# Enhanced constraint handling for reliability-constrained multiobjective testing resource allocation

Su, Zhaopin; Zhang, Guofu; Yue, Feng; Zhan, Dezhi; Li, Miqing; Li, Bin; Yao, Xin

*DOI:*
10.1109/TEVC.2021.3055538

# Enhanced Constraint Handling for Reliability-Constrained Multi-objective Testing Resource Allocation

Zhaopin Su, Guofu Zhang, Feng Yue, Dezhi Zhan, Miqing Li, Bin Li, *Member, IEEE,* and Xin Yao, *Fellow, IEEE*

***Abstract*—The multi-objective testing resource allocation problem (MOTRAP) is how to efficiently allocate the finite testing time to various modules, with the aim of optimizing system reliability, testing cost, and testing time simultaneously. To deal with this problem, a common approach is to use multi-objective evolutionary algorithms (MOEAs) to seek a set of trade-off solutions between the three objectives. However, such a trade-off set may contain a substantial proportion of solutions with very low reliability level, which consume lots of computational resources but may be valueless to the software project manager. In this paper, an MOTRAP model with a pre-specified reliability is first proposed. Then, new lower bounds on the testing time invested in different modules are theoretically deduced from the necessary condition for the achievement of the given reliability, based on which an exact algorithm for determining the new lower bounds is presented. Moreover, several enhanced constraint handling techniques (ECHTs) derived from the new bounds are successively developed to be combined with MOEAs to correct and reduce the constraint violation. Finally, the proposed ECHTs are evaluated in comparison with various state-of-the-art constraint solving approaches. The comparative results demonstrate that the proposed ECHTs can work well with MOEAs, make the search focus on the feasible region of the pre-specified reliability, and provide the software project manager with better and more diverse, satisfactory choices in test planning.**

***Index Terms*—Multi-objective testing resource allocation, reliability constraint, evolutionary algorithms, constraint handling.**

Manuscript received June 07, 2020; revised December 2, 2020. This work was supported in part by the Anhui Provincial Key Research and Development Program under Grant No. 202004d07020011, in part by the the National Natural Science Foundation of China under Grant No. U19B2044, in part by the MOE (Ministry of Education in China) Project of Humanities and Social Sciences under Grants 19YJC870021 and 18YJC870025, and in part by the Fundamental Research Funds for the Central Universities under Grants PA2020GDKC0015, PA2019GDQT0008, and PA2019GDPK0072.

Z. Su, G. Zhang (the corresponding author), F. Yue, and D. Zhan are with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230601, China, with the Intelligent Interconnected Systems Laboratory of Anhui Province (Hefei University of Technology), Hefei 230009, China, with the Key Laboratory of Knowledge Engineering with Big Data (Hefei University of Technology), Ministry of Education, Hefei 230601, China, and with the Anhui Province Key Laboratory of Industry Safety and Emergency Technology (Hefei University of Technology), Heifei 230601, China (e-mail: szp@hfut.edu.cn; zgf@hfut.edu.cn; yuefeng@hfut.edu.cn; dzzhan@mail.hfut.edu.cn).

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

B. Li is with the School of Information Science and Technology, University of Science and Technology of China, Hefei 230027, China (e-mail: binli@ustc.edu.cn).

X. Yao is with the Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China, and also with CERCIA, School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K. (e-mail: xiny@sustech.edu.cn).

## I. Introduction

SOFTWARE testing is an investigation conducted to detect software failures so that defects may be discovered and corrected and is of significant importance to the software project development. However, for some time past it has been widely accepted that almost half of the total amount of software development resources are expended in software testing [1]. In fact, the number of possible tests for even simple software modules is practically infinite, but the available testing time and resources are always tight, especially for large-scale, complex software projects. In such situations, a primary task for a software project manager is to develop a testing resource allocation (TRA) plan to derive the maximal system reliability from the available resources. Consequently, to date, a lot of research effort has been spent to formulate models and design algorithms to optimize the allocation of the limited testing resources [2].

In early studies, TRA was modeled as a single-objective optimisation problem, such as minimizing cost with the reliability or time constraint [1], or minimizing the consumed time with a reliability requirement [3]. Some traditional mathematical programming methods, such as Lagrange multipliers and dynamic programming, were used to solve the above models. In the last decade, TRA has been formulated as a multi-objective optimisation problem (MOP), called MOTRAP, which aims at maximizing reliability and minimizing cost and time [4]. It is clear that reliability, cost, and time are usually conflicting with each other. The traditional mathematical programming methods tend to be very susceptible to the discontinuity and the shape of the Pareto front and normally generate only a single non-dominated solution per run [5]. On the other side, multi-objective evolutionary algorithms (MOEAs) [6]–[8], such as NSGA-II [9] and MOEA/D [10], are less susceptible to the shape or continuity of the Pareto front and can manage a set of solutions and generate several elements of the Pareto optimal set in a single run [5]. Consequently, MOEAs have become very popular as a multi-objective optimizer of the MOTRAP [4], [11]–[13]. These MOEAs can provide the software project manager with a lot of additional choices that show different trade-offs between reliability, cost, and time, and hence can facilitate more informed planning of a testing period.

Although the above MOEAs can help the software project manager to organize the testing time in a more efficient way, in fact, the system reliability is always the primary driver of

software testing [1]. In traditional single-objective optimisation approaches, the reliability constraint has been widely used in different TRA models to make the final solution meet a certain reliability requirement. Unfortunately, the existing MOEAs [4], [11]–[13] for solving the MOTRAP have typically focused on the time constraint rather than the reliability constraint and produced a great number of solutions with very low reliability level, which may be valueless and unacceptable for the software project manager. In other words, only a small number of solutions in the trade-off set may be useful in practice. This leads to a big waste of computational efforts and a great deal of information redundancy, obviously deviating from the original intention of the MOTRAP. It should be noted that although we can consider cost and time, it is doubtless that the key issue for software testing is still reliability. The software project manager would be very interested in TRA schemes that can achieve the desired reliability for customer satisfaction. In such situations, the software project manager is eager to see that each solution in the trade-off set of the MOTRAP has the satisfactory reliability and is available for reference use in practical testing. Consequently, it should be natural for the reliability constraint to be further considered in the MOTRAP. This raises two new, open questions: How to make the solutions in the trade-off set meet the pre-specified reliability requirement? And how to make the search focus on the feasible region of the satisfactory reliability?

To answer the above questions, in this work, four aspects are studied: 1) The mathematical model for the reliability-constrained MOTRAP is constructed, considering the three objectives of reliability, cost, and time, and a variety of time and reliability constraints; 2) New lower bounds on the time invested in different modules are theoretically deduced from the necessary condition for the achievement of the pre-specified reliability, based on which an exact algorithm for determining the new lower bounds is developed; 3) Several enhanced constraint handling techniques (ECHTs) are developed on the basis of the new bounds to make up for the defects of MOEAs and make MOEAs explore in the acceptable solution region of the pre-specified reliability by correcting and reducing the constraint violation; 4) Three state-of-the-art constraint solving approaches designed specifically for the MOTRAP, two general constrained MOEAs for solving constrained MOPs (CMOPs), and 195 instances are used in our experimental studies.

