A Multi-objective Approach to Testing Resource Allocation in Modular Software Systems

Zai Wang, Ke Tang and Xin Yao

Abstract— **Nowadays, as the software systems become increasingly large and complex, the problem of allocating the limited testing-resource during the testing phase has become more and more difficult. In this paper, we propose to solve the testing-resource allocation problem (TRAP) using multi-objective evolutionary algorithms. Specifically, we formulate TRAP as two multi-objective problems. First, we consider the reliability of the system and the testing cost as two objectives. In the second formulation, the total testing-resource consumed is also taken into account as the third goal. Two multi-objective evolutionary algorithms, Non-dominated Sorting Genetic Algorithm II (NSGA2) and Multi-Objective Differential Evolution Algorithms (MODE), are applied to solve the TRAP in the two scenarios. This is the first time that the TRAP is explicitly formulated and solved by multi-objective evolutionary approaches. Advantages of our approaches over the state-of-the-art single-objective approaches are demonstrated on two parallel-series modular software models.**

keywords: Software Reliability; Multi-Objective Evolutionary Algorithm; Parallel-Series Modular Software System.

I. Introduction

Nowadays the software systems have become very large and complex. Despite of the decrease of the hardware costs, the costs of software systems have increased rapidly during the past several decades. In modern computer systems, software takes larger portion of the system cost than the hardware. Like in hardware development [1-3], reliability is a most important purpose in software development. There are many environmental factors in the software development process. The software testing-resource is a kind of entities which can be measured and controlled early in the software development cycle, and optimal allocation of testing resource can lead to an improvement of the reliability of a software system [4-5], Hence, the Test-Resource Allocation Problem (TRAP) plays an important part in the development of software systems.

Obviously, determining the optimal allocation of test-resource involves multiple considerations. For example, we would like to get a software with high reliability, but we also prefer minimizing the cost during the testing phase. At the same time, we may also aim to finish the testing phase as fast as we can (i.e. with the minimal time). Therefore, the TRAP is a multi-objective optimization problem, which concerns several incommensurable and often conflicting objectives. Though many researchers have carried out studies on the TRAP [6-13], most of them only considered reliability while ignoring the cost of the software system, i.e., they considered TRAP as a single-objective problem. Some other researchers did consider both the reliability and the cost [14-15], but they aggregated the two objectives to a scalar cost function, and optimized it in a single-objective way. Further, the total testing-resource consumed has never been considered in the optimization. Instead, a fixed resource budget is always defined in advance. In this paper, we suggest tackling the TRAP with Multi-Objective Evolutionary Algorithms (MOEAs). MOEAs, such as Strength Pareto Evolutionary Algorithm II (SPEA2) [16], Pareto Archived Evolution Strategy (PAES) [17], Nondominated Sorting Genetic Algorithm II (NSGA2) [18] and Multi-Objective Differential Evolution Algorithms (MODEs) [19-20], are a family of evolutionary approaches to multi-objective optimization problems. They have been proven to perform well on lots of real-world problems. To the best of our knowledge, MOEAs have never been applied to the TRAP before. We consider two scenarios. First, the reliability of the software and the testing cost. So the goal is to find a good trade-off between the reliability and the cost. Then, the total testing-resource consumed (i.e. the total time) is brought in as the third objective.

Compared to the existing single-objective approaches, which usually provide one single optimal solution at a time, MOEAs can offer a set of solutions that are well known as non-dominated solutions. Hence MOEAs can provide the designers a lot of choices with different level of trade-off between reliability and cost (in our first scenario), or between reliability, cost, and the total testing-resource consumed (in our second scenario). Efficacy of our proposed approaches are demonstrated on software reliability growth models (SRGMs) [21-22], which is a main type of software system models based on nonhomogeneous Poisson process (NHPP).

