Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

COMPUTATIONAL INVESTIGATIONS OF QUASIRANDOM SEQUENCES IN GENERATING TEST CASES FOR SPECIFICATION-BASED TESTS

Hongmei Chi

Department of Computer and Information Sciences
Florida A&M University,
Tallahassee, FL 32307-5100, U.S.A.

Edward L. Jones

Department of Computer and Information Sciences
Florida A&M University
Tallahassee, FL 32307-5100, U.S.A.

ABSTRACT

This paper presents work on generation of specification-driven test cases based on quasirandom (low-discrepancy) sequences instead of pseudorandom numbers. This approach is novel in software testing. This enhanced uniformity of quasirandom sequences leads to faster generation of test cases covering all possibilities. We demonstrate by examples that quasirandom sequences can be a viable alternative to pseudorandom numbers in generating test cases. In this paper, we present a method that can generate test cases from a decision table specification more effectively via quasirandom numbers. Analysis of a simple problem in this paper shows that quasirandom sequences achieve better data than pseudorandom numbers, and have the potential to converge faster and so reduce the computational burden. The use of different quasirandom sequences for generating test cases is presented in this paper.

1 INTRODUCTION

Specification-based testing of software is to increase the effectiveness of software testing (Muccini et al. 2004). A formal software specification is one of the most useful documents to have when testing software, since it is a concise and precise description of functionality. Specification-based testing focuses on obtaining test data from specification (Stocks and Carrington 1993). Generating test data to cover all specification is a challenge for a complex system (Goodenough and Gerhart 1975).

We are developing an approach to deriving test data from quasirandom sequences (Chi et al. 2006) instead of pseudorandom sequences. In that paper, only one of quasirandom sequences was considered. In the current paper, we investigate the convergence rate of generating test data from different quasirandom sequences. Quasirandom sequences (Tezuka 1995) are constructed to minimize the *discrepancy*, a measure of the deviation from uniformity and therefore quasirandom sequences are more uniformly distributed than pseudorandom sequences. This enhanced uniformity of quasirandom sequences leads to faster convergence rate in generating test data as well. In the past, pseudorandom number generators, such as linear congruential generators (Knuth 1997) have been used in the implementation of random testing. Recently, it has been recognized that the convergence rate of Monte Carlo approaches based on pseudorandom numbers is slow and that an important improvement of the convergence rate can be achieved by using quasi-Monte Carlo methods (Niederreiter 1992). Quasi-Monte Carlo methods are now successfully used in many scientific computational fields, such as computer graphics (Keller 1995) and computational finance (Papageorgiou and Traub 1997). To our knowledge, approaches of generating data in software testing are based on pseudorandom number generators. To take advantage of quasi-Monte Carlo methods, we explore to produce test data by using uniformly distributed sequences. This observation is the motivation for the investigation described in this paper.

We will conduct theoretical and empirical exploration of quasirandom sequences schemes appropriate for software testing problems and evaluate the effectiveness of different quasirandom sequences used in software testing. This approach is novel in software testing. The paper will explore benefits of quasi-random numbers for generating test cases in complex systems.

2 QUASIRANDOM SEQUENCES

Pseudorandom numbers are constructed to mimic the behavior of truly random numbers, whereas highly uniform numbers, called quasirandom numbers, are constructed to be as evenly distributed as is mathematically possible. Pseudorandom numbers are scrutinized via batteries of statistical tests that check for statistical independence in a variety of ways, and are also checked for uniformity of distribution, but not with excessively stringent requirements. Thus, one can think of computational random numbers as either those that possess considerable independence, such as pseudoran-

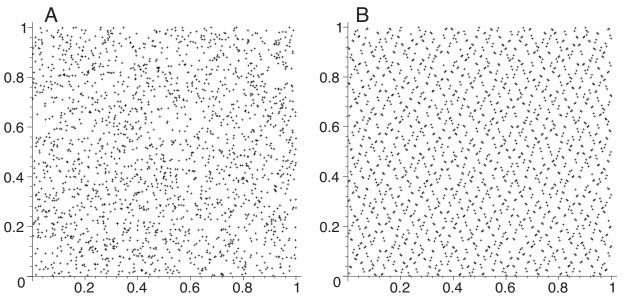


Figure 1: Comparison of Pseudorandom Numbers and Quasirandom Numbers in Two Dimensions (A: 2000 Pseudorandom Numbers, Linear Congruential Generator; B: 2000 Quasirandom Numbers, Soboĺ Sequence)

dom numbers; or those that possess considerable uniformity, such as quasirandom numbers (Niederreiter 1992).

