Bare Demo of IEEEtran.cls for IEEE Conferences

Yang Shi
Department of Electrical and
Computer Engineering
Portland State University Portland, Oregon 97201
Email: yangshi.psu@gmail.com

Homer Simpson
Twentieth Century Fox
Springfield, USA
Email: homer@thesimpsons.com

Peter Griffin
Starfleet Academy
San Francisco, California 96678–2391
Telephone: (800) 555–1212
Fax: (888) 555–1212

***Abstract*—The abstract goes here.**

I. Introduction

Software testing takes fifty percent of time in the overall software development cycle time. It is vital to have an efficient and high quality test data generation mechanism. Software automated testing aims at generating high quality test cases without human efforts. It is an active research area since thirty years ago and it is paid more attention recently. Coverage-driven software testing often refers to structural testing in which testing requirements are specified in terms of cover elements. A type of cover elements related to structure of SUT(System Under Test) is usually defined by to statement, branch or path of program. Another type of cover elements related to specification of SUT is usually defined by properties or functions. They are generally described in the form of formal language.

Coverage driven structural testing follows two principles: deterministic and statistical. In deterministic testing scheme, the quality of test set is measured directly by counting the number of cover elements being triggered. Higher quality indicates more cover elements being exercised. The quality of the test set is built up by evaluating the quality of each input vector. Automated Deterministic Structural Testing automate the process of constructing input vector set. A typical work includes the branch distance measure, approximation level etc*.

In statistical testing scheme, the quality of test set refers to the probability that this set could exercising each cover element at least once. In general, the test set is sampled from a probability distribution(or input profile) over the input domain space. An input distribution exhibiting a higher probability of exercising each cover element results in a higher quality of statistical test set.

Constructing such input profile can be accomplished by analyzing the program's control flow. One needs to identify each path of the program and its corresponding subset of input vectors. Then, proper probability is assigned to it. Apparently, It is infeasible work for non-toy program.

Automated Statistical Structural Testing(ASST) automate the process of constructing input profile.[Clark & Simon] use a search-based method to derive the optimal distribution. They demonstrate that the automated search is able to derive near-optimal distribution. The main problem is search efficiency: On one hand, Curse of uncertainty caused by fitness estimation could significantly decrease the convergence speed. On the other hand, a cover element corresponds to a small size of sub-domain or many non-consecutive sub-domains result in the search hard to detect the high fitness region.

To minimize the negative effects on search efficiency, We improve the original design from following aspects:

- Input distribution model: we use sum of weighted Gaussian distributions
- Search algorithm: we adopt cell based genetic algorithm approach.

The rest of this paper is organized as follows. Section II presents related work of statistical testing. Section III gives background knowledge and notations. Section IV gives brief review of cell based genetic algorithm. Section V represents the implementation of using cell base GA to search for near-optimal input profile. Section VI shows experiments and results. Section VII is the conclusion.

II. Problem Modeling

Suppose a black box system produces labels for a seeded input vector, and the input vector has a pre-defined domain range. Further, a probability distribution is constructed to generate input vectors for the system. The aim is to construct a probability distribution over the input domain space to satisfy some constraint. One of the them is described in the following: for each label can be produced by the system, the probability distribution guarantees that each label has equal probability being triggered.

Due to the reason that there is no information available to the system, the only method to gain the information is to draw samples from a probability distribution and then run the black box system against these samples to acquire the partial knowledge. With the learned knowledge, one can create or improve the probability distribution. This "trail and error" process is iterated a mount of times until the optimal/near optimal probability distribution is found.

Let's suppose the input of the system is denoted as x. The set of labels output by the system is denoted as $C = \{c_1, c_2 ... c_i\}$. Let S_{c_i} denotes the set of input vectors labeled as c_i. Furthermore, lets suppose $\bigcap_{j=1}^{i} S_{c_j} = \emptyset$.

