UNIVERSITY OF BIRMINGHAM
University of Birmingham Research at Birmingham

Constraint handling in NSGA-II for solving optimal testing resource allocation problems

Zhang, Guofu; Su, Zhaopin; Li, Miqing; Yue, Feng; Jiang, Jianguo; Yao, Xin

DOI:
10.1109/TR.2017.2738660

License:
None: All rights reserved

Document Version
Peer reviewed version

Citation for published version (Harvard):
Zhang, G, Su, Z, Li, M, Yue, F, Jiang, J & Yao, X 2017, 'Constraint handling in NSGA-II for solving optimal testing resource allocation problems', *IEEE Transactions on Reliability*, vol. 66, no. 4, pp. 1193-1212. https://doi.org/10.1109/TR.2017.2738660

Link to publication on Research at Birmingham portal

Publisher Rights Statement:
© 2017 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

General rights
Unless a licence is specified above, all rights (including copyright and moral rights) in this document are retained by the authors and/or the copyright holders. The express permission of the copyright holder must be obtained for any use of this material other than for purposes permitted by law.

•Users may freely distribute the URL that is used to identify this publication.
•Users may download and/or print one copy of the publication from the University of Birmingham research portal for the purpose of private study or non-commercial research.
•User may use extracts from the document in line with the concept of 'fair dealing' under the Copyright, Designs and Patents Act 1988 (?)
•Users may not further distribute the material nor use it for the purposes of commercial gain.

Where a licence is displayed above, please note the terms and conditions of the licence govern your use of this document.

When citing, please reference the published version.

Take down policy
While the University of Birmingham exercises care and attention in making items available there are rare occasions when an item has been uploaded in error or has been deemed to be commercially or otherwise sensitive.

If you believe that this is the case for this document, please contact UBIRA@lists.bham.ac.uk providing details and we will remove access to the work immediately and investigate.

Download date: 26. Apr. 2024

Constraint Handling in NSGA-II for Solving Optimal Testing Resource Allocation Problems

Guofu Zhang, *Member, IEEE,* Zhaopin Su, *Member, IEEE,* Miqing Li, Feng Yue, Jianguo Jiang, and Xin Yao, *Fellow, IEEE*

***Abstract*—In software testing, optimal testing resource allocation problems (OTRAPs) are important when seeking a good trade-off between reliability, cost, and time with limited resources. There have been intensive studies of OTRAPs using multi-objective evolutionary algorithms (MOEAs), but little attention has been paid to the constraint handling. This paper comprehensively investigates the effect of the constraint handling on the performance of nondominated sorting genetic algorithm II (NSGA-II) for solving OTRAPs, from both theoretical and empirical perspectives. The heuristics for individual repairs are first proposed to handle constraint violations in NSGA-II, based on which several properties are derived. Additionally, the z-score based Euclidean distance is adopted to estimate the difference between solutions. Finally, the above methods are evaluated and the experiments show several results. 1) The developed heuristics for constraint handling are better than the existing strategy in terms of the capacity and coverage values. 2) The z-score operation obtains better diversity values and reduces repeated solutions. 3) The modified NSGA-II for OTRAPs (called NSGA-II-TRA) performs significantly better than the existing MOEAs in terms of capacity and coverage values, which suggests that NSGA-II-TRA could obtain more and higher-quality testing-time allocation schemes, especially for large, complex datasets. 4) NSGA-II-TRA is robust according to the sensitivity analysis results.**

***Index Terms*—Software reliability, testing-resource allocation, multi-objective optimization, constraint handling, heuristics.**

I. Introduction

SOFTWARE testing, which aims to detect software failures, discover and correct defects, and thus achieve software quality for customer satisfaction, has played an essential role in the validation and verification of output in the software development industry [1]. Specifically, in the popular parallel-series modular software systems [2], [3], many parallel and serial modules need to be assigned available resources to cope with the testing. However, the number of possible tests for even simple modules is practically infinite. Accordingly, a natural and important issue in software testing is how to efficiently allocate the finite testing resources to modules. That is, finding an optimal testing resource allocation scheme for various modules to maximize the overall reliability of the entire software system. This is also known as Optimal Testing Resource Allocation Problems (OTRAPs) [4].

From a technical perspective, OTRAPs have been studied extensively. Most early research in this area addressed single-objective optimization problems, such as maximizing the reliability with a cost constraint [5], [6], minimizing the cost with the constraints on the reliability and time [7], [8], and minimizing the testing time with the required reliability [9], [10]. Nevertheless, the traditional linear programming or dynamic programming are mostly impractical for large-scale problems because of their computational complexity. Moreover, it is impossible for the final single solution to be optimal in terms of reliability, cost, and time altogether.

Recently, especially over the last decade, with the development of search-based software engineering, metaheuristic search techniques are applied to the difficult software engineering problems and to find near-optimal or good-enough solutions [11]–[16]. Particularly, researchers and practitioners have become deeply involved in OTRAPs with respect to multi-objective evolutionary algorithms (MOEAs) [17]–[22], such as harmonic distance based non-dominated sorting genetic algorithm II (called HaD-MOEA) [18] and multi-objective differential evolution on the basis of weighted normalized sum (WNS-MODE) [21]. These algorithms were expected to seek a good trade-off between reliability, cost, and time under the given time threshold, in which the system reliability is as high as possible, the consumed testing cost and time are as low as possible. Although these algorithms provided a lot of additional choices in comparison with the single-objective approaches, they ignored an important question: whether each solution is feasible? In essence, it is both impossible and unrealistic for the finite time to satisfy the testing needs of all the modules. Consequently, in these algorithms, there may exist a large number of infeasible solutions that violate the given time constraint. However, to the best of our knowledge, the constraint handling for repairing infeasible solutions in solving OTRAPs is quite limited.

In multi-objective approaches to OTRAPs, the constraint handling plays a valuable role in their exploration and exploitation. The fact was first mentioned in [17] and further discussed

Manuscript received –; revised –. This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 61573125, 61329302, and 61371155, in part by the Engineering and Physical Sciences Research Council under Grant No. EP/J017515/1, in part by the Anhui Provincial Natural Science Foundation under Grant Nos. 1608085MF131, 1508085MF132, and 1508085QF129, and in part by the Science and Technology Innovation Committee Foundation of Shenzhen under Grant No. ZDSYS201703031748284.

G. Zhang, Z. Su, F. Yue, and J. Jiang are with the School of Computer and Information, Hefei University of Technology, Hefei 230009, China (e-mail: zgf@hfut.edu.cn; szp@hfut.edu.cn; yuefeng@hfut.edu.cn; jgjiang@hfut.edu.cn).

M. Li is with the CERCIA, School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K. (e-mail: m.li.8@cs.bham.ac.uk).

X. Yao (the corresponding author) is with the Shenzhen Key Lab of Computational Intelligence, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China, and also with the CERCIA, School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K. (e-mail: xiny@sustc.edu.cn).

in [18], [19], [21]. Typically, in HaD-MOEA [18] and WNS-MODE [21], if the time threshold is violated in a solution, the infeasible solution will be repaired and migrated randomly into the feasible region of the search space by reducing each element value of the solution vector. However, no attempts have been made to link up the solution repairs with the nature of the used MOEAs themselves. Specifically, if infeasible solutions have to be repaired, would the repairs influence the balance between exploration and exploitation, and destroy the useful evolutionary information in solutions? Unfortunately, there is no theoretical analysis found to support the validity of such a direct reduction operation.

In addition to the above, the three optimization objectives reliability, cost, and time in OTRAPs have different scales and distributions. When estimating the difference between solutions, if we directly adopt the original solution samples, the difference calculation will mainly depend on the value of the testing time. Note that in OTRAPs, the system reliability which has the smallest value range is the most important, and thus it is inevitable that error will be incurred. In such a case, HaD-MOEA [18] and WNS-MODE [21] still simply summed the absolute normalized difference in each objective value of two individuals. This distance-estimation approach cannot reflect the true neighbor relationship among individuals in Euclidean space, when the number of objectives is equal to three or more.

To address these shortcomings, this paper takes into account the simple nondominated sorting genetic algorithm II (NSGA-II) [23] and investigates the potential constraint violations in the crossover and mutation operators, based on which heuristics for constraint handling are proposed. Our research findings suggest that the heuristics are computationally simple and maintain the original evolutionary characteristics in solutions. Thus, we recommend that the constraint handling in solving OTRAPs should be favorably combined with the evolutionary operators of the selected MOEAs. In addition, the Euclidean distance calculated on the standardized data has been frequently used to evaluate the difference between solutions [24]–[26]. In this paper, we adopt the most common z-scores in mathematical statistics [27] to make each objective value in solution samples have the same trend before calculating the Euclidean distance. Our experimental results on four parallel-series modular software systems with gradual complexities show that the improved NSGA-II for OTRAPs (called NSGA-II-TRA) outperforms the existing MOEAs on the capacity, coverage, and pure diversity values. This means that NSGA-II-TRA should be able to obtain superior testing-time allocation schemes. Moreover, NSGA-II-TRA seems quite robust from the sensitivity analysis results. This paper makes the following contributions.

1) A group of novel heuristics for constraint handling in the process of crossover and mutation operations are proposed. Compared with the existing work, the heuristics only repair the element values that cause the constraint violation rather than all the element values in the solution vector.
2) A theoretical analysis of the constraint handling is carried out, where several theorems are derived to provide evidence that the constraint handling maintains the previous evolutionary tendency of solutions.
3) The z-scores are adopted to standardize the solution samples with different scales of the reliability, cost, and time. Then, the standardized Euclidean distance is used to estimate the neighbor relationship among individuals instead of the traditional 1-normal distance in NSGA-II.

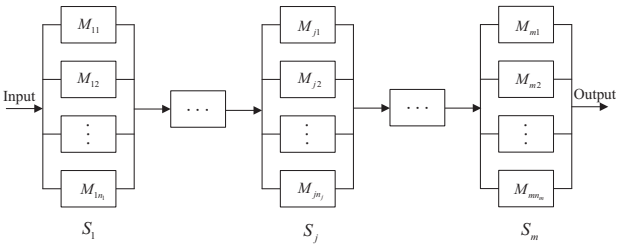


Fig. 1. The classical structure of a parallel-series modular software system.

The remainder of this paper is organized as follows. We first recall the multi-objective optimization model for OTRAPs in Section II. In Section III, we present our NSGA-II based search algorithm (i.e., NSGA-II-TRA) for OTRAPs, showing how NSGA-II-TRA initializes individuals, how NSGA-II-TRA repairs the infeasible individuals in the crossover and mutation operators to ensure the constraints and also preserve the evolutionary trend, and how NSGA-II-TRA standardizes the solution samples. After that, Section IV introduces performance metrics, provides an empirical evaluation of our approaches, and benchmarks NSGA-II-TRA against the existing algorithms for OTRAPs. Finally, Section V concludes this paper and outlines future directions of the research.

II. Problem Description

In software engineering, there mainly exist two popular models for simulating and describing real-world software systems: the parallel-series modular software model [28] and the architecture-based model [9], [10]. The architecture-based model is proposed to characterize the reliability of the object-oriented and component-based software systems by considering the system architecture, which is an important characteristic of the system depending on its operational profile [21]. The parallel-series modular software model is used to assess the reliability in light of the reliability block diagram that shows how the components of the system are reliability-wise connected. Although the parallel-series modular software model does not fully consider the system architecture in realistic applications, it is very simple and straightforward. Accordingly, following the previous work, in this paper we discuss the OTRAPs on the basis of the parallel-series modular software model.

Fig. 1 shows the basic structure of a complex parallel-series modular software system with m ($m \in \mathbb{N}$) serial subsystems, $S_1, \ldots, S_j, \ldots, S_m$. Each subsystem S_j ($j \in \{1, \ldots, m\}$) includes n_j ($n_j \in \mathbb{N}$) parallel modules, $M_{j1}, \ldots, M_{jn_j}$, which are tested separately [28].

In software testing, the available testing resources can be manpower, CPU hours, and executed test-cases [4]. In this

paper, we assume that the available testing resources are the given total testing time T^*, which is a time threshold and can be calculated according to the number of software testers and the working hours of each tester [18], [21]. For example, if 10 testers are available for testing the studied software system and each tester can spend up to 1,000 hours on the testing task, $T^* = 10 \cdot 1000 = 10,000$ (hours). We denote t_{jk} ($k \in \{1, \ldots, n_j\}$) as the possible testing time invested in a module M_{jk}.

The relationship between reliability and testing time can be described by software reliability growth models [29], [30], where the fault removing is generally recognized as a nonhomogeneous Poisson process. In these models, the failure intensity φ_{jk} of module M_{jk} can be calculated by

$$\varphi_{jk}(t_{jk}) = a_{jk} \cdot b_{jk} \cdot \exp(-b_{jk} \cdot t_{jk})$$

where a_{jk} is the mean value of the total errors in module M_{jk} and b_{jk} represents the rate of detected errors in M_{jk}. Given this, the achieved reliability of M_{jk} can be calculated by

$$r_{jk}(\lambda|t_{jk}) = \exp[-\varphi_{jk}(t_{jk}) \cdot \lambda]$$

where $\lambda \geq 0$ and denotes the period of workable time or the estimated life of the studied software system.

For a subsystem S_j, there may be n_j parallel modules to improve its performance. S_j cannot work only when all the parallel modules in S_j are unavailable. Therefore, the expected reliability of each S_j is $1 - \prod_{k=1}^{n_j}[1 - r_{jk}(\lambda|t_{jk})]$. Then, according to the multiplication rule, the achieved reliability of the software system is the total reliability of the m serial subsystems [18], that is,

$$R = \prod_{j=1}^{m}\{1 - \prod_{k=1}^{n_j}[1 - r_{jk}(\lambda|t_{jk})]\}$$

The testing cost refers to all the possible software testing expenditures. However, there is no hard and fast rules for prescribing how much software testing cost should be. What we can do is to estimate the amount being spent on testing and measuring quality, as well as the cost of corrections [31]. It is generally accepted that the testing cost consumed in a module M_{jk} is related to its reliability r_{jk}, and the higher reliability is required, the more testing cost will be consumed [32]. As a consequence, the possible testing cost of module M_{jk} is usually defined as [18], [32]

$$C_{jk}[r_{jk}(\lambda|t_{jk})] = c_1^{jk} \cdot \exp[c_2^{jk} \cdot r_{jk}(\lambda|t_{jk}) - c_3^{jk}]$$

where c_1^{jk}, c_2^{jk}, and c_3^{jk} control the increment rate of the testing cost corresponding to the reliability of M_{jk}. According to this, the possible total testing cost consumed by the system can be calculated by

$$C = \sum_{j=1}^{m}\sum_{k=1}^{n_j} C_{jk}[r_{jk}(\lambda|t_{jk})]$$

Recall the following circumstance from [18], [21]. The cycle of a software development process has become shorter because of the user request. Then, software testers have to shorten the testing phase, even if the system reliability may decrease slightly. In such a case, the total testing time expenditure should be considered. The actual consumed testing time of the system is the total amount of testing time consumed by all the modules in m serial subsystems [7], [33], that is,

$$T = \sum_{j=1}^{m}\sum_{k=1}^{n_j} t_{jk}$$

Taking reliability, cost, and time altogether, we describe the multi-objective optimization model for OTRAPs as the following tri-objective problem.

$$\begin{cases} \text{Maximize: } R \\ \text{Minimize: } C \\ \text{Minimize: } T \\ \text{Subject to:} \\ \sum\limits_{j=1}^{m}\sum\limits_{k=1}^{n_j} t_{jk} \leq T^* \\ t_{jk} \geq 0 \\ j = 1, \ldots, m \\ k = 1, \ldots, n_j \end{cases}$$

where $\sum_{j=1}^{m}\sum_{k=1}^{n_j} t_{jk} \leq T^*$ is the upper bound constraint and $t_{jk} \geq 0$ is the lower bound constraint. Note that here t_{jk} may be 0 because the scheduled T^* is limited and cannot satisfy the needs of all the modules. In addition, if t_{jk} and T^* are integers, the objective function R or C is just a nonlinear resource allocation problem which is NP-Hard [34]. Of course, when t_{jk} and T^* are real numbers, the above continuous multi-objective optimization model for OTRAPs also falls in the category of NP-Hard.

III. Improved NSGA-II for OTRAPs

There have been several attempts [17], [18] at using improved NSGA-II for OTRAPs. However, all such work tackles infeasibility in an arbitrary repair manner and uses unstandardized solution samples to evaluate the difference between solutions. Although there are modern MOEAs [35] in recent years, we prefer to improve the existing NSGA-II because of its simplicity and wider acceptance in many applied areas [36]. Moreover, it would be expected that the constraint handling proposed in this paper can also be used by alternative MOEAs. Specifically, we plan to enhance NSGA-II with more advanced genetic operators and statistical operations so that the improved NSGA-II can solve the constrained multi-objective OTRAPs effectively and efficiently.