The rest of this paper is organized as follows: The basic structure of the parallel-series modular software system is presented in section 2. Section 3 shows how we solve the testing-resource allocation problem with NSGA2 and MODE. In section 4, our multi-objective approaches are experimentally compared to some existing single-objective approaches on two examples of the parallel-series software systems. Finally, discussion and conclusions are presented in Section 5.

The authors are with the Nature Inspired Computation and Applications Laboratory, the Department of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China. Xin Yao is also with CERCIA, the school of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, U.K.(emails: wangzai@mail.ustc.edu.cn, ketang@ustc.edu.cn, x.yao@cs.bham.ac.uk).

978-1-4244-1823-7/08/$25.00

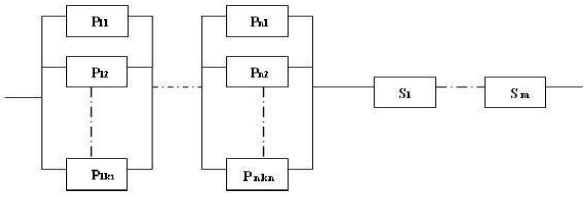

Fig. 1. The basic structure of a parallel-series modular software system

TABLE I

THE NOMENCLATURE OF THE SOFTWARE SYSTEM

T	total testing resource (or time)
T_i	testing resource allocated on module *i*
T_i*	optimal allocation resource on module *i*
C	the total testing cost
$C_i(R_i)$	cost function of module *i* according to the reliability of module *i* R_i
$m(t)$	mean value function in NHPP
R	system reliability
$R_i(x\|T_i)$	reliability function of module *i*, after a testing period T_i

II. PARALLEL-SERIES MODULAR SOFTWARE SYSTEM

A. Basic Structure of the Parallel-series Modular Software System

With the growth of the size of the software systems, a system is usually comprised of many modules in the designing and testing phase. The basic structure of a general parallel-series modular software system with *n* groups of parallel modules and *m* serial modules is shown in Fig.1.

B. The Performance of The Parallel-series Modular Software System

According to the software reliability literature, we adopt the following models to quantify the reliability of a parallel-series modular system and the testing cost. A summary of the nomenclature are presented in Table I.

For every modular in this system, the failure intensity of module *i* is $\lambda_i(T_i)$ and it can be calculated by:

$$\lambda_i(T_i) = a_i b_i exp(-b_i T_i) \tag{1}$$

Where a_i and b_i are constants. a_i is the mean value of the total errors in modular *i* and b_i describes the rate of detected errors in modular *i*. T_i is the testing time allocated to module *i*.

Thus the reliability of module *i* is

$$R_i(x|T_i) = exp\{-\lambda_i(T_i)x\}, x \geq 0 \tag{2}$$

After the T_i unit time of testing, The probability of no fault occuring in the interval $(T_i, T_i + x]$ is $R_i(x|T_i)$. From Eq. (1) and Eq. (2), we can find that the reliability of modular *i* is exponentially increasing to the resource allocated to the modular *i* [23].

TABLE II

PROBLEM.1:THE TESTING-RESOURCE ALLOCATION PROBLEM WITH TWO OBJECTIVES

(1) Maximize
$R(x|T) = \prod_{l=1}^{n}(1 - \prod_{i=1}^{k_l}[1 - R_{l_i}(x|T_{l_i})]) \prod_{j=1}^{m} R_j(x|T_j)$
(2)Minimize
$C(R_{l_i}, R_j) = \sum_{l=1}^{n}\sum_{i=1}^{k_l} C_{l_i}(R_{l_i}) + \sum_{j=1}^{m} C_j(R_j)$
Subject to
$\sum_{l=1}^{n}\sum_{i=1}^{k_l} T_{l_i} + \sum_{j=1}^{m} T_j \leq T$ and $T_{l_i}, T_j \geq 0$
The total testing-resource to be allocated is *T*

Based on the above two equations, the reliability of this parallel-series modular software system is calculated as following equation:

$$R(x|T) = \prod_{l=1}^{n}(1 - \prod_{i=1}^{k_l}[1 - R_{l_i}(x|T_{l_i})]) \prod_{j=1}^{m} R_j(x|T_j) \tag{3}$$