This paper is organized as follows. Section II presents an overview of the related work. Section III constructs the mathematical model of the reliability-constrained MOTRAP. In Section IV, the derivation of new lower bounds and an exact algorithm for determining the new bounds are developed. Section V details the proposed ECHTs on the basis of the new bounds. Experimental analyses are presented in Section VI. Conclusions are drawn in Section VII.

## II. Literature Review

This section focuses on most relevant work on evolutionary algorithms (EAs) based solutions to TRA, especially constraint handling techniques in MOEAs for solving the MOTRAP. Of course, other classical methods can be found in [2], [14].

### A. EAs Based TRA

In search-based software engineering, researchers and practitioners use EAs to find near-optimal, feasible solutions [15]. An early effort was from Dai et al. [16] who modeled TRA in terms of a weighted sum of reliability and cost and applied genetic algorithm (GA) to search good enough solutions. Kapur et al. [17] used GA to minimized cost under constrained resources and reliability. Gao and Xiong [18] improved Kapur et al.'s work with GA and local search strategy. Aggarwal et al. [19] introduced GA to maximize the total fault removal or minimize total testing resources. To deal with TRA under uncertainties, Chaudhary et al. [20] adopted differential evolution (DE) to maximize the number of detected faults in each testing period with the re-estimated modular parameters. Although the above EAs were empirically validated and technically feasible, it should be pointed out that it is impossible for a single solution of TRA to fully characterize the optimal balance between reliability, cost, and time.

### B. Constraint Handling Based MOEAs for MOTRAP

To our best knowledge, Wang et al. [4] made the first attempt to formulate TRA as an MOP and use an MOEA to solve it. Then, Yu et al. [13] embedded effective local search into MOEA/D and considered both reliability and cost. To reduce the impact of uncertainty in TRA models, Pietrantuono et al. [21] embedded Monte Carlo simulations into MOEAs and provided interval-solutions instead of point-solutions under different possible values of model parameters. However, there is no constraint solving approach in the above mentioned studies, which may weaken the performance of MOEAs.

In search-based software engineering, the individual-repair based constraint handling techniques have become increasingly important [22]. Wang et al. [4] and Yang et al. [12] randomly migrated infeasible solution into the feasible region of the search space by reducing each element value of the solution vector. This technique can ensure each repaired individual to be feasible but might destroy useful genetic information that should be inherited from parent individuals. To link up the constraint handling with the nature of the used NSGA-II, Zhang et al. [11] handled constraint violations on time within the crossover and mutation operators. Particularly, they only amend the gene values that are supposed to be changed, which is beneficial to preserve the useful evolutionary information in offspring.

It is clear that the MOTRAP belongs to CMOPs. To date, a lot of general constrained MOEAs have sprung up like mushrooms for solving CMOPs. Liu et al. [23] evaluated the performance of different combinations between indicator-based MOEAs and constraint-handling techniques. Liu and Wang [24] transformed CMOPs into constrained single objective optimisation problems and used DE and the feasibility rule to generate high-quality candidate solutions that are utilized as the original population of MOEAs. Fan et al. [25] proposed an integrated method, called MOEA/D-IEpsilon, to combine MOEA/D with an improved epsilon constraint-handling method that controls the relaxation of constraints by

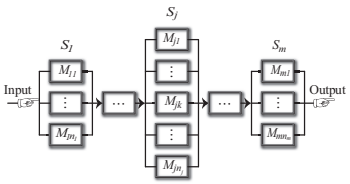



Fig. 1. The classical structure of a parallel-series modular software system.

adjusting the epsilon level dynamically according to the ratio of feasible to total solutions in the current population. Yang et al. [26] adjusted the epsilon level in line with the maximum and minimum constraint violation values of infeasible individuals to prevent MOEAs being converged to feasible solutions prematurely. To make good use of infeasible solutions, Tian et al. [27] presented a coevolutionary constrained multi-objective optimisation framework, called CCMO, to evolve two different populations simultaneously to solve the original problem and a helper problem derived from the original one, respectively, sharing useful information between the two populations. Peng et al. [28] also used one population to explore feasible regions and the other population to explore the entire space on the basis of a set of directed weight vectors to guide the search to a wide range of promising regions. Although the above constraint solving approaches have been validated to play a valuable role in the exploration and exploitation of MOEAs for solving CMOPs, few attempts have been made to link up the problem's knowledge with the corresponding constraint handling techniques.

Different from the previous studies, in this work we theoretically deduce the new lower bounds on the time invested in different modules on the basis of the pre-specified reliability to reduce the solution space, and at the same time, we develop an exact algorithm to determine the new lower bounds. Particularly, several ECHTs are proposed to utilize the theoretical lower bounds to correct and reduce the constraint violation.

## III. Problem Formulation

Following the practice in [4], [11], in this work, we discuss the MOTRAP on the basis of the parallel-series modular software model, which is very simple, straightforward, and widely-used in real-world software systems. The incorporation of the equally popular architecture-based model [3] is proposed as future work.

The typical structure of a parallel-series modular software system is shown in Fig. 1. As can be seen, the whole system has $m \in \mathbb{N}$ serial subsystems, $S_1, \ldots, S_m$, each of which contains $n_j \in \mathbb{N}$ $(j = 1, \ldots, m)$ modules, $M_{j1}, \ldots, M_{jn_j}$. In software testing, the available testing time $\mathcal{T}^* \in \mathbb{R}^+$ refers to the total working hours that can be calculated according to the number of software testers and the working hours of each tester [4], [12]. For further illustrations, we use $t_{jk} \in \mathbb{R}_0^+$ $(k = 1, \ldots, n_j)$ to denote the testing time invested in a module $M_{jk}$. Obviously, the actual consumed testing time $T \in \mathbb{R}^+$ of the whole system can be computed by [4].

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

The relationship between the reliability and the invested testing time is often described by the software reliability growth models (SRGMs) [1]. In SRGMs, the fault removing is recognized as a nonhomogeneous Poisson process, based on which the achieved reliability $r_{jk} \in [0, 1]$ of the module $M_{jk}$ can be evaluated as below [1].

$$r_{jk} = \exp\left[-\lambda a_{jk} b_{jk} \exp\left(-b_{jk} t_{jk}\right)\right] \tag{1}$$

where $\lambda \in \mathbb{R}^+$ is a predefined constant and denotes the period of workable time or the estimated system life; $a_{jk} \in \mathbb{R}^+$ represents the mean value of the total errors in $M_{jk}$; $b_{jk} \in \mathbb{R}^+$ is the rate of detected errors for $M_{jk}$. Notice that both $a_{jk}$ and $b_{jk}$ depict the time-reliability relationship for different $M_{jk}$ and their values are usually determined on the basis of a sequence of software failure times corresponding to $M_{jk}$ and maximum likelihood estimation [1].