For the purpose of illustration, we start by introducing the standard NSGA-II. Next, we show the constraint handling in population initialization, crossover operator, and mutation operator, respectively. After that, we investigate several fundamental properties of the constraint handling. Finally, we show how to use z-scores to standardize solution samples.

A. Conventional NSGA-II

The overall structure of the basic NSGA-II [23] is shown in Fig. 2. More specifically, the population is initialized on the basis of the problem range and constraints. Then, the current generation population is sorted fast according to non-domination levels into fronts. An individual is said to dominate

```
Create a random population of size N
Sort the population based on non-domination
Assign a rank for each non-dominated solution according to its non-domination level
Compute the crowding distance between any two neighbors in the population
repeat
    Select a parent population, P_t of size N from the new population
    Create an offspring, Q_t of size N by performing the crossover and mutation operations on P_t
    Create the mating pool G_t, of size 2 · N by combining P_t and Q_t
    Sort G_t according to the front rank and the crowding distance
    Select N best individuals from G_t to generate the new population P_{t+1}
until The maximum number of iterations is reached
```

Fig. 2. The psuedocode of the most basic NSGA-II.

```
τ ← 0
for j := 1 to m do
    for k := 1 to n_j do
        t_jk ← rand(0, T*)
        τ ← τ + t_jk
    end for
end for
if τ > T* then
    for j := 1 to m do
        for k := 1 to n_j do
            t̂_jk ← t_jk · rand(0, 1) · (T*/τ)
        end for
    end for
end if
```

Fig. 3. The heuristic for constraint handling in individual initialization.

another if the objective functions of this individual is no worse than the other and at least in one of the objective functions this individual is better than the other. In addition to the front rank, the crowding distance is calculated for each individual on the basis of the objective functions to measure how close this individual is to its neighbors, which aims to effectively maintain diversity and spread of solutions. To build up the parent population, the individuals are selected by using a binary tournament selection on the basis of the assigned front rank and the crowding distance. The selected parent population generates an offspring population after the operations of simulated binary crossover and polynomial mutation. Thereafter, the new population which consists of the current generation population and the offspring population is sorted again according to the front rank and the crowding distance. Only the best individuals in the combined population are selected to guarantee elitism and maintained to create the next generation. The algorithm repeats the above steps to precede the population's evolution until the maximum number of generations to decide termination is reached.

B. Heuristics for Constraint Handling

In constrained multi-objective optimization, the feasibility rules, which strongly favor feasible solutions, are very popular because of their simplicity and flexibility. The feasibility rules are very suitable to be coupled to any sort of selection mechanism relatively easily, but they may cause premature convergence [37], [38]. There are also other studies in which good infeasible solutions in the population are kept to preserve diversity, such as the classical Infeasibility Driven Evolutionary Algorithm (IDEA) [39], [40]. Nonetheless, the replacement process of IDEA requires to calculate the proportion of infeasible solutions to remain in the population for the next generation. Moreover, an additional optimization objective which consists on the constraint violation measure should be added. Thus, this approach will significantly increase the computation pressure.

For multi-objective optimization approaches to OTRAPs, the feasibility rules would be preferred. This is due to the fact that the multi-objective optimization approaches produce a solution set rather than a single solution. In the solution set, there exit a large amount of solutions which have low reliability or high cost. These unacceptable solutions are not available at all for OTRAPs, even if they are feasible. That is, the available region in the whole search space of OTRAPs is very tiny. Those impracticable solutions are equivalent to being "infeasible" for OTRAPs. For this reason, we recommend that all the infeasible solutions for OTRAPs should be repaired by constraint handling. It is expected to create more available solutions in practice so as to match with the original purpose of multi-objective approaches to OTRAPs.

In NSGA-II, the potential constraint violations may exit in the process of papulation initialization, crossover operator, and mutation operator. When handling the violations, the existing method HaD-MOEA [18] and WNS-MODE [21] changed all the element values in a solution by random reduction. These arbitrary repairs, no doubt, may destroy the important parent information inherited and retained in an individual, and alter the original evolutionary tendency of the individual. Furthermore, the changes of all the element values are prone to reduce the search ability of the algorithm to a simple, random walk and prevent population from converging into the optimal region [41]. Hence, in the next three subsections, we will try to answer the following questions: How to maintain the evolutionary tendency in constraint handling? How to preserve the parent information in individual repairs? And how to achieve a good balance between convergence and diversity in individual repairs?

1) Population Initialization: We use the following one-dimensional real value encoding to represent a possible testing-time allocation scheme.

$$\overbrace{[t_{11}\dots t_{1n_1}}^{S_1} \quad \dots \quad \overbrace{t_{j1}\dots t_{jn_j}}^{S_j} \quad \dots \quad \overbrace{t_{m1}\dots t_{mn_m}}^{S_m}]$$

where each partition represents a subsystem S_j and each element denotes the possible testing time t_{jk} invested in module M_{jk}.

The basic idea of population initialization with constraint handling is shown in Fig. 3, where $rand(0, T^*)$ represents a random number in $(0, T^*)$ and $rand(0, 1)$ denotes a random number in $(0, 1)$. Both $rand(0, T^*)$ and $rand(0, 1)$ are generated from a normal distribution. As Fig. 3 indicates, if the total amount of the testing time invested in all the modules is bigger than T^*, each t_{jk} is correspondingly scaled down to ensure that the initialized individual is feasible.

2) Simulated Binary Crossover: The crossover in NSGA-II selects genes randomly from two parents and produces two new offspring combining the information of the two parents.

Assume that p_1 and p_2 are the selected parents, $t_{jk}^{p_1}$ and $t_{jk}^{p_2}$ are two crossover-gene values in p_1 and p_2, o_1 and o_2 are the produced offspring from p_1 and p_2, and $t_{jk}^{o_1}$ and $t_{jk}^{o_2}$ are two gene values at the same position jk in p_1 and p_2, respectively. Then, the simulated binary crossover can be given as below [42]:

$$\begin{cases} t_{jk}^{o_1} \leftarrow \bar{y} - 0.5 \cdot \beta \cdot (y_2 - y_1) \\ t_{jk}^{o_2} \leftarrow \bar{y} + 0.5 \cdot \beta \cdot (y_2 - y_1) \end{cases}$$

where

$$\begin{cases} y_2 = \max\{t_{jk}^{p_1}, t_{jk}^{p_2}\} \\ y_1 = \min\{t_{jk}^{p_1}, t_{jk}^{p_2}\} \end{cases}$$

and

$$\bar{y} = 0.5 \cdot (y_1 + y_2) = 0.5 \cdot (t_{jk}^{p_1} + t_{jk}^{p_2})$$

From the above, we can obtain

$$0.5 \cdot (t_{jk}^{o_1} + t_{jk}^{o_2}) = \bar{y} = 0.5 \cdot (t_{jk}^{p_1} + t_{jk}^{p_2})$$

This is known as having the average property in which the average of the decoded parameter values is the same before and after the crossover operation [42]. Besides, $\beta \geq 0$ is called the spread factor and is a randomly generated number having the density that the probability of occurrence of $\beta \approx 1$ is more likely than any other β value.

The relationship among parents, the spread factor, and children is dedicated in Fig. 4. As can be seen, when $0 \leq \beta < 1$, $t_{jk}^{o_1}$ is enlarged relative to y_1 and $t_{jk}^{o_2}$ is reduced relative to y_2. Because y_1 and y_2 are feasible, both $t_{jk}^{o_1}$ and $t_{jk}^{o_2}$ are feasible. When $\beta = 1$, $t_{jk}^{o_1} = y_1$ and $t_{jk}^{o_2} = y_2$, it is clear that $t_{jk}^{o_1}$ and $t_{jk}^{o_2}$ are both feasible. When $\beta > 1$, $t_{jk}^{o_1}$ is shrunk relative to y_1 and $t_{jk}^{o_2}$ is enlarged relative to y_2. As a result, $t_{jk}^{o_1}$ may violate its lower bound constraint or $t_{jk}^{o_2}$ may violate its upper bound constraint. It should be noted that no matter whether $t_{jk}^{o_1} < 0$ or $t_{jk}^{o_2} > T^*$, when we plan to repair $t_{jk}^{o_1}$ and $t_{jk}^{o_2}$, we must first consider whether adjusting one of the children may make the other violate its constraints. For this purpose, see Fig. 4, we select the minimum interval from $(0, y_1)$ and (y_2, T^*) to deal with the repairs of offspring. Specifically, if $y_1 \leq T^* - y_2$, we first repair $t_{jk}^{o_1}$ in a random manner and then revise $t_{jk}^{o_2}$ according to the average property, namely,

$$\begin{cases} \hat{t}_{jk}^{o_1} \leftarrow rand(0, y_1) \\ \hat{t}_{jk}^{o_2} \leftarrow y_1 + y_2 - \hat{t}_{jk}^{o_1} \end{cases} \tag{1}$$

where $rand(0, y_1)$ represents a random number in $(0, y_1)$ with a normal distribution. Similarly, if $y_1 > T^* - y_2$, we repair $t_{jk}^{o_2}$ and $t_{jk}^{o_1}$ on the basis of the same idea:

$$\begin{cases} \hat{t}_{jk}^{o_2} \leftarrow rand(y_2, T^*) \\ \hat{t}_{jk}^{o_1} \leftarrow y_1 + y_2 - \hat{t}_{jk}^{o_2} \end{cases} \tag{2}$$

where $rand(y_2, T^*)$ denotes a random number in (y_2, T^*) with a normal distribution.

In addition to the above, another aspect we should not neglect is that in Fig. 4, $t_{jk}^{o_1}$ or $t_{jk}^{o_2}$ may be enlarged relative to y_1 or y_2, so the sum of gene values in o_1 or o_2 may violate the upper bound constraint T^*, and then o_1 or o_2 is also infeasible. As a consequence, we still need to check the feasibility of o_1 and o_2. For the convenience of presentation, we give the following notations. τ^{p_1}, τ^{p_2}, τ^{o_1}, and τ^{o_2} are the corresponding sum of all the gene values in the four individuals p_1, p_2, o_1, and o_2, respectively.

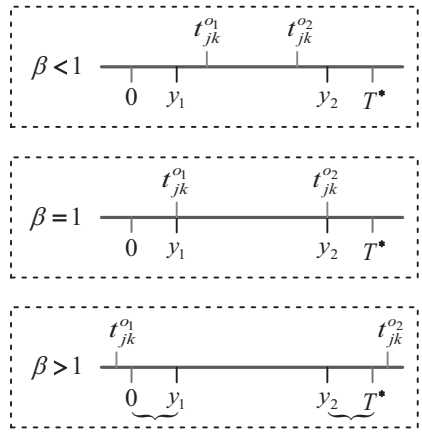


Fig. 4. The relationship among parents, the spread factor, and offspring.

Before checking τ^{o_1} and τ^{o_2}, two important aspects should be considered. First of all, if $\tau^{o_1} > T^*$, it is certain that $\tau^{o_2} < T^*$ because of the average property

$$\tau^{o_1} + \tau^{o_2} = \tau^{p_1} + \tau^{p_2} \leq 2 \cdot T^* \tag{3}$$

Similarly, if $\tau^{o_2} > T^*$, we have $\tau^{o_1} < T^*$. Consequently, if the upper bound constraint is violated, we just need to repair one of o_1 and o_2 rather than both o_1 and o_2. What is more, in the process of crossover, only the values of the selected crossover genes in an individual will change. It follows that all the infeasible cases are caused only by the crossover genes, but are irrelevant to other non-crossover genes. Of course, all the repair operations must be done only on the crossover genes to preserve the parent information in offspring. Based on this observation, we introduce some new notations. $\tau_\epsilon^{o_1}$ and $\tau_\epsilon^{o_2}$ are the sums of all the crossover-gene values in o_1 and o_2, respectively. $T_\epsilon^{o_1}$ and $T_\epsilon^{o_2}$ are the total amounts of the testing time which can be allocated to the crossover genes in o_1 and o_2, respectively, satisfying

$$\begin{cases} T_\epsilon^{o_1} = T^* - (\tau^{o_1} - \tau_\epsilon^{o_1}) \\ T_\epsilon^{o_2} = T^* - (\tau^{o_2} - \tau_\epsilon^{o_2}) \end{cases} \tag{4}$$

η is a scale factor, satisfying $0 \leq \eta \leq 1$, and is used to control the change amplitude of $t_{jk}^{o_1}$ or $t_{jk}^{o_1}$.

Now, if $\tau^{o_1} > T^*$, we have $\tau_\epsilon^{o_1} > T_\epsilon^{o_1}$. According to the average property, the values of the selected crossover genes in o_1 and o_2 can be repaired as follows:

$$\begin{cases} \hat{t}_{jk}^{o_1} \leftarrow t_{jk}^{o_1} \cdot \eta \cdot (T_\epsilon^{o_1} / \tau_\epsilon^{o_1}) \\ \hat{t}_{jk}^{o_2} \leftarrow t_{jk}^{o_1} + t_{jk}^{o_2} - \hat{t}_{jk}^{o_1} \end{cases}$$

where we use η to reduce $t_{jk}^{o_1}$. Admittedly, if each crossover-gene value in o_1 is decreased to too small numbers, the sum of crossover-gene values in the repaired o_2 may inevitably be

enlarged excessively because of the average property. Hence, it is possible that the repaired o_2 may violate the upper bound constraint. Taking the above factor into consideration, we need to figure out the value range of η. For the repaired o_2, the new sum of all the gene values is

$$\hat{\tau}^{o_2} = \tau^{o_2} - \tau_\epsilon^{o_2} + \hat{\tau}_\epsilon^{o_2}$$

where $\hat{\tau}_\epsilon^{o_2}$ is the new sum of all the crossover-gene values in the repaired o_2, satisfying

$$\hat{\tau}_\epsilon^{o_2} = \tau_\epsilon^{o_1} + \tau_\epsilon^{o_2} - \hat{\tau}_\epsilon^{o_1}$$

$\hat{\tau}_\epsilon^{o_1}$ is the new sum of all the crossover-gene values in the repaired o_1 and can be calculated as follows.

$$\begin{aligned}\hat{\tau}_\epsilon^{o_1} &= \sum \left[t_{jk}^{o_1} \cdot \eta \cdot (T_\epsilon^{o_1}/\tau_\epsilon^{o_1})\right] \\ &= \eta \cdot (T_\epsilon^{o_1}/\tau_\epsilon^{o_1}) \cdot \sum t_{jk}^{o_1} \\ &= \eta \cdot (T_\epsilon^{o_1}/\tau_\epsilon^{o_1}) \cdot \tau_\epsilon^{o_1} \\ &= \eta \cdot T_\epsilon^{o_1}\end{aligned}$$

Accordingly,

$$\begin{aligned}\hat{\tau}^{o_2} &= \tau^{o_2} - \tau_\epsilon^{o_2} + \tau_\epsilon^{o_1} + \tau_\epsilon^{o_2} - \eta \cdot T_\epsilon^{o_1} \\ &= \tau^{o_2} + \tau_\epsilon^{o_1} - \eta \cdot T_\epsilon^{o_1}\end{aligned}$$

To ensure $\hat{\tau}^{o_2} \leq T^*$, the following should hold:

$$\tau^{o_2} + \tau_\epsilon^{o_1} - \eta \cdot T_\epsilon^{o_1} \leq T^*$$

According to this, we have

$$\eta \geq (\tau^{o_2} + \tau_\epsilon^{o_1} - T^*)/T_\epsilon^{o_1}$$

Note that if $\tau^{o_2} + \tau_\epsilon^{o_1} - T^* \leq 0$, we set $\eta \leftarrow 0$ to ensure $\hat{t}_{jk}^{o_1} \geq 0$. If $\tau^{o_2} + \tau_\epsilon^{o_1} - T^* > 0$, we know that the crossover-gene values in o_1 cannot be reduced too small. Thus, we set the scale factor η to be $\max\{0, (\tau^{o_2} + \tau_\epsilon^{o_1} - T^*)/T_\epsilon^{o_1}\}$ to control the reduction degree of $t_{jk}^{o_1}$. Overall, we can rewrite the repair operations as follows:

$$\begin{cases} \eta \leftarrow \max\{0, (\tau^{o_2} + \tau_\epsilon^{o_1} - T^*)/T_\epsilon^{o_1}\} \\ \hat{t}_{jk}^{o_1} \leftarrow t_{jk}^{o_1} \cdot rand(\eta, 1) \cdot (T_\epsilon^{o_1}/\tau_\epsilon^{o_1}) \\ \hat{t}_{jk}^{o_2} \leftarrow t_{jk}^{o_1} + t_{jk}^{o_2} - \hat{t}_{jk}^{o_1} \end{cases} \tag{5}$$

where $rand(\eta, 1)$ denotes a random number in $(\eta, 1)$ which is drawn from a normal distribution. Here we use $rand(\eta, 1)$ to promote population diversity.