The $(1-\prod_{i=1}^{k_l}[1-R_{l_i}(x|T_{l_i})]$ is the reliability function of the *l*th parallel groups. There are *n* groups of parallel modules, So the total reliability of these *n* groups of parallel modules are $\prod_{l=1}^{n}(1-\prod_{i=1}^{k_l}[1-R_{l_i}(x|T_{l_i})])$ and $\prod_{j=1}^{m} R_j(x|T_j)$ presents the total reliability of the *m* series modules.

The cost function for individual module which we will adopt has been explained in [24]:

$$C_i(R_i) = H_i \times e^{B_i R_i - D_i} \tag{4}$$

The H_i, B_i and D_i are constants associated with the modular *i*. [24] has told us that the cost is exponentially increasing to the improved reliability of the single model.

The corresponding cost of the parallel-series modular software system will be

$$C(R_{l_i}, R_j) = \sum_{l=1}^{n}\sum_{i=1}^{k_l} C_{l_i}(R_{l_i}) + \sum_{j=1}^{m} C_j(R_j) \tag{5}$$

III. SOLVE THE TESTING-RESOURCE ALLOCATION PROBLEMS WITH MOEAS

A. Formulating TRAP as Multi-objective Optimization Problems

Based on the above equations in section 2.1, we can easily calculate the reliability and cost of a parallel-series modular system. We allocate testing-resource for the sake of getting an allocation of testing resource in an interval corresponding to highest reliability and lowest cost of a software system. However, Eqs. (1-5) denotes that the cost and reliability increase exponentially to the testing resource allocated to the system, i.e., getting a higher reliability means we need to pay more cost. Hence, we can consider the reliability and cost as two objectives and formulate a two-objective problem in Table II. Then we can use MOEAs to get a set of solutions which are non-dominated with one another.

From Table II, it can be observed that, as a two-objective problem, maximizing the reliability and minimizing the cost

TABLE III

PROBLEM.2:THE TESTING-RESOURCE ALLOCATION PROBLEM WITH THREE OBJECTIVES

(1) Maximize
$R(x|T) = \prod_{l=1}^{n}(1 - \prod_{i=1}^{k_l}[1 - R_{l_i}(x|T_{l_i})]) \prod_{j=1}^{m} R_j(x|T_j)$
(2)Minimize
$C(R_{l_i}, R_j) = \sum_{l=1}^{n} \sum_{i=1}^{k_l} C_{l_i}(R_{l_i}) + \sum_{j=1}^{m} C_j(R_j)$
(3)Minimize
$T_c = \sum_{l=1}^{n} \sum_{i=1}^{k_l} T_{l_i} + \sum_{j=1}^{m} T_j$
Subject to
$\sum_{l=1}^{n} \sum_{i=1}^{k_l} T_{l_i} + \sum_{j=1}^{m} T_j \leq T$ and $T_{l_i}, T_j \geq 0$
The total testing-resource to be allocated is T

of a software system are two purposes. Also we can find this problem has bound constraint of the testing-resource (in this paper means the testing time T).

Despite of the increase of the size and complexity of the software systems, the cycle of a software development process has become shorter and shorter and the testing time which can be allocated to the software system in the testing phase has been become insufficient day by day. So designers want to save the testing time eagerly, and lower reliability can sometimes be acceptable. So we consider the total testing-resource expenditure as a new target, and the three-objective problem can be formulated in Table III.

This problem is also a constrained multi-objective problem and has a bound constraint for the testing time. The designers can get a choice based on the testing time from the non-dominated solutions.