Let $P(x)$ be the discrete input distribution. The probability of a test input produced by $P(x)$ that triggers c_i is defined in the following:

$$p_{c_i} = \sum_{x \in S_{c_i}} P(x)$$

Namely, p_{c_i} is the probability of triggering any one of input vectors inside of S_{c_i}. In other words, p_{c_i} represents the level of confidence of an input distribution to trigger the cover element c_i by a sampled input vector.

From above equation, Calculating p_{c_i} requires to know each element in the S_{c_i} which can only be obtained by exhaustively running the system over the entire input domain space. Therefore it is necessary to estimate the probability. Suppose a test set which consists of n input vectors are labeled after run the system. Two outcomes are taken into consideration: sampled test input triggers c_i or it does not trigger c_i. Let p_* denotes the actual probability of triggering c_i. We can derive that the number of input vectors labeled c_i in the test set follows a binomial distribution defined as follows:

$$P(n_{c_i}) = \binom{n}{n_{c_i}} p_*^{n_{c_i}} \cdot (1 - p_*)^{(n - n_{c_i})}$$

In the view of frequentist statistics, the probability can be estimated by using the maximum likelihood estimation method(MLE) [**?**], and the unbiased estimator of binomial distribution is in the following:

$$\hat{p_{c_i}} = \frac{n_{c_i}}{n} \tag{5}$$

Therefore, the estimated probability of triggering label c_i by the input distribution can be estimated by the fraction of the number of input vectors triggered c_i in the test set and the size of the test set.

A. *Probability Distribution Modeling*

The input domain space is a set of input vectors. The construction of probability distribution can be viewed as the process of assign weights to each input vector. Then the probability of sampling a input vector is simply the fraction of weight assigned to the weight and the total weights. Then the optimization problem is to find a suitable set of weights for each input vector. Then the dimension of the optimization problem is equal to the size of input domain space, which is huge and impossible to optimize. Therefore, we divide the input domain space into several equivalent sub-domains, and the weights are only assigned to the sub-domain space rather than each input vector. If a sub-domain is selected, a input vector in the sub-domain is uniformly sampled.

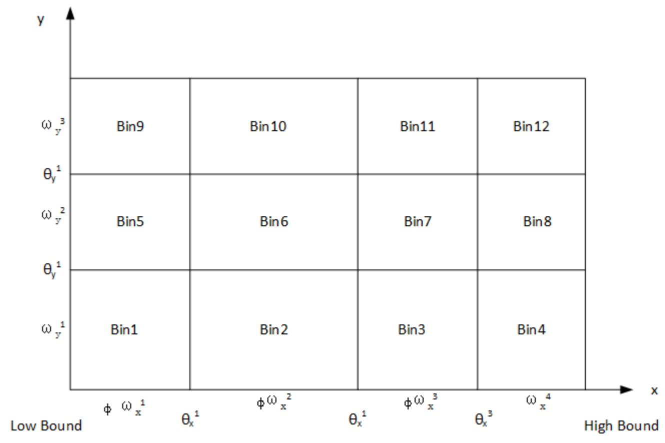

Fig. 1: Top view of input domain split into sub-domains for two variables

1) Split the input domain space: The domain range of each variable is divided into several equivalent sub-domains, and the sub-domain range of variable x_i is denoted as Δx_i. Then the elements with in the range defined in below form a sub-domain.

$$\begin{bmatrix} k_1 \\ : \\ k_d \end{bmatrix}^T \begin{bmatrix} \Delta_1 \\ : \\ \Delta_d \end{bmatrix} \leq \begin{bmatrix} x_1 \\ : \\ x_d \end{bmatrix} < \begin{bmatrix} x_1 + \Delta_1 \\ : \\ x_d + \Delta_d \end{bmatrix}$$

where k_1 represents the index of the sub-domain of variable x_i. Therefore the low bound of the sub-domain of x_i is $k_i * \Delta_i$. The high bound is the vector addition of x and Δ.