Similarly, if $\tau^{o_2} > T^*$, we have $\tau^{o_1} < T^*$. Then, o_2 and o_1 can be repaired as below:

$$\begin{cases} \eta \leftarrow \max\{0, (\tau^{o_1} + \tau_\epsilon^{o_2} - T^*)/T_\epsilon^{o_2}\} \\ \hat{t}_{jk}^{o_2} \leftarrow t_{jk}^{o_2} \cdot rand(\eta, 1) \cdot (T_\epsilon^{o_2}/\tau_\epsilon^{o_2}) \\ \hat{t}_{jk}^{o_1} \leftarrow t_{jk}^{o_1} + t_{jk}^{o_2} - \hat{t}_{jk}^{o_2} \end{cases} \tag{6}$$

The procedure of constraint handling based simulated binary crossover is presented in Fig. 5, where at most one of the situations $\tau^{o_1} > T^*$ or $\tau^{o_2} > T^*$ will occur.

```
Choose two parents p_1 and p_2
τ^{o_1} ← 0, τ^{o_2} ← 0, τ_ε^{o_1} ← 0, τ_ε^{o_2} ← 0, T_ε^{o_1} ← 0, T_ε^{o_2} ← 0
for j := 1 to m do
    for k := 1 to n_j do
        if jk is selected as a crossover gene then
            y_2 ← max{t_jk^{p_1}, t_jk^{p_2}}, y_1 ← min{t_jk^{p_1}, t_jk^{p_2}}, ȳ = 0.5·(y_1+y_2)
            Generate a random number β
            t_jk^{o_1} ← ȳ − 0.5 · β · (y_2 − y_1), t_jk^{o_2} ← ȳ + 0.5 · β · (y_2 − y_1)
            if t_jk^{o_1} < 0 || t_jk^{o_2} > T* then
                if y_1 ≤ T* − y_2 then
                    t_jk^{o_1} ← rand(0, y_1), t_jk^{o_2} ← y_1 + y_2 − t_jk^{o_1}
                else
                    t_jk^{o_2} ← rand(y_2, T*), t_jk^{o_1} ← y_1 + y_2 − t_jk^{o_2}
                end if
            end if
            τ_ε^{o_1} ← τ_ε^{o_1} + t_jk^{o_1}, τ_ε^{o_2} ← τ_ε^{o_2} + t_jk^{o_2}
        else
            t_jk^{o_1} ← t_jk^{p_1}, t_jk^{o_2} ← t_jk^{p_2}
        end if
        τ^{o_1} ← τ^{o_1} + t_jk^{o_1}, τ^{o_2} ← τ^{o_2} + t_jk^{o_2}
    end for
end for
if τ^{o_1} > T* then
    T_ε^{o_1} ← T* − (τ^{o_1} − τ_ε^{o_1}), η ← max{0, (τ^{o_2} + τ_ε^{o_1} − T*)/T_ε^{o_1}}
    for j := 1 to m do
        for k := 1 to n_j do
            if jk is a crossover gene then
                t̂_jk^{o_1} ← t_jk^{o_1} · rand(η, 1) · (T_ε^{o_1}/τ_ε^{o_1}), t̂_jk^{o_2} ← t_jk^{o_1} + t_jk^{o_2} − t̂_jk^{o_1}
            end if
        end for
    end for
end if
if τ^{o_2} > T* then
    T_ε^{o_2} ← T* − (τ^{o_2} − τ_ε^{o_2}), η ← max{0, (τ^{o_1} + τ_ε^{o_2} − T*)/T_ε^{o_2}}
    for j := 1 to m do
        for k := 1 to n_j do
            if jk is a crossover gene then
                t̂_jk^{o_2} ← t_jk^{o_2} · rand(η, 1) · (T_ε^{o_2}/τ_ε^{o_2}), t̂_jk^{o_1} ← t_jk^{o_1} + t_jk^{o_2} − t̂_jk^{o_2}
            end if
        end for
    end for
end if
```

Fig. 5. The simulated binary crossover operator with constraint handling.

3) Polynomial Mutation: The mutation in NSGA-II changes genes randomly from a parent to produce a new offspring. Assume that p is the selected parent, t_{jk}^p is the value of the selected mutation gene jk in p, o is the produced offspring from p, and t_{jk}^o is the gene value at the same position in o. Then, the polynomial mutation is performed as below [42]:

$$t_{jk}^o \leftarrow t_{jk}^p + \delta \cdot T^*$$

where $\delta \in (-1, 1)$ is a random number from a polynomial distribution.

It can be observed that t_{jk}^o may be reduced or enlarged relative to t_{jk}^p. First, if $t_{jk}^o < 0$, to maintain the trend of decreasing t_{jk}^p, we generate a random positive number from $(0, t_{jk}^p)$ to replace the old t_{jk}^o. That is,

$$\hat{t}_{jk}^o \leftarrow rand(0, t_{jk}^p) \tag{7}$$

where $rand(0, t_{jk}^p)$ is a random number in $(0, t_{jk}^p)$ with a normal distribution. Second, if $t_{jk}^o > T^*$, to maintain the

```
Choose a parent p
τ^o ← 0, τ^o_ε ← 0, T^o_ε ← 0
for j := 1 to m do
    for k := 1 to n_j do
        if jk is selected as a mutation gene then
            Generate a random number δ
            t^o_jk ← t^p_jk + δ · T*
            if t^o_jk < 0 then
                t^o_jk ← rand(0, t^p_jk)
            end if
            if t^o_jk > T* then
                t^o_jk ← rand(t^p_jk, T*)
            end if
            τ^o_ε ← τ^o_ε + t^o_jk
        else
            t^o_jk ← t^p_jk
        end if
        τ^o ← τ^o + t^o_jk
    end for
end for
if τ^o > T* then
    T^o_ε ← T* − (τ^o − τ^o_ε)
    for j := 1 to m do
        for k := 1 to n_j do
            if jk is a mutation gene then
                t̂^o_jk ← t^o_jk · rand(0, 1) · (T^o_ε / τ^o_ε)
            end if
        end for
    end for
end if
```

Fig. 6. The polynomial mutation operator with constraint handling.

enlargement tendency of t^p_{jk}, we create a random positive number from (t^p_{jk}, T^*) as a substitute for the previous t^o_{jk}, namely,

$$\hat{t}^o_{jk} \leftarrow rand(t^p_{jk}, T^*) \tag{8}$$

where $rand(t^p_{jk}, T^*)$ is a random number in (t^p_{jk}, T^*) with a normal distribution.

Even if each t^o_{jk} in o is feasible, it is possible that o may violate the upper bound constraint T^*. This is due to the fact that some mutation-gene values may be enlarged too much. Denote τ^o as the sum of all the gene values in o. When $\tau^o > T^*$, we are in line with the basic principle that we only repair the selected mutation-gene values rather than all the gene values to preserve the parent characteristics in offspring. For further discussions, we give the following notations. τ^o_ϵ is the sum of all the mutation-gene values in o. T^o_ϵ is the total amount of the testing time which can be allocated to the mutation genes, satisfying

$$T^o_\epsilon = T^* - (\tau^o - \tau^o_\epsilon) \geq 0 \tag{9}$$

Now, if $\tau^o > T^*$, we reduce all the mutation-gene values in o according to

$$\hat{t}^o_{jk} \leftarrow t^o_{jk} \cdot rand(0,1) \cdot (T^o_\epsilon / \tau^o_\epsilon) \tag{10}$$

where $rand(0,1)$ is a random coefficient in $(0,1)$ with a normal distribution and is used to promote diversity.

The detailed procedure of the polynomial mutation with constraint handling is presented in Fig. 6.

Generally speaking, the hurdles in solving OTRAPs arise from the challenge of searching a huge space to locate feasible solutions with acceptable allocation schemes. As mentioned earlier, the reason for this challenge is that for OTRAPs, the feasible and acceptable space is very tiny compare to the whole search space. The above heuristics for constraint handling are designed to keep every individual in the population as feasible as possible. This feasibility rule under tiny feasible space was found to be useful for the search algorithms in [18] and [21]. More specifically, to ensure the quality of the initial solutions, the first heuristic shown in Fig. 3 directs the selected infeasible individuals in the initial population to move to the feasible space randomly. Note that the feasibility of the initial solutions is a basic prerequisite to the evolution of the population, without which the subsequent crossover and mutation operators will be invalidated. The heuristic shown in Fig. 5 ensures the feasibility of each offspring after the crossover operator, which makes the mutation operator workable. Similarly, the heuristic shown in Fig. 6 ensures the feasibility of each offspring after the mutation operator and makes the repeated crossover operator workable. It is clear that the above heuristics in the crossover and mutation operators are interdependent, without one of which the other will be painful and ineffective. The mutual assistance and mutual benefit make the whole population always evolve in tiny feasible space during iterations. Additionally, the two heuristics maintain the previous evolutionary tendency of individuals on the basis of (1), (2), (7), and (8). Moreover, the randomization strategies in (5), (6), and (10) are introduced to promote population diversity.

C. Properties of Constraint Handling

To begin with, we prove that the new $\hat{t}^{o_1}_{jk}$ and $\hat{t}^{o_2}_{jk}$ generated by (1) and (2) are feasible and also maintain the previous evolutionary trend. That is, $\hat{t}^{o_1}_{jk}$ and $\hat{t}^{o_2}_{jk}$ satisfy the constraints and can still enclose the values of the two parents.

Theorem 1: No matter whether $t^{o_1}_{jk} < 0$ or $t^{o_2}_{jk} > T^*$, the new $\hat{t}^{o_1}_{jk}$ and $\hat{t}^{o_2}_{jk}$ in (1) and (2) satisfy

$$0 \leq \hat{t}^{o_1}_{jk} \leq y_1 < y_2 \leq \hat{t}^{o_2}_{jk} \leq T^*$$

Proof: In (1) and (2), as long as $t^{o_1}_{jk} < 0$ or $t^{o_2}_{jk} > T^*$, we repair offspring on the basis of the intervals $(0, y_1)$ and (y_2, T^*). Hence, we will prove the following cases.

Case 1: $y_1 \leq T^* - y_2$. From (1), we can easily obtain

$$0 \leq \hat{t}^{o_1}_{jk} \leq y_1 < y_2 \leq \hat{t}^{o_2}_{jk}$$

Considering (1) and $y_1 \leq T^* - y_2$ altogether,

$$\hat{t}^{o_2}_{jk} \leq T^* - \hat{t}^{o_1}_{jk}$$

Because $\hat{t}^{o_1}_{jk} \geq 0$, we have

$$\hat{t}^{o_2}_{jk} \leq T^*$$

Case 2: $y_1 > T^* - y_2$. According to (2), we can easily find that

$$\hat{t}^{o_1}_{jk} \leq y_1 < y_2 \leq \hat{t}^{o_2}_{jk} \leq T^*$$

and

$$\hat{t}_{jk}^{o_1} = y_1 + y_2 - rand(y_2, T^*)$$

Recalling $y_1 > T^* - y_2$, we have

$$\hat{t}_{jk}^{o_1} > T^* - rand(y_2, T^*)$$

Since $T^* - rand(y_2, T^*) \geq 0$, then

$$\hat{t}_{jk}^{o_1} \geq 0$$

■

Similarly, from (7) and (8), it can be directly observed that the new $\hat{t}_{jk}^{o}$ satisfies the constraints and maintains the original evolutionary characteristics.

Theorem 2: The new $\hat{t}_{jk}^{o}$ in (7) has

$$0 \leq \hat{t}_{jk}^{o} \leq t_{jk}^{p} \leq T^*$$

Theorem 3: The new $\hat{t}_{jk}^{o}$ in (8) satisfies

$$t_{jk}^{p} \leq \hat{t}_{jk}^{o} \leq T^*$$

Next, we prove that the scale factor η created in (5) or (6) is in $[0, 1]$.

Theorem 4: The scale factor η in (5) satisfies $0 \leq \eta \leq 1$.

Proof: We show the implications separately. On one hand, in (5), if $\tau^{o_2} + \tau_\epsilon^{o_1} - T^* \leq 0$, we can easily obtain

$$\eta = 0$$

On the other hand, if $\tau^{o_2} + \tau_\epsilon^{o_1} - T^* > 0$, we have $\eta > 0$ because of $T_\epsilon^{o_1} > 0$. Now, we need to prove $\eta \leq 1$. For convenience, here we give a proof by contradiction. We assume that $\eta > 1$, then

$$\tau^{o_2} + \tau_\epsilon^{o_1} - T^* > T_\epsilon^{o_1}$$

Substituting (4) into the above inequality, we can obtain

$$\tau^{o_1} + \tau^{o_2} > 2 \cdot T^*$$

which is inconsistent with (3), so we have

$$\eta \leq 1$$

■

Theorem 5: The scale factor η in (6) is also in $[0, 1]$.

Proof: The result can be proved in the same way as shown in Theorem 4. ■

In addition, we prove that, in (5) or (6), no matter how greatly the crossover-gene values in o_1 or o_2 have been decreased, the sum of all the gene values in the repaired o_2 or o_1 will not go beyond T^*. We also prove that the new o repaired according to (10) satisfies the constraint.

Theorem 6: If $\tau^{o_1} > T^*$, the new o_2 created by (5) satisfies $\hat{\tau}^{o_2} \leq T^*$.

Proof: As mentioned earlier,

$$\begin{aligned}\hat{\tau}^{o_2} &= \tau^{o_2} - \tau_\epsilon^{o_2} + \hat{\tau}_\epsilon^{o_2} \\ &= \tau^{o_2} - \tau_\epsilon^{o_2} + (\tau_\epsilon^{o_1} + \tau_\epsilon^{o_2} - \hat{\tau}_\epsilon^{o_1}) \\ &= \tau^{o_2} + \tau_\epsilon^{o_1} - \hat{\tau}_\epsilon^{o_1}\end{aligned}$$

In (5), if $\eta = 0$, we certainly have

$$\tau^{o_2} + \tau_\epsilon^{o_1} - T^* \leq 0$$

Thus,

$$\hat{\tau}^{o_2} \leq T^* - \hat{\tau}_\epsilon^{o_1}$$

Since $\hat{\tau}_\epsilon^{o_1} \geq 0$, then

$$\hat{\tau}^{o_2} \leq T^*$$

If $0 < \eta \leq 1$, from (5) we have

$$\eta = (\tau^{o_2} + \tau_\epsilon^{o_1} - T^*)/T_\epsilon^{o_1}$$

In (5), each $\hat{t}_{jk}^{o_1}$ has

$$\hat{t}_{jk}^{o_1} \geq t_{jk}^{o_1} \cdot \eta \cdot (T_\epsilon^{o_1}/\tau_\epsilon^{o_1})$$

Hence,

$$\sum \hat{t}_{jk}^{o_1} \geq \sum [t_{jk}^{o_1} \cdot \eta \cdot (T_\epsilon^{o_1}/\tau_\epsilon^{o_1})]$$

Since

$$\sum \hat{t}_{jk}^{o_1} = \hat{\tau}_\epsilon^{o_1}$$

and

$$\begin{aligned}\sum [t_{jk}^{o_1} \cdot \eta \cdot (T_\epsilon^{o_1}/\tau_\epsilon^{o_1})] &= \eta \cdot (T_\epsilon^{o_1}/\tau_\epsilon^{o_1}) \cdot \sum t_{jk}^{o_1} \\ &= \eta \cdot (T_\epsilon^{o_1}/\tau_\epsilon^{o_1}) \cdot \tau_\epsilon^{o_1} \\ &= \eta \cdot T_\epsilon^{o_1} \\ &= (\tau^{o_2} + \tau_\epsilon^{o_1} - T^*)/T_\epsilon^{o_1} \cdot T_\epsilon^{o_1} \\ &= \tau^{o_2} + \tau_\epsilon^{o_1} - T^*\end{aligned}$$

then

$$\hat{\tau}_\epsilon^{o_1} \geq \tau^{o_2} + \tau_\epsilon^{o_1} - T^*$$

That is,

$$\tau^{o_2} + \tau_\epsilon^{o_1} - \hat{\tau}_\epsilon^{o_1} \leq T^*$$

Recalling $\hat{\tau}^{o_2} = \tau^{o_2} + \tau_\epsilon^{o_1} - \hat{\tau}_\epsilon^{o_1}$, we have

$$\hat{\tau}^{o_2} \leq T^*$$

■

Theorem 7: If $\tau^{o_1} > T^*$, the new o_1 created by (5) satisfies $\hat{\tau}^{o_1} \leq T^*$.

Proof: In light of the average property, we have

$$\hat{\tau}^{o_1} + \hat{\tau}^{o_2} = \tau^{o_1} + \tau^{o_2}$$

From (3), we obtain

$$\hat{\tau}^{o_1} + \hat{\tau}^{o_2} \leq 2 \cdot T^*$$

From Theorem 6, it is easily obtained that

$$\hat{\tau}^{o_1} \leq T^*$$

■

Theorem 8: If $\tau^{o_2} > T^*$, the new o_1 created by (6) satisfies $\hat{\tau}^{o_1} \leq T^*$.