From Fig. 1 we can see that pseudorandom numbers tend to show clustering effects while quasirandom numbers are uniformly distributed. Pseudorandom numbers are only a substitute for true random numbers; while quasirandom numbers tends to more uniformly distributed. There are many applications that do not really require randomness, but instead require numbers that uniformly cover the sample space. Quasirandom sequences are more suitable for such applications. In particular, fewer quasi-random samples are needed to achieve a similar level of accuracy as obtained by using pseudo-random sequences (Loh 2003; Spanier and Maize 1994). Since the convergence rate of Monte Carlo methods is asymptotically $\mathbf{O}(N^{-\frac{1}{2}})$, where N is the number of samples, while QMC methods can have an error bound which behaves approximately $\mathbf{O}(N^{-1})$.

The original construction of quasirandom sequences is related to the Weyl sequence and the van der Corput sequence (Kuipers and Niederreiter 1974). Weyl sequence is based on irrational numbers while the van der Corput sequence is a one-dimension quasirandom sequence based on digital inversion. This digital inversion method is the central idea behind the construction of current quasirandom sequences, such as Halton, Faure and Soboĺ (Soboĺ 1967) sequences. Niederreiter (Niederreiter 1992) extended this method to arbitrary bases and dimensions.

Before we begin our discussion of the use of quasirandom numbers to generate test cases, it behooves us to describe, in detail, the standard and widely accepted methods of quasirandom number generation. The reason for this is that many scrambling methods have been designed specifically for one class of quasirandom sequence.

Well-distributed Sequence

An infinite sequence $\{x_i\}$ is well-distributed, if we have

$$\lim_{n\longrightarrow\infty} D_n^*(z_1, z_2, \ldots, z_n) = 0, \qquad (1)$$

where $z_i = (x_i, x_{i+1}, \ldots, x_{i+s-1}) \in (0,1]^s$ for any positive integer s and D_n^* is the star-discrepancy, a measure for uniformity. Therefore, if a sequence is well-distributed, it is a low-discrepancy sequence. A well-distributed sequence has a slightly stronger property than uniformly distributed (quasirandom) sequences, namely, well-distributed sequences are a subset of quasirandom sequences.

Weyl sequences (Tezuka 1995) are well-distributed and used this paper for numerical experiments . The definition of Weyl sequence is as follows:

Definition 1 *If θ is an irrational number, then the Weyl sequence $n * \theta \pmod 1$, n=1,2,3,...., is uniformly distributed.*

Here $\pmod 1$ is operation of keeping the fraction part of any number, for example $2.345 \pmod 1 = 0.345$. The Weyl sequence is easy to implement and well-distributed. For different θ, the different dimensions of the Weyl sequence can be generated.

The Van der Corput Sequence

The radical inverse function, which is the basis for the Van der Corput sequence, is also central to many other methods of quasirandom number generation. First we give the definition of the radical inverse function: let $b \geq 2$ be an integer, and n a non-negative integer. Let $n = n_1 + n_2 b + ... + n_w b^w$ be the b-adic representation of n, and define the vector $\mathbf{n} = (n_1, n_2, ..., n_w)^T$. Then the radical inverse function, $\phi_b(\mathbf{n})$, is defined as

$$\phi_b(\mathbf{n}) = \frac{n_1}{b} + \frac{n_2}{b^2} + ... + \frac{n_w}{b^w}.$$

When b is prime, $\phi_b(\mathbf{n})$ is the nth term of the van der Corput sequence. The radical inverse function simply reverses the digit expansion of n, and places it to the right of the “decimal” point. Moreover, moving from $\phi_b(\mathbf{n})$ to $\phi_b(\mathbf{n+1})$ can be implemented with rightward-carry addition of $1/b$, and thus is very efficiently implemented. The Van der Corput sequence is a basic one-dimensional quasirandom sequence.

The Halton Sequence

The Halton sequence is based on the Van der Corput sequence. It can be thought of as the natural s-dimensional extension of the van der Corput sequence, and an s-tuple of the Halton sequence is defined as $(\phi_{b_1}(\mathbf{n}), ..., \phi_{b_s}(\mathbf{n}))$, where the bases, $b_1, b_2, ..., b_s$, are pairwise coprime.

