International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:3, No:11, 2009

Construct Pairwise Test Suites Based on the Bak-Sneppen Model of Biological Evolution

Jianjun Yuan, Changjun Jiang

***Abstract*—**Pairwise testing, which requires that every combination of valid values of each pair of system factors be covered by at lease one test case, plays an important role in software testing since many faults are caused by unexpected 2-way interactions among system factors. Although meta-heuristic strategies like simulated annealing can generally discover smaller pairwise test suite, they may cost more time to perform search, compared with greedy algorithms. We propose a new method, improved Extremal Optimization (EO) based on the Bak-Sneppen (BS) model of biological evolution, for constructing pairwise test suites and define fitness function according to the requirement of improved EO. Experimental results show that improved EO gives similar size of resulting pairwise test suite and yields an 85% reduction in solution time over SA.

***Keywords*—**Covering Arrays, Extremal Optimization, Simulated Annealing, Software Testing.

I. Introduction

PAIRWISE testing is a practical software testing approach, which requires that, for each pair of factors (parameters) of a system, every combination of valid values of these two factors be covered by at lease one test case [1], [2]. For instance, a system with three factors as shown below: factor A has values 0 and 1, factor B has values 2 and 3, and factor C has values 4 and 5. Four tests (rows) in Table 1 cover all pairwise interactions.

Pairwise testing provides a systematic approach to identify and isolate faults since many faults are caused by unexpected 2-way interactions among system factors [3]. Dalal et al. present empirical results that the testing of all pairwise interactions in a software system detects a large percentage of the existing faults [4].

If a set of tests covers all pairwise interactions, this set is called a pairwise test suite. Table I is an example of a pairwise test suite. An important problem in pairwise testing is to find as small pairwise test suite as possible to reduce test cost. Greedy algorithms [1], [2], [5]–[7] and meta-heuristic approaches [8]–[11] dominate the techniques for constructing pairwise test suites because mathematical methods can only be applied in some special cases. Greedy algorithms begin with an empty set T and add one test at a time according to some greedy policies. Final T is a solution. Meta-heuristic approaches start with individual or population test suite(s) and transform this or these test suite(s) according to various search strategies. Optimal answer may be found at last. Although meta-heuristic strategies generally discover better answers, they cost more time to perform search [8]. Extremal Optimization (EO) is a promising approach to reduce search cost.

TABLE I
THE PAIRWISE COVERAGE IS ACHIEVED WITHIN THE FOUR TESTS

Test no.	A	B	C
1	0	2	4
2	0	3	5
3	1	2	5
4	1	3	4

EO is a recently developed local search heuristic method [12] based on the Bak-Sneppen (BS) model of biological evolution [13]. The BS model demonstrates self-organized criticality, a tendency for systems in statistical physics to organize them-selves into non-equilibrium states. Unlike most optimization techniques, EO focuses on removing poor components of solutions instead of favoring the good ones. Moreover, instead of evolving populations, EO performs updates on a single solution, which is similar to Simulated Annealing (SA) [14]. Its performance proves competitive with, and often superior to, more elaborate stochastic optimization approaches. EO has been applied to graph partitioning, the traveling salesman problem, and so on [12]. However, it is not been applied to construct pairwise test suites to the best of our knowledge.

The major contributions to this paper are summarized as follows:

- This paper is the first literature that EO and its improvement have been applied to construct pairwise test suites to the best of our knowledge;
- Based on the problem of constructing pairwise test suites, this paper presents local fitness function and global fitness

This research was partially supported by the National Natural Science Foundation of China (60534060, 90718012, and 90818023), the National High-Tech Research and Development Plan (863) of China (2007AA01Z136, 2007AA01Z149, and 2009AA01Z401), and Shanghai Science and Technology Research Plan (07JC14016).

Jianjun Yuan is with Department of Computer Science and Technology, The Key Laboratory of "Embedded System and Service Computing" Ministry of Education of China, Tongji University, Shanghai, 201804, P.R.China (phone: +86-15821050499; e-mail: yjj802401@gmail.com).

Changjun Jiang is with Department of Computer Science and Technology, The Key Laboratory of "Embedded System and Service Computing" Ministry of Education of China, Tongji University, Shanghai, 201804, P.R.China (e-mail: cjjiang@online.sh.cn).

function, which guarantees the application of improved EO efficiently;
- Various experiments have been done on constructing pairwise test suites by adopting SA and improved EO. The results show that improved EO is a promising approach for constructing pairwise test suites.