Proof: The proof is similar to Theorem 6. ■

Theorem 9: If $\tau^{o_2} > T^*$, the new o_2 created by (6) satisfies $\hat{\tau}^{o_2} \leq T^*$.

Proof: The proof is similar to Theorem 7. ■

Theorem 10: If $\tau^{o} > T^*$, the new o created by (10) satisfies $\hat{\tau}^{o} \leq T^*$.

Proof: The sum of all the gene values in the new o is

$$\hat{\tau}^{o} = \tau^{o} - \tau_\epsilon^{o} + \hat{\tau}_\epsilon^{o}$$

From (10), each $\hat{t}^o_{jk}$ has

$$\hat{t}^o_{jk} \leq t^o_{jk} \cdot (T^o_\epsilon / \tau^o_\epsilon)$$

Thus,

$$\sum \hat{t}^o_{jk} \leq \sum [t^o_{jk} \cdot (T^o_\epsilon / \tau^o_\epsilon)]$$

Since

$$\sum \hat{t}^o_{jk} = \hat{\tau}^o_\epsilon$$

and

$$\begin{aligned} \sum [t^o_{jk} \cdot (T^o_\epsilon / \tau^o_\epsilon)] &= T^o_\epsilon / \tau^o_\epsilon \cdot \sum t^o_{jk} \\ &= T^o_\epsilon / \tau^o_\epsilon \cdot \tau^o_\epsilon \\ &= T^o_\epsilon \end{aligned}$$

then

$$\hat{\tau}^o_\epsilon \leq T^o_\epsilon$$

Therefore,

$$\hat{\tau}^o \leq \tau^o - \tau^o_\epsilon + T^o_\epsilon$$

Substituting (9) into the above inequality, we have

$$\hat{\tau}^o \leq T^*$$

■

Last, we prove that the proposed heuristics for constraint handling is computationally simple, considering the computational complexity to be the number of the required operations with different values of m and n_j.

Theorem 11: The worst case complexity of the individual initialization shown in Fig. 3 is $\Theta(n)$.

Proof: In Fig. 3, it is clear that there are a total of $\sum_{j=1}^m n_j$ gene values in an individual, so the number of operations required to generate an individual is $\sum_{j=1}^m n_j$. If the sum of all the gene values is bigger than the upper bound constraint, all the gene values will be changed, implying that the number of repair operations is also $\sum_{j=1}^m n_j$. In summary, the total number of operations is between $\sum_{j=1}^m n_j$ and $2 \cdot \sum_{j=1}^m n_j$. Hence, the worst case complexity of the individual initialization in Fig. 3 is $\Theta(n)$. ■

Theorem 12: The worst case complexity of the simulated binary crossover shown in Fig. 5 is $\Theta(n)$.

Proof: In Fig. 5, no matter whether an element is selected as a crossover gene, all the $\sum_{j=1}^m n_j$ gene values in an offspring must be created from the two parents. Thus, the number of operations required to generate two offspring is just $2 \cdot \sum_{j=1}^m n_j$. For the two offspring, it is possible that each gene value may violate the lower bound constraint or the upper bound constraint, which results in $2 \cdot \sum_{j=1}^m n_j$ repair operations. Worse, the sum of all the gene values in an offspring would go beyond T^*, and thus the two offspring have to be repaired again. To sum up, the total number of operations is between $2 \cdot \sum_{j=1}^m n_j$ and $6 \cdot \sum_{j=1}^m n_j$. Accordingly, the worst case complexity of the simulated binary crossover in Fig. 5 is $\Theta(n)$. ■

Theorem 13: The worst case complexity of the polynomial mutation shown in Fig. 6 is $\Theta(n)$.

Proof: In Fig. 6, the number of operations required to create an offspring from the selected parent is just $\sum_{j=1}^m n_j$. Moreover, it can be easily observed that the number of possible repair operations for feasibility is at most $2 \cdot \sum_{j=1}^m n_j$. On the whole, the total number of operations is between $\sum_{j=1}^m n_j$ and $3 \cdot \sum_{j=1}^m n_j$. It follows that the worst case complexity of the polynomial mutation in Fig. 6 is $\Theta(n)$. ■

D. Solution Sample Standardization

Before calculating the crowding distance between neighbors, we adopt the z-scores [27] in mathematical statistics to standardize the solution samples due to the different scales of R, C, and T. For a papulation of size N, there are N values, $x_1, \cdots, x_N$, for an objective in solution samples, and then each value x_ι ($\iota \in \{1, \ldots, N\}$) will be standardized as below:

$$x'_\iota \leftarrow \frac{x_\iota - \bar{x}}{s}$$

where:

$$\bar{x} = \frac{1}{N} \sum_{\iota=1}^{N} x_\iota$$

is the sample mean of the population, and

$$s = \sqrt{\frac{1}{N-1} \sum_{\iota=1}^{N} (x_\iota - \bar{x})^2}$$

is the standard deviation of the population. Note that the value of x'_ι represents the fluctuation between the raw score and the population mean in units of the standard deviation, so x'_ι may be negative when x_ι is below the mean.

Having the normalized solution samples in hand, we first compute the standardized Euclidean distance between any two different individuals on a nondominated front. Next, the two individuals that have the biggest Euclidean distance are set to the boundary points; if there are several pairs of individuals with the biggest Euclidean distance, set all of them to the boundary points. After that, we estimate the crowing degree of an individual with the harmonic mean distance proposed in [18].

Assume that the standardized Euclidean distances between an individual and its κ-nearest neighbors are $d_1, d_2, \ldots, d_\kappa$, respectively. If an individual is a boundary point, the crowing distance of this individual will be set to the maximum value; otherwise, the harmonic mean distance associated with this individual can be calculated by [18]

$$d = \frac{\kappa}{\frac{1}{d_1} + \frac{1}{d_2} + \ldots + \frac{1}{d_\kappa}}$$

It is known that d is able to reach the maximum value if and only if $d_1 = d_2 = \ldots = d_\kappa$ [43]. This means that when an individual and its neighbors are distributed more evenly, the individual will be assigned a bigger crowding distance. This is just in line with the diversity-maintenance goal that we aspire to achieve.

IV. Performance Evaluation

In this section, the proposed constraint handling based NSGA-II for OTRAPs (henceforth called NSGA-II-TRA) is compared with existing approaches to OTRAPs. For the sake of illustration, we first introduce the basic parameter settings

and the appropriate performance metrics. Next, we demonstrate the effectiveness of the constraint handling. Thirdly, we evaluate the z-score operation. After that, we draw a comparison between NSGA-II-TRA, HaD-MOEA [18], and WNS-MODE [21], which are all multi-objective optimization approaches to OTRAPs with different constraint handling techniques. Finally, we conduct sensitivity analysis with respect to the five modular parameters.

A. Parameter Settings and Performance Metrics

Four parallel-series modular software systems with gradual complexities are considered in our experiments: a simple system with 5 sub-systems and 10 modules, a complex system with 6 sub-systems and 14 modules, a large system with 7 sub-systems and 20 modules, and a larger system with 11 sub-systems and 30 modules, as shown in Fig. 7. Compared with the existing work [17], [18], [21], the given four systems in Fig. 7 are significantly large in problem size. This can help us evaluate the efficiency of different algorithms (in particular, the proposed NSGA-II-ERA) more comprehensively and draw a more general conclusion. Following the previous studies, we assume that $T^* = 50,000, \lambda = 50$, $T^* = 70,000, \lambda = 100$, $T^* = 85,000, \lambda = 200$, and $T^* = 150,000, \lambda = 200$ are available for testing the simple, complex, large, and larger systems, respectively.

The modular parameters are recalled from [18] and listed in Table I, where the serial module indicates that this module belongs to a subsystem that has only one module (e.g., M_{11} in Fig. 7(a)) and the parallel module represents that this module belongs to a subsystem with more than one module (e.g., M_{31} in Fig. 7(a)). For each system, we generated 30 different instances randomly from a normal distribution according to the pre-defined intervals in Table I. Besides, to make the comparisons as fair as possible, the baseline settings of all the algorithmic parameters that were recommended in [18], [21] are adopted and shown in Table II.

In [18], [21], the hypervolume indicator is used to evaluate the quality of the whole solution set. However, as mentioned earlier, most of the solutions in the solution set are impracticable and unavailable for OTRAPs. Consequently, measuring the quality of solutions mainly in terms of the valueless solutions that account for the vast majority of all the solutions is very one-sided and biased for OTRAPs. To this end, in this paper, we adopt the classical coverage metric [44] as the main performance measure to compare the quality of solutions obtained by different approaches. Assuming that A and B are the final solution sets achieved by two different MOEAs, a point a in A covers a point b in B if a is not worse than b on any objective. $\zeta(A, B)$ denotes the percentage of set B that is covered by points in A. $\zeta(A, B) > \zeta(B, A)$ is used to indicate a win for the algorithm that produces A. A series of such tests is counted as statistically significant if an algorithm is the winner suitably often [44]. Besides, OTRAPs are aiming at achieving software reliability for customer satisfaction. Thus, we also adopt the capacity measure [45], namely, the numbers of satisfactory non-dominated solutions. We evaluate the capacity and coverage values of the satisfactory solution sets, regarding solutions whose reliability is higher than the given threshold. Furthermore, good convergence, sometimes, may result in repeated testing allocation schemes or very similar schemes, which are redundant to software testers. Hence, we adopt the norm-based pure diversity metric to evaluate the accumulation of the dissimilarity in the solution set [46].

In our experiments, all the algorithms' codes are written in VISUAL C++ 2013. Each test instance is repeated for 30 independent runs with different random seeds on a PC with Intel 2.50 GHz CPU and 10 GB of RAM.

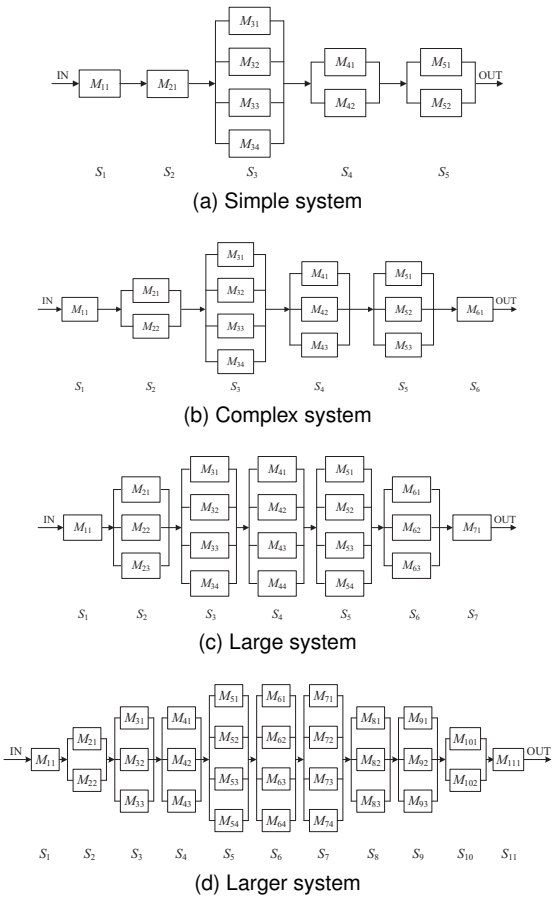


Fig. 7. Structure of the four parallel-series modular software systems.

TABLE I
MODULAR PARAMETERS IN ALL THE NUMERICAL EXPERIMENTS

Module	a_{jk}	b_{jk}	c_1^{jk}	c_2^{jk}	c_3^{jk}
Serial	[30.0, 35.0]	[5.8E-3, 6.2E-3]	[3.4, 3.55]	[6.0, 6.2]	[4.0, 4.1]
Parallel	[200.0, 350.0]	[3.0E-4, 9.0E-4]	[3.4, 3.55]	[6.0, 6.2]	[4.9, 5.1]

TABLE II
BASIC ALGORITHMIC PARAMETERS IN ALL THE NUMERICAL EXPERIMENTS

Parameter	NSGA-II	HaD-MOEA	WNS-MODE	NSGA-II-TRA
PS	200	200	200	200
NG	500	500	500	500
CP	0.9	0.9	0.1	0.9
MP/F	0.1	0.1	1.4	0.1

PS: population size. NG: number of generations. CP: crossover probability. MP: mutation probability. F: mutation factor.

B. Comparing Different Constraint Handling Techniques

The repair mechanisms in [18] and [21] (henceforth called Existing Strategy) execute the same repair operations to check the whole population after the crossover and mutation operators, respectively. On the contrary, the proposed constraint handling in this paper (henceforth called Our Strategy) is embedded in the whole process of the crossover and mutation operators. Moreover, the two heuristics (shown in Fig. 5 and Fig. 6) for constraint handling in the crossover and mutation operators are as intimate as kins, without one of which the other will lose its effects for the reason that the essential conditions for constraint handling in (3) or (9) may not hold. Accordingly, in this section, we compare the whole constraint handling schemes rather than each single evolutionary operator, using the standard NSGA-II shown in Fig. 2.

To measure the convergence ability, Tables III and IV show the coverage values of the standard NSGA-II on OTRAPs with different numbers of generations (i.e., NG), population size (i.e., PS), repair strategies, and systems, respectively, regarding the mean and standard deviation values. The better result regarding the mean value between the two compared strategies for each test instance is highlighted in boldface. We used the Wilcoxon rank-sum test [47] at a 0.05 significance level to measure the significance of the differences among the results obtained by the two strategies. It was found that all the differences between the two strategies are statistically significant. Besides, it can be observed that with the increase of NG or PS, the coverage values of Our Strategy vs Existing Strategy are much better than those of Existing Strategy vs Our Strategy on almost all the test instances. In particular, on most of the test instances for the complex, large, and larger systems, the coverage values of Existing Strategy vs Our Strategy are almost only half of the corresponding values of Our Strategy vs Existing Strategy. This observation suggests that Our Strategy should be able to keep a better convergence and obtain higher-quality testing-time allocation schemes.

For further illustrations of exploration ability, we select solutions with a high reliability ($R \geq 0.99$) in each test instance to evaluate whether Our Strategy can achieve higher reliability, lower cost, and less time. Tables V and VI show the capacity and coverage values of the two repair strategies, respectively, regarding solutions whose reliability is higher than or equal to 0.99 in all the 30 runs for each test instance. It can be seen that Our Strategy obtains much more satisfactory testing time allocation schemes than Existing Strategy, especially under small NG and PS. Additionally, the coverage values of Our Strategy vs Existing Strategy are much greater than those of Existing Strategy vs Our Strategy. On most of the test instances, the satisfactory solutions of Our Strategy can completely cover the solutions of Existing Strategy but not be covered by the latter at all. Our Strategy achieved a very high reliability under lower cost and less time, but Existing Strategy often failed in exploration, especially for the complex, large, and larger systems. As a result, Our Strategy can explore more and higher quality testing time allocation schemes and works much better in promoting convergence than Existing Strategy. An explanation is that Existing Strategy might destroy useful genetic information that should be inherited from parents because of mutating all the gene values in offspring to ensure feasibility. On the contrary, Our Strategy only repairs the gene values that are supposed to be changed.

C. Evaluating the Z-score Operation

Since the z-score operation is adopted to estimate the distance between neighbours more reasonably, in this subsection, we evaluate the diversity of the solution set to analyze the effectiveness of the z-score operation.

To measure the diversity quality, Tables VII and VIII show the pure diversity values [46] of the basic NSGA-II and the z-score operation based NSGA-II on OTRAPs with different numbers of generations, population size, and systems, respectively, regarding the mean and standard deviation values. The results are analyzed using the Wilcoxon rank-sum test [47]. It can be observed that the z-score operation improves the diversity of the solution set on almost all the test instances. When NG is fixed, the improvement becomes greater with the increase of PS. When PS is fixed, the improvement has not changed much with the increase of NG. This implies that the z-score operation estimates the distance between neighbors more accurately when the population size is bigger. To conclude, the z-score operation reduces the repeated or similar solutions and can provide more information to software testers.

D. Comparing MOEAs

The existing HaD-MOEA [18] and WNS-MODE [21] are both multi-objective optimization approaches and used the constraint handling to improve their performance for solving OTRAPs. In this subsection, on the basis of the termination criterion of 100,000 evaluations, we compared our IR-NSGA-TRA with WNS-MODE and HaD-MOEA to further verify our approaches on OTRAPs.