The Soboĺ Sequence

The Soboĺ sequence can be thought of as a permutation of the binary van der Corput sequence, $\phi_2(\mathbf{n})$, in each dimension. The nth element of the jth dimension of the Soboĺ sequence, x_n^j, can be generated by

$$x_n^{(j)} = n_1\nu_1^{(j)} \oplus n_2\nu_2^{(j)} \oplus ... \oplus n_w\nu_w^{(j)}. \qquad (2)$$

Here $\nu_i^{(j)}$ is the ith direction numbers in the jth dimension and each $\nu_i^{(j)}$ is a binary fraction. There is another commonly used expression for $\nu_i^{(j)}$ obtained by using the integer $m_i^{(j)} = \nu_i^{(j)} * 2^i$. Thus, the choice of q initial direct numbers $\nu_i^{(j)}$ becomes the problem of choosing q integers such that $m_i^{(j)} < 2^i$. The initial direction numbers, $\nu_i^{(j)} = \frac{m_i^{(j)}}{2^i}$, in the recurrence, where $i \leq q$, can be chosen through the $m_i^{(j)}$'s, which can be arbitrary odd integers less than 2^i. For $i > q$, these direction numbers are generated using the following q-term recurrence relation

$$\nu_i^{(j)} = a_1\nu_{i-1}^{(j)} \oplus a_2\nu_{i-2}^{(j)} \oplus ...a_q\nu_{i-q+1}^{(j)} \oplus \nu_{i-q}^{(j)} \oplus (\nu_{i-q}^{(j)}/2^q). \quad (3)$$

In the above the bits, a_i, come from the coefficients of a degree q primitive polynomial over the finite field, $\mathbf{F}_2$. Clearly, one should use a different primitive polynomial to generate the Soboĺ direction numbers for each different dimension.

The Faure Sequence

The Faure sequence can also be thought of as a permutation of the van der Corput sequence, $\phi_b(\mathbf{n})$, in each dimension. The nth element of the Faure sequence in the jth dimension, $x_n^{(j)}$, is generated by

$$x_n^{(j)} = \phi(C^{(j)}\mathbf{n}). \quad (4)$$

In equation (4), $\mathbf{n} = (n_1, ..., n_w)^T$ is the vector of coefficients in the b-adic representation of the integer n. The generator matrix for the jth coordinate, $C^{(j)} = P^{j-1}$ for $(1 \leq j \leq s)$, is chosen among powers of the Pascal matrix P, which is defined with $b \geq s$, b prime, and $1 \leq j \leq b$ as:

$$P^{j-1} = \binom{r-1}{k-1}(j-1)^{(r-k)}) \pmod{b}, \quad k \geq 1, r \geq 1. \quad (5)$$

Here k is the row index and r is the column index of the Pascal matrix, and so the sequences $(\phi_b(P^{j_1}\mathbf{n}), \phi_b(P^{j_2}\mathbf{n}), ..., \phi_b(P^{j_s}\mathbf{n}))$, with $j_1, j_2, ..., j_s$ being distinct integers between 0 and $b-1$, are s-dimensional Faure sequences. It is important to mention that the Soboĺ sequence can also be defined in terms of a generator matrix.

Following the approach in our previous paper (Chi et al. 2006), we compare the effectiveness of generating test cases by the other quasirandom sequences with well-distributed sequences.

3 SPECIFICATION-BASED TESTS

Although a formal software specification is one of the most useful document to have when testing software, most of software specifications are stated informally in practice and that leaves a lot of ambiguities. Additional specification notations are needed to clarify these statements. A decision table is a rule-based specification in which responses are specified in terms of combinations of conditions met by input data. The decision table is a specification technique that can be used as the basis for test case design (Glora et al. 1995; Jones 2006). In this section we show by an example how a decision table can provide the basis for defining specification-based tests. We also show that how quasirandom sequences produce the test data based on the decision table. One measure for test case effectiveness is a ratio (full functional coverage) of the number of rules triggered by the set of test data to the number of rules in the decision table. This full functional coverage measures the thoroughness of testing based on the specification.