A subsystem $S_j$ may contain more than one module (i.e., $n_j \geq 1$). These parallel modules are used to improve the performance of $S_j$ due to the fact that $S_j$ can work permanently until all the parallel modules in $S_j$ are no longer available. Accordingly, the expected reliability $R_j \in [0, 1]$ of $S_j$ is [1]

$$R_j = 1 - \prod_{k=1}^{n_j} \left(1 - r_{jk}\right) \tag{2}$$

Furthermore, according to the multiplication rule, the achieved reliability $R \in [0, 1]$ of the whole system is the total reliability of the $m$ serial subsystems [1], [4], namely,

$$R = \prod_{j=1}^{m} R_j \geq \mathcal{R}^*$$

where $\mathcal{R}^* \in [0, 1]$ is the pre-specified reliability [1].

The testing cost refers to the amount being spent on testing and measuring quality as well as the cost of corrections. Usually, the potential cost $C_{jk} \in \mathbb{R}^+$ derived from the achieved reliability $r_{jk}$ of $M_{jk}$ is defined as follows [16]:

$$C_{jk} = c_1^{jk} \exp\left(c_2^{jk} r_{jk} - c_3^{jk}\right)$$

where $c_1^{jk}, c_2^{jk}, c_3^{jk} \in \mathbb{R}^+$ are cost increment rates corresponding to $M_{jk}$. According to this, the total testing cost $C \in \mathbb{R}^+$ consumed by the whole system is [4], [16]

$$C = \sum_{j=1}^{m} \sum_{k=1}^{n_j} C_{jk}$$

Finally, the tri-objective optimisation model for the reliability-constrained MOTRAP can be listed as below.

$$\begin{aligned} \min_{\mathbf{t}} \quad & f(\mathbf{t}) = (1 - R, C, T) \\ \text{s.t.} \quad & \mathbf{t} = (t_{11}, \ldots, t_{mn_m}) \in \mathbb{R}^D \\ & R \geq \mathcal{R}^* \\ & T \leq \mathcal{T}^* \\ & 0 \leq t_{jk} \leq \mathcal{T}^*, j \in \{1, \ldots, m\}, k \in \{1, \ldots, n_j\} \end{aligned} \tag{3}$$

where $\mathbf{t}$ is a solution of the MOTRAP; $D = \sum_{j=1}^{m} n_j$ is the number of decision variables; $\mathcal{R}^*$ is the lower bound constraint for $R$; $\mathcal{T}^*$ is the upper bound constraint for $T$ and $t_{jk}$. In (3), it is expected that at least $\mathbf{t}$ is able to achieve the desired reliability $\mathcal{R}^*$ within the available testing time $\mathcal{T}^*$.

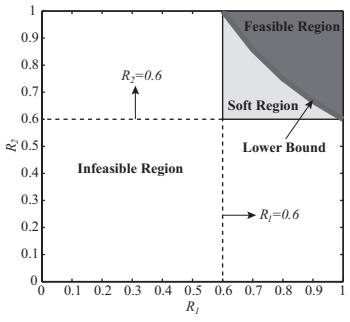



Fig. 2. An illustration of the solution space of a 2-D product inequality $R_1R_2 \geq 0.6$.

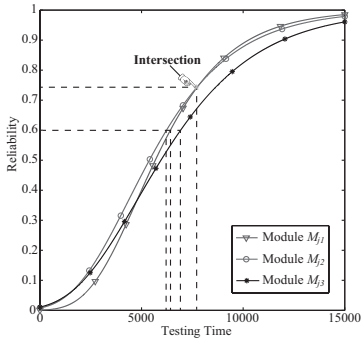



Fig. 3. An illustration of the time-reliability relationship for modules with different $a_{jk}$ and $b_{jk}$ in $S_j$ under $\mathcal{R}^* = 0.6$.

## IV. New Lower Bounds on Testing Time

In this section, we try to utilize the new constraints in (3) to derive new bounds to reduce the solution space. It is highly expected that, the derived new bounds can help MOEAs to restrict solutions to a specific, acceptable range, and avoid solutions wandering into infeasible or unacceptable regions.

First, from (3), we can obtain that

$$\prod_{j=1}^{m} R_j \geq \mathcal{R}^* \tag{4}$$

Obviously, (4) is an $m$-dimensional continuous product inequality whose solutions are enclosed by an $m$-dimensional space. When $m$ is big, it is very difficult to accurately determine the hard range of each $R_j$. Accordingly, a compromise approach is to find a soft, but reasonable region for $R_j$. Since $\forall j \in \{1,\ldots,m\}$, $R_j \in [0,1]$, at least we can obtain the following necessary conditions, intuitively and immediately from (4):

$$\forall j \in \{1,\ldots,m\}, R_j \geq \mathcal{R}^* \tag{5}$$

That is, each subsystem's reliability must reach to at least $\mathcal{R}^*$. This is because as long as $\exists j \in \{1,\ldots,m\}$, $R_j < \mathcal{R}^*$, it is impossible for (4) to hold, even if each of the other subsystems' reliability is equal to 1. To provide visual illustrations, Fig. 2 shows the solution space of a 2-D product inequality $R_1R_2 \geq 0.6$ (i.e., $m = 2$ and $\mathcal{R}^* = 0.6$). It can be observed that in the whole solution space, the feasible region is very tiny and most areas are infeasible. Particularly, the yellow rectangle enclosed by $R_1 \geq 0.6$ and $R_2 \geq 0.6$ can just cover the lower bound and the feasible region, exclude most of the infeasible solutions from the rectangle, and can be viewed as the best soft region for solution exploration. Based on this observation, we take into account (5) as a reasonable soft condition for achieving $\mathcal{R}^*$.