The remainder of this paper is organized as follows. In Section 2, we introduce the mathematical model of pairwise test suites, main techniques to construct the test suites, base EO, and its improvement. In Section 3, we describe improved EO for constructing pairwise test suites. Section 4 provides the results of applying our algorithms to a collection of models. Section 5 concludes this paper and discusses future work.

II. Background

Before discussing the various methods for constructing pairwise test suites we establish some mathematical models for interaction test suites. Finally base EO and its improvement are introduced.

A. Mathematical Model

A pairwise test suite is a t-way interaction test suite where $t = 2$. A t-way interaction test suite is a mathematical structure, called a covering array.

Definition 1 A *covering array*, $CA(N; t, k, |v|)$, is an $N \times k$ array from a set, v, of values (symbols) such that every $N \times t$ subarray contains all tuples of size t (t-tuples) from the $|v|$ values at least once [8].

The strength of a covering array is t, which defines, for example, 2-way (pairwise) or 3-way interaction test suite. The k columns of this array are called factors, where each factor has $|v|$ values. In general, most software systems do not have the same number of values for each factor. A more general structure can be defined that allows variability of $|v|$.

Definition 2 A *mixed level covering array*, $MCA(N; t, k, (|v_1|,|v_2|,\ldots, |v_k|))$, is an $N \times k$ array on $|v|$ values, where $|v| = \sum_{i=1}^{k} |v_i|$, with the following properties: (1) Each column i ($1 \le i \le k$) contains only elements from a set S_i of size $|v_i|$. (2) The rows of each $N \times t$ subarray cover all t-tuples of values from the t columns at least once [8].

A shorthand notation is used to describe mixed level covering arrays by combining entries with equally sized value ranges. For instance, a software system has 9 factors, 4 of which each take on 3 values, while the other 5 are binary. The pairwise test suite of this model can be written as $MCA(N; 2, 3^4 2^5)$. The symbol of k can be dropped since it can be obtained by adding the superscripts.

B. Constructing Pairwise Test Suites

The main techniques for constructing pairwise test suites can be classified three categories: mathematical methods, greedy algorithms, and meta-heuristic strategies. Mathematical methods for generating pairwise test suites usually require that each factor has the same number of values, which restrict the universality of this kind of methods. The use of orthogonal arrays belongs to this category [15], [16]. Greedy algorithms start with an empty set T and add one test at a time according to some policies like covering the most uncovered pairs [2]. Final T is a solution. The representatives of greedy algorithms include the Automatic Efficient Test Case Generator (AETG) [2], the In Parameter Order (IPO) algorithm [1], [6], [7], the Test Case Generator (TCG) [17] and the Deterministic Density Algorithm (DDA) [5]. Meta-heuristic approaches begin with individual or population test suite(s) and transform this or these test suite(s) according to various search strategies. The final answer will be found when stopping conditions are met. This kind of approaches mainly includes hill climbing [8], great deluge algorithm [8], simulated annealing [8]–[10], tabu search [18] and genetic algorithms [9], [19]. In general, greedy algorithms run faster, while meta-heuristic strategies can discover better answers at higher costs [8].

C. Base EO and its Improvement

Suppose that a solution candidate S consists of some variables. S_{best} is the best solution currently. Base EO is performed on S as follows [14]:

Step 1: Generate an initial S at random, and set S_{best} equal to S;
Step 2: Compute local fitness related to each variable of S;
Step 3: Rank the variables by their fitness and select the worst to be updated;
Step 4: Select S' in the neighborhood of S such that the worst variable must be modified;
Step 5: Set S equal to S';
Step 6: If global fitness of S' is better than the previous best, S_{best}, replace S_{best} with S';
Step 7: Repeat from Step 2 to Step 6 when stopping conditions are met.