B. Evolutionary Multi-objective Optimization Approaches to TRAP

Given the two multi-objective formulations, it is obviously that MOEAs are promising approaches to the TRAP. In this work, we suggest using NSGA2 and MODE as the specific problem solver. Proposed by Deb et al. [18], NSGA2 is one of the most famous MOEAs and has been extensively used in various real-world problems. MODE [20] is a recently proposed approach and show some advantages over NSGA2 in some aspects. Due to space constraints, the details of NSGA2 and MODE may not be described in this paper, interested readers are referred to the original publications. In the next subsection, we briefly describe some issues regarding the application of MOEAs to the TRAP.

C. Two Implementation Details

When using MOEAs to solve the TRAPs, two particular issues need to be addressed, i.e., how to code and generate solutions

1) The Coding Scheme: In modular software systems, the total testing resource comprises testing resource consumed in every module. The testing resource of the parallel-series modular software system in this paper is the total testing time T. Thus, we can construct a chromosome using a list of time every modular consumes. The sum of the elements in the list is T. In other words, for a system with n modules, we have:

$$p = (t_1, t_2, ..., tn)$$

Subject to $sum(t_i) = T$

TABLE IV

REPAIR THE INFEASIBLE OFFSPRING a'

Assume a' is re-assigned to a new chromosome a''.
First we calculate the sum of elements in a':$total = sum(a')$.
For problem.1, the a'' is:
a''=$(t''_1, t''_2, ...t''_n) = ((t'_1 \times T/total), (t'_2 \times T/total), ...(t'_n \times T/total))$.
For problem.2, the a'' is:
a''=$(t''_1, t''_2, ...t''_n) = ((t'_1 \times T \times rand(1)/total)$,
$(t'_2 \times T \times rand(1)/total), ...(t'_n \times T \times rand(1)/total))$.

2) A Repairing Operator: Assume there is a chromosome $a = (t_1, t_2, ..., t_n)$, after the crossover between a and another chromosome b or mutation on a, there will be a new chromosome $a' = (t'_1, t'_2...t'_n)$. If the sum of elements in a' exceeds the pre-defined total resource T, the a' is an infeasible solution and will be repaired. In this work, we repair the infeasible solution following the method presented in Table IV.

IV. COMPARE THE MOEAS WITH OTHER ALGORITHMS ON TWO PARALLEL-SERIES SOFTWARE SYSTEM MODELS

In this section, we experimentally compare our multi-objective approaches with traditional single-objective approaches on two examples. Three state-of-the-art single-objective methods are chosen for comparison. The first algorithm is devised by Dai and Xie [25], and denoted as D-M algorithm. The second is proposed by Yang and Xie [23], denoted as Y-X algorithm. The last method is devised by Tom and Murthy [26] and denoted as T-M algorithm.

In the case study, we adopt a simple software model with only two modules and a complex model with eight modules. In the literature, the D-M and T-M algorithms have been applied to the simple and complex models, respectively. And the Y-M algorithm has been applied to both. For the sake of a fair comparison, we adopt the same structure and parameter settings of the two models as in [25]. To be specific, the structures of the simple model and the complex model are demonstrated in Fig.2 and Fig.3, respectively. Parameter settings of the systems are given in Tables V and VI, respectively. When considering the total testing resource as the third objective, the maximal total testing resource is also required to be pre-defined. Since this objective has never been considered as an objective in the literature, we assume that 10 persons are prepared to test this simple model, and 23 persons for complex model. The deadline of testing time for every person from now on is 1000 hours. Thus, the total testing time is calculated by $10 \times 1000 = 10000$ hours for simple model and $23 \times 1000 = 23000$ for complex model.