Putting (2) and (5) together, we can obtain that $\forall j \in \{1,\ldots,m\}$, $\prod_{k=1}^{n_j}(1-r_{jk}) \leq 1-\mathcal{R}^*$. Similarly, we can achieve the following necessary condition for each $S_j$: $\exists k \in \{1,\ldots,n_j\}$, $1-r_{jk} \leq \sqrt[n_j]{1-\mathcal{R}^*}$. That is, for each $S_j$, there exists at least a module $k \in \{1,\ldots,n_j\}$, satisfying $r_{jk} \geq 1-\sqrt[n_j]{1-\mathcal{R}^*}$. Taking this and (1) together, we can obtain that in $S_j$, at least $\exists k \in \{1,\ldots,n_j\}$,

$$t_{jk} \geq \frac{\ln\left[\frac{-\lambda a_{jk} b_{jk}}{\ln\left(1-\sqrt[n_j]{1-\mathcal{R}^*}\right)}\right]}{b_{jk}}$$

Notice that different $a_{jk}$ and $b_{jk}$ will result in different $t_{jk}$. For instance, Fig. 3 depicts the time-reliability relationship for modules with different $a_{jk}$ and $b_{jk}$ in terms of SRGMs. It can be observed that, the module's reliability increases with the testing time, showing different growth rates. In other words, to achieve the same reliability, the time consumed by modules in $S_j$ is completely different. Since the total available time is always tight, the greatest strength of a wise choice is that the module which consumes the least testing time in $S_j$ can be chosen to save the total available time as much as possible. This is because saving the total testing time is of great significance to shorten the testing cycle and save the testing cost, especially human resource costs in practice [1]. For this purpose, we denote

$$\tau_{jk} = \frac{\ln\left[\frac{-\lambda a_{jk} b_{jk}}{\ln\left(1-\sqrt[n_j]{1-\mathcal{R}^*}\right)}\right]}{b_{jk}} \tag{6}$$

as the needed time for each module in $S_j$ to achieve the required reliability. Then, $\tau_j^* = \min_k(\tau_{jk})$ represents the minimal time derived from the most time-efficient module in $S_j$. Obviously, this special module should be prioritized in TRA to save the limited testing time. Without loss of generality, we denote the special module as $M_{j\kappa_j^*}$ ($\kappa_j^* \in \{1,\ldots,n_j\}$), namely, $\kappa_j^* = \arg\min_k(\tau_{jk})$.

It should be noted that $M_{j\kappa_j^*}$ is not necessarily always the right choice. See Fig. 3, under the required $\mathcal{R}^* = 0.6$, $M_{j2}$ consumes the least time, followed by $M_{j1}$ and $M_{j3}$. However, the curves of $M_{j1}$ and $M_{j2}$ has an intersection, behind which $M_{j1}$ can achieve higher reliability with lower testing time than $M_{j2}$ and $M_{j3}$. In other words, $M_{j1}$ is easier to achieve a higher reliability with lower time than $M_{j2}$ and is obviously a better choice. Therefore, for each $M_{j\kappa_j^*}$ in $S_j$, we need to further check whether the reliability curve of $M_{j\kappa_j^*}$ has an intersection with each of the reliability curves of the other modules in $S_j$. Assume that $M_{j\kappa_j^*}$ has an intersection with $M_{jk}$ ($k \neq \kappa_j^*$), whose time value is denoted as $\vec{\tau}_{jk}$. On the basis of (1), $\vec{\tau}_{jk}$ can be calculated as below.

$$\vec{\tau}_{jk} = \frac{\ln(a_{j\kappa_j^*} b_{j\kappa_j^*}) - \ln(a_{jk} b_{jk})}{b_{j\kappa_j^*} - b_{jk}} \tag{7}$$

**Algorithm 1** The exact algorithm for determining the new lower bounds.

**Input:** $\mathcal{R}^*$
**Output:** $\tau_{jk}^L$ ($j \in \{1,\dots,m\}$, $k \in \{1,\dots,n_j\}$)
**for** $j := 1$ to $m$ **do**
  **for** $k := 1$ to $n_j$ **do**
    compute $\tau_{jk}$ according to (6)
  **end for**
**end for**
**for** $j := 1$ to $m$ **do**
  $\tau_j^* \leftarrow \min_k(\tau_{jk}), \kappa_j^* \leftarrow \arg\min_k(\tau_{jk})$
**end for**
**for** $j := 1$ to $m$ **do**
  **for** $k := 1$ to $n_j$ **do**
    **if** $M_{jk}$ intersects with $M_{j\kappa_j^*}$ ($k \neq \kappa_j^*$) **then**
      calculate $\vec{\tau}_{jk}$ based on (7)
      **if** $\vec{\tau}_{jk} > \tau_j^*$ **then**
        $\kappa_j^* \leftarrow k, \tau_j^* \leftarrow \vec{\tau}_{jk}$
      **end if**
    **end if**
  **end for**
**end for**
**for** $j := 1$ to $m$ **do**
  **for** $k := 1$ to $n_j$ **do**
    **if** $k = \kappa_j^*$ **then**
      $\tau_{jk}^L \leftarrow \tau_j^*$
    **else**
      $\tau_{jk}^L \leftarrow 0$
    **end if**
  **end for**
**end for**

If $\vec{\tau}_{jk} > \tau_j^*$, execute $\kappa_j^* \leftarrow k$ and $\tau_j^* \leftarrow \vec{\tau}_{jk}$ to update the most time-efficient module.

Having $\kappa_j^*$ and $\tau_j^*$ in hand, we redefine the lower bound constraint for each $t_{jk}$ as $\tau_{jk}^L$ to classify and prioritize different modules. Then, for $\forall j \in \{1,\dots,m\}$ and $\forall k \in \{1,\dots,n_j\}$, we have

$$\tau_{jk}^L = \begin{cases} \tau_j^*, k = \kappa_j^* \\ 0, k \neq \kappa_j^* \end{cases} \tag{8}$$

Importantly, these new bounds can tell us which modules are time-efficient or time-consuming to achieve the pre-specified reliability. Similarly, for the entire system, the lower bound constraint for $T$ can be defined as $\mathcal{T}^L = \sum_{j=1}^{m}\sum_{k=1}^{n_j}\tau_{jk}^L$.

The basic idea for determining the new lower bounds is shown in Algorithm 1. It can be easily seen that the worst-case complexity of Algorithm 1 is $O(D)$. Solvers can use Algorithm 1 to prepare the new lower bounds automatically. Now, it can be obtained that each $t_{jk}$ must satisfy

$$t_{jk} \geq \tau_{jk}^L \tag{9}$$

and

$$\mathcal{T}^L \leq \sum_{j=1}^{m}\sum_{k=1}^{n_j} t_{jk} \leq \mathcal{T}^* \tag{10}$$

## V. Constraint Handling Techniques Using the New Bounds

The goal of this work is to design ECHTs for the reliability-constrained MOTRAP which can be combined with MOEAs to make up for the defects of MOEAs and improve the search effectiveness. Following the practice of existing studies [4], [11], we consider individual initialization, simulated binary crossover, and polynomial mutation as the basic evolutionary operators in which infeasible individuals with very low reliability level may be produced. We embed ECHTs in the process of the above evolutionary operators to correct and reduce the constraint violation in MOEAs. More specifically, the proposed ECHTs (i.e., Algorithms 1-4) are driven by both time and reliability. Algorithm 1 is an important basic part of the proposed ECHTs. In Algorithms 2-4, once an individual is infeasible in timeline, it is repaired according to both $\tau_{jk}^L$ and $\mathcal{T}^*$, in which $\tau_{jk}^L$ is determined by Equation (8) to ensure that those most time-efficient modules can be chosen first. As such, a high system reliability with low time consumption can be achieved.