To measure the convergence quality, the coverage values of IR-NSGA-II, WNS-MODE, and HaD-MOEA on OTRAPs under different instances and software systems are shown in Table IX, which are analyzed using the Wilcoxon rank-sum test [47]. For all the 120 testing instances, the coverage values of IR-NSGA-TRA vs WNS-MODE are far greater than those of WNS-MODE vs IR-NSGA-TRA. In particular, IR-NSGA-TRA covers almost half of the solutions obtained by WNS-MODE on each testing instance, especially in the case of larger system. Besides, the coverage values of IR-NSGA-TRA vs HaD-MOEA are much better than those of HaD-MOEA vs IR-NSGA-TRA on 111 instances, but are inferior on only 2 instances. Thus, we can say that IR-NSGA-TRA outperforms WNS-MODE and HaD-MOEA in terms of convergence on most test instances.

Table X show the capacity and coverage values of IR-NSGA-TRA, WNS-MODE, and HaD-MOEA on OTRAPs under different instances and software systems, respectively, regarding solutions whose system reliability is higher than or equal to 0.99 in all the 30 runs for each test instance. It can be observed that for all the 120 testing instances, the capacity

TABLE III
COVERAGE VALUES (MEAN AND STANDARD DEVIATION) OF THE BASIC NSGA-II ON OTRAPS UNDER $NG = 250$, DIFFERENT PS, REPAIR STRATEGIES, AND SOFTWARE SYSTEMS

PS	Simple system		Complex system		Large system		Larger system	
	$\zeta(A,B)$	$\zeta(B,A)$	$\zeta(A,B)$	$\zeta(B,A)$	$\zeta(A,B)$	$\zeta(B,A)$	$\zeta(A,B)$	$\zeta(B,A)$
50	**0.2140(9.54%)**	0.1780(7.80%)	**0.2967(9.10%)**	0.1953(5.48%)	**0.3107(13.48%)**	0.2100(8.83%)	**0.3013(11.96%)**	0.2420(11.84%)
100	**0.2203(5.14%)**	0.1523(5.64%)	**0.3387(10.10%)**	0.1787(4.04%)	**0.3283(10.00%)**	0.1667(6.54%)	**0.3867(14.13%)**	0.1557(6.91%)
150	**0.2298(5.78%)**	0.1456(3.83%)	**0.3224(9.19%)**	0.1769(4.15%)	**0.3687(10.29%)**	0.1473(5.43%)	**0.3540(14.83%)**	0.1696(8.80%)
200	**0.2110(4.28%)**	0.1435(2.98%)	**0.3335(7.00%)**	0.1543(4.07%)	**0.3482(11.59%)**	0.1807(10.85%)	**0.3562(10.13%)**	0.1568(7.13%)
250	**0.2079(4.96%)**	0.1216(2.69%)	**0.3361(7.82%)**	0.1613(5.03%)	**0.3653(13.53%)**	0.1396(6.50%)	**0.3963(11.60%)**	0.1395(6.66%)
300	**0.2211(3.36%)**	0.1277(2.69%)	**0.3294(7.74%)**	0.1734(5.61%)	**0.3468(9.05%)**	0.1588(8.55%)	**0.3975(16.03%)**	0.1537(8.60%)
350	**0.2032(3.86%)**	0.1266(3.06%)	**0.3310(8.11%)**	0.1678(5.00%)	**0.3630(13.74%)**	0.1696(10.73%)	**0.3749(11.75%)**	0.1713(7.12%)
400	**0.2156(3.23%)**	0.1212(2.21%)	**0.3499(5.48%)**	0.1573(2.22%)	**0.3678(12.44%)**	0.1651(10.40%)	**0.3825(14.49%)**	0.1978(10.70%)
450	**0.2087(3.08%)**	0.1243(2.86%)	**0.3500(5.84%)**	0.1541(3.73%)	**0.3767(13.10%)**	0.1623(9.62%)	**0.3561(11.58%)**	0.1876(11.54%)
500	**0.2151(2.93%)**	0.1187(2.30%)	**0.3345(5.23%)**	0.1741(3.51%)	**0.3763(14.68%)**	0.1754(10.73%)	**0.3817(12.97%)**	0.1730(8.78%)

A and B denote the solution sets obtained by Our Strategy and Existing Strategy, respectively.

TABLE IV
COVERAGE VALUES (MEAN AND STANDARD DEVIATION) OF THE BASIC NSGA-II ON OTRAPS UNDER $PS = 250$, DIFFERENT NG, REPAIR STRATEGIES, AND SOFTWARE SYSTEMS

NG	Simple system		Complex system		Large system		Larger system	
	$\zeta(A,B)$	$\zeta(B,A)$	$\zeta(A,B)$	$\zeta(B,A)$	$\zeta(A,B)$	$\zeta(B,A)$	$\zeta(A,B)$	$\zeta(B,A)$
50	**0.1960(5.95%)**	0.1648(4.50%)	**0.3039(9.86%)**	0.2009(6.06%)	**0.3583(12.13%)**	0.1937(8.25%)	**0.3181(15.97%)**	0.2365(11.07%)
100	**0.2272(6.48%)**	0.1328(3.72%)	**0.3135(6.40%)**	0.1863(6.22%)	**0.3224(11.20%)**	0.1835(7.63%)	**0.3357(15.83%)**	0.2029(9.73%)
150	**0.2385(5.23%)**	0.1335(3.83%)	**0.3356(6.76%)**	0.1763(3.96%)	**0.3864(12.78%)**	0.1497(7.20%)	**0.3869(11.09%)**	0.1525(7.74%)
200	**0.2112(4.31%)**	0.1312(3.57%)	**0.3419(6.37%)**	0.1639(4.39%)	**0.3697(10.86%)**	0.1543(6.20%)	**0.4308(17.83%)**	0.1627(13.53%)
250	**0.2079(4.96%)**	0.1216(2.69%)	**0.3361(7.82%)**	0.1613(5.03%)	**0.3653(13.53%)**	0.1396(6.50%)	**0.3963(11.60%)**	0.1395(6.66%)
300	**0.2007(5.02%)**	0.1303(3.49%)	**0.3275(8.10%)**	0.1663(5.64%)	**0.3191(9.87%)**	0.1660(6.80%)	**0.3689(10.38%)**	0.1677(9.66%)
350	**0.1999(3.78%)**	0.1233(3.09%)	**0.3352(5.35%)**	0.1520(4.58%)	**0.3447(13.83%)**	0.1659(9.44%)	**0.3460(13.71%)**	0.1912(11.16%)
400	**0.2177(4.13%)**	0.1259(2.75%)	**0.3268(7.46%)**	0.1685(4.67%)	**0.3045(10.59%)**	0.1975(9.68%)	**0.3685(8.30%)**	0.1473(6.27%)
450	**0.2075(4.23%)**	0.1241(2.41%)	**0.3225(6.85%)**	0.1664(4.17%)	**0.3681(14.88%)**	0.1551(8.77%)	**0.3496(12.63%)**	0.1664(11.00%)
500	**0.2192(5.64%)**	0.1253(3.02%)	**0.3456(5.63%)**	0.1551(3.00%)	**0.3065(12.36%)**	0.1748(10.60%)	**0.3077(11.98%)**	0.2243(16.17%)

A and B denote the solution sets obtained by Our Strategy and Existing Strategy, respectively.

TABLE V
CAPACITY AND COVERAGE VALUES OF THE BASIC NSGA-II ON OTRAPS UNDER $NG = 250$, DIFFERENT PS, REPAIR STRATEGIES, AND SOFTWARE SYSTEMS, REGARDING THE SOLUTIONS WHOSE RELIABILITY IS HIGHER THAN 0.99 IN ALL THE 30 RUNS

PS	Simple system				Complex system				Large system				Larger system			
	$\lvert A\rvert$	$\lvert B\rvert$	$\zeta(A,B)$	$\zeta(B,A)$	$\lvert A\rvert$	$\lvert B\rvert$	$\zeta(A,B)$	$\zeta(B,A)$	$\lvert A\rvert$	$\lvert B\rvert$	$\zeta(A,B)$	$\zeta(B,A)$	$\lvert A\rvert$	$\lvert B\rvert$	$\zeta(A,B)$	$\zeta(B,A)$
50	64	4	100.00%	3.13%	42	0	100.00%	0.00%	24	0	100.00%	0.00%	0	0	0.00%	0.00%
100	141	7	100.00%	0.00%	80	0	100.00%	0.00%	55	0	100.00%	0.00%	2	0	100.00%	0.00%
150	218	13	100.00%	0.00%	142	0	100.00%	0.00%	130	0	100.00%	0.00%	11	0	100.00%	0.00%
200	267	31	100.00%	0.00%	177	0	100.00%	0.00%	125	0	100.00%	0.00%	20	0	100.00%	0.00%
250	366	38	100.00%	0.00%	178	0	100.00%	0.00%	185	0	100.00%	0.00%	51	0	100.00%	0.00%
300	369	33	100.00%	0.00%	220	2	100.00%	0.00%	186	0	100.00%	0.00%	89	0	100.00%	0.00%
350	480	84	100.00%	2.71%	270	3	100.00%	0.00%	271	0	100.00%	0.00%	111	0	100.00%	0.00%
400	522	143	98.60%	2.49%	309	0	100.00%	0.00%	247	0	100.00%	0.00%	133	0	100.00%	0.00%
450	572	184	100.00%	2.45%	293	0	100.00%	0.00%	310	0	100.00%	0.00%	188	0	100.00%	0.00%
500	632	261	100.00%	0.00%	300	6	100.00%	0.00%	359	0	100.00%	0.00%	229	0	100.00%	0.00%

A and B denote the satisfactory solution sets obtained by Our Strategy and Existing Strategy, respectively.

TABLE VI
CAPACITY AND COVERAGE VALUES OF THE BASIC NSGA-II ON OTRAPS UNDER $PS = 250$, DIFFERENT NG, REPAIR STRATEGIES, AND SOFTWARE SYSTEMS, REGARDING THE SOLUTIONS WHOSE RELIABILITY IS HIGHER THAN 0.99 IN ALL THE 30 RUNS

NG	Simple system				Complex system				Large system				Larger system			
	$\lvert A\rvert$	$\lvert B\rvert$	$\zeta(A,B)$	$\zeta(B,A)$	$\lvert A\rvert$	$\lvert B\rvert$	$\zeta(A,B)$	$\zeta(B,A)$	$\lvert A\rvert$	$\lvert B\rvert$	$\zeta(A,B)$	$\zeta(B,A)$	$\lvert A\rvert$	$\lvert B\rvert$	$\zeta(A,B)$	$\zeta(B,A)$
50	40	0	100.00%	0.00%	1	0	100.00%	0.00%	1	0	100.00%	0.00%	0	0	0.00%	0.00%
100	213	9	100.00%	1.88%	54	0	100.00%	0.00%	16	0	100.00%	0.00%	0	0	0.00%	0.00%
150	281	2	100.00%	0.00%	124	0	100.00%	0.00%	82	0	100.00%	0.00%	0	0	0.00%	0.00%
200	325	33	100.00%	1.23%	190	0	100.00%	0.00%	127	0	100.00%	0.00%	9	0	100.00%	0.00%
250	366	38	100.00%	0.00%	178	0	100.00%	0.00%	185	0	100.00%	0.00%	51	0	100.00%	0.00%
300	363	58	96.55%	7.44%	211	3	100.00%	0.00%	202	0	100.00%	0.00%	106	0	100.00%	0.00%
350	364	105	98.10%	1.65%	238	10	100.00%	0.00%	236	1	100.00%	0.00%	179	0	100.00%	0.00%
400	357	141	97.87%	1.96%	216	14	100.00%	0.00%	235	0	100.00%	0.00%	230	2	100.00%	0.00%
450	350	145	99.31%	2.00%	251	15	100.00%	0.40%	247	6	100.00%	0.00%	270	11	100.00%	0.00%
500	367	168	98.81%	35.69%	238	26	100.00%	0.00%	252	1	100.00%	0.00%	242	24	100.00%	0.00%

A and B denote the satisfactory solution sets obtained by Our Strategy and Existing Strategy, respectively.

TABLE VII
PURE DIVERSITY VALUES (MEAN AND STANDARD DEVIATION) OF THE BASIC NSGA-II, Z-SCORE OPERATION BASED NSGA-II ON OTRAPS UNDER $NG = 250$, DIFFERENT PS, AND SOFTWARE SYSTEMS

PS	Simple system		Complex system	
	Z-score	Basic	Z-score	Basic
50	4.202e+07±9.555e+06	4.222e+07±7.368e+06	4.324e+07±1.022e+07	4.283e+07±9.53e+06
100	**7.228e+07±8.929e+06**	5.375e+07±7.477e+06	**7.247e+07±1.243e+07**	6.326e+07±6.45e+06
150	**7.875e+07±9.571e+06**	6.684e+07±7.353e+06	**9.277e+07±9.921e+06**	7.809e+07±9.138e+06
200	**8.746e+07±5.506e+06**	7.12e+07±7.931e+06	**1.089e+08±9.291e+06**	8.436e+07±8.07e+06
250	**1.022e+08±8.119e+06**	7.901e+07±5.947e+06	**1.236e+08±1.129e+07**	9.226e+07±1.02e+07
300	**1.098e+08±1.006e+07**	8.212e+07±7.483e+06	**1.338e+08±8.753e+06**	9.846e+07±8.925e+06
350	**1.182e+08±8.745e+06**	8.82e+07±8.055e+06	**1.429e+08±7.093e+06**	1.062e+08±1.042e+07
400	**1.258e+08±1.009e+07**	9.214e+07±7.119e+06	**1.547e+08±9.425e+06**	1.088e+08±8.431e+06
450	**1.331e+08±8.812e+06**	9.663e+07±8.598e+06	**1.631e+08±8.185e+06**	1.198e+08±9.562e+06
500	**1.408e+08±1.042e+07**	1.029e+08±8.199e+06	**1.708e+08±9.419e+06**	1.214e+08±9.337e+06
PS	Large system		Larger system	
	Z-score	Basic	Z-score	Basic
50	3.699e+07±1.482e+07	4.082e+07±1.063e+07	3.075e+07±3.327e+07	3.632e+07±3.26e+07
100	**7.444e+07±1.067e+07**	6.507e+07±1.026e+07	1.206e+08±1.981e+07	1.017e+08±2.513e+07
150	**9.801e+07±1.165e+07**	7.79e+07±1.052e+07	1.527e+08±1.854e+07	1.239e+08±1.516e+07
200	**1.167e+08±1.323e+07**	8.808e+07±9.527e+06	**1.811e+08±1.791e+07**	1.33e+08±1.842e+07
250	**1.334e+08±1.217e+07**	9.845e+07±1.019e+07	**2.083e+08±2.109e+07**	1.469e+08±1.356e+07
300	**1.436e+08±8.006e+06**	1.047e+08±9.595e+06	**2.266e+08±1.859e+07**	1.661e+08±1.587e+07
350	**1.594e+08±1.316e+07**	1.111e+08±9.606e+06	**2.453e+08±1.603e+07**	1.725e+08±1.593e+07
400	**1.655e+08±1.22e+07**	1.153e+08±7.927e+06	**2.728e+08±1.739e+07**	1.866e+08±1.673e+07
450	**1.784e+08±1.11e+07**	1.209e+08±1.079e+07	**2.825e+08±1.584e+07**	1.881e+08±1.581e+07
500	**1.864e+08±1.045e+07**	1.292e+08±1.018e+07	**3.011e+08±1.392e+07**	1.959e+08±1.611e+07

TABLE VIII
PURE DIVERSITY VALUES (MEAN AND STANDARD DEVIATION) OF THE BASIC NSGA-II, Z-SCORE OPERATION BASED NSGA-II WITH Z-SCORES ON OTRAPS UNDER $PS = 250$, DIFFERENT NG, AND SOFTWARE SYSTEMS

NG	Simple system		Complex system	
	Z-score	Basic	Z-score	Basic
50	**9.438e+07±8.685e+06**	7.768e+07±8.131e+06	**1.046e+08±9.712e+06**	9.062e+07±9.635e+06
100	**9.616e+07±8.036e+06**	8.011e+07±7.986e+06	**1.156e+08±1.438e+07**	9.377e+07±1.011e+07
150	**1.001e+08±9.968e+06**	7.981e+07±6.951e+06	**1.152e+08±7.874e+06**	9.248e+07±9.03e+06
200	**1.023e+08±9.467e+06**	8.139e+07±8.751e+06	**1.181e+08±1.004e+07**	9.016e+07±8.044e+06
250	**1.022e+08±8.119e+06**	7.901e+07±5.947e+06	**1.236e+08±1.129e+07**	9.226e+07±1.02e+07
300	**1.04e+08±7.981e+06**	8.015e+07±7.47e+06	**1.276e+08±1.113e+07**	9.483e+07±8.871e+06
350	**1.036e+08±8.409e+06**	7.735e+07±7.148e+06	**1.246e+08±1.089e+07**	9.174e+07±1.137e+07
400	**1.032e+08±9.468e+06**	7.855e+07±6.871e+06	**1.227e+08±8.809e+06**	9.187e+07±9.191e+06
450	**1.055e+08±1.108e+07**	7.956e+07±8.839e+06	**1.252e+08±9.534e+06**	9.361e+07±9.968e+06
500	**1.026e+08±8.206e+06**	7.903e+07±7.145e+06	**1.265e+08±9.638e+06**	9.366e+07±9.818e+06
NG	Large system		Larger system	
	Z-score	Basic	Z-score	Basic
50	1.049e+08±8.821e+06	8.948e+07±8.299e+06	1.435e+08±1.398e+07	1.348e+08±1.359e+07
100	1.196e+08±1.069e+07	9.239e+07±9.053e+06	1.773e+08±1.74e+07	1.489e+08±1.46e+07
150	**1.24e+08±1.231e+07**	9.526e+07±1.08e+07	**1.89e+08±1.572e+07**	1.434e+08±1.532e+07
200	**1.275e+08±1.159e+07**	9.769e+07±1e+07	**1.962e+08±1.226e+07**	1.515e+08±1.748e+07
250	**1.334e+08±1.217e+07**	9.845e+07±1.019e+07	**2.083e+08±2.109e+07**	1.469e+08±1.356e+07
300	**1.287e+08±1.145e+07**	9.615e+07±9.994e+06	**2.098e+08±1.643e+07**	1.401e+08±1.198e+07
350	**1.351e+08±1.146e+07**	9.735e+07±9.643e+06	**2.092e+08±1.497e+07**	1.458e+08±1.607e+07
400	**1.37e+08±1.024e+07**	9.832e+07±1.126e+07	**2.182e+08±2.204e+07**	1.52e+08±1.777e+07
450	**1.366e+08±1.33e+07**	9.557e+07±8.701e+06	**2.174e+08±1.223e+07**	1.455e+08±1.502e+07
500	**1.372e+08±9.728e+06**	9.686e+07±1.069e+07	**2.227e+08±1.825e+07**	1.519e+08±2.11e+07

values of NSGA-II-TRA are far greater than those of WNS-MODE on 111 instances and HaD-MOEA on 103 instances, respectively. In the case of larger system, both WNS-MODE and HaD-MOEA can almost find no satisfactory solution. The coverage values of NSGA-II-TRA vs WNS-MODE are better than those of WNS-MODE vs NSGA-II-TRA on 95 instances. In addition, the coverage values of NSGA-II-TRA vs HaD-MOEA are much better than those of HaD-MOEA vs NSGA-II-TRA on 111 instances in which NSGA-II-TRA can almost completely cover the solutions given by HaD-MOEA.