Table 1: Payroll Specification

Calculate employee pay, including overtime paid at 1.5 times the hourly rate of hourly employees for time in excess of 40 hours. Salaried employees are not paid overtime, nor do they lose pay when they work less than the normal work week of 40 hours. Hourly employees earn less than 10 per hour.

```
 CONDITIONS                                      | DECISION RULES
------------------------------------------------ --------------
 hours>40                                        |  N Y N Y
 rate<10                                         |  Y Y N N
------------------------------------------------ --------------
 ACTIONS                                         | ACTION RULES
------------------------------------------------ --------------
 pay = hours * rate; pay = pay;                  |  X - - -
 pay = 1.5 * rate * (hours - 40) + 40 * rate;    |  - X - -
 pay = 40 * rate;                                |  - - X X
 overtime  = 0;                                  |  X - X X
 overtime = 1.5 * rate * (hours - 40);           |  - X - -
------------------------------------------------ --------------
```

Figure 2: Payroll Decision Table (DT1) Based on the Narrative Specification in Table 1

Consider the narrative specification in Table 1 (Jones 2005), which specifies software to compute the weekly pay of employees. The first step in transforming this narrative specification is to identify the stimuli and responses. From the specification, we can deduce that the necessary stimuli (input data) are the hours worked and the hourly salary rate. According to the specification, the software must determine whether an employee is hourly (rate $\leq$10) or salaried, and whether the employee has exceeded 40 hours of work (hours $\geq$ 40). Figure 2 is a summary of all rules and actions for Pay Specification in Table 1.

When testing complicated software with a static specification, it is difficult to determine manually whether each rule has been covered and if there are anomalies in the decision table specification (Chang et al. 2000). Jones (2005) has developed a tool that uses test data to identify

Table 2: Payroll Specification (Extended from Table 1)

Calculate employee pay, including overtime paid at x times the hourly rate. Salaried employees are paid overtime only if they work more than 50 hours, but they do not lose pay when they work less than the normal work week of 40 hours. Hourly employees earn less than \$30 per hour. Employees who work more than 50 hours receive 1.5 times the hourly rate for each overtime hour. Employees who work more than 60 hours are paid 1.5 times the hourly rate for hours up to 60, and 1.6 times the hourly rate for each hour after 60. Employees who work more than 70 hours are paid 1.5 times the hourly rate for hours up to 60, 1.6 times the hourly rate for hours between 60 and 70, and 1.7 times the hourly rate for each hour after 70. Those who work more than 80 hours receive 1.5 times the hourly rate for hours up to 60, 1.6 times the hourly rate for hours between 60 and 70, 1.7 times the hourly rate for hours between 70 and 80, plus a bonus of \$100.

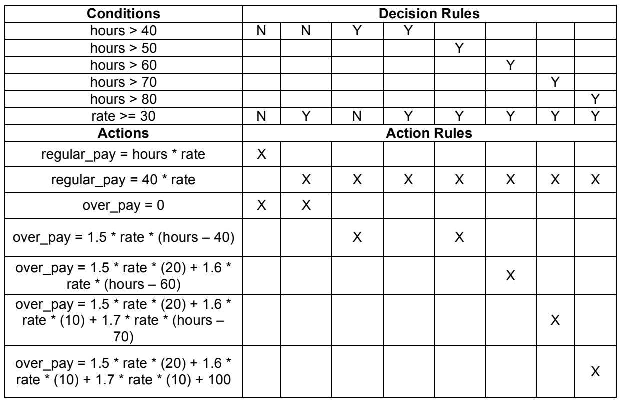
specification anomalies, while using the specification to determine adequacy of the test data. We use quasirandom sequences to provide the test data, and functional coverage as the criterion for measuring the test data. The procedure is simple: according Figure 2, we generate two-dimension test data sets (hour, rate), and check functional coverage to see how many decision table rules are satisfied (covered) by one or more test data pairs. The measure of interest for comparing pseudo-random and quasirandom generation of data sets is the number of test data needed to reach functional coverage of 1. The numerical results are shown in Section 4.

4 NUMERICAL EXPERIMENTS

The numerical experiments are base on Decision Tables in Fig. 2 and Fig. 3 . In order to compare the effectiveness of quasirandom numbers, we use different quasirandom sequences and well-distributed sequences to produce the test data.

The quasirandom number generator is Halton Soboĺ and Faure sequences and we used the same implementation in Fox (1986) and Morokoff and Caflish (1994). The Weyl sequence we used in this paper is same as in Chi et al. (2006). The pseudorandom number generator we used in this paper is one of linear congruential generators in Numerical Recipe in C (Press et al. 1992). This generator is defined as following:

Definition 2 *The linear congruential generator determined by* $x_n = ax_{n-1} \pmod{m}$ *with* $a = 16807$ *and* $m = 2^{31} - 1$ *has a period of* $2^{31} - 2$.