2) Probability Distribution Model: To accommodate the sub-domain based probability distribution, the model is chosen to be the *mixture of Multi-variable uniform distributions* (MUD). The generalized mixture of uniform distributions has the following form:

$$U(\mathbf{x}, \theta_1, \theta_2, ..., \theta_n) = \sum_{i=1}^{j} \phi_i u_i(x, \theta_{i*2-1}, \theta_{i*2})$$

where ϕ_i refers to the weight of i'th component, and the total weight is summed up to 1.

$u_i(\mathbf{x}, \theta_{i*2-1}, \theta_{i*2})$ represents the i-th component of MUD. It is a multi-dimensional uniform distribution with low bound denoted as θ_{i*2-1} and up bound denoted as θ_{i*2}.

Mapping from the sub-domain to MUD, each uniform component corresponds to a sub-domain space where each input vector in the sub-domain is sampled uniformly; The low bound and high bound are matched into the low end and high end of the uniform distribution. The weight ϕ_i of i-th component in MUD is calculated by the equation in below.

$$\phi_i = \frac{\prod_{d=1}^{dim} \omega_d^i}{\sum_{i=1}^{j} \prod_{d=1}^{dim} \omega_d^i} \tag{1}$$

where dim represents the dimension of the input domain space. ϕ_i represents the total weight of the i-th bin. The total weight is the product of weights in each dimension. Each ϕ_i is normalized so that all of the weights in MUD is summed up to 1.

B. The Memetic MOGA

The goal of the problem is to optimize the weights ω for each sub-domain. From the empirical study

C. Fitness Estimation

D. Local Search

E. A Case Study

III. Application: Statistical Structural Testing

We start from essential definitions

Definition 1: A Branch Cover Element c_i is an indication of a branch being taken or not taken by running SUT with a given input vector.

Definition 2: An Instrumented SUT is the SUT with inserted lines of code into each branch which output branch cover elements being triggered by running SUT against a set of input vectors. Formally, An SUT has n inputs, denoted as $\mathbf{v} = (v_1, v_2...v_n)$. Each input has a domain d_{v_i} = $[L_{d_{v_i}}, U_{d_{v_i}}]$. The input domain space D has $k = |d_{v_1}| * |d_{v_2}| * ... * |d_{v_n}|$ input vectors. An SUT which contains n branches has $2 * n$ branch cover elements denoted as $C = \{c_1, c_2...c_n, c_{2n}\}$. An Instrumented SUT takes an input vector $\mathbf{x_i}$, output a set of cover elements $c_{x_i} = \{c_{x_i} | \forall c \in C, x_i\ triggers\ c\}$

In this article, We only consider the SUTs which inputs are independent from each other.

Definition 3: An input profile denoted as $f(x)$ is a probability distribution of selecting input vectors x_i over input domain space.The probability of triggering a cover element c_i by sampling from $f(x)$ is as follow:

$$P_{c_i} = \sum_{x \in s} f(x), s \subseteq D\ and\ x \in s\ triggersc_i$$

.

Definition 4: Quality of statistical input vector set is a measurement of goodness of a test set with respect to branch cover elements. We denote size of input vector set n. The quality q_{c_i} with respect to cover element c_i is defined as follow:

$$q_{c_i} = 1 - (1 - P_{c_i})^n$$

Usually, we set $q = q_{c_1} = q_{c_2} = ... = q_{c_n}$.To improve the quality of statistical input vector set, one can either increase the size of test set or can find an input profile that has a higher p_{c_i} value. However, a high efficient input vector set requires less number of input vectors to achieve an equivalent level of quality with respect to all of cover elements. Search based statistical testing aims at searching for a well-suited input profile which satisfies the user's requirement on efficiency in terms of τ_{c_i}. Statistical test data generation requires following steps:

1) **Searching for a well-suited input distribution**: Normally, we preset the target value of p_{c_i} as τ_{c_i}. A search algorithm is conducted to find a qualified $f(x)$ s.t $\forall c_i \in C, p_{c_i} >= \tau_{c_i}$.
2) **Determining the size of test vector set**: With given p_{c_i} from last step, the size of input vectors n can be derived from following equation

$$n = \frac{log(1-q)}{log(1-p_{c_i})}, \quad where\ p_{c_i} = \min\{p_{c_k}, c_k \in C\}$$

