include:*

- *Instruction,*
- *Operand-I,*
- *Operand-II,*
- *Result,*
- *Round-Mode,*

- *Round-Occur,*
- *Interrupt-enable-mode,*
- *Status register mode and so forth.*

The domain of the attribute Instruction *is:* fadd, fsub, fdiv, fmul, fabs, fneg, fsel, fmr*, etc. A possible partition of this domain is to a set of arithmetic instructions and a set of non-arithmetic instructions, with the values* fadd, fsub, fdiv, fmul*,... and* fabs, fneg, fsel, fmr*,... respectively.*

□

The state-space D is defined as the cartesian product of the Domains $D_1,\ldots,D_n$. A coverage model is a partition of D. Each of the elements of the partition – which are subsets of the possible assignments – is called a *coverage goal*, and we say that such a goal is *covered* by a test-suite if at least one of the tests results in a valuation corresponding to that goal. The coverage goals of a cross-product functional coverage model CM is the Cartesian product of the partitions $P_1,\ldots,P_n$. Not all resulting goals are *legal*, however. For example, the cross-product of a *fabs* Instruction and a negative Result is an illegal event.

Example 2. *Consider once again the floating-point unit. We define the following cross-product coverage model, based on partitioning the domain of only two variables:*

- *Let v_1 and v_2 represent the operands attributes (corresponding to* Operand-I *and* Operand-II *above).*
- *The respective domains of v_1 and v_2 are D_1 and D_2, and are equal to 128 bits each.*
- *The respective partitions P_1 and P_2 are equal to:*

  $\{\pm 0, \pm\infty, \pm NaN, \pm Norm, \pm Denorm\}$*. There is a mapping between each value in D_1, D_2 to one of these partitions.*

The coverage model is the Cartesian product of P_1 and P_2 and hence has 10^2 elements.

Note that there are other attributes in the tested system, such as Round-mode*,* Result*, and so forth. But these attributes are not partitioned for this coverage model.*

□

A logical error (i.e., 'bug') is defined as a (possibly empty) subset $B \subseteq D$. The goal of a coverage model is to increase the probability of finding at least one of the elements of B for a given test effort. The coverage model expresses the tester's presumed knowledge on the system's structure and functionality. For instance, in the above example the numbers in the domain of v_1 and v_2 are partitioned to 10 categories which are known by the tester to activate different parts of the logic.

We now describe four different *testing models*, each of which assumes a specific functional coverage model scheme, various assumptions on the error distribution (B), the cost of directed sampling, the bound (if exists) on the number of tests, and so forth. In the first two models the goal is to maximize the probability of finding an error. In the last two the goal is to minimize the cost of achieving full coverage.

Given the initial parameters of each such testing model, we derive a probabilistic model for each of the above mentioned strategies and compare them in order to find the optimal strategy. Note that we have multiple objectives (probability to find the bug and cost), which complicates the definition of the 'best' strategy.

Chapter 3

Testing Model 1

3.1 Model definition

In this model the tested system is modeled by a set of Boolean variables (bits, from hereon) $V = v_1, \ldots, v_n$, and correspondingly, both the error distribution B and the coverage model are given as sets of bits. Since these are binary variables, the partition of the domains is such that a variable's domain can either be partitioned to {0},{1} or not at all. We call the former set the *coverage-model visible* bits, and the latter set the *coverage-model invisible* bits, and denote the latter by *ci*. This terminology reflects the fact that the former set of bits are those that we are interested in observing their values, whereas the latter set are those bits that we ignore. Hence, there are $2^{|V \setminus ci|}$ goals to cover, each of which contains $2^{|ci|}$ assignments.

Example 3. *Consider a system with 5 bits $v_1, \ldots, v_5$, from which $ci = \{v_1, v_4\}$ are coverage-model invisible. This means that the state-space is partitioned by the coverage model to eight coverage goals (2^{5-2}), one goal for each possible assignment to the visible bits. In other words, each goal is an abstraction, referring to the invisible bits as 'don't cares'. For example the partition $v_2 = 0, v_3 = 0, v_5 = 0$ contains the 4 assignments:*

$$v_1 = 0, v_2 = 0, v_3 = 0, v_4 = 0, v_5 = 0$$
$$v_1 = 0, v_2 = 0, v_3 = 0, v_4 = 1, v_5 = 0$$
$$v_1 = 1, v_2 = 0, v_3 = 0, v_4 = 0, v_5 = 0$$
$$v_1 = 1, v_2 = 0, v_3 = 0, v_4 = 1, v_5 = 0$$

□

To model the error distribution B, the model assumes a partition of the bits to *bug visible bits* and *bug invisible bits*, the latter denoted by *bi*. The former have a specific Boolean value, whereas the latter are 'don't cares', i.e., each assignment that agrees with the bug visible bits corresponds to a bug. Hence, there are $2^{|bi|}$ assignments that expose the error. Since this definition of a set of bugs correspond to the set defined by the cross-product of the *bi* variables, we refer to it as a *product bug*. Every tested system has a (possible empty) union of product bugs. For simplicity, in this testing model we assume a single product-bug, and a single cross-product coverage model.

Example 4. *Consider the system described in the previous example with 5 bits* $v_1, \ldots, v_5$. *Assume* $bi = \{v_1, v_2\}$, *i.e., these are bug invisible bits, hence there are* 2^2 *erroneous assignments.*

□

3.2 Test generation strategies

- *Directed*: A strategy in which the test is directed to a given coverage goal by means of additional constraints to the CSP modeling of the environment. This strategy has a high cost due to the additional effort of modeling the constraints that impose the selection of a sample from a given goal and also additional cost of solving the more-constrained model.

- *Random*: A strategy in which tests are chosen in random. The Random strategy has a low cost.[1]

In this testing model we compare a directed sampling strategy that achieves 100% coverage (i.e., using $2^{|bi \setminus ci|}$ samples), and a random strategy with the number of samples $2^{|V \setminus ci|}$. The question is: given the sets *ci* and *bi*, how do the two sampling strategies compare?

[1] In practice uniformly distributed samples are very hard to generate, since an automatically generated test is a result of solving a CSP problem, and consecutive solutions are similar to each other (only few variable values are different between consecutive solutions). To improve the distribution such solvers introduce randomness in various parts of the algorithm, but still uniformity is not guaranteed. Nevertheless in our analysis we consider this distribution as being uniform.

Results for this model

Our results in this section show that:

- When the cost of directed sampling is disregarded (i.e., considered to be equal to that of random testing), directed testing is always better or equal to the random strategy.
- When the cost of directed sampling is higher than some threshold, random testing is better.

3.3 The probability to expose an error with the Directed and Random sampling strategies

As explained above, there are $2^{|V \setminus ci|}$ goals that we want to cover, each of which contains $2^{|ci|}$ assignments. There are $2^{|bi|}$ assignments that expose the error, distributed evenly over $2^{|bi \setminus ci|}$ goals. Hence, the number of errors in each such goal is equal to:

$$\frac{2^{|bi|}}{2^{|bi \setminus ci|}} = 2^{|bi \cap ci|}, \tag{3.1}$$

whereas there is no error in the other goals. First consider the directed sampling strategy:

- The probability to find an error in a goal of the first set of goals (those goals that contain assignments that expose the error) is:

$$\frac{2^{|bi \cap ci|}}{2^{|ci|}} = 2^{-|ci \setminus bi|} . \tag{3.2}$$

- The overall probability to find an error is:

$$1 - \left(1 - 2^{-|ci \setminus bi|}\right)^{2^{|bi \setminus ci|}} . \tag{3.3}$$

Example 5. *We demonstrate the formulas above for two cases:*

- *The case $ci \subseteq bi$: $V = \{v_1, v_2, v_3, v_4, v_5\}$, $bi = \{v_1, v_2\}$ and $ci = \{v_1\}$. There are $2^{|V|-|ci|} = 2^{5-1} = 16$ goals to cover, and in each goal there are $2^{|ci|} = 2$ assignments, $2^{|bi|} = 4$ of them expose the error. Only $2^{|bi-ci|} = 2$ of 16 partitions contain $2^{|bi \cap ci|} = 2$ errors. The probability to find an error in one sample is $2^{-|ci-bi|} = 2^0 = 1$. The overall probability to find an error, according to (3.3), is:*

$$1 - (1 - 2^{-|ci \setminus bi|})^{2^{|bi \setminus ci|}} = 1 - (1 - 2^0)^{2^1} = 1 \; .$$

- *The case $ci \not\subseteq bi$: $V = \{v_1, v_2, v_3, v_4, v_5\}$, $bi = \{v_1, v_2\}$ and $ci = \{v_1, v_3\}$. There are $2^{|V \cap ci|} = 2^{5-2} = 8$ goals to cover, in each there are $2^{|ci|} = 4$ assignments, $2^{|bi|} = 4$ of which expose the error. Only $2^{|bi \setminus ci|} = 2$ of 8 goals contain $2^{|bi \cap ci|} = 2$ errors. The probability to expose an error in one of them is $2^{-|ci \cap bi|} = 2^{-1} = 0.5$. The overall probability to expose one or more errors, according to (3.3), is:*

$$1 - (1 - 2^{-|ci \setminus bi|})^{2^{|bi \setminus ci|}} = 1 - (1 - 2^{-1})^{2^1} = 0.75 \; .$$

Hence the probability to find an error with this sampling strategy is maximal (=1) when $ci \subseteq bi$. □

Now consider the *Random* sampling strategy. As before, we will make $2^{|V \setminus ci|}$ samples (equivalent to the number of partitions to cover). The overall probability to expose an error is then:

$$1 - \left(1 - 2^{|bi \setminus V|}\right)^{2^{|V \setminus ci|}} = 1 - \left(1 - 2^{-(|V \setminus bi|)}\right)^{2^{|V \setminus ci|}} \; . \tag{3.4}$$

With the Random sampling strategy the overall probability to expose an error when $|bi| = |ci|$ and $|V| \to \infty$ is:

$$1 - e^{-1} \approx 0.63 \; ,$$

whereas with the Directed sampling strategy the overall probability to expose an error in the case $bi = ci$ is equal to 1.