However, Step 3 may cause the search to become stuck in local optima in some cases. Therefore Boettcher and Percus proposed an improvement of base EO, called τ-EO, to deal with this [20]. In τ-EO the worst variable ranks first and the best variable ranks n, where n is the number of variables in the solution S. We select a variable ranking k to be updated according to a probability distribution $P(k) \propto k^{-\tau}$, where $1 \le k \le n$, and τ is a constant that determines how stochastic or deterministic the selection should be [19]. For higher values of τ, τ-EO is more likely to get stuck in local optima because it does without a selection function at all. With lower values of τ, τ-EO sometimes changes better values of a solution in order to explore a larger part of the search space. If τ is set to zero, τ-EO performs random search. Therefore τ-EO can jump out of near-optimal solutions when τ is set appropriately [14].

III. Improved EO for Constructing Pairwise Test Suites

The main difficulties of applying EO to construct pairwise test suites are defining fitness function wisely. This section begins with the definitions of local and global fitness function.

Improved EO for constructing pairwise test suites has been proposed subsequently.

A. Fitness Function

EO can be treated as a kind of meta-heuristic approach. The objective of transform a test suite is to obtain a pairwise test suite that cover all pairwise interactions between system factors when meta-heuristic approaches have been applied to construct pairwise test suites. EO achieves this goal through changing every variable in a solution. In a test suite a value can be regarded as a variable in EO. Local fitness is the contribution of a value to the objective of constructing pairwise test suites. For a value, v_i, of a factor f_i in relation to an individual factor, f_j, $L_{i,j} = (r_{i,j} / (| v_i | | v_j |))$ indicates the fraction of uncovered pairs involving v_i and a value of factor f_j, where $r_{i,j}$ is the number of uncovered pairs involving v_i and a value of factor f_j, $| v_i |$, $| v_j |$ are respective the number of values of f_i, j_j, and $i \neq j$. Local fitness, $L(v_i)$, for a value, v_i, of a factor f_i is computed as

$$L(v_i) = \sum_{\substack{j=1 \\ i \neq j}}^{k} L_{i,j}, \quad (1)$$

where k is number of the factors.Global fitness, G, for a test suite is computed as

$$G = \sum_{i=1}^{n} L(v_i), \quad (2)$$

where n is number of the values in this test suite.

A test suite is a pairwise test suite when $G = 0$. According to the definition of local fitness function, it can denote the contribution of a value to the goal of constructing a pairwise test suite, which makes it possible to generate pairwise test suites by EO. Defining global fitness as the sum of local fitness also reduces the cost of computation, which improves the performance of EO further. Therefore it is appropriate to determine fitness function like this.

B. Improved Extremal Optimization

Improved EO for constructing pairwise test suites consists of two layers: outer search and inner search. Because the minimum size of a pairwise test suite cannot be known ahead of time, the outer search repeatedly calls inner search, each time with a different N, where N is the size of a test suite. The outer search chooses values for N and either accepts or rejects them according to the results of an inner search.

The pseudo-code of the outer search is shown in Fig.1 [11]. It takes an upper and lower bound on the size of the pairwise test suite and performs a binary search within this range. In the first place, at each size the inner search, improvedEO, attempts to build an array S' -- a test suite (line 4), and the return value is the last array found and its global fitness must be checked to determine whether it is a solution (line 5). In the second place, the outer search is responsible for returning the smallest pairwise test suite constructed, so it must keep a copy of the best solution (line 6).

```
binarySearch (lower, upper, t, k, v)
1  S = Φ
2  N = floor (( lower + upper) / 2 )
3  while upper ≥ lower do
4      S′ = improvedEO (N, t, k, v)
5      if S′ [global fitness] = 0 then
6          S = S′
7          upper = N − 1
8      else
9          lower = N + 1
10     end if
11     N = floor (( lower + upper) / 2 )
12 end while
13 return S
```

Fig. 1 Pseudo-code for outer search

The pseudo-code for the inner search, improvedEO, is shown in Fig.2. The inner search starts with an initial array S (line 1) and assigns S to S_{best} (line 2). Each value in S is then computed and ranks by its local fitness (line 4-5), and the mth value is selected according to a probability distribution $P(m) \propto m^{-\tau}$ (line 6), where the value of τ is determined in next section. Subsequently an array S' in the neighborhood of S is discovered such that the selected value must be updated (line 7) and S is assigned to S' (line 8). Set S_{best} equal to S' when global fitness of S' is smaller than that of S_{best} (line 9). The iterations continue until a stabilization criterion is met (line 3); that is, the algorithm has found a solution or seems not to be making further progress. Finally the array S_{best} is returned (line 11). S [global fitness] stores global fitness of S, which avoid repeating the calculation of its global fitness in outer search.