A. Results of The Simple Model

1) Parameter Settings: When we use NSGA2 to solve TRAP on this simple model, the parameters of NSGA2 are

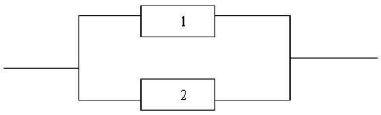

Fig. 2. The structure of a two-unit parallel modular software system

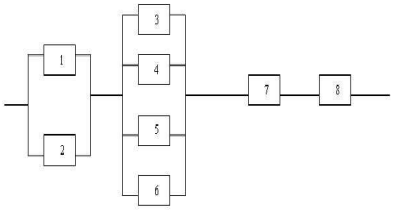

Fig. 3. The structure of the complex modular software system

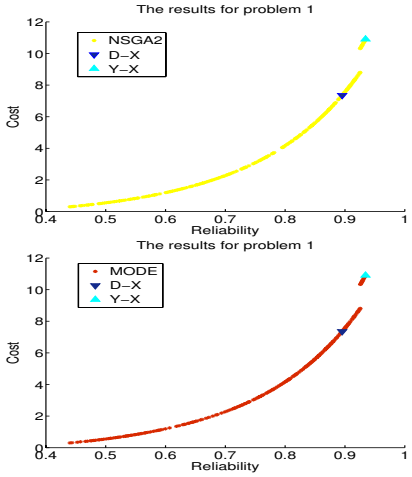

Fig. 4. The results for problem.1 on simple model

assigned as follow: crossover probability: 0.9; mutation probability: 0.1; the terminate generations: 200. The population size is assigned as 200 for problem.1 and 1000 for problem.2.

For MODE, parameter valued as: scaling factor *F*=0.3; crossover probability *CR* =0.3. The population size is same as the enactment of NSGA2.

When comparing MOEAs with D-X algorithm which combine the reliability and cost to a single objective with the sensitivity weight K_1 and K_2, the K_1 is set as 1.1 and K_2 as 0.3.

2) Results on Problem.1: The NSGA2, MODE, D-X, Y-X are adopted to solve this problem, the results of the above algorithms are shown in Fig.4.

From Fig.4, we can find that NSGA2 and MODE get a set of solutions which are non-dominated by each other. Thus as designers of software systems, we can choose a solution with respect to the reliability and cost easily from the pareto-front

TABLE V

THE VALUES OF THE PARAMETERS FOR SIMPLE MODEL

	Modular1	Modular2
a_i	249.22	199.48
$b_i(10^{-4}/h)$	5.9464	5.8379
H_i	5.05	4.95
B_i	6	6.3
D_i	5	5.1

TABLE VI

THE PARAMETERS OF COMPLEX MODEL

Modules	a_i	b_i	H_i	B_i	D_i
1	210	0.00051	3.493	6.011	4.97
2	199	0.00059	3.503	6.12	4.93
3	453	0.00048	3.498	6.012	4.955
4	345	0.00058	3.498	6.001	4.997
5	258	0.00063	3.499	6.002	4.995
6	221	0.00074	3.5015	6.15	4.97
7	33.99	0.00597	3.495	6.01	4.98
8	32.32	0.00593	3.500	6.005	4.01

TABLE VII

COMPARE MOEAS WITH Y-X AND D-X ALGORITHM

Cost factor	Algorithm	T_1*	T_2*	Reliability	Cost(units)
Consider	NSGA2	9402.37	597.63	0.8953	7.354
Consider	MODE	9403.49	596.51	0.8953	7.355
consider	D-X	9404.34	595.66	0.8953	7.356
Consider	NSGA2	0	10000	0.9344	10.903
Consider	MODE	0	10000	0.9344	10.903
Without	Y-X	0	10000	0.9344	10.903

of NSGA2 and MODE.

It might be interesting to check whether the results achieved by D-X and Y-X algorithms really lie on the pareto front of the problem, So we ran MOEAs for 30 times and calculate the mean value of the results. Then, the solutions corresponding to the same reliability achieved by D-X (0.8953) and Y-X (0.9344) are picked out and presented in TABLE VII, respectively. From TABLE VII, we can observe that MOEAs manage to provide almost the same solutions as the compared single-objective approaches, while they also provide many other candidate solutions to the decision maker.

In this work, the NSGA2 and MODE can get the highest reliability with the $T_1* = 0$ and $T_2* = 10000$ every time, so the mean value of these results is also $\overline{T_1*} = 0$ and $\overline{T_2*} = 10000$ when comparing with Y-X algorithm.