Conditions	Decision Rules							
hours > 40	N	N	Y	Y				
hours > 50					Y			
hours > 60						Y		
hours > 70							Y	
hours > 80								Y
rate >= 30	N	Y	N	Y	Y	Y	Y	Y
Actions	**Action Rules**							
regular_pay = hours * rate	X							
regular_pay = 40 * rate		X	X	X	X	X	X	X
over_pay = 0	X	X						
over_pay = 1.5 * rate * (hours – 40)			X		X			
over_pay = 1.5 * rate * (20) + 1.6 * rate * (hours – 60)						X		
over_pay = 1.5 * rate * (20) + 1.6 * rate * (10) + 1.7 * rate * (hours – 70)							X	
over_pay = 1.5 * rate * (20) + 1.6 * rate * (10) + 1.7 * rate * (10) + 100								X

Figure 3: Payroll Decision Table 2 (DT2) Based on the Narrative Specification in Table2

All results are shown in Table 3 is the floor of the average of five different runs. For pseduorandom number generator, we use different seeds for each run. For each quasirandom sequence, different number(100, 200, 500, 1000, 1500) of initial points are skipped and this method are proposed at Morokoff and Caflish (1994) and Owen (2004). The results in our previous paper (Chi et al. 2006) show that well-distributed sequences significantly converges faster, i.e., covers all rules with fewer test data. We compared different quasirandom sequences with well-distributed sequences for generating test data and results are listed in Table 3. It is not hard to see that the convergence rate of quasirandom sequences is faster than pseudorandom sequences and slower than well-distributed sequences (Weyl sequences). When the number of rules in decision table is 4 pairs, we could not see any advantage for quasirandom sequences. However, when the number of rules increases to 8 pairs, the advantage of generating test data by quasirandom sequences is fast to cover all cases. This experiment is preliminary and we need to apply this method to more complicated cases requiring large, complex decision table specifications.

5 CONCLUSIONS

A scheme for generating test data via various quasirandom sequences is proposed. The advantage of this scheme is that we can provide test data based on a specification automatically. This scheme is an alternative to generate test data manually or from pseudorandom numbers. Our numerical results, though preliminary , are promising. Should our observations about faster convergence (full coverage with fewer test data) hold, uniformly distributed test data generation may offer economical advantages over quasirandom and pseudo-random testing. A broader question is whether uniformly distributed sequences testing is supe-

Table 3: Test Results for Decision Tables (# Test Data for Full Functional Coverage)

	Decision Table DT1		Decision Table DT 2	
	# rules	# test data pairs	# rules	# test data pairs
pseudorandom	4	6	8	29
Halton	4	5	8	19
Faure	4	5	8	18
Soboĺ	4	5	8	17
Weyl	4	5	8	11

rior to pseudo-random testing, in terms of efficiency and effectiveness. Addressing this question may require a replication of past studies such as in Abdurazik et al. (2000) and Podgurski et al. (1999) and more other complicated cases requiring large, complex decision table specifications.

In the future, we will extend the study given in this paper to support the test-driven specifications when applied to scrambled quasirandom sequences (Hong and Hickernell 2003) and optimal quasirandom sequences (Chi et al. 2005). Unlike pseudorandom number generators, there are only a few common choices for quasirandom number generation. However, by scrambling a quasirandom sequence, one can produce a family of related quasirandom sequences. Finding one or a group of optimal quasirandom sequences within this family is an interesting problem, as such optimal quasirandom sequences can be quite useful for enhancing the performance of ordinary quasi-Monte Carlo. Ongoing work includes the development of a library of scrambled and optimal quasirandom generation routines to support specification-based test generation.

REFERENCES

Abdurazik, A. and P. Ammann and W. Ding and J. Offutt. 2000. Evaluation of Three Specification- Based Testing Criteria.*Sixth IEEE International Conference on Complex Computer Systems (ICECCS'00)* 179-187.

Chang, K. H. and S. Liao and R. Chapman. 2000. Test Scenario Geneneration Based On Formal Specification And Usage. *International Journal of Software Engineering and Knowledge Engineering*. 10(2):1-17.

Chi, H. and M. Mascagni and T. Warnock. 2005. On the Optimal Halton Sequences. *Mathematics and Computers in Simulation*. 70(1): 9-21.