IV. Experiment

Having presented two research questions, we then describe experiments aimed to answer each research question. Results, analysis, and threats to validity follow.

A. Research Questions

It is shown that some meta-heuristic search like Simulated Annealing (SA) can generate smaller pairwise test suite [8], [10]. Furthermore, the structure of improved EO algorithm for constructing pairwise test suites is similar to that of SA for the same work. The first objective of experiments, hence, is to tell that whether the size of the pairwise test suite found by improved EO is competent with that discovered by SA. The second objective is to determine that whether the performance

of improved EO for constructing pairwise test suites surpasses that of SA for this thing, which is more important in this paper.

```
improvedEO (N, t, k, v)
1  S = initialState (N, t, k, v)
2  S_best = S
3  until stabilized ( S [global fitness]) do
4       compute the local fitness for each value of S
5     rank values as v_1 , v_2 , ..., v_m , … , v_n , such that L(v_1)≥
          L(v_2) ≥ …≥ L(v_m) ≥ …≥ L(v_n), where n is
          number of the values in S
6      select a value ranking k to be updated according to a
          probability distribution P(m)∝m^-τ,
          supposing that the value selected is v_m
7      find S′ in the neighborhood of S such that v_m must be
          modified
8       S = S′
9      if  S′ [global fitness] < S_best [global fitness]  then S_best
          = S′
10 end until
11 return S_best
```

Fig. 2 Pseudo-code for inner search.

These lead to the following research questions.

Q1: Whether is the size of the pairwise test suite found by improved EO competent with that discovered by SA?

Q2: Whether does improved EO take less time to find the smallest pairwise test suite than SA?

B. Results and Analysis

We have implemented improved EO for generating pairwise test suites. Moreover, the program of SA for dealing with the same work has been implemented. A distinct integer starting with zero is assigned to each value of factors, which is seen in the example in Table 1. Both programs use the same data structure to store an integer (a rank) to represent a *t*-set, where *t* is 2, which is a standard combinatorial technique and provide a general way to represent *t*-sets with any strength *t* [21]. Supposing that the factor, f_i, has the largest number, $| v_i |$, of values, the initial lower bound is the product of *t* and $| v_i |$. The initial upper bound is five times the $| v_i |^t$. The inner search is terminated after 10000 iterations.

In simulated annealing a worse solution S' is accepted with probability $e^{(g(S)-g(S'))/T}$, where T is the controlling temperature of the simulation, S is the old solution, and g(S) represents global fitness of a solution S. The temperature is lowed by a decrement value when the algorithm performs. With the temperature decreasing, the probability of accepting a worse solution drops. Allowing a bad move, which accepts a worse solution, helps to keep the solution from being stuck in local optima and to search larger space. The algorithm stops when a feasible solution is found, whose global fitness is zero, or the current solution can not be improved further. We use an initial temperature of 0.5 that decreases by 0.001% every iteration.

In improved EO we use $\tau = 1 + 1/\ln(n)$ and $P(m) = ((\tau - 1)/(1 - n^{1-\tau}))m^{-\tau}$, where n is number of the values in a test suite [22], [23]. When more than one value has the same local fitness, they rank lexicographically for value tie-breaking. Both programs are written in C+ + and run on Windows XP using an INTEL Pentium Dual-Core 1.73 GHZ processor with 1GB of memory. The results, shown in Table 2, are average of 10 runs. Our experiments perform on data come from [7], [17].

The results listed in Table II can address two research questions proposed in this paper. For one thing, the smallest size of pairwise test suite found by both SA and improved EO is similar, which answers Q1, that is, the capability of improved EO for discovering the smallest size of pairwise test suite is competent with SA.