3) Results on Problem.2: For this problem, we consider the total testing resource expenditure as the third objective. Y-X algorithm and D-X algorithm set the total testing resource as constant, hence they can not be used to solve problem.2. However we can draw the results of Y-X and D-X algorithms with the solutions of NSGA2 and MODE when the total testing resource is set to be 10000 hours. The results are presented in Fig.5.

From Fig.5, we can find that the solutions obtained by D-X and Y-X still approximately lie on the pareto front obtained by MODE. That means MODE still manage to

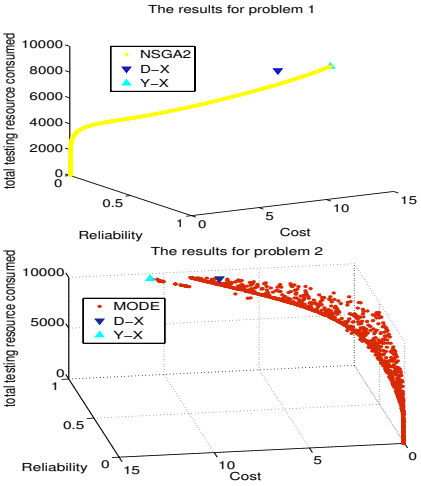

Fig. 5. The results for problem.2 on simple model

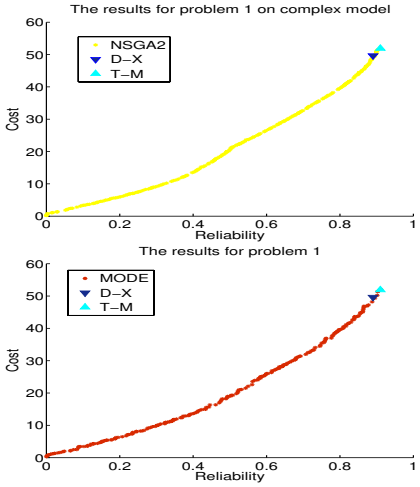

Fig. 6. the results for problem.1 on complex model

provide a set of good solutions in this scenario. The results obtained by NSGA2 are different. NSGA2 offer us a set of solutions with a shape of curve, i.e., there is a unique value of reliability and cost corresponding to a fixed value *T*. As we know, there should be a set of choices with a fixed *T*, as can be observed on MODE. There might be two reasons for NSGA2 not providing a set of appropriate solutions :
a):Too close relation between the reliability and testing-time consumed.
b):Too early strong convergence to the pareto-front.

When MODE is utilized, the problem *b* does not exist and the DE strategy can indicate us to get a good spread of the non-dominated solutions.

B. Results of The Complex Model

1) Parameter Settings: The parameter settings for approaches on this complex model are same as the above settings on the simple model, expect for the population size is assigned as 3000 for problem.2.

2) Results on Problem.1: Y-X algorithm cannot work on this model, So we use T-M algorithm which consider this system as a distributed computing system [26]. The results of the approaches on problem.1 are shown in Fig.6.

To quantitavely compare MOEAs, D-X and T-M algorithms, we adopt the same procedure as in the previous subsection. To be specific, we averaged the results of MOEAs over 30 runs, picked out the ones with the same reliability value as D-X and T-M, and present them in TABLE VIII. Again, it can be observed that the MOEAs are able to find the pareto optimial solutions, including those obtained via single-objective methods.

Table XIII tells us that NSGA2 and MODE get better solutions than the D-X algorithm. MODE outperforms NSGA2 when getting the highest reliability.

TABLE VIII
COMPARE MOEAS WITH Y-X AND T-M ALGORITHM ON SIMPLE MODEL

Alogrithm	Reliability	Cost(units)
NSGA2	0.895725	49.46
MODE	0.894875	49.33
D-X	0.89	49.65
NSGA2	0.9055	51.88
MODE	0.9118	52.04
T-M	0.91	52.64

3) Results on Problem.2: The T-M algorithm and D-X algorithm also set the total testing resource as constant, so they cannot be used to solve problem.2. However we can draw the results of T-M and D-X algorithms with the solutions of MOEAs when the total testing resource is set to be 23000. Solutions of above algorithms are shown in the following Fig.7.