Is the Directed sampling strategy always superior to the random one? To answer this question we compare in the next subsection equations (3.3) and (3.4).

3.4 A comparison of the Directed and Random sampling strategies

Assume the following cost of a single sample, in each of the sampling strategies:

- *Random*: cost = 1.
- *Directed*: cost = C.

As defined in Sect. 3.1, there are $2^{|bi \setminus ci|}$ goals with erroneous assignments. The total number of directed samples is equal to $2^{|bi \setminus ci|} * n$ where n is a number of samples in each subset ($n \geq 1$)[2]. A total sampling cost in this case is equal to $2^{|bi \setminus ci|} * n * C$.

At the same time we can make $2^{|bi \setminus ci|} * n * C$ random samples instead of $2^{|bi \setminus ci|} * n$ directed samples (C random samples instead of a single directed sample).

We can compare the two strategies in the following way.

$$\text{P(not to find an error with directed sampling)} = \left(1 - \frac{1}{2^{(|ci \setminus bi|)}}\right)^{2^{(|bi \setminus ci|)} * n}$$

$$\text{P(not to find an error with random sampling)} = \left(1 - \frac{1}{2^{(|V \setminus bi|)}}\right)^{2^{(|bi \setminus ci|)} * n * C}$$

Since $|V| \geq |ci|$, the probability not to find an error is lower in one directed sample than it is in one random sample.

However, the probability not to find an error with the Random strategy decreasing when increasing C (the probability not to find an error with the Directed strategy stays unchanged). Therefore start from some value of C the probability not to find the error with the Random strategy becomes lower than it is with the Directed strategy.

We now find a value of C for which the probability not to find an error is equal in both strategies, assuming the overall cost is the same:

[2]For simplicity of the model assume that with directed testing we perform an equal number of tests in each subset.

$$\left(1-\frac{1}{2^{(|ci\backslash bi|)}}\right)^{2^{(|bi\backslash ci|)}*n} = \left(1-\frac{1}{2^{(|V\backslash bi|)}}\right)^{2^{(|bi\backslash ci|)}*n*C}. \tag{3.5}$$

We take logarithm on both sides to make it easier to extract C:

$$\ln\left(1-\frac{1}{2^{(|ci\backslash bi|)}}\right)^{2^{(|bi\backslash ci|)}*n} = \ln\left(1-\frac{1}{2^{(|V\backslash bi|)}}\right)^{2^{(|bi\backslash ci|)}*n*C}. \tag{3.6}$$

We now have

$$2^{(|bi\backslash ci|)} * n * \ln\left(1-\frac{1}{2^{(|ci\backslash bi|)}}\right) = 2^{(|bi\backslash ci|)} * n * C * \ln\left(1-\frac{1}{2^{(|V\backslash bi|)}}\right)$$

which implies that

$$C = \frac{\ln\left(1-\frac{1}{2^{(|ci\backslash bi|)}}\right)}{\ln\left(1-\frac{1}{2^{(|V\backslash bi|)}}\right)}. \tag{3.7}$$

If a cost of one directed sample is higher than C, then the random strategy is better. Otherwise the directed sampling is preferable.

This result is conditional on decreasing the probability not to find an error when increasing C. The term $1-\frac{1}{2^{(|V\backslash bi|)}}$ is smaller than 1, and therefore increasing C leads to decreasing of the probability not to find an error:

$$\left(1-\frac{1}{2^{(|V\backslash bi|)}}\right)^{2^{(|bi\backslash ci|)}*n*(C+1)} = \left[\left(1-\frac{1}{2^{(|V\backslash bi|)}}\right)^{C} * \left(1-\frac{1}{2^{(|V\backslash bi|)}}\right)\right]^{2^{(|bi\backslash ci|)}*n}.$$

Therefore the probability not to find an error with the Random strategy monotonic decreases when increasing C.

Equation (3.7) shows that either random or directed sampling is better, depending on the value of C. Hence, there is no advantage to a *hybrid* strategy (i.e., switching between these two strategies).

Chapter 4

Testing Model 2

4.1 Model definition

In this testing model we assume a partition of the space of all possible assignments to k goals. Denote by N_i the number of assignments in goal i, and let $N \doteq \sum_{i=1}^{k} N_i$. Assume that the error is such that it is only exposed within a single goal. The goal parameters are:

- $n_i \leq N_i$ is the number of assignments in goal i that expose the error, assuming the error is in goal i.
- p_i denotes the probability that the bug is in goal i, where $\sum_{i=1}^{k} p_i = 1$.

4.2 Test generation strategies

The four strategies that we study in this model are called *Random, Round-Robin, Hybrid* and *Directed*:

- *Directed*: See Sect. 3.2. In this testing model, however, there is no restriction on whether the number of tests in each coverage goal is the same.
- *Random*: See Sect. 3.2.

- *Round-Robin*: A variant of the directed strategy, by which the coverage goals are sampled following a cyclic order. It is assumed that the cycles are complete, i.e., each goal is sampled an equal number of times.

- *Hybrid*: A combination of the previous two strategies.

Directed sampling is commonly used with the aid of expert knowledge as to which goal to sample next. In Sect. 4.5 we will study automatic guidance of this strategy that leads to optimality in terms of the expected cost until exposing the bug.

Random sampling is important in case we have no information about the bug distribution or the probability to expose it in each goal.

Round-Robin sampling, like directed sampling, has a high cost due to the additional effort of directing the samples through additional constraints. This strategy is commonly used in the verification world due to its simplicity, and the fact that it does not rely on expert's knowledge as to where to look next.

Hybrid sampling raises the problem of finding the optimal point to switch between the Random and Round-Robin strategies. Further, this point can be different if we determine a-priory the overall cost that we are willing to invest in testing. In other words, knowing the number of tests left and the number of uncovered goals, allows us to compute the optimal point of changing the strategy.

We can conduct a sampling process in this model in two different ways:

- The *recurring* case - sampling with replacement (assignments that were previously sampled are not removed from the sampling space).

- The *non-recurring* case - sampling without replacement (assignments that were previously sampled are removed from the sampling space).

The next two sections are dedicated to these two ways of performing the samples, with the Round-Robin and Random strategies.

Results for this model

Our results show that:

- Recurring case: When the cost of a random sample and a single round-robin sample are assumed to be equal, it can still be the case that random testing is better.
- Recurring case: There are some cases in which the Hybrid strategy is preferable over both pure strategies.
- Nonrecurring case: Here we derive formulas for computing the probability to find an error in the Round-Robin and Random strategies, but are unable to compare between them.
- Both recurring and non-recurring cases: exists Directed strategy that leads to minimal expected cost of a testing process.

4.3 Random, Round-Robin and Hybrid sampling strategies: recurring case

Assume that there is a bound M on the number of allowed samples. With the Random strategy, the next tested assignment is chosen randomly, hence so is the goal i to which this sample belongs. With the Round-Robin strategy, each goal i is sampled $C = \frac{M}{k}$ times.[1]

4.3.1 The probability to expose an error

The Random strategy

- The probability not to find the bug in one sample if the bug is in goal i:

$$\mathrm{P}(\text{not to find a bug in one sample} \mid \text{bug is in goal } i) =$$

$$= \mathrm{P}(\text{not to find a bug in one sample} \mid \text{bug is in goal } i, \text{we sampled goal } i) *$$

$$* \mathrm{P}(\text{we sampled goal } i \mid \text{bug is in goal } i) +$$

[1] For simplicity of the model assume that M is a multiple of k.

+P(not to find a bug in one sample | bug is in goal i, we did not sample goal i)$*$

$$*\text{P(we didn't sample goal } i \mid \text{bug is in goal } i) =$$

$$= \left(1 - \frac{n_i}{N_i}\right) * \frac{N_i}{N} + 1 * \left(1 - \frac{N_i}{N}\right) =$$

$$= \left(\frac{N_i}{N} - \frac{n_i}{N_i} * \frac{N_i}{N}\right) + \left(1 - \frac{N_i}{N}\right) = 1 - \frac{n_i}{N} \tag{4.1}$$

- The probability not to find the bug in one sample based on (4.1):

$$\text{P(not to find a bug in one sample)} =$$
$$= \sum_{i=1}^{k} \text{P(not to find a bug in one sample} \mid \text{bug is in goal } i) * p_i =$$
$$= \sum_{i=1}^{k} \left(1 - \frac{n_i}{N}\right) * p_i \tag{4.2}$$

- The overall probability to find the error in M samples:

$$1 - \sum_{i=1}^{k} \left(1 - \frac{n_i}{N}\right)^M * p_i \ . \tag{4.3}$$

The Round-Robin strategy

- The probability not to find the error in one directed sample:

$$\text{P(not to find a bug in one sample in goal } i) =$$
$$= \text{P(not to find a bug in one sample in goal } i \mid \text{bug is in goal } i) * p_i =$$
$$= \left(1 - \frac{n_i}{N_i}\right) * p_i \tag{4.4}$$

- The overall probability to expose the error in M samples:

$$1 - \sum_{i=1}^{k} \left(1 - \frac{n_i}{N_i}\right)^{\frac{M}{k}} * p_i \ . \tag{4.5}$$

Under what sampling strategy is the probability to find an error better? The answer depends on the parameters of the coverage model and distribution of the assignments in it that expose the error. The examples below demonstrate this dependency.