In brief, NSGA-II-TRA outperforms WNS-MODE and HaD-MOEA on both the quantity and the quality of the available testing-time allocation schemes. However, an important aspect that should not be ignored is that the coverage difference between NSGA-II-TRA and HaD-MOEA which are both developed based on NSGA-II is smaller than that between Our Strategy and Existing Strategy in the first experiment. The reason is that better diversity inevitably brings about a more

TABLE IX
COVERAGE VALUES (MEAN AND STANDARD DEVIATION) OF IR-NSGA-II, WNS-MODE, AND HAD-MOEA ON OTRAPS UNDER DIFFERENT INSTANCES AND SYSTEMS

Instance	Simple system				Complex system			
	$\zeta(A,B)$	$\zeta(B,A)$	$\zeta(A,C)$	$\zeta(C,A)$	$\zeta(A,B)$	$\zeta(B,A)$	$\zeta(A,C)$	$\zeta(C,A)$
1	**0.5125(3.60%)**	0.0667(1.45%)	**0.1953(3.98%)**	0.0558(1.31%)	**0.5340(3.79%)**	0.1035(3.58%)	**0.2677(8.34%)**	0.0917(3.16%)
2	**0.4828(3.34%)**	0.0658(2.42%)	**0.2162(4.57%)**	0.0520(1.63%)	**0.5500(3.35%)**	0.1052(3.85%)	**0.2190(5.75%)**	0.1032(3.43%)
3	**0.4815(3.14%)**	0.0707(3.01%)	**0.2043(5.35%)**	0.0463(2.10%)	**0.5215(3.20%)**	0.1150(3.22%)	**0.2290(6.69%)**	0.1132(3.21%)
4	**0.5402(2.95%)**	0.0780(2.07%)	**0.2213(4.48%)**	0.0492(1.64%)	**0.5555(3.35%)**	0.1020(1.77%)	**0.2188(5.01%)**	0.1082(2.29%)
5	**0.5593(2.25%)**	0.0512(1.78%)	**0.2535(5.33%)**	0.0427(1.37%)	**0.5240(3.42%)**	0.1150(3.67%)	**0.2225(6.26%)**	0.1117(3.49%)
6	**0.5098(3.67%)**	0.0690(3.25%)	**0.2243(6.15%)**	0.0548(2.95%)	**0.5387(3.73%)**	0.1043(3.63%)	**0.2152(6.61%)**	0.0907(3.19%)
7	**0.4987(3.66%)**	0.0657(2.29%)	**0.1977(5.38%)**	0.0587(2.41%)	**0.5760(3.13%)**	0.1008(4.00%)	**0.1923(6.52%)**	0.1033(3.64%)
8	**0.5372(3.26%)**	0.0733(1.93%)	**0.2590(6.23%)**	0.0590(1.58%)	**0.5138(4.26%)**	0.1197(3.85%)	**0.2030(11.03%)**	0.1135(4.12%)
9	**0.5462(2.79%)**	0.0587(2.28%)	**0.2053(4.07%)**	0.0580(1.97%)	**0.5972(3.34%)**	0.1077(5.87%)	**0.1853(6.09%)**	0.1355(4.62%)
10	**0.5462(2.80%)**	0.0910(4.27%)	**0.2063(5.83%)**	0.0495(1.93%)	**0.5455(3.58%)**	0.1053(6.43%)	**0.2500(7.76%)**	0.0987(4.02%)
11	**0.5462(2.81%)**	0.0713(2.42%)	**0.2257(5.26%)**	0.0530(1.81%)	**0.5505(3.92%)**	0.1348(5.46%)	**0.1723(7.68%)**	0.1335(5.64%)
12	**0.5462(2.82%)**	0.0705(2.13%)	**0.2240(4.78%)**	0.0472(1.47%)	**0.5695(3.11%)**	0.0983(2.58%)	**0.2293(4.61%)**	0.0985(2.64%)
13	**0.5462(2.83%)**	0.0687(1.96%)	**0.2492(5.62%)**	0.0580(1.91%)	**0.5672(3.03%)**	0.0887(3.68%)	**0.1945(6.47%)**	0.1088(4.11%)
14	**0.5462(2.84%)**	0.0558(1.45%)	**0.2308(6.18%)**	0.0408(1.11%)	**0.5697(3.47%)**	0.1168(4.20%)	**0.2282(7.34%)**	0.1220(3.58%)
15	**0.5462(2.85%)**	0.0615(1.89%)	**0.2013(4.16%)**	0.0660(1.65%)	**0.5110(3.26%)**	0.1260(4.73%)	**0.2148(5.59%)**	0.1153(4.22%)
16	**0.5462(2.86%)**	0.0553(1.66%)	**0.1888(3.72%)**	0.0613(2.26%)	**0.5532(3.44%)**	0.1265(4.97%)	**0.2105(7.42%)**	0.1162(4.26%)
17	**0.5462(2.87%)**	0.0545(1.72%)	**0.2310(5.31%)**	0.0465(1.12%)	**0.5177(3.75%)**	0.1223(3.25%)	**0.2160(9.57%)**	0.1190(4.27%)
18	**0.5462(2.88%)**	0.0693(2.57%)	**0.2238(7.80%)**	0.0513(1.73%)	**0.5647(3.21%)**	0.1047(3.67%)	**0.2367(7.05%)**	0.0995(3.41%)
19	**0.5462(2.89%)**	0.0635(2.37%)	**0.2662(6.04%)**	0.0530(1.74%)	**0.5597(3.18%)**	0.1022(3.76%)	**0.2077(4.52%)**	0.1040(2.84%)
20	**0.5462(2.90%)**	0.0528(1.51%)	**0.1907(4.91%)**	0.0422(2.22%)	**0.5187(2.71%)**	0.1080(3.05%)	**0.2363(8.47%)**	0.1015(2.30%)
21	**0.5462(2.91%)**	0.0838(2.46%)	**0.2200(4.85%)**	0.0662(1.88%)	**0.5272(3.39%)**	0.1203(4.10%)	**0.2272(6.89%)**	0.0950(3.51%)
22	**0.5462(2.92%)**	0.0677(3.91%)	**0.2207(5.21%)**	0.0523(1.83%)	**0.5550(2.93%)**	0.1108(4.01%)	**0.2275(6.94%)**	0.1070(3.94%)
23	**0.5462(2.93%)**	0.0650(1.84%)	**0.2588(6.31%)**	0.0595(2.01%)	**0.5518(3.71%)**	0.1097(4.32%)	**0.2323(8.07%)**	0.1067(3.41%)
24	**0.5462(2.94%)**	0.0595(1.72%)	**0.1938(3.41%)**	0.0772(2.22%)	**0.5600(3.16%)**	0.0928(1.77%)	**0.2087(6.33%)**	0.0997(2.67%)
25	**0.5462(2.95%)**	0.0553(1.80%)	**0.1893(4.20%)**	0.0633(1.94%)	**0.5817(3.94%)**	0.1058(3.11%)	**0.2462(9.09%)**	0.1033(2.84%)
26	**0.5462(2.96%)**	0.0635(1.99%)	**0.2132(5.24%)**	0.0533(1.56%)	**0.5488(2.94%)**	0.0983(4.08%)	**0.2133(5.53%)**	0.1008(3.49%)
27	**0.5462(2.97%)**	0.0577(2.27%)	**0.2118(4.65%)**	0.0500(2.06%)	**0.5493(3.39%)**	0.1143(3.32%)	**0.2245(7.98%)**	0.1138(2.79%)
28	**0.5462(2.98%)**	0.0775(3.36%)	**0.2328(5.63%)**	0.0512(2.07%)	**0.5377(4.06%)**	0.1080(4.98%)	**0.2328(7.30%)**	0.1068(3.91%)
29	**0.5462(2.99%)**	0.0482(1.98%)	**0.2152(4.64%)**	0.0420(1.47%)	**0.5448(3.53%)**	0.1292(5.87%)	**0.2157(9.27%)**	0.1142(6.04%)
30	**0.5462(2.10%)**	0.0642(1.83%)	**0.2360(5.62%)**	0.0605(2.03%)	**0.5302(4.55%)**	0.1050(3.47%)	**0.2232(7.98%)**	0.0987(3.25%)

Instance	Large system				Larger system			
	$\zeta(A,B)$	$\zeta(B,A)$	$\zeta(A,C)$	$\zeta(C,A)$	$\zeta(A,B)$	$\zeta(B,A)$	$\zeta(A,C)$	$\zeta(C,A)$
1	**0.5833(4.55%)**	0.0988(3.94%)	0.1692(7.15%)	0.1512(6.66%)	**0.5995(4.01%)**	0.1315(4.41%)	**0.3362(13.86%)**	0.0958(7.60%)
2	**0.6048(6.28%)**	0.0915(5.00%)	**0.1978(9.31%)**	0.1362(5.92%)	**0.6253(4.04%)**	0.1040(3.08%)	**0.2983(12.93%)**	0.1330(12.42%)
3	**0.6032(5.46%)**	0.0815(3.57%)	**0.2355(10.08%)**	0.1283(5.77%)	**0.6313(4.58%)**	0.1093(4.80%)	**0.3130(12.82%)**	0.0933(5.99%)
4	**0.5857(4.30%)**	0.0863(4.20%)	**0.1873(7.23%)**	0.1357(5.28%)	**0.6167(4.39%)**	0.1288(4.94%)	**0.2695(17.50%)**	0.1665(17.78%)
5	**0.5838(4.64%)**	0.1185(5.88%)	**0.2138(7.70%)**	0.1627(9.05%)	**0.6240(5.66%)**	0.1138(4.57%)	**0.2497(13.56%)**	0.1773(17.17%)
6	**0.5865(4.92%)**	0.1018(4.66%)	**0.2398(9.77%)**	0.1422(6.34%)	**0.5920(5.71%)**	0.1310(5.67%)	**0.3403(18.42%)**	0.1207(9.92%)
7	**0.5768(4.97%)**	0.1137(5.58%)	**0.2230(9.99%)**	0.1503(6.87%)	**0.6033(5.16%)**	0.1347(4.90%)	**0.3445(13.30%)**	0.1048(7.41%)
8	**0.5610(4.26%)**	0.1087(3.98%)	**0.2198(10.64%)**	0.1410(6.39%)	**0.6127(4.48%)**	0.1185(4.21%)	**0.2767(11.18%)**	0.1213(10.55%)
9	**0.5952(5.44%)**	0.1048(5.33%)	0.1275(5.68%)	**0.1860(6.25%)**	**0.6192(4.46%)**	0.1170(3.94%)	**0.2875(16.35%)**	0.1492(13.20%)
10	**0.5963(4.56%)**	0.0943(3.95%)	**0.2378(9.52%)**	0.1378(5.66%)	**0.6168(5.34%)**	0.1220(4.92%)	**0.2768(12.70%)**	0.1318(8.91%)
11	**0.5952(6.33%)**	0.0958(5.97%)	**0.2262(10.92%)**	0.1357(8.51%)	**0.6187(5.89%)**	0.1190(4.67%)	**0.3277(16.99%)**	0.1627(17.75%)
12	**0.5995(3.65%)**	0.1003(4.20%)	**0.2252(8.07%)**	0.1422(5.01%)	**0.6220(5.68%)**	0.1345(6.68%)	**0.3010(16.67%)**	0.1288(10.65%)
13	**0.5952(3.88%)**	0.0868(3.54%)	**0.2222(9.80%)**	0.1260(4.96%)	**0.6192(5.59%)**	0.1220(4.99%)	**0.2545(14.56%)**	0.1430(10.22%)
14	**0.6063(4.66%)**	0.0892(3.74%)	**0.2067(6.77%)**	0.1383(6.01%)	**0.5955(5.46%)**	0.1187(3.97%)	**0.3257(16.13%)**	0.1097(10.53%)
15	**0.5875(4.88%)**	0.0977(5.73%)	**0.2097(9.03%)**	0.1363(6.24%)	**0.6050(5.22%)**	0.1290(5.61%)	**0.3050(17.14%)**	0.1410(10.11%)
16	**0.6058(6.02%)**	0.1032(7.73%)	0.1998(9.27%)	0.1737(9.65%)	**0.6067(5.32%)**	0.1380(5.65%)	**0.3145(22.33%)**	0.1645(14.01%)
17	**0.5813(6.25%)**	0.1143(5.43%)	0.2005(12.83%)	0.1720(10.19%)	**0.6330(3.91%)**	0.1110(3.55%)	**0.2823(13.79%)**	0.1442(11.81%)
18	**0.5652(4.26%)**	0.1122(4.56%)	**0.2265(10.39%)**	0.1507(6.66%)	**0.6107(7.01%)**	0.1178(5.38%)	**0.2340(13.52%)**	0.1650(16.15%)
19	**0.6047(3.88%)**	0.0855(3.67%)	**0.2140(7.95%)**	0.1383(4.61%)	**0.6278(5.77%)**	0.1120(4.01%)	**0.3215(13.88%)**	0.1060(9.52%)
20	**0.6062(4.56%)**	0.0898(4.01%)	**0.1945(6.97%)**	0.1485(4.78%)	**0.6298(3.90%)**	0.1092(3.38%)	**0.3047(14.92%)**	0.1245(10.99%)
21	**0.6122(4.99%)**	0.0817(3.13%)	**0.2213(6.89%)**	0.1157(4.96%)	**0.6248(5.08%)**	0.1132(4.90%)	**0.2762(12.81%)**	0.1282(12.52%)
22	**0.6002(4.54%)**	0.0950(5.23%)	**0.1980(8.37%)**	0.1473(6.93%)	**0.5912(5.50%)**	0.1615(7.24%)	**0.3493(18.62%)**	0.1275(12.27%)
23	**0.5837(4.91%)**	0.1017(3.61%)	0.1857(11.16%)	0.1672(7.28%)	**0.6060(5.99%)**	0.1228(4.57%)	**0.3627(18.97%)**	0.1182(12.52%)
24	**0.6005(3.72%)**	0.0768(2.36%)	**0.2055(9.20%)**	0.1292(3.85%)	**0.5970(4.61%)**	0.1268(4.28%)	**0.3103(9.73%)**	0.1003(6.42%)
25	**0.6063(5.33%)**	0.0800(2.94%)	0.1673(6.19%)	0.1410(4.63%)	**0.6238(4.76%)**	0.1133(3.49%)	**0.3580(17.01%)**	0.1097(10.98%)
26	**0.6113(5.12%)**	0.1037(6.39%)	0.1918(10.11%)	0.1678(8.38%)	**0.6447(7.52%)**	0.1145(5.40%)	**0.2628(17.18%)**	0.1880(16.19%)
27	**0.5743(3.93%)**	0.1193(5.43%)	0.2070(10.88%)	0.1557(7.48%)	**0.6075(3.98%)**	0.1352(4.74%)	**0.3320(17.44%)**	0.1183(13.04%)
28	**0.5970(5.66%)**	0.1005(4.49%)	0.1590(6.91%)	0.1645(7.97%)	**0.6308(6.68%)**	0.1175(5.41%)	**0.3163(20.59%)**	0.1752(18.82%)
29	**0.6065(4.77%)**	0.0942(5.34%)	**0.1902(9.94%)**	0.1487(10.36%)	**0.6155(4.73%)**	0.1303(4.33%)	**0.3362(15.64%)**	0.1135(10.49%)
30	**0.6107(4.79%)**	0.0965(4.03%)	**0.2140(7.34%)**	0.1382(6.27%)	**0.6118(6.46%)**	0.1330(6.46%)	**0.3220(22.73%)**	0.1650(20.59%)

A, B, and C denote the solution sets obtained by IR-NSGA-II, WNS-MODE, and HaD-MOEA, respectively.