From Fig.7, we can see that a fixed value of testing-resource consumed denoted as T_i corresponds to a set of solutions which comprise a range of reliability and cost. So as designers of software systems, we can get lots of choices in specified reliability interval with lower testing-resource expenditure.

V. CONCLUSION

In this paper, we first formulate two multi-objective testing-resource allocation problems on parallel-series modular software system, and try a new attempt to use MOEAs (NSGA2 and MODE) to solve these two problems. Two main observations can be made by comparing MOEAs to state-of-the-art single-objective algorithms. First, the MOEAs provide the best results that has been obtained with single-objective approaches, while MOEAs also offer the designers of software systems a set of appropriate choices. Second, our approach can even handle the scenario where the total testing resource expenditure is involved as the third objective. In this way, we manage to obtain solutions with acceptable

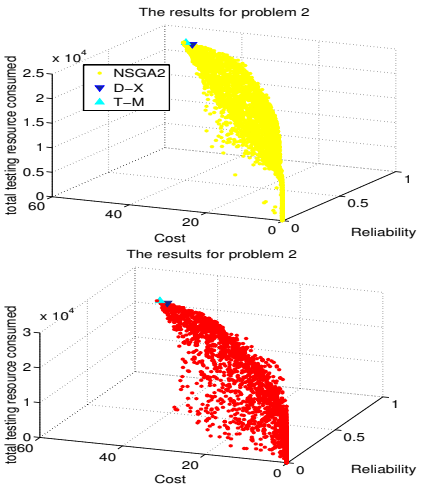

Fig. 7. The results for problem.2 on complex model

reliability, cost and total time expenditure, which might be of more help to designers and conners of software systems. To summarize, MOEAs are more applicable and successful than the single-objective algorithms.

Although we focus on the parallel-series software systems in this paper, we believe that the general idea are presented here is also applicable to many other modular systems such as star structure, circular structure, etc. we will investigate these issues in our future work.

ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China (Grant No. 60428202), The Engineering and Physical Sciences Research Council (EPSRC Grant No. EP/D052785/1) and The Fund for Foreign Scholars in University Research and Teaching Programs (Grant No. B07033).

REFERENCES

[1] C. Y. Huang, J. H. Lo, S. Y. Kuo and M. R. Lyu, "Optimal Allocation of Testing Resources for Modular Software Systems," In *Proceedings of the Thirteenth IEEE International Symposium on Software Reliability Engineering*, 2002.

[2] C. Y. Huang, J. H. Lo, S. Y. Kuo and M. R, "Optimal Allocation of Testing-Resource Considering Cost, Reliability, and Testing-Effort," Dependable Computing, 2004. Proceedings. 10th *IEEE Pacific Rim International Symposium* on 3-5 March 2004, pp. 103-112.

[3] Y. S. Dai, X. L. Wang, "Optimal Resource Allocation on Grid Systems For Maximizing Service Reliability Using A Genetic Algorithms," In *Reliability Engineering and System Safety*, 2006.

[4] X. M. Zhang, M. Y. Shin, H. Pham, 2001. "Exploratory Analysis of Environmental Factors for Enhancing the Software Reliability Assessment," J. Syst. Software 57 (1), pp. 73-78.

[5] J. Tian, 1999. "Measurement and Continuous Improvement of Software Reliability Throughout Software Life-Cycle," J. Syst. Software 47 (2C3), pp. 189-195.

[6] P. Kubat and H. S. Koch, "Managing Test-Procedure to Achieve Reliable Software," *IEEE Transcations on Reliability*, Vol, 32, No. 3, pp. 299-303, 1983.