Example 6. *Assume a partition with $N = 1500$, 2 goals and $M = 40$. The following table presents various values of n_1, n_2, N_1 and N_2, and the corresponding probabilities.*

n_1	N_1	p_1	n_2	N_2	p_2	*Random*	*Round-Robin*
250	*1000*	*0.5*	*50*	*500*	*0.5*	*0.8708*	*0.9376*
50	*1000*	*0.5*	*250*	*500*	*0.5*	*0.8708*	*0.8207*
200	*800*	*0.5*	*100*	*700*	*0.5*	*0.9667*	*0.9755*
100	*800*	*0.5*	*200*	*700*	*0.5*	*0.9667*	*0.9647*

□

The Hybrid strategy

The Hybrid sampling strategy makes random samples and then continues with Round-Robin sampling. Denote by D the total number of Round-Robin samples (hence, we make $M - D$ Random samples). The probability to find the error with the Hybrid strategy may be higher than both Random and Round-Robin, as we will later demonstrate with an example.

The overall probability not to expose an error in k goals with D directed and $M - D$ random samples is equal to:

$$f(D) \doteq \sum_{i=1}^{k} \left(1 - \frac{n_i}{N}\right)^{M-D} * \left(1 - \frac{n_i}{N_i}\right)^{\frac{D}{k}} * p_i \ . \tag{4.6}$$

Using this formula, the following example demonstrates that the Hybrid strategy can be better than both Random and Round-Robin.

Example 7. *Assume a partition with $N = 1500$, 2 goals and $M = 40$. The following table presents various values of n_1, n_2, N_1 and N_2, and the corresponding probabilities.*

n_1	N_1	p_1	n_2	N_2	p_2	*Random*	*Round-Robin*	*Hybrid*
100	*100*	*0.01*	*200*	*9900*	*0.99*	*0.55*	*0.35*	*0.5598 (D=2)*
10	*100*	*0.4*	*200*	*9900*	*0.6*	*0.3482*	*0.5524*	*0.5575 (D=30)*

□

In order to find the optimal value of D, discrete optimization tools can be used, for a given values of n_i, N_i, p_i for all $i = \{1, \ldots, k\}$. In Appendix A we show that approximating the optimal value by considering D as a continuous number is infeasible.

We can prove, however, that there is a single optimal value of D.

Proposition 1. *The probability not to find an error with the Hybrid sampling is a convex function with a single point of minimum.*

Proof. We use the following equivalence:

$$\begin{aligned}
&\left(1-\frac{n_i}{N}\right)^{M-D} * \left(1-\frac{n_i}{N_i}\right)^{\frac{D}{k}} = \\
&\frac{\left(1-\frac{n_i}{N}\right)^{M}}{\left(1-\frac{n_i}{N}\right)^{D}} * \left(1-\frac{n_i}{N_i}\right)^{\frac{D}{k}} = \\
&\left(1-\frac{n_i}{N}\right)^{M} * \left[\frac{\left(1-\frac{n_i}{N_i}\right)}{\left(1-\frac{n_i}{N}\right)^{k}}\right]^{\frac{D}{k}} .
\end{aligned} \tag{4.7}$$

Rewriting (4.6) according to (4.7) yields:

$$f(D) \doteq \sum_{i=1}^{k} \left(1-\frac{n_i}{N}\right)^{M} * \left[\frac{\left(1-\frac{n_i}{N_i}\right)}{\left(1-\frac{n_i}{N}\right)^{k}}\right]^{\frac{D}{k}} * p_i \ . \tag{4.8}$$

The right component of (4.7) can be either smaller or greater than 1. For a given i, (4.7) is a convex function of D. Since a sum of convex functions is convex, then $f(D)$ is convex, which means that, because D is finite, it has a single finite optimal value. □

In some cases it is possible to compute the optimal value without invoking a discrete optimization algorithm. Denote by $ratio_i$ the ratio $\frac{\left(1-\frac{n_i}{N_i}\right)}{\left(1-\frac{n_i}{N}\right)^{k}}$. Then, from (4.8) we can see that:

- If $ratio_i > 1$ for all i, the random sampling is better than any hybrid one, i.e., $D = 0$ is the optimal point.
- If $ratio_i < 1$ for all i the directed sampling is better than any hybrid one, i.e., $D = M$ is the optimal point.

The following example shows that in other cases the optimal value of D depends on M.

Example 8. *Assume a partition with* $N = 17000$, $k = 5$, $p_i = \frac{1}{5}$*:*

n_i	N_i	$ratio_i$
2	*4000*	*1.000088*
2	*7000*	*1.000303*
3	*1500*	*0.998881*
10	*2500*	*0.998935*
10	*2000*	*0.997932*

The optimal value of D was computed numerically for various values of M as can be seen in Fig. ***4.1****. As we can see in this figure, up to some value of M it is better to use only directed sampling, and afterwards some mix is optimal.*

□

Note that the discussion in this section assumed a single parameter D, and hence $\frac{D}{k}$ samples in each coverage goal. The more general case, in which a different number of samples for each goal is possible, will be studied as part of Sect. 4.5, which considers the Directed strategy.

4.3.2 The probability to expose an error with a cost

We now extend the previous analysis to the case that there is an associated cost C with the Round-Robin strategy. Hence, we can make C random samples for the price of one directed sample. We assume that there is testing budget and attempt to find the best strategy that still satisfies this budget. The probability *not* to find an error under the Hybrid strategy now is equal to:

$$f(D) \doteq \sum_{i=1}^{k} \left(1 - \frac{n_i}{N}\right)^{M - C*D} * \left(1 - \frac{n_i}{N_i}\right)^{\frac{D}{k}} * p_i$$

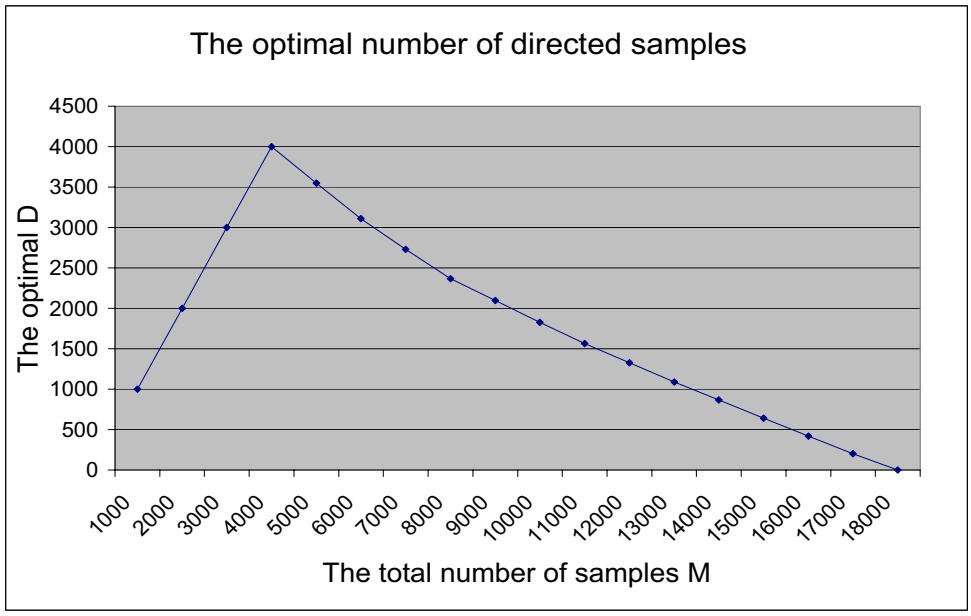

***Figure 4.1**: The optimal number of directed samples in example 8*

The probability to expose an error under the Hybrid strategy – based on (4.8) – is:

$$f(D) \doteq \sum_{i=1}^{k} \left(1 - \frac{n_i}{N}\right)^M * \left(\frac{\left(1 - \frac{n_i}{N_i}\right)}{(1 - \frac{n_i}{N})^{C*k}}\right)^{\frac{D}{k}} * p_i \ . \tag{4.9}$$

As in the previous section, in some cases it is possible to compute the optimal value without invoking a discrete optimization algorithm. We define

$$ratio_i^C \doteq \left(\frac{\left(1 - \frac{n_i}{N_i}\right)}{(1 - \frac{n_i}{N})^{C*k}}\right)^{\frac{D}{k}} .$$

Then we can see that:

- if $ratio_i^C > 1$ for all i, then the optimal point is when $D = 0$.
- if $ratio_i^C < 1$ for all i, then the optimal point is when $D = \frac{M}{k}$.

Note that sufficiently increasing C can make all ratios be larger than 1, and hence encourages the use of the random strategy, as expected.