Chi, H and E. L. Jones and D. Evans. 2006. Generating Test Data for Specification-based Tests via Quasirandom Sequences. *Lecture Notes in Computer Science*.3994: 773-780.

Fox, B. 1986. Implementation and relative efficiency of quasirandom sequence generators", *ACM Trans. on Mathematical Software*. 12:362-376.

Glora, N. and H. Pu and W. O. Rom. 1995. Evaluation of Process Tools in Systems Analysis. *Information and Technology*. 37: 1191-1126.

Goodenough, J. B. and S. L. Gerhart. 1975. Toward a theory of test data selection. *Proceedings of the international conference on Reliable software*. 493-510.

Hong, H. S. and F. J. Hickernell. 2003. Algorithm 823: Implementing Scrambled Digital Sequences. *ACM Transactions on Mathematical Software*. 29 (2): 95-109.

Jones, E. L. 2005. Automated Support for Test-driven Specification. *Proceedings of the 9th IASTED International Conference on Software Engineering and Applications* . 218-223.

Jones, E.L. 2006. Test-Driven Specification: Paradigm and Automation. *44th ACM Southeast Conference, March 10-12, Melbourne, Florida*.

Keller, A. 1995. A Quasi-Monte Carlo Algorithm for the Global Illumination Problem in the Radiosity Setting. *Lecture Notes in Statistics (Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing)*. 106: 239-251.

Knuth, D. E. 1997. *The Art of Computer Programming, vol. 2: Seminumerical Algorithms*. New York, Cambridge University Press.

Kuipers, L. and H. Niederreiter. 1974. *Uniform Distribution of Sequences*. New York, John Wiley and Sons.

Loh, W. L. 2003. On the asymptotic distribution of scrambled net quadrature. *Annals of Statistics*. 31: 1282-1324.

Morokoff, W.J. and R.E. Caflish. 1994. Quasirandom sequences and their discrepancy. *SIAM Journal on Scientific Computing*. 15: 1251-1279.

Muccini, H. and A. Bertolino and P. Inverardi. 2004. Using Software Architecture for Code Testing. *IEEE Trans. on Software Engineering*. 30(3): 160-17.

Niederreiter, H. 1992. *Random Number Generations and Quasi-Monte Carlo Methods*. Philadelphia, SIAM.

Owen, A. 2004. Halton Sequences Avoid the Origin. *Research Report, Stanford University*. Available online via `<http://www-stat.stanford.edu/˜owen/reports/halton.pdf>` [accessed July 12, 2006].

Papageorgiou, A. and J. Traub. 1997. Beating Monte Carlo. *RISK*. 9:63-65.

Podgurski, A. and W. Masri and Y. McCleese and F. G. Wolff and C. Yang. 1999. Estimation of software reliability by stratified sampling, *ACM Trans. Softw. Eng. Methodol.* 8(3): 263-283.

Press, W. H. and S. A. Teukolsky and W. T. Vetterling and B. P. Flannery. 1992 *Numerical Recipes in C*. Reading, Massachusetts. Addison-Wesley.

Soboĺ. I. M. 1967. Uniformly distributed sequences with additional uniformity properties. *USSR Comput. Math. and Math. Phy.* 16: 236-242.

Spanier, J. and E. Maize. 1994. Quasirandom methods for estimating integrals using relatively small sampling. *SIAM Review*. 36: 18-44.

Stocks, P. and D. Carrington. 1993. Test template framework: A specification-based testing case study. *Proceedings of Int. Symp. Software Testing and Analysis, Cambridge, MA* . 11-18.

Tezuka, S. 1995. *Uniform Random Numbers, Theory and Practice*. IBM Japan, Kluwer Academic Publishers.

AUTHOR BIOGRAPHIES

HONGMEI CHI is an assistant professor of Computer & Information Sciences at Florida A &M University. She earned a Ph.D in Computer Science from Florida State University in 2004. Her current research interests include Monte Carlo simulation, software testing, computational biology and network security. Her e-mail address is <hchi@cis.famu.edu>, and her web page is <www.cis.famu.edu/˜hchi>.

EDWARD L. JONES is associate professor and chairman of the Department of Computer & Information Sciences at Florida A&M University. He earned the Ph.D. in Computer Science from the University of North Carolina at Chapel Hill. His current research interests include software engineering, software testing, and technology transfer. He is a member of the ACM and the IEEE Computer Society . His e-mail address is <ejones@cis.famu.edu>, and his web page is <www.cis.famu.edu/˜ejones>.