TABLE II
COMPARISONS OF EXPERIMENTAL RESULTS BETWEEN SA AND IMPROVED EO

	Minimum tests in test suite		**Run time in seconds**	
	SA	**Improved EO**	**SA**	**Improved EO**
***CA*(*N*; 2, 4, 3)**	9	9	23	11
***CA*(*N*; 2, 100, 4)**	46	45	811	102
***CA*(*N*; 2, 13, 3)**	19	20	147	29
***MCA*(*N*; 2, $4^{15}3^{17}2^{29}$)**	32	31	223	38
***MCA*(*N*; 2, $4^{1}3^{39}2^{35}$)**	22	24	190	34
***MCA*(*N*; 2, $5^{1}3^{8}2^{2}$)**	18	19	121	22
***MCA*(*N*; 2, $7^{1}6^{1}5^{1}4^{5}3^{8}2^{3}$)**	41	37	354	47
***MCA*(*N*; 2, $5^{1}4^{4}3^{11}2^{5}$)**	23	23	182	25
***MCA*(*N*; 2, $6^{1}5^{1}4^{6}3^{8}2^{3}$)**	33	34	267	31
Sum	243	242	2318	339

For another, improved EO yields an 85% reduction in run time over SA. The principles of SA and improved EO can account for this great difference. In contrast to SA, which only considers the macroscopic behavior of computational systems (i.e. global fitness function), and does not study the micro mechanism of solution, improved EO simulates a complex multi-particle system in statistical physics. Both the collective behavior and the individual states of particles are considered simultaneously during improved EO. Therefore it is not

surprising that improved EO outperform SA for generating pairwise test suites a lot, which also answers Q2.

C. Threats to Validity

There are mainly three kinds of threats to our findings: external, internal and construct validity.

Threats to external validity [24] are conditions that limit the ability to generalize the results of our experiments. The major external threat in this paper is our choice of models of covering arrays, i.e. data from [7], [17]. We cannot guarantee that these models accurately represent real software systems.

Threats to internal validity are conditions that can affect the dependent variables of the experiment without the researcher's knowledge. There are two main threats to internal validity. Firstly, we may bias the results by setting poor parameters for some algorithms such as SA and improved EO. Secondly, although we have verified the results of every run, we cannot be completely sure that the implementations are correct translations from pseudo-code, nor that there are not bugs in these programs.

Threats to construct validity are that we have considered both run time and the size of the resulting test suite, but we have ignored other metrics that are important in some cases. For instance, varieties of *t*-sets, i.e. different values of *t*, in the early rows are desirable if we do not expect to finish pairwise testing.

V. Conclusion

Generating the smallest size test suite in the shortest time is a vital task in pairwise testing. A new approach for generating pairwise test suites has been proposed in this paper by applying improved EO, which is a novel search optimization method based on Bak-Sneppen model of biological evolution. Both local fitness function and global fitness function have also been defined in order to meet the requirement of improved EO, which is another contribution of this paper. Experimental results show that improved EO yields an 85% reduction in solution time and preserves the ability discovering the smallest size test suite, compared with SA. Therefore improved EO is a promising approach for pairwise testing.

In the future we will make extensive experiments to demonstrate our findings and improve techniques in this paper further. In addition, devising new approaches for constructing pairwise test suites is also an important goal of our following work.

References

[1] K. C. Tai and Y. Lei, "A test generation strategy for pairwise testing," *IEEE Transactions on Software Engineering*, vol. 28, pp. 109-111, January 2002.

[2] D. M. Cohen, S. R. Dalal, M. L. Fredman, and G. C. Patton, "The aetg system: an approach to testing based on combinatorial design," *IEEE Transactions on Software Engineering*, vol. 23, pp. 437-444, July 1997.

[3] D. R. Kuhn, D. R. Wallace, and A. M. Gallo, "Software fault interactions and implications for software testing," *IEEE Transactions on Software Engineering*, vol. 30, pp. 418-421, June 2004.

[4] S. R. Dalal et al. "Model-based testing in practice," in *Proceedings of the International Conference on Software Engineering*, pp. 285-294, 1999.

[5] C. J. Colbourn, M. B. Cohen, and R. C. Turban, "A deterministic density algorithm for pairwise interaction coverage," in *Proceedings of IASTED International Conference on Software Engineering*, pp. 345-352, 2004.

[6] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, "IPOG: a general strategy for t-way software testing," in *Proceedings of IEEE International Conference on the Engineering of Computer-Based Systems*, pp. 549-556, 2007.

[7] L. Yu and K. C. Tai, "In-parameter-order: a test generation strategy for pairwise testing," in *Proceedings of the 3rd IEEE International High-Assurance Systems Engineering Symposium*, pp. 254-261, 1998.