4.3.3 Expected number of samples until error detection

We now consider the expected number of samples until error detection with the Random and Round-Robin strategies. The total cost is then simply this number multiplied by the respective cost.

The Round-Robin strategy

Let M denote the number of samples until the error is detected. The expected value of M is:

$$E(M) = \sum_{j=1}^{\infty} j * p(M = j) = \sum_{j=1}^{\infty} j \sum_{i=1}^{k} p(M = j | bug\ in\ i) * p_i$$

We denote by l the index of the round, hence:

$$\sum_{l=0}^{\infty} \sum_{i=1}^{k} (l * k + i) * p(M = l * k + i | bug\ in\ i) * p_i =$$

$$= \sum_{l=0}^{\infty} \sum_{i=1}^{k} (l * k + i) * \left(1 - \frac{n_i}{N_i}\right)^l * \frac{n_i}{N_i} * p_i =$$

$$\sum_{i=1}^{k} \left[\sum_{l=0}^{\infty} (l * k + i) * \left(1 - \frac{n_i}{N_i}\right)^l * \frac{n_i}{N_i} \right] * p_i =$$

$$\sum_{i=1}^{k} \left[\sum_{l=0}^{\infty} i * \left(1 - \frac{n_i}{N_i}\right)^l * \frac{n_i}{N_i} + \sum_{l=0}^{\infty} l * k * \left(1 - \frac{n_i}{N_i}\right)^l * \frac{n_i}{N_i} \right] * p_i =$$

$$= \sum_{i=1}^{k} \left[i * \frac{n_i}{N_i} * \sum_{l=0}^{\infty} \left(1 - \frac{n_i}{N_i}\right)^l + k * \sum_{l=0}^{\infty} l * \left(1 - \frac{n_i}{N_i}\right)^l * \frac{n_i}{N_i} \right] * p_i \ .$$

Since $\sum_{l=0}^{\infty} \left(1 - \frac{n_i}{N_i}\right)^l$ is a geometric series with a base smaller than 1, this is equal to:

$$\sum_{i=1}^{k} \left[i * \frac{n_i}{N_i} * \left(\frac{1}{1 - \left(1 - \frac{n_i}{N_i}\right)} \right) + \right.$$

$$+k * \left(1 - \frac{n_i}{N_i}\right) * \sum_{l=0}^{\infty} l * \left(1 - \frac{n_i}{N_i}\right)^{l-1} * \frac{n_i}{N_i}\Bigg] * p_i \ .$$

Since $\sum_{l=0}^{\infty} l * \left(1 - \frac{n_i}{N_i}\right)^{l-1} * \frac{n_i}{N_i}$ is the formula for computing the expected value over a geometric-distribution with $p = \frac{n_i}{N_i}$, this is equivalent to:

$$\begin{aligned} & \sum_{i=1}^{k} \left[i * \frac{n_i}{N_i} * \frac{N_i}{n_i} + k * \left(1 - \frac{n_i}{N_i}\right) * \frac{N_i}{n_i} \right] * p_i = \\ = & \sum_{i=1}^{k} \left[i + k * \left(\frac{N_i}{n_i} - 1\right) \right] * p_i = \\ = & \sum_{i=1}^{k} i * p_i + k * \sum_{i=1}^{k} \left(\frac{N_i}{n_i}\right) * p_i - k \ . \end{aligned} \tag{4.10}$$

As we can see from (4.10), the order of goals in the Round-Robin sampling strategy affects the expected number of samples until finding the error. In particular, the best strategy is to order the goals in decreasing values of p_i (this will minimize the left component of (4.10)). This matches our intuition: starting with the goal with the highest probability to contain the error, reduces the overall search time.

A special case

Now assume a special case: the probability of the error to be in goal i is equal for all i, i.e. $p_i = \frac{1}{k}$ for all i. The other parameters remain unchanged.

Claim 1. *If the probability of the error to be in goal i is equal for all i, then the expected number of samples until error detection is independent of the ordering of the goals during a Round-Robin sampling process.*

Assigning $p_i = \frac{1}{k}$ in (4.10) yields:

$$\begin{aligned} E(M) &= \sum_{i=1}^{k} i * \frac{1}{k} + k * \sum_{i=1}^{k} \left(\frac{N_i}{n_i}\right) * \frac{1}{k} - k = \\ &= \frac{k*(k+1)}{2} * \frac{1}{k} + \sum_{i=1}^{k} \frac{N_i}{n_i} - k = \\ &= \frac{1-k}{2} + \sum_{i=1}^{k} \frac{N_i}{n_i} \, . \end{aligned} \tag{4.11}$$

As we can see from (4.11), the result in this case does not depend on the ordering of the different goals in the cycle. The obtained result is non-intuitive, because one would expect that the values of n_i, N_i affect the best order. In other words, it seems reasonable that starting from the goal in which the chances of detecting the error, if it is indeed there, are the highest. But (4.11) tells us that this is not the case: all orders lead to the same expected value.

The Random strategy

The expected number of samples until error detection with the Random strategy:

$$\begin{aligned} E(M) &= \sum_{j=1}^{\infty} j * p(M=j) = \sum_{j=1}^{\infty} j \sum_{i=1}^{k} p(M=j/bug\ in\ i) * p_i = \\ &= \sum_{j=1}^{\infty} j \sum_{i=1}^{k} \left(1 - \frac{n_i}{N}\right)^{j-1} * \frac{n_i}{N} * p_i = \\ &= \sum_{i=1}^{k} \frac{N}{n_i} * p_i \, . \end{aligned} \tag{4.12}$$

We can see from (4.10) and (4.12) that the optimal strategy, in terms of the expected number of samples until error detection, depends on the values of n_i, N_i for all i. Hence, there is no dominant strategy in this case.

4.4 The Random and Round-Robin sampling strategies: non-recurring case

In this section we consider the Round-Robin and Random sampling strategies when samples are removed from the search space before continuing to the next sample.

4.4.1 The probability to expose an error

The probability to expose the error in M samples:

- The Random strategy:

$$1 - \sum_{i=1}^{k} \left(\prod_{j=0}^{S} \left(1 - \frac{n_i}{N-j} \right) \right) * p_i \ , \tag{4.13}$$

  where $S = \min(M, N - n_1, \ldots, N - n_k) - 1$. The reason we need to limit the number of rounds to S is that it is impossible to sample more than $M - 1$ or $N - n_i - 1$ without detecting the bug, for all i.

- The Round-Robin strategy:

$$1 - \sum_{i=1}^{k} \left(\prod_{j=0}^{S} \left(1 - \frac{n_i}{N_i-j} \right) \right) * p_i \ . \tag{4.14}$$

  where $S = \min(\frac{M}{k} - 1, N_1 - n_1 + 1, \ldots, N_k - n_k + 1)$. The reason we need to limit the number of rounds to S is that it is possible that one of the goals will be emptied from its assignments before reaching the value $\frac{M}{k}$.

Like in the recurring case, there is no possibility to compare the Random and the Directed sampling strategy in the non-recurring case. The optimal strategy depends on the parameters of coverage model.

4.5 The Directed strategy

4.5.1 Reduction to a search problem

Now consider the Directed sampling strategy: we can directly sample any goal i at any time t in order to find a bug. In this case we can reduce our problem to an

equivalent search problem.

The following problem corresponds to a class of Multi-Armed Bandit problems (Simple Family of Alternative Bandit Processes), presented by John Gittins in [4].

Definition 1 (Search problem (Problem 3 in [4]))**.** *A stationary object is hidden in one of n boxes. The probability that a search of box i finds the object if it is in box i is q_i. The probability that the object is in box i is p_i, and changes by Bayes' theorem as successive boxes are searched. The cost of a single search of box i is C_i. How should the boxes be sequenced for search so as to minimize the expected cost of finding the object?*

Our problem can be reduced to a search problem as follows.

Our problem: The searched 'object' is a single error that exists in one of k goals. The probability that a sampling of goal i finds the error conditional on the error being in goal i is $\frac{n_i}{N_i}$. The probability that the error is in goal i is p_i and changes by Bayes' theorem after each sample. Denote by C_i the cost of a single sample in goal i. In what order should we sample the goals in order to minimize the expected cost of finding the error? This problem is equivalent to the search problem above.

In Sect. 4.5.2 we will discuss the optimal strategy that leads the expected sampling cost until bug detection to minimum. We will examine the optimal strategy in the recurring and the non-recurring cases of our model.

4.5.2 The optimal strategy in terms of cost: recurring case

In the formulas below we use $\frac{n_i}{N_i}$ instead of q_i as in the original formulas in [4].

Assume $C_i = const$ is the cost of a single sample in goal i. Let $C(i, j)$ be the total cost of j samples in goal i ($C(i, j) = j * C_i$). The expected overall cost of finding an error by [4] is:

$$\sum_{i=1}^{n} p_i * \sum_{j=1}^{\infty} \big(1 - \frac{n_i}{N_i}\big)^{j-1} * \frac{n_i}{N_i} * C(i, j) \ . \tag{4.15}$$

Gittins shows how to associate a value with each goal, called the *index* of that goal, which is then used for choosing the next goal to search.