Note that these ECHTs are problem-specific and work only for the reliability-constrained MOTRAP. Algorithms 1-4 are closely connected with each other and are an organic whole of four, without one of which the others will be ineffective. The primary aim of Algorithms 2-4 is to utilize the new lower bounds determined by Algorithm 1 to make the search as close as possible to the feasible region, thereby a higher chance to produce feasible solutions with acceptable reliability level. In addition, the proposed ECHTs do not suggest any novel selection strategy due to the fact that no new infeasible individual will be created during selection. However, it should be noted that these ECHTs cannot guarantee that the repaired individual satisfies (4). This is because Constraint Set (5) is only a necessary condition for achieving the desired $\mathcal{R}^*$. Hence, the constrained dominance relation is suggested in mating selection or environmental selection. More specifically, an individual is said to dominate the other individual if any of the following conditions is true: this individual is feasible and the other individual is not; both the two individuals are infeasible but the constraint violation degree of this individual is smaller than that of the other individual; both the two individuals are feasible but this individual is no worse than the other individual for all objectives and at the same time, this individual is strictly better than the other individual in at least one objective.

### A. Individual Initialization

Each individual represents a solution $\mathbf{t}$ and each gene denotes a decision variable $t_{jk}$. The procedure of individual initialization with ECHTs is shown in Algorithm 2, where $rand(\tau_{jk}^L, \mathcal{T}^*)$ represents a random number in $[\tau_{jk}^L, \mathcal{T}^*]$ which is generated from a normal distribution. As shown in Algorithm 2, each $t_{jk}$ is generated randomly between the new lower bounds and the upper bounds: $t_{jk} \leftarrow rand(\tau_{jk}^L, \mathcal{T}^*)$. Once the sum of all $t_{jk}$ is beyond $\mathcal{T}^*$, each $t_{jk}$ will be scaled down as below.

$$\hat{t}_{jk} \leftarrow \tau_{jk}^L + (t_{jk} - \tau_{jk}^L)\frac{\mathcal{T}^* - \mathcal{T}^L}{\sum_{j=1}^{m}\sum_{k=1}^{n_j} t_{jk} - \mathcal{T}^L}$$

in which $\mathcal{T}^L$ is used to achieve a reasonable reduction proportion, maintain the previous diversity, and ensure the required reliability level. Notice that the new $\hat{t}_{jk}$ can always obey Constraint Sets (9) and (10). Moreover, it can be immediately seen that the worst-case complexity of Algorithm 2 is $O(D)$.

**Algorithm 2** The ECHTs for individual initialization.

**Input:** $\tau_{jk}^L$ ($j \in \{1,\dots,m\}$, $k \in \{1,\dots,n_j\}$) and $\mathcal{T}^*$
**Output:** a feasible individual
**for** $j := 1$ to $m$ **do**
  **for** $k := 1$ to $n_j$ **do**
    $t_{jk} \leftarrow rand(\tau_{jk}^L, \mathcal{T}^*)$
  **end for**
**end for**
**if** $\sum_{j=1}^{m}\sum_{k=1}^{n_j} t_{jk} > \mathcal{T}^*$ **then**
  **for** $j := 1$ to $m$ **do**
    **for** $k := 1$ to $n_j$ **do**
      $\hat{t}_{jk} \leftarrow \tau_{jk}^L + (t_{jk} - \tau_{jk}^L)\frac{\mathcal{T}^* - \mathcal{T}^L}{\sum_{j=1}^{m}\sum_{k=1}^{n_j} t_{jk} - \mathcal{T}^L}$
    **end for**
  **end for**
**end if**

### B. Simulated Binary Crossover

During the process of crossover, the infeasibility is caused only by the revised genes rather than non-revision genes. Accordingly, constraint handling must be operated only on the updated genes to preserve the essential genetic information in children.

The procedure of the ECHTs based simulated binary crossover is illustrated in Algorithm 3, where $sum^{o_1}, sum^{o_2}$ are the sums of all the gene values; $sum_{cs}^{o_1}, sum_{cs}^{o_2}$ are the sums of all the crossover-gene values; $\mathcal{T}_{cs}^{o_1}, \mathcal{T}_{cs}^{o_2}$ are the available testing time for the crossover genes; $\mathcal{T}_{cs}^L$ denotes the least testing time needed by all the crossover genes; $CR \in [0,1]$ is the predefined crossover rate; $rand(0,1)$, $rand(\tau_{jk}^L, y^l)$, and $rand(y^h, \mathcal{T}^*)$ represent random numbers in $[0,1]$, $[\tau_{jk}^L, y^l]$, and $[y^h, \mathcal{T}^*]$, respectively, which are generated from a normal distribution; $\beta \in \mathbb{R}_0^+$ is a uniformly sampled random number [9].

In Algorithm 3, $t_{jk}^{o_1}$ may be decreased and $t_{jk}^{o_2}$ may be enlarged after the crossover. Thus, it is possible that $t_{jk}^{o_1} < \tau_{jk}^L$ or $t_{jk}^{o_2} > \mathcal{T}^*$. In such cases, we choose the smaller of $y^l - \tau_{jk}^L$ and $\mathcal{T}^* - y^h$ to control the change range of gene values and avoid potential constraint violations in the two children. If $y^l - \tau_{jk}^L \le \mathcal{T}^* - y^h$, $t_{jk}^{o_1} \leftarrow rand(\tau_{jk}^L, y^l)$ and $t_{jk}^{o_2} \leftarrow y^h + y^l - t_{jk}^{o_1}$. Otherwise, $t_{jk}^{o_2} \leftarrow rand(y^h, \mathcal{T}^*)$ and $t_{jk}^{o_1} \leftarrow y^h + y^l - t_{jk}^{o_2}$. It can be found that the new $t_{jk}^{o_1}$ and $t_{jk}^{o_2}$ obey (9). When the sum of gene values violates $\mathcal{T}^*$, all the crossover-gene values will be reduced according to $\mathcal{T}_{cs}^L$ to maintain the previous evolutionary tendency and approach the acceptable reliability level. More specifically, if $sum^{o_1} > \mathcal{T}^*$ or $sum^{o_2} > \mathcal{T}^*$, each crossover-gene should be amended as follows:

$$\begin{cases} \hat{t}_{jk}^{o_1} \leftarrow \tau_{jk}^L + (t_{jk}^{o_1} - \tau_{jk}^L)\frac{\mathcal{T}_{cs}^{o_1} - \mathcal{T}_{cs}^L}{sum_{cs}^{o_1} - \mathcal{T}_{cs}^L} \\ \hat{t}_{jk}^{o_2} \leftarrow t_{jk}^{o_1} + t_{jk}^{o_2} - \hat{t}_{jk}^{o_1} \end{cases}$$

or

$$\begin{cases} \hat{t}_{jk}^{o_2} \leftarrow \tau_{jk}^L + (t_{jk}^{o_2} - \tau_{jk}^L)\frac{\mathcal{T}_{cs}^{o_2} - \mathcal{T}_{cs}^L}{sum_{cs}^{o_2} - \mathcal{T}_{cs}^L} \\ \hat{t}_{jk}^{o_1} \leftarrow t_{jk}^{o_1} + t_{jk}^{o_2} - \hat{t}_{jk}^{o_2} \end{cases}$$

It can be observed that the new $\hat{t}_{jk}^{o_1}$ and $\hat{t}_{jk}^{o_2}$ obey (9) and (10), and moreover, the worst-case complexity of Algorithm 3 is $O(D)$.