There exists a notable problem on the evaluation of p_{c_i}. The exact p_{c_i} calculation requires to exhaustively run SUT over the entire input domain D in order to retrieve the maximum subset of D covers c_i. It is most of time unrealistic even for a small program. Instead of exact calculation, we could estimate p_{c_i} from samples. Suppose n samples are drawn from $f(x)$. after running SUT against the sample set, there are n_{c_i} input vectors exercising c_i. The estimation of p_{c_i} is as follow:

$$\hat{p_{c_i}} = \frac{n_{c_i}}{n}$$

To increase the accuracy of estimation, ensemble average method can be adopted. The detail algorithm is described in the later sections.

IV. Cell Based Genetic Algorithm

Cell based genetic algorithm is proposed by Amer [3]. Their objective is to automatically generate a proper directives for random test generators in order to test circuits that are written by SystemC. The directives are cells over the input domain with assigned weights indicating the importance of a cell in terms of number of functional cover points being covered. The set of input vectors are randomly generated but guided by the directives. In this section, we give a briefly review on general GA, cell based encoding scheme and the Inter-Cell crossover operator.

A. Cell Based Encoding Scheme

Definition 5: A cell represents a sub-input-domain d_i, $d_i \subseteq D$. A cell consists of a lower bound vector $\mathbf{L_i}$, an upper bound vector $\mathbf{H_i}$ and a weight parameter w_i.

A genotype is encoded as an array of n disjoint cells. Each gene represented as a cell contains three attributes: $(\mathbf{L_i}, \mathbf{H_i}, w_i)$.

B. Inter-Cell Crossover

Given two chromosomes $Chrome_i$ and $Chrome_j$, Inter-Cell Crossover creates new chromosomes by two set operations listed in below:

- $Chrome_i \cup Chrome_j$: One chromosome is created by the union operation. The weight of new merged cell is proportional to the weight and size of each cell involved in the union process.

- $Chrome_i \cap Chrome_j$: The other chromosome is created by the intersection operation. The weight of new intersected cell is the average of weights of the overlapped cells.

As with the standard genetic algorithm, cell based GA selects chromosomes for reproduction by roulette wheel selection. Its mutation operators helps evolution process from preventing the premature convergence. Elitism is adopted to ensure the evolution process does not discard the currently best solutions by copying a small potion of the best solutions into the next generation.

V. Searching Input Distribution by CBGA

Given a population of initial input distributions $P = \{f_1(x), f_2(x), ...f_n(x)\}$, our goal is to find an optimal input distribution that maximizes $p_{c_i}\ where\ p_{c_i} = \min\{p_{c_k}, c_k \in C\}$. As mentioned above, GA consists of genotype encoding, genetic operators, and fitness evaluation. In what follows, we discuss the implementation details of each component in solving the input distribution optimization problem.

A. Representation of Input Distribution

The input distribution $f(x)$ is represented as a sum of weighted Gaussian distributions. Formally, let n denotes the number of Gaussian distributions; $\mu_\mathbf{i}$, $\sigma_\mathbf{i}$ and a_i denotes the mean vector, standard deviation vector and weight of i'th Gaussian distribution. The equation of $f(x)$ is defined as follow:

$$f(x) = \sum_{i=1}^{n} a_i N_i(x, \mu_\mathbf{i}, \sigma_i)$$

The total number of parameters for GA to optimize is: $P = 3 * |\mu_\mathbf{i}| * n + n$.

GA can be benefit from using sum of weighted Gaussian distributions in the following reasons:

1) It is suited for branch cover elements with discontinuous sub-input domain. Each Gaussian component is represented as a gene in its genotype. As evolution keeps moving forward, a Gaussian component which takes effect on a solitary but high fitness sub-input-domain space could be kept in the natural selection phase.