After j unsuccessful searches in goal i, the appropriate index of goal i is:

$$v_i(j) = \sup_{N>j} \frac{\sum_{r=j+1}^{N} p_i * \left(1 - \frac{n_i}{N_i}\right)^{r-1} * \frac{n_i}{N_i}}{(N-j) * C_i} \ . \tag{4.16}$$

Since $\left(1-\frac{n_i}{N_i}\right)^{r-1}$ is a decreasing function of r the maximum is reached at $N = j+1$:

$$v_i(j) = \frac{p_i * \left(1 - \frac{n_i}{N_i}\right)^{j} * \frac{n_i}{N_i}}{C_i} \ . \tag{4.17}$$

This means that it is sufficient to compute a value $v_i(j)$ for any goal i only for one step forward, which means that this is a *greedy strategy*:

Definition 2 (Greedy strategy)**.** *A* greedy strategy *is a strategy that makes the locally optimum choice at each stage with the hope of finding the global optimum.*

It follows from (4.17) that the greedy strategy is an optimal strategy in the recurring case with a fixed cost.

In the formulas (4.16, 4.17) Gittins uses a-priory probabilities p_i, that in fact change after j unsuccessful searches in the box i into posterior probabilities p_i' by Bayes' theorem proportionally to $p_i * \left(1 - \frac{n_i}{N_i}\right)^{j}$. From (4.17) it follows that the optimal policy is that of conforming to the index $p_i' * \frac{n_i}{N_i}/C_i$.

If we adopt the convention that the a-priory probabilities p_i change according to Bayes' theorem then the optimal policy for a search problem is a policy conforming to the index $p_i * \frac{n_i}{N_i}/C_i$ (Theorem 8.1 [4]). The optimality of this policy is proved in various papers (for example in [6]).

Following Theorem 8.1 in [4] under the optimal policy the goal i is sampled next if its index is:

$$\frac{p_i * \frac{n_i}{N_i}}{C_i} = \max_j \frac{p_j * \frac{n_j}{N_j}}{C_j} \ . \tag{4.18}$$

Conclusion: The greedy strategy is optimal if a cost of one sample is fixed ($C_i = const$).

4.5.3 The optimal strategy in terms of cost: non-recurring case

Search problem (non-recurring case): Suppose now that the probability of finding an error on the j-th search of goal i conditioned on not finding it before, and provided it is in box i is $\frac{n_i}{N_i-j+1}$, and that the cost of the j-th search of goal i is C_{ij}.

The expected total cost required to find the object by [4] is now:

$$\mathbf{E}\left\{ \sum_{i=1}^{n} p_i * \sum_{j=1}^{\infty} \prod_{r=1}^{j-1} \left(1 - \frac{n_i}{N_i - r}\right) * \frac{n_i}{N_i - j} * C(i,j) \right\}, \tag{4.19}$$

where $C(i,j)$ is the total cost of j searches in goal i. Note that unlike the recurring case, which was discussed in Sect. 4.5.2, here $C(i,j)$ is not necessarily equal to $j * C_i$ where C_i is some fixed cost, because there is a different cost for each sample in the same goal.

Therefore the optimal strategy for the search problem is that based on an index, which now takes the form:

$$v_i(x_i(j)) = \sup_{N>j} \frac{p_i \mathbf{E}\left\{ \sum_{r=j+1}^{N} \prod_{s=j+1}^{r-1} (1 - \frac{n_i}{N_i-s+1}) * \frac{n_i}{N_i-r+1} | x_i(j) \right\}}{\mathbf{E}\left\{ \sum_{r=j+1}^{N} C_{i,r} | x_i(j) \right\}}, \tag{4.20}$$

where $x_i(j)$ denotes that a goal i is searched at the j-th time, N is an integer-valued stopping variable for goal i and p_i is a posterior probability that an error is in goal i.

Conclusion: In the generalized case the optimal sampling strategy is not greedy - we choose the next goal by the evaluation of the expected benefit over time N.

However we want to prove the following claim.

Claim 2. *The greedy strategy is optimal relatively to the expected cost until error detection when the cost of a single sample in each goal is non-decreasing.*

Proof. [2] A greedy strategy is optimal in non-recurring case when:

$$\frac{p_i * \frac{n_i}{N_i-j+1}}{C_{i,j}} \geq \frac{p_i * \frac{n_i}{N_i-j+1} + p_i * \left(1 - \frac{n_i}{N_i-j+1}\right) * \frac{n_i}{N_i-j}}{C_{i,j} + C_{i,j+1}}$$

[2] The claim was proved by Laurent Fourner from IBM.

First, recall the inequality:
For any $a, b, c, d > 0$

$$\frac{a}{c} < \frac{a+b}{c+d} < \frac{b}{d} \Leftrightarrow \frac{a}{c} < \frac{b}{d} \,. \tag{4.21}$$

Using (4.21) we can express a condition for optimality of the greedy strategy in the following manner:

$$\frac{p_i * \frac{n_i}{N_i - j + 1}}{C_{i,j}} \geq \frac{p_i * \left(1 - \frac{n_i}{N_i - j + 1}\right) * \frac{n_i}{N_i - j}}{C_{i,j+1}} \,. \tag{4.22}$$

From (4.22) it follows that if:

$$C_{i,j+1} \geq C_{i,j} * \frac{N_i - j + 1 - n_i}{N_i - j} \,, \tag{4.23}$$

then the subexpression of (4.20)

$$\frac{\prod_{r=1}^{j-1} \left(1 - \frac{n_i}{N_i - r}\right) * \frac{n_i}{N_i - j}}{C_{i,r}}$$

is a decreasing function of r and, like in the recurring case, the maximum in (4.20) is reached at N=j+1.

□

Conclusion: The non-decreasing cost of directed samples in each goal i leads to the optimality of the greedy strategy in the non-recurring case in our model.

4.5.4 The optimal strategy in terms of probability to expose an error

Gittin's proof only considers the optimal expected cost. In this section we show that Gittin's indices are optimal also in terms of the probability to expose an error, assuming the cost is fixed.

Denote by $S = \{S_1, S_2, \ldots, \}$ a sampling strategy, where S_i denotes a decision made at time $i \geq 1$. Denote by $P(S, t)$ the probability to expose an error with strategy S up to time t.

Proposition 2. *Let S be a greedy sampling strategy following the indices in (4.18) and using a constant cost ($C_i = C$ for all i). Then the probability to expose an error up to any time t under S is maximal.*

This proposition is correct for both the recurring and nonrecurring case. The proof for the recurring case appears in Appendix B[3].

A proof of Proposition 2: non-recurring case

Proof. ($\Rightarrow$) Denote by S the greedy strategy as specified in the proposition, and by S' any other strategy. Strategies can be seen as a series of decisions. Assume we are considering the suffixes of the decisions made by the two strategies, starting from the first time they made a different decision. Call that time 't'. Denote by D_t an error distribution at time t, by $u(i)_t$ a number of unsuccessful samples in goal i up to time t. Let p_i^t denote the probability of the error being in goal i at time t, for any goal i. Note that the value of p_i^t depends on the order of samples prior to time t. Let P(find an error $\mid S_t = i$) denote a probability to find an error in goal i at time t with the strategy S.

We prove ($\Rightarrow$) by induction.

Base: $t = 1$. Since S is greedy and P(find an error $\mid S_1 = i$) $= \max_j p_j^t * \frac{n_j}{N_j - u(j)_{t-1}}$, P(find an error $\mid S_1 = i$) $\geq$ P(find an error $\mid S_1' \neq i$) is true for $t = 1$. Therefore $P(S, 1) \geq P(S', 1)$.

Assumption: Assume that $P(S, t) > P(S', t)$ is true for any time $n \leq t$.

Step: Now we prove that $P(S, t+1) > P(S', t+1)$ is true.

Assume $S_t = i$ (that is, S decides on goal i at time t). After unsuccessful search in goal i at time t the new error distribution D_{t+1} at time $t+1$ is computed as follows. The probability p_j of an error being in any goal j changes by Bayes' theorem:

$$\begin{aligned} p_j^{t+1} &= \text{P(error is in goal } j \text{ at time } t+1) = \\ &= \text{P(error is in goal } j \text{ at time } t) * \frac{1}{\text{bug was not found at time } t \text{ in goal } i} \\ &= \text{P(error is in goal } j \text{ at time } t) * \frac{1}{1 - p_i^t * \frac{n_i}{N_i}} \ . \end{aligned} \tag{4.24}$$

After adding constraints[4] the probability to find an error if the error is in goal i is equal now to $\frac{n_i}{N_i - u(i)_{t-1} - 1}$ and the probabilities to find an error if the error is in goal j stay unchanged for any goal $j \neq i$ ($\frac{n_j}{N_j - u(j)_{t-1}} = \frac{n_j}{N_j - u(j)_t}$).

Under a greedy strategy S we sample at time $t+1$ a goal with the highest probability to expose an error. Assume such a goal at time $t+1$ is the goal k.

The following lemma is proved in Appendix B for the recurring case:

[3]The proof for the recurring case is by Laurent Fournier from IBM.

[4]Recall that in the non-recurring case we add constraints in order to exclude the assignment that have already been sampled before.

Lemma 1. *Let i, j, k, m be indices of goals.*

$$\begin{aligned}&\Big\{\Big(P(\textit{find an error in goal } i \textit{ at time } t) > P(\textit{find error in goal } j \textit{ at time } t)\Big)\wedge\\&\Big(P(\textit{error is in goal } k \textit{ at time } t) > P(\textit{error is in goal } m \textit{ at time } t)\Big)\Big\}\Rightarrow\\&\Big(P(\textit{error is in goal } k \textit{ at time } t+1 \mid \textit{error was not found in goal } i \textit{ at time } t) >\\&P(\textit{error is in goal } m \textit{ at time } t+1 \mid \textit{error was not found in goal } j \textit{ at time } t)\Big)\ .\end{aligned} \tag{4.25}$$

This lemma holds in the nonrecurring case as well, although there is a minor change in the proof, because the probability at each step changes also due to the change in the sizes of the goals N_i.