**Algorithm 3** The ECHTs for simulated binary crossover.

**Input:** two mating individuals $p_1$ and $p_2$
**Output:** two child individuals $o_1, o_2$
$sum^{o_1} \leftarrow 0, sum^{o_2} \leftarrow 0, sum_{cs}^{o_1} \leftarrow 0, sum_{cs}^{o_2} \leftarrow 0$
$\mathcal{T}_{cs}^{o_1} \leftarrow 0, \mathcal{T}_{cs}^{o_2} \leftarrow 0, \mathcal{T}_{cs}^L \leftarrow 0$
**for** $j := 1$ to $m$ **do**
  **for** $k := 1$ to $n_j$ **do**
    **if** $rand(0,1) \le CR$ **then**
      $y^h = \max(t_{jk}^{p_1}, t_{jk}^{p_2}), y^l = \min(t_{jk}^{p_1}, t_{jk}^{p_2})$
      $t_{jk}^{o_1} \leftarrow \frac{1}{2}[y^h + y^l - \beta(y^h - y^l)], t_{jk}^{o_2} \leftarrow y^h + y^l - t_{jk}^{o_1}$
      **if** $t_{jk}^{o_1} < \tau_{jk}^L$ or $t_{jk}^{o_2} > \mathcal{T}^*$ **then**
        **if** $y^l - \tau_{jk}^L \le \mathcal{T}^* - y^h$ **then**
          $t_{jk}^{o_1} \leftarrow rand(\tau_{jk}^L, y^l), t_{jk}^{o_2} \leftarrow y^h + y^l - t_{jk}^{o_1}$
        **else**
          $t_{jk}^{o_2} \leftarrow rand(y^h, \mathcal{T}^*), t_{jk}^{o_1} \leftarrow y^h + y^l - t_{jk}^{o_2}$
        **end if**
      **end if**
      $sum_{cs}^{o_1} \leftarrow sum_{cs}^{o_1} + t_{jk}^{o_1}, sum_{cs}^{o_2} \leftarrow sum_{cs}^{o_2} + t_{jk}^{o_2}$
      $\mathcal{T}_{cs}^L \leftarrow \mathcal{T}_{cs}^L + \tau_{jk}^L$
    **else**
      $t_{jk}^{o_1} \leftarrow t_{jk}^{p_1}, t_{jk}^{o_2} \leftarrow t_{jk}^{p_2}$
    **end if**
    $sum^{o_1} \leftarrow sum^{o_1} + t_{jk}^{o_1}, sum^{o_2} \leftarrow sum^{o_2} + t_{jk}^{o_2}$
  **end for**
**end for**
**if** $sum^{o_1} > \mathcal{T}^*$ **then**
  $\mathcal{T}_{cs}^{o_1} \leftarrow \mathcal{T}^* - (sum^{o_1} - sum_{cs}^{o_1})$
  **for** $j := 1$ to $m$ **do**
    **for** $k := 1$ to $n_j$ **do**
      **if** $t_{jk}^{o_1}$ is a crossover gene **then**
        $\hat{t}_{jk}^{o_1} \leftarrow \tau_{jk}^L + (t_{jk}^{o_1} - \tau_{jk}^L)\frac{\mathcal{T}_{cs}^{o_1} - \mathcal{T}_{cs}^L}{sum_{cs}^{o_1} - \mathcal{T}_{cs}^L}$
        $\hat{t}_{jk}^{o_2} \leftarrow t_{jk}^{o_1} + t_{jk}^{o_2} - \hat{t}_{jk}^{o_1}$
      **end if**
    **end for**
  **end for**
**end if**
**if** $sum^{o_2} > \mathcal{T}^*$ **then**
  $\mathcal{T}_{cs}^{o_2} \leftarrow \mathcal{T}^* - (sum^{o_2} - sum_{cs}^{o_2})$
  **for** $j := 1$ to $m$ **do**
    **for** $k := 1$ to $n_j$ **do**
      **if** $t_{jk}^{o_2}$ is a crossover gene **then**
        $\hat{t}_{jk}^{o_2} \leftarrow \tau_{jk}^L + (t_{jk}^{o_2} - \tau_{jk}^L)\frac{\mathcal{T}_{cs}^{o_2} - \mathcal{T}_{cs}^L}{sum_{cs}^{o_2} - \mathcal{T}_{cs}^L}$
        $\hat{t}_{jk}^{o_1} \leftarrow t_{jk}^{o_1} + t_{jk}^{o_2} - \hat{t}_{jk}^{o_2}$
      **end if**
    **end for**
  **end for**
**end if**

### C. Polynomial Mutation

The procedure of the ECHTs based polynomial mutation is presented in Algorithm 4, where $sum^o$ denotes the sum of all the gene values; $sum_{mt}^o$ represents the sum of all the mutation-gene values; $\mathcal{T}_{mt}^o$ is the available testing time for the mutation genes; $\mathcal{T}_{mt}^L$ represents the least testing time needed by all the mutation genes; $MR \in [0,1]$ is the given mutation rate; $rand(\tau_{jk}^L, t_{jk}^p)$ and $rand(t_{jk}^p, \mathcal{T}^*)$ represent random numbers in $[\tau_{jk}^L, t_{jk}^p]$ and $[t_{jk}^p, \mathcal{T}^*]$, respectively, which are generated from a normal distribution; $\delta \in (-1,1)$ is a random number generated from a polynomial distribution [9].