TABLE X
CAPACITY AND COVERAGE VALUES OF IR-NSGA-II, WNS-MODE, AND HAD-MOEA ON OTRAPS UNDER DIFFERENT INSTANCES AND SYSTEMS

Instance	Simple system							Complex system						
	$\|A\|$	$\|B\|$	$\|C\|$	$\zeta(A,B)$	$\zeta(B,A)$	$\zeta(A,C)$	$\zeta(C,A)$	$\|A\|$	$\|B\|$	$\|C\|$	$\zeta(A,B)$	$\zeta(B,A)$	$\zeta(A,C)$	$\zeta(C,A)$
1	52	15	7	66.67%	44.23%	100.00%	1.92%	14	0	0	100.00%	0.00%	100.00%	0.00%
2	20	2	0	50.00%	0.00%	100.00%	0.00%	58	21	1	95.24%	48.28%	100.00%	0.00%
3	33	16	0	31.25%	96.97%	100.00%	0.00%	28	2	0	50.00%	46.43%	100.00%	0.00%
4	116	58	94	72.41%	92.24%	100.00%	6.90%	73	16	10	81.25%	19.18%	100.00%	0.00%
5	141	62	156	90.32%	50.35%	100.00%	3.55%	28	11	0	63.64%	39.29%	100.00%	0.00%
6	68	47	4	70.21%	85.29%	100.00%	2.94%	68	16	4	93.75%	25.00%	100.00%	0.00%
7	51	16	4	75.00%	19.61%	100.00%	13.73%	109	40	74	90.00%	58.72%	100.00%	2.75%
8	88	52	75	88.46%	78.41%	100.00%	5.68%	32	5	0	0.00%	28.13%	100.00%	0.00%
9	130	60	170	76.67%	65.38%	99.41%	21.54%	155	54	181	90.74%	35.48%	99.45%	12.90%
10	41	18	9	50.00%	82.93%	100.00%	12.20%	37	5	0	100.00%	10.81%	100.00%	0.00%
11	131	61	103	95.08%	41.22%	100.00%	6.87%	67	23	22	78.26%	22.39%	100.00%	10.45%
12	6	0	0	100.00%	0.00%	100.00%	0.00%	101	33	68	84.85%	31.68%	100.00%	1.98%
13	134	65	107	89.23%	41.04%	100.00%	1.49%	77	20	2	80.00%	38.96%	100.00%	0.00%
14	16	1	0	0.00%	6.25%	100.00%	0.00%	78	26	41	69.23%	74.36%	100.00%	3.85%
15	66	34	30	85.29%	42.42%	100.00%	3.03%	7	0	0	100.00%	0.00%	100.00%	0.00%
16	114	51	128	88.24%	43.86%	97.66%	19.30%	88	38	20	84.21%	45.45%	100.00%	2.27%
17	82	33	21	75.76%	53.66%	100.00%	10.98%	33	1	0	0.00%	63.64%	100.00%	0.00%
18	108	49	94	91.84%	79.63%	98.94%	17.59%	90	32	46	81.25%	20.00%	100.00%	1.11%
19	203	95	258	94.74%	59.11%	100.00%	36.95%	97	29	49	89.66%	25.77%	100.00%	0.00%
20	50	12	10	66.67%	52.00%	100.00%	8.00%	1	0	0	100.00%	0.00%	100.00%	0.00%
21	128	60	106	76.67%	42.19%	100.00%	2.34%	42	16	0	43.75%	57.14%	100.00%	0.00%
22	141	61	158	77.05%	54.61%	99.37%	18.44%	64	20	1	75.00%	54.69%	100.00%	0.00%
23	115	42	29	95.24%	38.26%	100.00%	0.87%	66	17	9	82.35%	33.33%	100.00%	1.52%
24	79	40	26	80.00%	75.95%	100.00%	1.27%	86	20	20	95.00%	12.79%	100.00%	1.16%
25	110	65	151	84.62%	74.55%	98.01%	15.45%	141	64	142	87.50%	55.32%	100.00%	5.67%
26	81	30	56	76.67%	50.62%	98.21%	13.58%	53	5	0	80.00%	52.83%	100.00%	0.00%
27	39	20	0	60.00%	64.10%	100.00%	0.00%	72	23	15	73.91%	40.28%	100.00%	1.39%
28	61	34	4	64.71%	83.61%	100.00%	8.20%	20	1	0	0.00%	10.00%	100.00%	0.00%
29	22	5	0	20.00%	27.27%	100.00%	0.00%	64	18	9	72.22%	45.31%	100.00%	0.00%
30	123	47	73	89.36%	59.35%	100.00%	0.81%	23	7	0	57.14%	0.00%	100.00%	0.00%

Instance	Large system							Larger system						
	$\|A\|$	$\|B\|$	$\|C\|$	$\zeta(A,B)$	$\zeta(B,A)$	$\zeta(A,C)$	$\zeta(C,A)$	$\|A\|$	$\|B\|$	$\|C\|$	$\zeta(A,B)$	$\zeta(B,A)$	$\zeta(A,C)$	$\zeta(C,A)$
1	8	0	0	100.00%	0.00%	100.00%	0.00%	0	0	0	0.00%	0.00%	0.00%	0.00%
2	39	5	3	100.00%	17.95%	100.00%	2.56%	38	0	0	100.00%	0.00%	100.00%	0.00%
3	37	0	0	100.00%	0.00%	100.00%	0.00%	95	0	0	100.00%	0.00%	100.00%	0.00%
4	14	1	0	100.00%	0.00%	100.00%	0.00%	21	0	0	100.00%	0.00%	100.00%	0.00%
5	19	0	0	100.00%	0.00%	100.00%	0.00%	7	0	0	100.00%	0.00%	100.00%	0.00%
6	24	0	0	100.00%	0.00%	100.00%	0.00%	0	0	0	0.00%	0.00%	0.00%	0.00%
7	16	2	0	0.00%	18.75%	100.00%	0.00%	21	0	0	100.00%	0.00%	100.00%	0.00%
8	24	0	0	100.00%	0.00%	100.00%	0.00%	55	0	0	100.00%	0.00%	100.00%	0.00%
9	72	15	23	86.67%	36.11%	100.00%	0.00%	22	0	0	100.00%	0.00%	100.00%	0.00%
10	33	2	0	50.00%	6.06%	100.00%	0.00%	72	0	2	100.00%	0.00%	100.00%	0.00%
11	35	4	0	75.00%	8.57%	100.00%	0.00%	0	0	0	0.00%	0.00%	0.00%	0.00%
12	52	10	10	100.00%	21.15%	100.00%	0.00%	45	0	0	100.00%	0.00%	100.00%	0.00%
13	35	3	0	33.33%	31.43%	100.00%	0.00%	49	0	0	100.00%	0.00%	100.00%	0.00%
14	38	3	0	66.67%	31.58%	100.00%	0.00%	0	0	0	0.00%	0.00%	0.00%	0.00%
15	0	0	0	0.00%	0.00%	0.00%	0.00%	3	0	0	100.00%	0.00%	100.00%	0.00%
16	59	12	13	75.00%	27.12%	100.00%	1.69%	23	0	0	100.00%	0.00%	100.00%	0.00%
17	12	0	0	100.00%	0.00%	100.00%	0.00%	77	1	0	100.00%	5.19%	100.00%	0.00%
18	28	5	0	40.00%	78.57%	100.00%	0.00%	0	0	0	0.00%	0.00%	0.00%	0.00%
19	35	1	2	0.00%	11.43%	100.00%	0.00%	13	0	0	100.00%	0.00%	100.00%	0.00%
20	45	1	0	0.00%	8.89%	100.00%	0.00%	58	0	0	100.00%	0.00%	100.00%	0.00%
21	14	0	0	100.00%	0.00%	100.00%	0.00%	58	0	0	100.00%	0.00%	100.00%	0.00%
22	49	12	0	75.00%	32.65%	100.00%	0.00%	14	0	0	100.00%	0.00%	100.00%	0.00%
23	36	4	0	50.00%	55.56%	100.00%	0.00%	9	0	0	100.00%	0.00%	100.00%	0.00%
24	34	0	0	100.00%	0.00%	100.00%	0.00%	0	0	0	0.00%	0.00%	0.00%	0.00%
25	73	13	42	69.23%	39.73%	100.00%	2.74%	20	0	0	100.00%	0.00%	100.00%	0.00%
26	61	4	4	100.00%	6.56%	100.00%	0.00%	97	0	0	100.00%	0.00%	100.00%	0.00%
27	33	4	0	50.00%	45.45%	100.00%	0.00%	24	0	0	100.00%	0.00%	100.00%	0.00%
28	0	0	0	0.00%	0.00%	0.00%	0.00%	15	0	0	100.00%	0.00%	100.00%	0.00%
29	54	5	12	60.00%	38.89%	100.00%	0.00%	60	0	0	100.00%	0.00%	100.00%	0.00%
30	21	0	0	100.00%	0.00%	100.00%	0.00%	0	0	0	0.00%	0.00%	0.00%	0.00%

A, B, and C denote the satisfactory solution sets obtained by IR-NSGA-II, WNS-MODE, and HaD-MOEA, respectively.

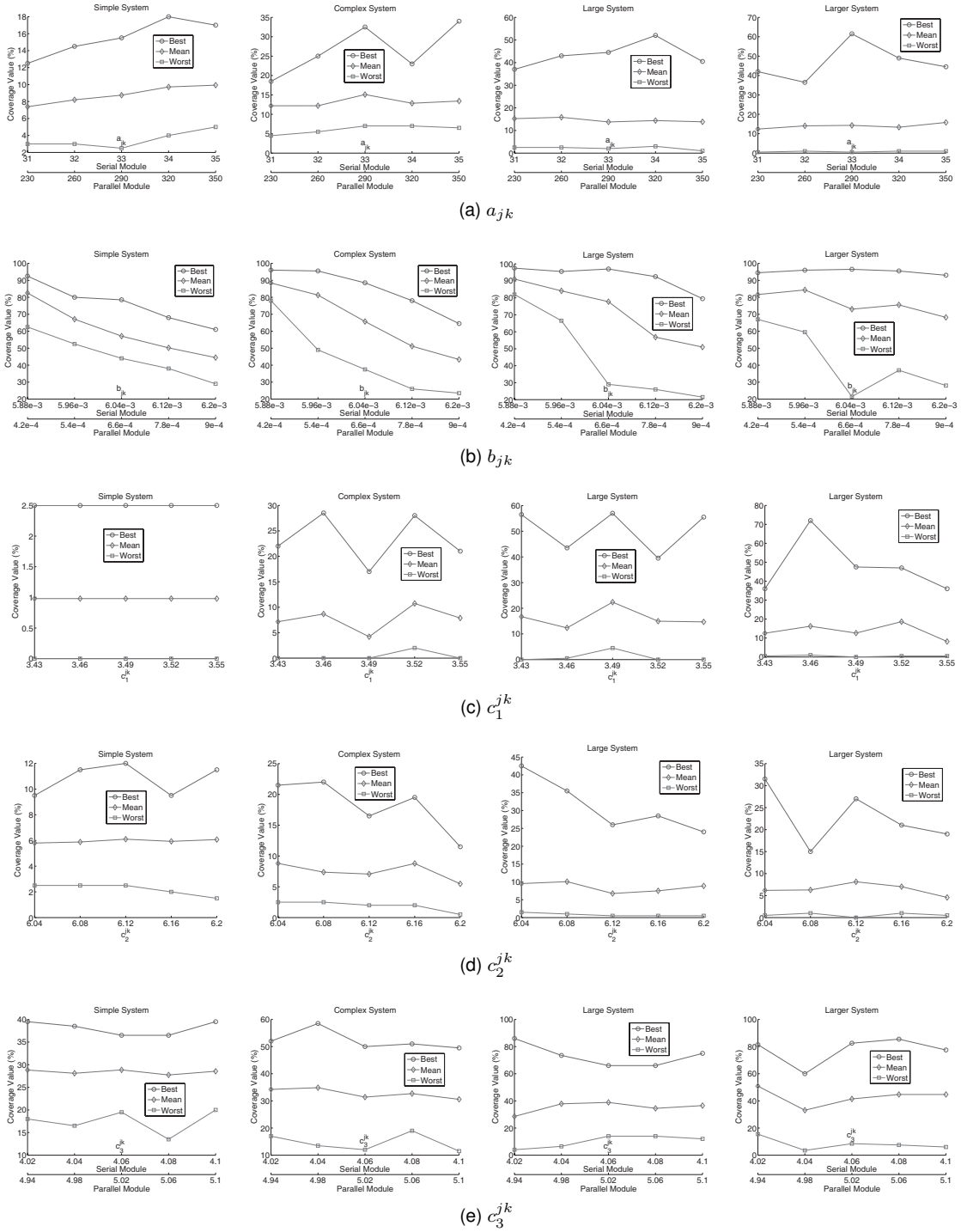


Fig. 8. Sensitivity analysis for NSGA-II-TRA on the five modular parameters, respectively.

even distribution of solutions and thus reduces the number of repeated and similar solutions to a certain extent. This suggests that excessive pursuit of high diversity values may not be beneficial for MOEAs to solve OTRAPs.

E. Sensitivity Analysis

To evaluate the sensitivity of NSGA-II-TRA with respect to the five modular parameters in Table I, we examine how much the coverage value fluctuates with the increase of the value of the modular parameters in the pre-defined intervals. From 30 independent runs and 100,000 evaluations in each run, the relative changes of the coverage values corresponding to the increases of a_{jk}, b_{jk}, c_1^{jk}, c_2^{jk}, and c_3^{jk}, respectively, are shown in Fig. 8, in which the coverage value is the percentage of the solutions in the previous case that is covered by the solutions in the current case.

In Fig. 8(a), the initial case is that the values of a_{jk} are 30.0 for each serial module and 200.0 for each parallel module, respectively. The average coverage value changes little and is

often small with the increase of the value of a_{jk}. The reason is that a_{jk} is the mean value of the total errors in M_{jk}, and with the increase of the value of a_{jk}, the same amount of consumed testing time may result in lower reliability. It means that the obtained solutions under bigger values of a_{jk} hardly cover the solutions under smaller values of a_{jk}.

In Fig. 8(b), the initial case is that the values of b_{jk} are 0.0058 for each serial module and 0.0003 for each parallel module, respectively. The average coverage value is often great with the increase of the value of b_{jk}. This is because b_{jk} is the rate of detected errors in M_{jk}, and with the increase of the value of b_{jk}, the same amount of consumed testing time may produce higher reliability. This implies that the obtained solutions under bigger values of b_{jk} may easily cover the solutions under smaller values of b_{jk}. Another noteworthy aspect is that the average coverage value in Fig. 8(b) is in a downward trend, which suggests that when b_{jk} increases to a certain value, it may be difficult for the reliability to be greatly improved. This is due to the fact that the number of possible tests for even simple modules is practically infinite.

In Fig. 8(c), the initial case is that the value of c_1^{jk} is 3.4 for each module. In Fig. 8(d), the initial case is that the value of c_2^{jk} is 6.0 for each module. It can be seen that, in Fig. 8(c) and Fig. 8(d), the average coverage value and its fluctuations are small. An explanation is that with the increase of the value of c_1^{jk} or c_2^{jk}, the same amount of consumed testing time may generate higher cost. This means that it is difficult for the obtained solutions under bigger values of c_1^{jk} or c_2^{jk} to cover the solutions under smaller values of c_1^{jk} or c_2^{jk}.