Proof. The left-hand side of B.2 in our case is equivalent at time t to:

$$\begin{aligned}&p_i^t * \tfrac{n_i}{N_i - u(i)_{t-1}} > p_j^t * \tfrac{n_j}{N_j - u(j)_{t-1}} \wedge\\&p_k^t > p_m^t\ .\end{aligned} \tag{4.26}$$

It follows from (4.26) that

$$\frac{1}{1 - p_i^t * \frac{n_i}{N_i - u(i)_{t-1}}} > \frac{1}{1 - p_j^t * \frac{n_j}{N_j - u(j)_{t-1}}}\ . \tag{4.27}$$

The probabilities of having an error in goal $k,\ m$ at time $t+1$ change as follows:

$$p_k^{t+1} = \mathrm{P}(\text{error is in goal } k \text{ at time } t+1) =$$

$$\mathrm{P}(\text{error is in goal } k \text{ at time } t+1 \mid \text{error was not found in goal } i \text{ at time } t)\ .$$

According to (4.24), this is equal to:

$$p_k^t * \frac{1}{1 - p_i^t * \frac{n_i}{N_i - u(i)_{t-1}}}\ ,$$

and similarly,

$$p_m^{t+1} = \mathrm{P}(\text{error is in goal } m \text{ at time } t+1) =$$

$$\mathrm{P}(\text{error is in goal } m \text{ at time } t+1 \mid \text{error was not found in goal } j \text{ at time } t) =$$

$$p_m^t * \frac{1}{1 - p_j^t * \frac{n_j}{N_j - u(j)_t}}\ .$$

We now need a lemma that is proved in Appendix B (the proof of which does not rely on the model being recurring or non-recurring):

Lemma 2. *If a goal with the highest probability of finding the error is sampled at time t without success, then the probability of having an error in the other goals at time $t+1$ increases maximally compared to any other choice at time t.*

Following Lemma 2, for all $k \neq i$ p_k^{t+1} increases maximally if at time t we sampled the goal with the highest probability to find the error.

Then it follows from (4.26) and (4.27) that at time $t+1$:

$$p_k^t * \frac{1}{1 - p_i^t * \frac{n_i}{N_i - u(i)_{t-1}}} > p_m^t * \frac{1}{1 - p_j^t * \frac{n_j}{N_j - u(j)_{t-1}}}$$

or put another way:

$$p_k^{t+1} > p_m^{t+1} \ ,$$

which is equivalent to the right-side of Lemma 1.

□

The probabilities to find an error in goal k if it is in goal k or in goal m if it is in goal m stay unchanged in time $t+1$. That is,

$$\frac{n_k}{N_k - u(k)_{t-1}} = \frac{n_k}{N_k - u(k)_t} \ ,$$

$$\frac{n_m}{N_m - u(m)_{t-1}} = \frac{n_m}{N_m - u(m)_t} \ .$$

Following Lemma 1

$$p_k^{t+1} * \frac{n_k}{N_k - u(k)_t} > p_m^{t+1} * \frac{n_m}{N_m - u(m)_t} \ .$$

Therefore we obtain:

P(find an error in goal k at time $t+1$ | an error was not found in goal i at time t) >

P(find an error in goal m at time $t+1$ | an error was not found in goal j at time t) ,

and the following property holds:

$$P(S, t{+}1) > P(S', t{+}1 | S'_t = i, S'_{t+1} \neq k) > P(S', t{+}1 | S'_t \neq i, S'_{t+1} \neq k) = P(S', t{+}1) \ .$$

□

Chapter 5

Testing Model 3

5.1 Model definition

The aim in this testing model is to cover exactly N goals with minimum cost, assuming each goal contains the same number of assignments. This model assumes recurring sampling.

5.2 Test generation strategies

- *Random*: See Sect. 3.2.
- *Directed*: See Sect. 3.2.
- *Hybrid*: A strategy that combines the two previous ones. The problem is to find the optimal point to switch between two strategies.

Assume the following cost of a single sample, in each of the generation strategies:

- *Random*: cost = 1.
- *Directed*: cost = C.
- *Hybrid*: the cost is calculated as 1 as long as the testing is random, and C otherwise.

5.3 Expected sampling cost

We analyze the cost of covering N samples with the three strategies:

1. *Random.* The assignments for the next sample are chosen randomly over the space of all goals. Let i denote the number of goals already covered. Hence, there are $N - i$ goals left. Also, let S_{N-i} denote the number of samples required for covering the first of the remaining $N-i$ goals. S_{N-i} is distributed Geometrically, and its expected value is:

$$E(S_{N-i}) = \frac{1}{p_i} = \frac{N}{N-i}. \tag{5.1}$$

The expected total cost for covering N goals with the random sampling strategy is:

$$E(total) = \sum_{i=0}^{N-1} 1 * E(S_{N-i}) = \sum_{i=0}^{N-1} \frac{N}{N-i}. \tag{5.2}$$

Driving N to infinity yields

$$\lim_{N\to\infty} \sum_{i=0}^{N-1} \frac{N}{N-i} = N * \log N. \tag{5.3}$$

2. *Directed.* The number of samples for covering N goals is exactly N and the total cost of sampling is:

$$E(total) = N * C. \tag{5.4}$$

3. *Hybrid.* In this strategy first N_1 goals are covered by the random strategy, and then $N_2 = N - N_1$ goals are covered with the directed sampling strategy. The expected total cost is:

$$E(total) = E(totalN_1) + E(totalN_2) = \sum_{i=0}^{N_1} \frac{N}{N-i} + N_2 * C. \tag{5.5}$$

5.4 Optimal sampling strategy in terms of expected sampling cost

It is obvious that the *Directed* strategy is better than the *Random* one up to a given, computable cost C. Otherwise the *Random* strategy is preferred. The *Hybrid* strategy uses the advantages of both strategies. This strategy combines first N_1 cheap random samples with N_2 expensive directed samples in order to achieve the optimal result. The problem is to find the best value of N_1 (in other words, when to switch between the strategies, which we will call the *switch* point). From (5.5) the switch point is reached when:

$$C \sim \frac{N}{N - N_1}. \tag{5.6}$$

In other words, the switch point is when the average cost of samples until covering a new goal with the random sampling equals to the cost of one directed sample. The switch point can be computed from (5.6) a-priory. There is no advantage to deciding the size of N_1 dynamically, i.e., during the testing process in this case.

Chapter 6

Testing Model 4

6.1 Model definition

In the previous model we defined a testing model with a fixed cost for performing directed sampling. In Testing Model 4 we consider a more general and realistic model.

As in the previous model, here we also need to cover N goals. In this model, however, after covering one or more goals, constraints are added to rule out their associated assignments, and random sampling continues from there with a different cost. When constraints are added this way, we say that the testing process passed a *level*. The cost of sampling in level i is denoted by C_i (the cost changes in each level because of the different number of constraints). Let N_i be the number of goals covered at level i. Then $N = \sum_{i=1}^{k} N_i$ for some $k \geq 1$.

6.2 Expected sampling cost

The total number of goals to be covered after i levels is $N - \sum_{l=1}^{i} N_l$. Let j denote the number of new goals that have already been covered in level i. There are $N - \sum_{l=1}^{i-1} N_l - j$ remaining goals. Like in Model 3, the number of samples required for

covering the first of the remaining $N - \sum_{l=1}^{i-1} N_l - j$ uncovered goals is distributed geometrically. The expected number of samples until covering a single goal is thus:

$$\frac{N - \sum_{l=1}^{i-1} N_l}{N - \sum_{l=1}^{i-1} N_l - j}.$$

The expected number of samples until covering N_i j goals in level i is equal to:

$$\sum_{j=0}^{N_i - 1} \frac{N - \sum_{l=1}^{i-1} N_l}{N - \sum_{l=1}^{i-1} N_l - j}.$$

The expected total cost of covering N goals in this model is:

$$E(N) = \sum_{i=1}^{k} E(N_i) = C_1 * \sum_{j=0}^{N_1 - 1} \frac{N}{N - j} + C_2 * \sum_{j=0}^{N_2 - 1} \frac{N - N_1}{N - N_1 - j} + \ldots + C_k * N_k \ , \quad (6.1)$$

where k is not known a-priori.

Like in testing model 3, the optimal strategy is the one that minimizes the total cost of covering the N goals. The problem is to find all the points $N_1, \ldots, N_k$ from equation (6.1) for switching among these strategies in order to reach the optimal result.

6.3 Two sampling strategies

We now consider two sampling strategies, neither of which is optimal. Both strategies calculate, after covering a new goal, whether it is better to add constraints that remove all covered goals so far (that were not yet removed), or to keep sampling without adding such constraints. The difference between the two strategies is that the first one calculates the expected cost of covering *all* the remaining goals with both options, whereas the second strategy considers only the expected cost of covering the next uncovered goal.