In Algorithm 4, it is possible that $t_{jk}^o$ violates the lower or upper bound after the mutation. If it happens, $t_{jk}^o$ will be repaired within the appropriate range to maintain the previous evolutionary trend. Once $t_{jk}^o < \tau_{jk}^L$, $\hat{t}_{jk}^o \leftarrow rand(\tau_{jk}^L, t_{jk}^p)$. On the other side, if $t_{jk}^o > \mathcal{T}^*$, $\hat{t}_{jk}^o \leftarrow rand(t_{jk}^p, \mathcal{T}^*)$. It is clear that the new $\hat{t}_{jk}^o$ satisfies (9). In addition, if $sum^o$ is beyond $\mathcal{T}^*$, each mutation-gene value will be scaled down on the basis of $\mathcal{T}_{mt}^L$ to preserve the genetic characteristics in offspring and

**Algorithm 4** The ECHTs for polynomial mutation.

**Input:** a selected individual $p$
**Output:** a produced offspring $o$
$sum^o \leftarrow 0$, $sum^o_{mt} \leftarrow 0$, $\mathcal{T}^o_{mt} \leftarrow 0$, $\mathcal{T}^L_{mt} \leftarrow 0$
**for** $j := 1$ to $m$ **do**
  **for** $k := 1$ to $n_j$ **do**
    **if** $rand(0,1) \leq MR$ **then**
      $t^o_{jk} \leftarrow t^p_{jk} + \delta(\mathcal{T}^* - \tau^L_{jk})$
      **if** $t^o_{jk} < \tau^L_{jk}$ **then**
        $t^o_{jk} \leftarrow rand(\tau^L_{jk}, t^p_{jk})$
      **end if**
      **if** $t^o_{jk} > \mathcal{T}^*$ **then**
        $t^o_{jk} \leftarrow rand(t^p_{jk}, \mathcal{T}^*)$
      **end if**
      $sum^o_{mt} \leftarrow sum^o_{mt} + t^o_{jk}, \mathcal{T}^L_{mt} \leftarrow \mathcal{T}^L_{mt} + \tau^L_{jk}$
    **else**
      $t^o_{jk} \leftarrow t^p_{jk}$
    **end if**
    $sum^o \leftarrow sum^o + t^o_{jk}$
  **end for**
**end for**
**if** $sum^o > \mathcal{T}^*$ **then**
  $\mathcal{T}^o_{mt} \leftarrow \mathcal{T}^* - (sum^o - sum^o_{mt})$
  **for** $j := 1$ to $m$ **do**
    **for** $k := 1$ to $n_j$ **do**
      **if** $t^o_{jk}$ is a mutation gene **then**
        $\hat{t}^o_{jk} \leftarrow \tau^L_{jk} + (t^o_{jk} - \tau^L_{jk}) \frac{\mathcal{T}^o_{mt} - \mathcal{T}^L_{mt}}{sum^o_{mt} - \mathcal{T}^L_{mt}}$
      **end if**
    **end for**
  **end for**
**end if**

TABLE I
MODEL PARAMETERS FOR DIFFERENT SOFTWARE SYSTEMS

| System | $\lambda$ | $\mathcal{T}^*$ | $\mathcal{R}^*$ | $m$ | $n_j$ |
|---|---|---|---|---|---|
| Complex | 200 | 150,000 | 0.65 | 11 | $\{1,2,3,3,4,4,4,3,3,2,1\}$ |
| Large | 200 | 230,000 | 0.65 | 16 | $\{1,2,3,3,3,4,4,5,5,4,4,3,3,3,2,1\}$ |
| Larger | 200 | 560,000 | 0.65 | 30 | $\{1,2,2,2,3,3,3,3,4,4,4,4,5,5,5,$ $5,5,5,4,4,4,4,3,3,3,3,2,2,2,1\}$ |

TABLE II
RANGE OF THE MODULAR PARAMETER VALUES

| 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] |

achieve the satisfactory reliability level:

$$\hat{t}^o_{jk} \leftarrow \tau^L_{jk} + (t^o_{jk} - \tau^L_{jk}) \frac{\mathcal{T}^o_{mt} - \mathcal{T}^L_{mt}}{sum^o_{mt} - \mathcal{T}^L_{mt}}$$

It can be easily found that the new $\hat{t}^o_{jk}$ obeys (9) and (10) and the worst-case complexity of Algorithm 4 is $O(D)$.

## VI. PERFORMANCE EVALUATION

In this section, the proposed ECHTs were evaluated through experiments, which addressed the following three research questions.

- Research Question 1 (RQ1): Do the proposed ECHTs outperform the state-of-the-art constraint solving methods designed specifically for the MOTRAP?
- Research Question 2 (RQ2): Are the proposed ECHTs superior to the general constrained MOEAs especially under different reliability constraints?
- Research Question 3 (RQ3): Are the proposed ECHTs robust against the change of values of modular parameters?

Basic parameter settings and appropriate performance metrics were first introduced. Then, to address RQ1, in the initial experiment the ECHTs were compared with the state-of-the-art constraint solving methods. The second experiment addressed RQ2 by further comparing different constrained MOEAs under different reliability constraints. The final experiment conducted sensitivity analysis with respect to the five modular parameters to address RQ3.

### A. Parameter Settings and Performance Metrics

All the experiments were done on the basis of the following three parallel-series modular software systems: a complex system with 11 sub-systems and 30 modules, a large system with 16 sub-systems and 50 modules, and a larger system with 30 sub-systems and 100 modules. The model parameters for different systems were illustrated in Table I. Different from the previous studies, the employed three systems with gradual complexities are significantly large in problem size, which can help us to evaluate the efficiency of different constraint solving methods more comprehensively and draw a more general conclusion. Note that in this work we set $\lambda = 200$ for all the three systems, implying that each system is expected to work continuously for at least 200 hours. The pre-defined intervals of modular parameter values were listed in Table II, in which "Serial" means that the module belongs to a subsystem that has only one module and "Parallel" indicates that the module belongs to a subsystem with more than one module. These intervals were determined by the collected parameter values of realistic modular systems that were estimated according to the software failure times and maximum likelihood estimation, and have been widely used as the typical benchmark intervals in the previous studies [4], [11], [12], [16], [17]. We generated every instance randomly in the above settings, according to the given model parameters and intervals of modular parameter values.

To compare the overall performance of the tested algorithms, the popular hypervolume (HV) metric [29] was used on account of its good properties. HV calculates the volume of the objective space between the obtained non-dominated solution set and a reference point. The HV value can provide a comprehensive information about the convergence and diversity of the whole solution set. Usually, a larger HV value means a better quality of the solution set. Note that the choice of the reference point is crucial for the calculation of HV. To determine the reference point, for each instance, we combined the solution sets of all the runs of all the compared algorithms, removed the duplicate solutions, sorted the left solutions according to the non-domination relationship, and removed all the dominated solutions. Then, we set the reference point to 1.1 times the worst value of each objective in the combined non-dominated solution set. The above method has been found to be suitable because of the good balance between the convergence and diversity of the solution set [30].