2) It creates a smooth fitness landscape due to the bell shaped curve of each Gaussian component and the averaging effect of the sum operation.

Sampling a PDF composed by sum of n weighted Gaussian distributions can be divided into a two-step process[2]. The first step is to choose a Gaussian component. we use roulette wheel selection method. The probability of choosing the i'th component N_i is computed by the following equation:

$$p_i = \frac{w_i}{\sum_{k=1}^{n} w_k}$$

The second step is to sample the selected Gaussian component N_i. This is a standard process which can be accomplished by many piratical methods. For instance, the Box-Muller method[4].

B. Encoding

As we discussed above, a genotype g_i which has n genes are encoded as follow:

$$G_i = [(\mathbf{L_1}, \mathbf{H_1}, w_1),\ (\mathbf{L_2}, \mathbf{H_2}, w_2)\ ...\ (\mathbf{L_i}, \mathbf{H_i}, w_i),\ (\mathbf{L_n}, \mathbf{H_n}, w_n)]$$

where $\mathbf{L_i}, \mathbf{H_i} \in \mathbb{R}^d, w_i \in \mathbb{R}+, d = |\mathbf{H_i}|$. Each Gaussian component is mapped into a corresponding cell with same index. The following equations convert the parameters of i'th cell into parameters of i'th Gaussian component:

1) $a_i = w_i$,
2) $\mu_\mathbf{i} = \frac{\mathbf{L_i}+\mathbf{H_i}}{2}$,
3) $\sigma_\mathbf{i} = (\mathbf{H_i} - \mu_\mathbf{i})$

We require the Gaussian component to be centered at the mean of the cell and samples drawn from the Gaussian should have the probability of 68.26% hitting the region of the cell. The weight of Gaussian component is equivalent to the weight of the cell.

C. Recombination

The crossover operator selects two parent solutions randomly from the population and recombines them to form two offspring. We use *Inner-Cell Crossover* as crossover operator. Specially, Suppose two genotypes G_1 and G_2 performs Inner Cell Crossover. Gene g_{12} indicated as the gene at position one of G_2 overlaps with g_{11} and g_{12}, as shown in figure x. After inner cell crossover, $New\ G_1$ is created by union operator, $New\ G_2$ is created by intersect operator. The value of their attributes are listed in table below:

	$\mathbf{L}_1$	$\mathbf{H}_1$	w_1	$\mathbf{L}_2$	$\mathbf{H}_2$	w_2
$New\ G_1$	$\mathbf{L}_{g_{11}}$	$\mathbf{H}_{g_{21}}$	K	-	-	-
$New\ G_2$	$\mathbf{L}_{g_{12}}$	$\mathbf{H}_{g_{11}}$	$\frac{w_{g_{11}}+w_{g_{12}}}{2}$	$\mathbf{L}_{g_{21}}$	$\mathbf{H}_{g_{12}}$	$\frac{w_{g_{21}}+w_{g_{12}}}{2}$

$$K = \frac{(w_{g_{11}}*(\mathbf{H}_{g_{11}}-\mathbf{L}_{g_{11}})+w_{g_{21}}*(\mathbf{H}_{g_{21}}-\mathbf{L}_{g_{21}})+w_{g_{12}}*(\mathbf{H}_{g_{12}}-\mathbf{L}_{g_{12}}))}{\mathbf{H}_{g_{21}}-\mathbf{L}_{g_{11}}}$$

D. Mutation

The mutation operator mutates a genotype to create a new solution. We adopt the following mutation operators from CBGA, which are:

- insert or delete a cell
- Shift or adjust a cell
- Change cell's weight

All of the operations should not violate the disjoint property of each cell. When a genotype is chosen for mutation, one of the above five operators is randomly selected by performing roulette wheel selection method. The weight of each operator is pre-defined by user.

E. Fitness Evaluation

VI. Experimental Results

SUT introduction
how to initialize
what are the parameters
experiment result
Compare table

VII. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.