6.3.1 Strategy I

Two things happen when we stay in level i:

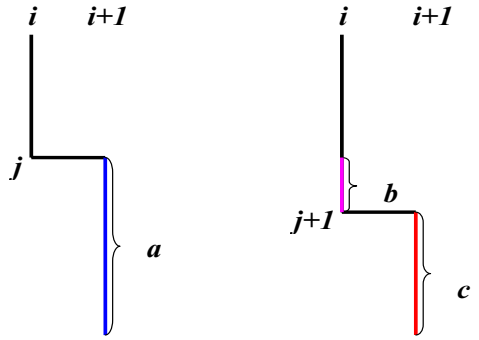

Figure 6.1: Illustrating the two options of switching to level $i+1$ after j or $j+1$ samples in level i.

- The number of goals that are covered increases, and since we do not rule them out, the expected number of samples until covering the next goal – and hence the cost – increases.
- Switching to level $i+1$ later, decreases the cost of finding the remaining uncovered goals, because there will be less goals in the search space in level $i+1$, and less goals to cover. Hence the cost decreases.

We would like to find the point in which the increase in cost due to the first effect is higher than the reduction in cost due to the second effect. Fig. **6.1** illustrates the two options in level i: whether to stay for the next goal $j+1$ in level i (right) or switch to level $i+1$ (left). We would like to switch if the cost of the segment a in the figure is smaller than the cost of the segments $b+c$.

- a is the cost of covering $N - (\sum_{l=1}^{i-1} N_l) - j$ uncovered goals in level $i+1$,
- b is the cost of covering goal $j+1$ in level i, and, finally,
- c is the cost of covering $N - (\sum_{l=1}^{i-1} N_l) - j - 1$ uncovered goals in level $i+1$.

Thus, we switch if $a < b + c$, or, equivalently, if:

$$C_{i+1} * \sum_{t=0}^{N-\sum_{l=1}^{i-1} N_l - j - 1} \frac{N - \sum_{l=1}^{i-1} N_l - j}{N - \sum_{l=1}^{i-1} N_l - j - t} < C_i * \frac{N - \sum_{l=1}^{i-1} N_l}{N - \sum_{l=1}^{i-1} N_l - j} + C_{i+1} * \sum_{t=0}^{N-\sum_{l=1}^{i-1} N_l - j - 2} \frac{N - \sum_{l=1}^{i-1} N_l - j - 1}{N - \sum_{l=1}^{i-1} N_l - j - 1 - t} . \quad (6.2)$$

6.3.2 Strategy II

In this strategy after covering a new goal, we make a decision if to cover the next goal in level i or in level $i + 1$. We move to level $i + 1$ if:

$$C_{i+1} * \frac{N - \sum_{l=1}^{i-1} N_l - j}{N - \sum_{l=1}^{i-1} N_l - j} < C_i * \frac{N - \sum_{l=1}^{i-1} N_l}{N - \sum_{l=1}^{i-1} N_l - j} . \quad (6.3)$$

The two sides of this test reflect the cost of covering the next goal in levels $i + 1$ and i, respectively. Note that the left-hand side is equal to C_{i+1}. The reason that the expected number of samples until covering the next goal is 1 is that when moving to the next level, all goals that were previously covered are excluded by constraints. Hence any sample hits a new goal.

Chapter 7

Related work

Prior work that we are aware of used the same two measures that we use here: the probability to find an error if it exists and the cost of finding it. However, they all focused on random testing, and ignored the possibility of a coverage model combined with directed sampling as in our testing models. In the rest of this section, then, we describe various measures of testing quality.

Fault simulation is a common method for estimating the quality of a testing method, the quality of a given set of tests, or the progress of a testing effort. It consists of simulating a circuit in the presence of faults which are artificially inserted into the circuit by the tester, and checking how many of them can be discovered by the test method. With the increase in the number of gates in the circuit, the number of possible faults increases, which results in high computational time for a complete fault simulation. To overcome the high computational requirements, statistical sampling methods have been proposed. In *fault sampling*, a subset of the total faults known as *fault sample* is randomly selected from a set of all possible faults and used for fault simulation. The percentage of faults found, known as the *sample coverage*, can be used to predict within a small error range the *fault coverage*, which is the percentage of real faults that are covered by the test vectors in the design[1]. This technique is

[1]Fault coverage should not be confused with coverage related to a coverage model.

frequently used to evaluate the quality of fault-coverage models.

Fault simulation and sampling is not the only way to estimate fault coverage. For example, the CAD-tool PROTEST (Probabilistic Testability Analysis), presented in [8], estimates the *fault detection probability* (the probability to cover an error) and the necessary number of tests, called the *test lengths*, for achieving the required fault coverage. The fault detection probability is computed with a simple model: it is computed according to the probability of some logical component having a specific value assuming that there exists a *single path* from this component to a primary output (this assumption is not realistic). This can be thought of as estimating the value of n_i/N_i in our second model, i.e., estimating the probability of detecting an error in a given partition (a logical component in their case). Their work focuses on random sampling only.

In [7] the STAtistical Fault Analysis (STAFAN) is proposed as an alternative to fault simulation. In this analysis, two probabilistic characteristics of circuit nodes, called *controllability* and *observability*, are estimated from signal statistics obtained from fault-free simulation. The unbiased fault detection probability on a path is estimated as a product of the controllability of the nodes of the path and the observability of this path on an output. Hence, not only overall fault coverage for a given set of input vectors and the number of faults is estimated, but also a list of potential faults that are not likely to be found with this test-set is given (these are the faults that their computed detection probability is low).

In [2] a simple model for projecting the cost of ATPG is based on two phases: in Phase I the distribution of faults detected per number of tests is exponential, whereas in phase II it is close to a linearly decreasing function. In other words, in Phase II each additional test tends to detect fewer faults relatively to Phase I. As a result, the cost/effectiveness of test generation in Phase I is better than in Phase II. The author claims that in the general case, the crossover from Phase I to Phase II is a gradual process. The empirical results show that it occurs typically after 65% – 85% of fault covering.

Their work somewhat reminds of our 4th model, in the sense that in the 4th model we also attempt to find the switching point(s): we switch when the probability to cover a new goal in level i is lower than the probability to cover the new goal in level $i+1$ we switch to level $i+1$. The difference is that while in [2] switching means to stop testing, we add constraints and continue testing. In our model the switching point is not constant, rather it is computed according to various assumptions about the cost.

An accurate fault coverage probabilistic model is presented in [3]. A cost-based analysis finds a stoping point in fault simulation, evaluates fault coverage at this point and predicts fault coverage for longer test length. The prediction process is defined as follows: first, a relatively small number of random input vectors is chosen, and several check-points in the test process are added by a tester. At each check-point the *stopping criteria* are used to decide whether to stop or continue the fault simulation. The authors define two stopping criteria for fault simulation: a threshold for cost/benefit ratio and a lower bound on the desired fault coverage. If one of them is met, the fault simulation is stopped, the fault coverage is estimated, and the prediction for the longer test length is calculated. Otherwise, the additional tests are used to continue the fault simulation. In addition to the estimation of the expected fault coverage, a lower bound on the variance of this value is estimated as well. Variance's value is used for computing a more precise expected fault coverage, which predicts more accurately the actual fault coverage.

Bibliography

[1] *http://www.haifa.ibm.com/projects/verification/coverage_advisor/.*

[2] Prabhakar Goel, *Test generation cost analysis and projections*, Proceedings of the 17^{th} Design Automation Conference, 1980, pp. 77–84.

[3] Shashank K. Mehta Hailong Cui, Sharad C. Seth, *Modeling fault coverage of randome test patterns*, Journal of Electronic Testing: Theory and Applications **19** (2003), 271–284.

[4] Gittins John, *Multi-armed bandit allocation indexes*, Prentice-Hall, Englewood Cliffs, N.J., 1989.

[5] Shmuel Ur Avi Ziv Oded Lachish, Eitan Marcus, *Hole analysis for functional coverage data*, 39^{th} Annual ACM IEEE Design Automation Conference, 2002, pp. 807–812.

[6] Weber Richard, *On the gittins index for multiarmed bandits*, The Annals of Applied Probability **2** (1992), no. 4.

[7] Vishwani D. Agrawal Sunil K. Jain, *Stafan: An alternative to fault simulation*, **2** (1985), 38–44.

[8] Hans-Joachim Wunderlich, *Protest: A tool for probabilistic testability analysis*, 22^{th} Design Automation Conference, 1985, pp. 204–211.

Appendix A

Optimal number of directed samples in the Hybrid strategy

We want to find the strategy that leads to a minimum the probability not to find an error - an optimal point of the function $f(D)$. We can approximate this discrete function to a continues one. A natural technique for solving this problem is by computing $\frac{\partial f(D)}{\partial D}$:

$$\begin{aligned}\frac{\partial f(D)}{\partial D} &= \sum_{i=1}^{k} p_i * \Big[\Big(\big(1-\frac{n_i}{N}\big)^M * \big(1-\frac{n_i}{N}\big)^{-D}\Big)' * \Big(1-\frac{n_i}{N_i}\Big)^{\frac{D}{k}} + \\ &\qquad + \Big(1-\frac{n_i}{N}\Big)^{M-D} * \Big(\big(1-\frac{n_i}{N_i}\big)^{\frac{D}{k}}\Big)'\Big] = \\ &= \sum_{i=1}^{k} p_i * \Big[\big(1-\frac{n_i}{N}\big)^M \Big(-\ln\big(1-\frac{n_i}{N}\big) * \big(1-\frac{n_i}{N}\big)^{-D}\Big) + \\ &\qquad + \Big(1-\frac{n_i}{N}\Big)^{M-D} * \ln\Big(1-\frac{n_i}{N_i}\Big) * \Big(1-\frac{n_i}{N_i}\Big)^{\frac{D}{k}}\Big] = \\ &= \sum_{i=1}^{k} p_i * \Big(1-\frac{n_i}{N}\Big)^{M-D} * \Big[\ln\Big(1-\frac{n_i}{N_i}\Big) * \Big(1-\frac{n_i}{N_i}\Big)^{\frac{D}{k}} - \\ &\qquad - \ln\Big(1-\frac{n_i}{N}\Big)\Big]. \qquad \text{(A.1)}\end{aligned}$$

We want to find a value of D for which this derivation is equal to 0 (the minimum

of the function $f(D)$. The difficulty of finding an optimal point is in the sum of components: each of them may have any value. Thus we are unable to extract an optimal value of D from (A.1). We can find the minimal value of $f(D)$ for a given partition only by a numeric solution.