In Fig. 8(e), the initial case is that the values of c_3^{jk} are 4.0 for each serial module and 4.9 for each parallel module, respectively. It is clear that the average coverage value in Fig. 8(e) is higher than that in Fig. 8(c) and Fig. 8(d). This is because with the increase of the value of c_3^{jk}, the same amount of consumed testing time may bring lower cost, and thus the obtained solutions under bigger values of c_3^{jk} may easily cover the solutions under smaller values of c_3^{jk}. However, the average coverage value in Fig. 8(e) is lower than that in Fig. 8(b), which suggests that the effect of c_3^{jk} on testing cost is smaller than that of b_{jk} on reliability. This is in line with the natural characteristic of these two parameters.

In general, the curves for the coverage value fluctuate a little between some ranges. These fluctuations may be attributed to the fact that NSGA-II-TRA gets trapped in local optima. However, the changes are small and the sensitivities are at acceptable levels. NSGA-II-TRA seems robust on the five modular parameters. The goodness of the solutions obtained by NSGA-II-TRA does not fluctuate greatly with the change of the parameter values.

V. Conclusions and Outlook

Software testing is recognized as the most costly and resource-consuming part of software development [21]. One of the most challenging problems that arise in the software testing phase is the optimal testing resource (or time) allocation. This involves allocating limited testing resources to each module in a software system such that the reliability of the entire system is maximized, the testing cost is minimized, and the consumed testing time is minimized simultaneously [18]. In this paper, we have developed and evaluated a NSGA-II and constraint handling based multi-objective optimization approach (called NSGA-II-TRA) that finds high quality testing-time allocation schemes effectively. The strength of our approach is founded upon two main components:

1) We devise a group of novel heuristics for constraint handling in solving OTRAPs. Unlike previous work, these heuristics only repair the values of the selected genetic genes rather than all the gene values in offspring. By applying these heuristics, NSGA-II-TRA is able to preserve parental characteristics in offspring and also maintain useful genetic information. This is helpful to strengthen the effectiveness of genetic operators in NSGA-II-TRA and to improve the quality of solutions.
2) We introduce the z-scores and use the standardized Euclidean distance to estimate the neighbor relationship among individuals more accurately. By applying this approach, NSGA-II-TRA is able to reduce repeated or similar solutions.

Altogether, these components allow NSGA-II-TRA to make good performance gains over the existing MOEAs (e.g., WNS-MODE and HaD-MOEA) on the capacity, coverage, and pure diversity values. The experimental results illustrate that NSGA-II-TRA is robust on the modular parameters and obtains more and higher quality satisfactory solutions that are not dominated by those obtained by WNS-MODE and HaD-MOEA on almost all the testing instances, especially when the problem size is large. Therefore, compared to WNS-MODE and HaD-MOEA, NSGA-II-TRA can provide a software tester with a lot of additional better choices to achieve different practical goals on reliability, cost, and time, and thus benefit organizing the whole testing phase.

We do not imply in this paper that NSGA-II-TRA is always superior to WNS-MODE and HaD-MOEA. From a more practical viewpoint, this work can be seen as an initial step toward a more reasonable guide to solving OTRAPs and evaluating the solution quality from the perspective of multi-objective optimization, which may be helpful for designers and testers of software systems. This work still has some limitations to be improved in future research: our analysis was based only on four simulated software systems; these algorithms (i.e., WNS-MODE, HaD-MOEA, and NSGA-II-TRA) are not evaluated on real-world test instances and compared with a real-world testing resource allocator that is actually in use by a human manager; the strengths and weaknesses of these algorithms are not studied on OTRAPs in comparison with alternative MOEAs. In addition to the above, it would be interesting to investigate the following topics:

1) Software testing is a costly and unavoidable task, so more constraint conditions such as reliability, budget, and skill constraints could be considered in OTRAPs.
2) The results in this paper show that valuable Pareto-optimal solutions for OTRAPs are in the upper boundary region and thus raise an open question: how to make MOEAs search only in the pre-defined acceptable region

for user satisfaction?

3) In practice, users may put forward higher requirement of reliability or functionality. This could lead to the change of the number of modules during the optimization process, which is a dynamic constrained multi-objective optimization problem, needing more constraint-handling techniques and population-maintenance methods.

ACKNOWLEDGMENT

The authors would like to thank Dr. M. Yang and Xiaofen Lu for their fruitful discussions. They would also like to thank Prof. K. Deb for the open C code of the original multi-objective NSGA-II on the website of Kanpur Genetic Algorithms Laboratory, Dr. Y. Hu for the programs of WNS-MODE [21], and Dr. H. Wang for the matlab code of computing the pure diversity value [46]. They would also like to thank the Editors and anonymous referees for their insightful comments and suggestions. The work reported in this paper was initiated when the first author visited the last author at the University of Birmingham, U.K.

REFERENCES

[1] E. T. Barr, M. Harman, P. Mcminn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey", *IEEE Transactions on Software Engineering*, vol. 41, no. 5, pp. 507–525, 2015.

[2] R. Tavakkoli-Moghaddam, J. Safari, and F. Sassani, "Reliability optimization of series-parallel systems with a choice of redundancy strategies using a genetic algorithm," *Reliability Engineering & System Safety*, vol. 93, no. 4, pp. 550–556, 2008.

[3] F. Yue, G. Zhang, Z. Su, Y. Lu, and T. Zhang, "Multi-software reliability allocation in multimedia systems with budget constraints using Dempster-Shafer theory and improved differential evolution," *Neurocomputing*, vol. 169, pp. 13–22, 2015.

[4] H. Ohtera and S. Yamada, "Optimal allocation and control problems for software-testing resources," *IEEE Transactions on Reliability*, vol. 39, no. 2, pp. 171–176, 1990.

[5] R. H. Hou, S. Y. Kuo, and Y. P. Chang, "Efficient allocation of testing resources for software module testing based on the hyper-geometric distribution software reliability growth model," *IEEE Transactions on Reliability*, vol. 45, no. 4, pp. 541–549, 1996.

[6] M. Xie and B. Yang, "Optimal testing-time allocation for modular systems," *International Journal of Quality & Reliability Management*, vol. 18, no. 8, pp. 854–863, 2001.

[7] C. Y. Huang and M. R. Lyu, "Optimal testing resource allocation, and sensitivity analysis in software development," *IEEE Transactions on Reliability*, vol. 54, no. 4, pp. 592–603, 2005.

[8] P. C. Jha, D. Gupta, B. Yang, P. K. Kapur, "Optimal testing resource allocation during module testing considering cost, testing effort and reliability," *Computers & Industrial Engineering*, vol. 57, no. 3, pp. 1122–1130, 2009.

[9] R. Pietrantuono, S. Russo, and K. S. Trivedi, "Software reliability and testing time allocation: An architecture-based approach," *IEEE Transactions on Software Engineering*, vol. 36, no. 3, pp. 323–337, 2010.

[10] L. Fiondella and S. S. Gokhale, "Optimal allocation of testing effort considering software architecture," *IEEE Transactions on Reliability*, vol. 61, no. 2, pp. 580–589, 2012.

[11] D. Gong, W. Zhang, and X. Yao, "Evolutionary generation of test data for many paths coverage based on grouping," *Journal of Systems & Software*, vol. 84, no. 12, pp. 2222–2233, 2011.

[12] D. Gong, C. Zhang, T. Tian, and Z. Li, "Reducing scheduling sequences of message-passing parallel programs," *Information & Software Technology*, vol. 80, pp. 217–230, 2016.

[13] T. Tian and D. Gong, "Test data generation for path coverage of message-passing parallel programs based on co-evolutionary genetic algorithms," *Automated Software Engineering*, vol. 23, no. 3, pp. 469–500, 2016.

[14] D. Gong, G. Zhang, X. Yao, and F. Meng, "Mutant reduction based on dominance relation for weak mutation testing," *Information & Software Technology*, vol. 81, pp. 82–96, 2017.

[15] X. Yao, "Some recent work on multi-objective approaches to search-based software engineering," in *Proceedings of the 5th International Symposium on Search Based Software Engineering*, St. Petersburg, Russia, August 24–26, pp. 4–15, 2013.

[16] L. L. Minku, D. Sudholt, and X. Yao, "Improved evolutionary algorithm design for the project scheduling problem based on runtime analysis," *IEEE Transactions on Software Engineering*, vol. 40, no. 1, pp. 83–102, 2013.

[17] Z. Wang, K. Tang, and X. Yao, "A multi-objective approach to testing resource allocation in modular software systems," in *Proceedings of the IEEE Congress on Evolutionary Computation*, Hong Kong, China, June 1–6, pp. 1148–1153, 2008.

[18] Z. Wang, K. Tang, and X. Yao, "Multi-objective approaches to optimal testing resource allocation in modular software systems," *IEEE Transactions on Reliability*, vol. 59, no. 3, pp. 563–575, 2010.

[19] M. Pavithra, "Optimal testing resource allocation problems in software system using heuristic algorithm," *Bonfring International Journal of Software Engineering and Soft Computing*, vol. 2, no. 4, pp. 1–9, 2012.

[20] S. Yu, F. Dong, and B. Li, "Optimal testing resource allocation for modular software systems based-on multi-objective evolutionary algorithms with effective local search strategy," in *Proceedings of the IEEE Workshop on Memetic Computing*, Singapore, April 16–19, pp. 1–8, 2013.

[21] B. Yang, Y. Hu, and C. Y. Huang, "An architecture-based multi-objective optimization approach to testing resource allocation," *IEEE Transactions on Reliability*, vol. 64, no. 1, pp. 497–515, 2015.

[22] Y. Lu, F. Yue, G. F. Zhang, Z. P. Su, and Y. Q. Wang, "Model and solution to testing resource dynamic allocation for series-parallel software systems," *Journal of Software*, vol. 27, no. 8, pp. 1964–1977, 2016.

[23] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[24] K. Deb and J. Sundar, "Reference point based multi-objective optimization using evolutionary algorithms," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, Seattle, WA, USA, July. 08–12, pp. 635–642, 2006.

[25] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Reliability Engineering & System Safety*, vol. 91, no. 9, pp. 992–1007, 2006.

[26] M. Li, S. Yang, K. Li, and X. Liu, "Evolutionary algorithms with segment-based search for multiobjective optimization problems," *IEEE Transactions on Cybernetics*, vol. 44, no. 8, pp. 1295–1313, 2013.

[27] R. J. Larsen and M. L. Marx, *An Introduction to Mathematical Statistics and Its Applications*, 3rd ed. Essex, England: Prentice Hall, 2000.

[28] W. Kuo and R. Wan, "Recent advances in optimal reliability allocation," *IEEE Transactions on Systems Man & Cybernetics Part A: Systems & Humans*, vol. 37, no. 2, pp. 143–156, 2007.

[29] S. Yamada, H. Ohtera, and H. Narihisa, "Software reliability growth models with testing-effort," *IEEE Transactions on Reliability*, vol. 35, no. 1, pp. 19–23, 1986.

[30] S. Yamada, J. Hishitani, and S. Osaki, "Software-reliability growth with a Weibull test-effort: A model and application," *IEEE Transactions on Reliability*, vol. 42, no. 1, pp. 100–106, 1993.

[31] M. Jorgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Transactions on Software Engineering*, vol. 33, no. 1, pp. 33–53, 2007.

[32] A. Kumar and K. Malik, "Voting mechanisms in distributed systems," *IEEE Transactions on Reliability*, vol. 40, no. 5, pp. 593–600, 1991.

[33] M. R. Lyu, S. Rangarajan, and A. P. A. van Moorsel, "Optimal allocation of test resources for software reliability growth modeling in software development," *IEEE Transactions on Reliability*, vol. 51, no. 2, pp. 183–192, 2002.

[34] K. M. Bretthauer and B. Shetty, "The nonlinear resource allocation problem," *Operations Research*, vol. 43, no. 4, pp. 670–683, 1995.

[35] H. Wang, L. Jiao, and X. Yao, "Two_Arch2: An improved two-archive algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 4, pp. 524–541, 2015.

[36] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009.

[37] Y. G. Woldesenbet, B. G. Tessema, and G. G. Yen, "Constraint handling in multi-objective evolutionary optimization," in *Proceedings of the IEEE Congress on on Evolutionary Computation*, Swissôtel The Stamford, Singapore, Septemper. 25–28, pp. 3077–3084, 2007.

[38] E. Mezura-Montes abd C. A. C. Coello, "Constraint-handling in nature-inspired numerical optimization: Past, present and future," *Swarm & Evolutionary Computation*, vol. 1, no. 4, pp. 173–194, 2011.

[39] H. K. Singh, A. Isaacs, T. Ray, and W. Smith, "Infeasibility driven evolutionary algorithm (IDEA) for engineering design optimization," in *Proceedings of the 21st Australasian Joint Conference on Artificial Intelligence*, Auckland, New Zealand, December 1–5, pp. 104–115, 2008.

[40] H. K. Singh, A. Isaacs, T. T. Nguyen, T. Ray, and X. Yao, "Performance of infeasibility driven evolutionary algorithm (IDEA) on constrained dynamic single objective optimization problems," in *Proceedings of the IEEE Congress on Evolutionary Computation*, Trondheim, Norway, May 18–21, pp. 3127–3134, 2009.

[41] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, 1st ed. Chichester, England: John Wiley & Sons, 2001.

[42] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, no. 2, pp. 115–148, 1995.

[43] J. M. Steele, *The Cauchy-Schwarz Master Class: An Introduction to the Art of Mathematical Inequalities*, Cambridge, England: Cambridge University Press, 2004.

[44] E. Zitzler and L. Thiele, "Multiobjective evolutionary algprithms: A comparsion case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[45] S. Jiang, Y.-S. Ong, J. Zhang, and L. Feng, "Consistencies and contradictions of performance metrics in multiobjective optimization," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2391–2404, 2014.

[46] H. Wang, Y. Jin, and X. Yao, "Diversity assessment in many-objective optimization," *IEEE Transactions on Cybernetics*, vol. 47, no. 6, pp. 1510–1522, 2017.

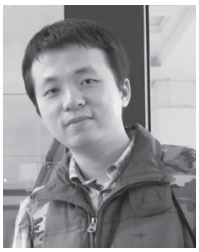
[47] P. Sprent and N. C. Smeeton, *Applied Nonparametric Statistical Methods*, 3rd ed. London, England: Chapman and Hall/CRC Press, 2000.

Guofu Zhang (M'11) received the B.S. and Ph.D. degrees in computer science from Hefei University of Technology, Hefei, China, in 2002 and 2008, respectively.

He is currently an Associate Professor with School of Computer and Information, Hefei University of Technology. His current research interests include multi-agent systems, evolutionary computation, and search-based software engineering.

Zhaopin Su (M'14) received the B.S. and Ph.D. degrees in computer science from Hefei University of Technology, Hefei, China, in 2003 and 2008, respectively.

She is currently an Associate Professor with School of Computer and Information, Hefei University of Technology. Her current research interests covers evolutionary computation, disaster emergency decision-making, and multimedia security.



Miqing Li received the Ph.D. degree in computer science from the Department of Computer Science, Brunel University London, UK in 2015. He is currently a research fellow in CERCIA, School of Computer Science, University of Birmingham, UK.

His research focuses on evolutionary multi- and many-objective optimisation, including algorithm design, performance assessment, test problem construction and visualisation, and their application in diverse fields.

Feng Yue received the B.S., M.S., and Ph.D. degrees in computer science from Hefei University of Technology, Hefei, China, in 2004, 2009, and 2015, respectively.

He is currently an Associate Professor with the School of Computer and Information, Hefei University of Technology. His current research interests include software engineering, multi-agent systems, and evolutionary computation.

Jianguo Jiang received the M.S. degree in computer science from Hefei University of Technology, Hefei, China, in 1989.

He is currently a Professor with the School of Computer and Information, Hefei University of Technology, China. His current research interests include automatic control, image processing, and multi-agent systems.

Xin Yao (M'91-SM'96-F'03) received the B.Sc. degree from the University of Science and Technology of China (USTC), Hefei, China, in 1982, the M.Sc. degree from the North China Institute of Computing Technology, Beijing, China, in 1985, and the Ph.D. degree from USTC in 1990.

He is currently a Chair Professor of Computer Science at the Southern University of Science and Technology, Shenzhen, China, and a part-time Professor of Computer Science at the University of Birmingham, UK. He is an IEEE Fellow, and a Distinguished Lecturer of IEEE Computational Intelligence Society (CIS). His major research interests include evolutionary computation, ensemble learning, and their applications in software engineering. His research won the 2001 IEEE Donald G. Fink Prize Paper Award, 2010, 2016, and 2017 IEEE Transactions on Evolutionary Computation Outstanding Paper Awards, 2010 BT Gordon Radley Award for Best Author of Innovation (Finalist), 2011 IEEE Transactions on Neural Networks Outstanding Paper Award, and many other best paper awards. He received the prestigious Royal Society Wolfson Research Merit Award in 2012 and the IEEE CIS Evolutionary Computation Pioneer Award in 2013. He was the the President (2014-15) of IEEE CIS, and the Editor-in-Chief (2003-08) of IEEE Transactions on Evolutionary Computation.