[7] P. Kubat, "Assessing Reliability of Modular Software," *Operation Research Letters*, Vol. 8, No. 1, pp. 35-41, 1989.

[8] H. Ohtera and S. Yamada, "Optimal Allocation and Control Problems for Software-Testing Resource," *IEEE Transcations on Reliablity*, Vol. 39, No. 2, pp. 171-176, 1990.

[9] S. Yamada, T. Lehimori and M. Nishiwaki, "Optimal Allocation Policies for Testing Resource Based On a Software Reliability Growth Model," *Mathemarical and Computer Modelling*, Vol. 22, pp. 295-301, 1995.

[10] Y. W. Leung, "Dynamic Resource Allocation for Software Module Testing," *The Journal of the Systems and Software*, Vol. 37, No. 2, pp. 129-139, May 1997.

[11] Y. W. Leung, "Software Reliablity Allocation Under Uncertain Operational Profits," *Journal of the Operational Research Society*, Vol.48, No. 4, pp. 401-411, April 1997.

[12] Y. W. Leung, "Optimal Reliability Allocation for Modular Software System Designed for Multiple Customers," *IEICE Transcations on Information and Systems*, Vol. 79, No. 12, pp. 1655-1662, December 1996.

[13] R. H. Huo, S. Y. Kuo and Y. P. Chang, "Efficient Allocation of Testing Reaources for Software Module Testing Based on the Hyper-Geometric Distribution Software Reliability Growth Model," *IEEE Transcations on Reliability*, Vol. 45, No. 4, pp. 541-549, Dec 1996.

[14] B. Yang and M. Xie, "Testing-resource Allocation for Redundant Software Systems," *Proceedings of the 1999 Pacific Rim International Symposium on Dependable Computing(PRDC"99)*, Dec 1999.

[15] B. Yang and M. Xie, "Optimal Testing-Time Allocation for Modular Systems, " *International Journal of Quality and Reliability Management*, Vol. 18, No. 8, pp. 854-863, 2001.

[16] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," In K. Giannakoglou et al., editor, *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pp. 95-100, Athens, Greece, 2002.

[17] J. D. Knowles and D. W. Corne, "Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy," In *Evolutionary Computation*, vol. 8, no. 2, pp. 149-172, 2000.

[18] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," In *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, April 2002.

[19] C. S. Chang, D. Y. Xu. and H. B. Quek. "Pareto-Optimal Set Based Multiobjective Tuning of Fuzzy Automatic Train Operation for Mass Transit System," In *IEE Proceedings on Electric Power Applications*, September 1999.

[20] V. L. Huang, P. N. Suganthan, A. K. Qin, and S. Baskar, "Multi-Objective Differential Evolution with External Archive and Harmonic Distance-Based Diversity Measure," Nanyang Technological University, Singapore, Tech. Rep. TR-07-01, 2006.

[21] M. R. Lyu(1996). *Handbook of Software Reliabilty Engineering*, McGraw Hill.

[22] O. Berman and N. Ashrafi, "Optimization Models for Reliability of Modular Software Systems," *IEEE Transcations on Software Engineering*, Vol.19, No.11, pp. 1119-1123, Nov. 1993.

[23] B. Yang, M. Xie, 2000. "A Study of Operational and Testing Reliability in Software Reliability Analysis," *Reliab. Eng. Syst. Safety* 70, pp. 323-329.

[24] A. Kumar, K. Malik, "Voting Mechanisms in Distributed Systems," *IEEE Transcations on Reliability* 40 (5), pp. 593-600, 1991.

[25] Y. S. Dai, M. Xie, K. L. Poh, B. Yang, "Optimal Testing-Resource Allocation with Genetic Algorithm for Modular software Systems," *The Journal of Systems and Software*, Vol. 66, pp. 47-55, 2003.

[26] P. A. Tom, C. S. R. Murthy, "Algorithms for Reliability-Oriented Module Allocation in Distributed Computing Systems," *J.Syst.Software* 40(2), pp. 125-138, 1998.