Appendix B

A proof of Proposition 2: recurring case

Proposition 2. *Let S be a greedy sampling strategy following the indices in (4.18) and using a constant cost ($C_i = C$ for all i). Then the probability to expose an error up to any time t under S is maximal.*

The Proposition 2 for recurring case is proved by Laurent Fournier from IBM.

Proof. ($\Rightarrow$) Denote by S the greedy strategy as specified in the proposition, and by S' any other strategy. Strategies can be seen as a series of decisions. Assume we are considering the suffixes of the decisions made by the two strategies, starting from the first time they made a different decision. Call that time 't'. Denote by D_t an error distribution at time t. Let p_i^t denote the probability of the error being in goal i at time t, for any goal i. Note that the value of p_i^t depends on the order of samples prior to time t. Let $P(S_t = i)$ denote a probability to find an error in goal i at time t with the strategy S.

Assume in contradiction that exists another policy S' that $P(S', t') > P(S, t')$.

Assume that all decisions $S_i = S'_i$ until time $t-1$ and at time t we make first different decision $S_t \neq S'_t$.

Since S is greedy, the decision $S_t = i$ is better than any other decision $S'_t \neq i$ at time t. Since $p_i^t * \frac{n_i}{N_i} = \max_j p_j^t * \frac{n_j}{N_j}$ and $P(S, t-1) = P(S', t-1)$, $P(S, t) > P(S', t)$. We proved that $P(S, t)$ is maximal up to time t. Now we want to prove that this property holds for $t+1$.

Assume the second after $p_i^t * \frac{n_i}{N_i}$ maximal value of probability to expose an error at time t is $p_j^t * \frac{n_j}{N_j}$. In the proof we use the following lemma (this lemma is equivalent to Lemma 2 in Sec. 4.5.4):

Lemma 2. *If goal with the highest probability of finding an error in it is sampled at time t, then the probability of having an error in the other goals at time $t+1$ increases maximally compared to any other choice at time t.*

Proof. The Lemma 2 is proved by Bayes' theorem:

$$
\begin{aligned}
& p_j^{t+1} = \\
& = \text{P(error is in goal } j \text{ at time } t+1 \mid \text{error was not found at time } t \text{ in goal } i) = \\
& = \frac{\begin{array}{c}\text{P(error was not found at time } t \text{ in goal } i \mid \text{error is in goal } j \text{ at time } t) * \\ * \text{P(error is in goal } j \text{ at time } t)\end{array}}{\text{P(error was not found at time } t \text{ in goal } i)} = \\
& = \frac{\text{P(error is in goal } j \text{ at time } t)}{\text{P(error was not found at time } t \text{ in goal } i)}
\end{aligned}
\tag{B.1}
$$

It follows that p_j^{t+1} increases maximally at time $t+1$ if at time t an error was not found in goal i. □

Therefore the probability to find an error in goal j increases maximally at time

$t+1$ if at time t we sampled goal i. If at time $t+1$ a goal i still has the highest value of $p_i^{t+1} * \frac{n_i}{N_i}$, then $S_{t+1} = i$ and:

$$P(S, t+1) = P(S, t+1 | S_t = i, S_{t+1} = i) > P(S, t+1 | S_t = i, S_{t+1} \neq i) >$$

$$> P(S, t+1 | S_t \neq i, S_{t+1} \neq i) \geq P(S', t+1 | S'_t \neq i, S'_{t+1} \neq i) = P(S', t+1)$$

Otherwise, $S_{t+1} = j$:

$$P(S, t+1) = P(S, t+1 | S_t = i, S_{t+1} = j) > P(S, t+1 | S_t = i, S_{t+1} \neq j) >$$

$$> P(S, t+1 | S_t \neq i, S_{t+1} \neq j) \geq P(S', t+1 | S'_t \neq i, S'_{t+1} \neq j) = P(S', t+1)$$

As we can see, a decision S_{t+1} leads to maximal probability to expose an error up to time $t+1$. We proved that $P(S, t+1) > P(S', t+1)$. Now we prove that the left side of our claim holds for time $t+2$.

Since S is greedy, we select at time $t+1$ a goal with the probability to find an error higher than in a goal selected under S'. For comparison of $P(S, t+2)$ and $P(S', t+2)$ we use the following lemma (this lemma is equivalent to Lemma 1 in Sec. 4.5.4):

Lemma 1. *Let i, j, k, m be indices of goals.*

$$\begin{aligned}
&\Big\{\Big(\text{P(find an error in goal } i \text{ at time } t) > \text{P(find bug in goal } j \text{ at time } t)\Big) \wedge \\
&\Big(\text{P(error is in goal } k \text{ at time } t) > \text{P(error is in goal } m \text{ at time } t)\Big)\Big\} \Rightarrow \\
&\Big(\text{P(error is in goal } k \text{ at time } t+1 \mid \text{error was not found in goal } i \text{ at time } t) > \\
&\text{P(error is in goal } m \text{ at time } t+1 \mid \text{error was not found in goal } j \text{ at time } t)\Big) .
\end{aligned} \tag{B.2}$$

Proof. Assume at time t an error was not found in goal i:

$$
\begin{aligned}
&p_k^{t+1} = \\
&= \text{P(error is in goal } k \text{ at time } t+1 \mid \text{error was not found at time } t \text{ in goal } i) = \\
&\quad \text{P(error was not found at time } t \text{ in goal } i \mid \text{error is in goal } k \text{ at time } t) * \\
&= \frac{*\text{P(error is in goal } k \text{ at time } t)}{\text{P(error was not found at time } t \text{ in goal } i)} = \\
&= \frac{\text{P(error is in goal } k \text{ at time } t)}{\text{P(error was not found at time } t \text{ in goal } i)}
\end{aligned}
\tag{B.3}
$$

Similarly:

$$
\begin{aligned}
&p_m^{t+1} = \\
&= \text{P(error is in goal } m \text{ at time } t+1 \mid \text{error was not found at time } t \text{ in goal } j) = \\
&\quad \text{P(error was not found at time } t \text{ in goal } j \mid \text{error is in goal } m \text{ at time } t) * \\
&= \frac{*\text{P(error is in goal } m \text{ at time } t)}{\text{P(error was not found at time } t \text{ in goal } j)} = \\
&= \frac{\text{P(error is in goal } m \text{ at time } t)}{\text{P(error was not found at time } t \text{ in goal } j)}
\end{aligned}
\tag{B.4}
$$

Therefore the following result obtained:

$$
\begin{aligned}
&\frac{\text{P(error is in goal } k \text{ at time } t)}{\text{P(error was not found in goal } i \text{ at time } t)} > \\
&\frac{\text{P(error is in goal } m \text{ at time } t)}{\text{P(error was not found in goal } i \text{ at time } t)} > \\
&\frac{\text{P(error is in goal } m \text{ at time } t)}{\text{P(error was not found in goal } j \text{ at time } t)}
\end{aligned}
$$

or put another way:

$$
\text{P(error is in goal } k \text{ at time } t+1 \mid \text{error was not found in goal } i \text{ at time } t) >
$$

P(error is in goal m at time $t+1$ | error was not found in goal j at time t) .

□

Now assume $S_{t+1} = j$ and a goal with the maximal probability to find an error is sampled under S at time $t+2$. Thus, by Lemma 2:

$$\begin{aligned}&\text{P(find an error in goal } j \text{ at time } t+1\ ) > \text{P(an error in goal } l \text{ at time } t+1) \wedge \\ &\text{P(error is in goal } n \text{ at time } t+1) > \text{P(error is in goal } m \text{ at time } t+1) \Rightarrow \\ &\text{P(find an error in goal } n \text{ at time } t+2) > \text{P(find an error in goal } m \text{ at time } t+2)\end{aligned} \tag{B.5}$$

Since $S'_t \neq i$, $S'_{t+1} \neq j$ and the fact that S is greedy follows:

$$P(S, t+2 | S_t = i, S_{t+1} = j) > P(S', t+2 | S'_t \neq i, S'_{t+1} \neq j) .$$

Therefore the Proposition 2 holds for $t+2$:

$$P(S, t+2) > P(S', t+2)$$

The proof may be continued by induction using Lemmas 1 and 2. The Proposition 2 holds for any time t in contradiction with assumption that exists time $n \geq t$ such that under non-greedy strategy S' $P(S', n) > P(S, n)$.

□