[8] M. B. Cohen, P. B. Gibbons, W. B. Mugridge, and C. J. Colbourn, "Constructing test suites for interaction testing," in *Proceedings of the 25th International Conference on Software Engineering*, pp. 38-48, 2003.

[9] J. Stardom, "Metaheuristics and the search for covering and packing arrays," Master dissertation, Department of Mathematics, Simon Fraser University, Canada, 2001.

[10] B. Stevens, "Transversal covers and packings," Ph.D. dissertation, Department of Mathematics, University of Toronto, Canada, 1998.

[11] B. J. Garvin, M. B. Cohen, and M. B. Dwyer, "An improved meta-heuristic search for constrained interaction testing," in *Proceedings of International Symposiumon Search-Based Software Engineering*, pp. 13-22, 2009.

[12] S. Boettcher and A. G. Percus, "Extremal optimization: methods derived from co-evolution," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 825-832, 1999.

[13] P. Bak and K. Sneppen, "Punctuated equilibrium and criticality in a simple model of evolution," *Physical Review Letters*, Vol. 71, pp. 4083-4086, December 1993.

[14] M. Martin, E. Drucker, and W. D. Potter, "Genetic algorithm, extremal optimization, and particle swarm optimization applied to the discrete network configuration problem," in *Proceedings of International Conference on Genetic and Evolutionary Methods*, pp. 129-134, 2008.

[15] R. Brownlie, J. Prowse, and M. S. Padke, "Robust testing of AT&T PMX/StarMAIL using OATS," *AT&T Technical Journal*, vol. 71, pp. 41-47, May 1992.

[16] R. Mandl, "Orthogonal latin squares: an application of experiment design to compiler testing," *Communications of the ACM*, vol. 28, pp. 1054-1058, October 1985.

[17] T. W. Tung and W. S. Aldiwan, "Automating test case generation for the new generation mission software system," in *Proceedings of IEEE Aerospace Conference*, pp. 431-437, 2000.

[18] K. Nurmela, "Upper bounds for covering arrays by tabu search," *Discrete Applied Mathematics*, vol. 138, pp. 143-152, March 2004.

[19] S. A. Ghazi and M. A. Ahmed, "Pair-wise test coverage using genetic algorithms," in *Proceedings of Congress on Evolutionary Computation*, pp. 1420-1424, 2003.

[20] S. Boettcher and A. G. Percus, "Extremal optimization for graph partitioning," *Physical Review E*, vol. 64, pp. 1-13, July 2001.

[21] M. B. Cohen, "Designing test suites for software interaction testing," Ph.D. dissertation, Department of Computer Science, The University of Auckland, New Zealand, 2004.

[22] S. Boettcher and A. G. Percus, "Optimizing through co-evolutionary avalanches," *Lecture Notes in Computer Science*, vol. 1917, pp. 447-456, 2000.

[23] S Boettcher and M. Grigni, "Jamming model for the extremal optimization heuristic," *Journal of Physics A-Mathematical and General*, vol. 35, pp. 1109-1123, January 2002.

[24] C. Wohlin et al., *Experimentation in software engineering: an introduction*, Kluwer Academic Publishers, Norwell, MA, USA, 2000.

Jianjun Yuan received the B.S. degree in 1998 from Hubei University, China, and the M.S. degree in 2005 from Huazhong University of Science & Technology, China, both in computer science and engineering. He is currently pursuing the Ph.D. degree in the Department of Computer Science and Technology at Tongji University, China. His research interests include software testing and evolutionary computing.

Changjun Jiang received the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 1995 and conducted post-doctoral research at the Institute of Computing Technology, Chinese Academy of Sciences, in 1997. He is a Professor with the Department of Computer Science and Technology, Tongji University, Shanghai, China. He has taken in over 20 projects supported by National Natural Science Foundation, National Key Technologies R&D Program, National Key Basic Research Developing Program, and other key projects at provincial or ministerial levels. He has published more than 100 papers in domestic and international academic journals and conference proceedings, including *IEEE Transactions on System, Man and Cybernetics, Information Sciences* and so on. Furthermore, he has published four books (supported by Science Publishing Foundation of the Chinese Academy of Science). His current areas of research are concurrent theory, Petri net, and formal verification of software. He has also been engaged in research of software engineering.