The classical coverage (CV) metric [31], which provides a direct comparison of how many solutions obtained by one

TABLE III
HV (MEAN AND STANDARD DEVIATION) RESULTS OF NSGA-II UNDER $\mathcal{R}^* = 0.65$ AND DIFFERENT CONSTRAINT SOLVING METHODS, SYSTEMS, AND INSTANCES. THE BETTER RESULTS REGARDING THE MEAN FOR EACH INSTANCE IS HIGHLIGHTED IN BOLDFACE. "†" AND "‡" INDICATE THAT THE DIFFERENCE BETWEEN THE PEER METHOD AND OUR METHOD IS STATISTICALLY AND PRACTICAL SIGNIFICANT, RESPECTIVELY

| Instance | Complex System | | | Large System | | | Larger System | | |
|---|---|---|---|---|---|---|---|---|---|
| | Our Method | Time Method | Random Method | Our Method | Time Method | Random Method | Our Method | Time Method | Random Method |
| 1 | **3.51E+6(1.049E+5)** | 3.16E+6(1.67E+5)†‡ | 2.703E+6(2.911E+5)†‡ | **4.673E+6(6.689E+4)** | 3.946E+6(3.748E+5)†‡ | 3.65E+6(4.19E+5)†‡ | **2.049E+7(6.669E+5)** | 1.632E+7(1.386E+6)†‡ | 1.609E+7(1.357E+6)†‡ |
| 2 | **2.744E+6(9.28E+4)** | 2.416E+6(2.098E+5)†‡ | 1.958E+6(2.293E+5)†‡ | **4.604E+6(5.266E+4)** | 3.848E+6(3.872E+5)†‡ | 3.593E+6(3.58E+5)†‡ | **1.505E+7(4.981E+5)** | 6.88E+6(5.4E+6)†‡ | 7.376E+6(5.392E+6)†‡ |
| 3 | **2.893E+6(4.973E+4)** | 2.6E+6(2.153E+5)†‡ | 2.263E+6(2.139E+5)†‡ | **4.191E+6(6.408E+4)** | 3.366E+6(3.862E+5)†‡ | 3.105E+6(3.399E+5)†‡ | **1.959E+7(4.675E+5)** | 1.43E+7(1.956E+6)†‡ | 1.506E+7(1.591E+6)†‡ |
| 4 | **2.422E+6(4.523E+4)** | 2.113E+6(2.289E+5)†‡ | 1.673E+6(2.146E+5)†‡ | **3.611E+6(2.431E+4)** | 2.857E+6(3.308E+5)†‡ | 2.564E+6(3.691E+5)†‡ | **1.858E+7(5.051E+5)** | 1.22E+7(4.326E+6)†‡ | 1.12E+7(5.243E+6)†‡ |
| 5 | **2.894E+6(4.892E+4)** | 2.489E+6(2.287E+5)†‡ | 2.224E+6(2.504E+5)†‡ | **4.547E+6(5.222E+4)** | 3.619E+6(5.035E+5)†‡ | 3.484E+6(3.995E+5)†‡ | **1.983E+7(5.354E+5)** | 1.551E+7(3.205E+6)†‡ | 1.528E+7(3.107E+6)†‡ |
| 6 | **3.297E+6(7.359E+4)** | 3.008E+6(2.143E+5)†‡ | 2.521E+6(3.274E+5)†‡ | **5.321E+6(6.452E+4)** | 4.693E+6(3.526E+5)†‡ | 4.303E+6(4.025E+5)†‡ | **1.915E+7(5.328E+5)** | 1.241E+7(3.696E+6)†‡ | 1.233E+7(4.376E+6)†‡ |
| 7 | **3.325E+6(7.792E+4)** | 3.048E+6(1.908E+5)†‡ | 2.72E+6(2.354E+5)†‡ | **4.643E+6(9.255E+4)** | 3.942E+6(3.746E+5)†‡ | 3.683E+6(3.419E+5)†‡ | **1.782E+7(3.786E+5)** | 1.264E+7(3.658E+6)†‡ | 1.388E+7(8.81E+5)†‡ |
| 8 | **3.482E+6(1.191E+5)** | 3.095E+6(2.033E+5)†‡ | 2.706E+6(2.175E+5)†‡ | **4.356E+6(9.33E+4)** | 3.822E+6(2.138E+5)†‡ | 3.391E+6(2.866E+5)†‡ | **1.77E+7(5.093E+5)** | 1.316E+7(1.285E+6)†‡ | 1.326E+7(1.402E+6)†‡ |
| 9 | **2.502E+6(6.164E+4)** | 2.206E+6(1.44E+5)†‡ | 1.778E+6(2.525E+5)†‡ | **4.982E+6(8.306E+4)** | 4.37E+6(3.162E+5)†‡ | 3.988E+6(3.611E+5)†‡ | **2.034E+7(4.236E+5)** | 1.515E+7(1.502E+6)†‡ | 1.523E+7(3.115E+6)†‡ |
| 10 | **2.916E+6(6.649E+4)** | 2.697E+6(1.322E+5)†‡ | 2.484E+6(1.472E+5)†‡ | **4.74E+6(1.165E+5)** | 4.277E+6(3.582E+5)†‡ | 3.848E+6(3.094E+5)†‡ | **2.603E+7(6.656E+5)** | 2.043E+7(4.125E+6)†‡ | 2.118E+7(1.417E+6)†‡ |
| 11 | **2.775E+6(5.284E+4)** | 2.444E+6(2.155E+5)†‡ | 2.053E+6(2.331E+5)†‡ | **4.959E+6(6.67E+4)** | 4.18E+6(3.649E+5)†‡ | 3.723E+6(3.81E+5)†‡ | **1.855E+7(5.036E+5)** | 1.279E+7(3.735E+6)†‡ | 1.274E+7(4.511E+6)†‡ |
| 12 | **4.089E+6(1.217E+5)** | 3.767E+6(2.464E+5)†‡ | 3.368E+6(2.289E+5)†‡ | **4.895E+6(8.287E+4)** | 4.151E+6(3.578E+5)†‡ | 3.727E+6(3.544E+5)†‡ | **1.794E+7(3.712E+5)** | 1.343E+7(1.41E+6)†‡ | 1.372E+7(2.857E+6)†‡ |
| 13 | **3.752E+6(9.243E+4)** | 3.509E+6(1.483E+5)†‡ | 3.125E+6(2.364E+5)†‡ | **4.537E+6(4.927E+4)** | 3.88E+6(3.564E+5)†‡ | 3.417E+6(4.25E+5)†‡ | **1.626E+7(4.22E+5)** | 7.827E+6(5.382E+6)†‡ | 8.79E+6(4.792E+6)†‡ |
| 14 | **3.448E+6(7.398E+4)** | 3.203E+6(1.851E+5)†‡ | 2.788E+6(2.44E+5)†‡ | **3.648E+6(2.652E+4)** | 2.912E+6(2.68E+5)†‡ | 2.711E+6(4.196E+5)†‡ | **1.621E+7(4.464E+5)** | 1.012E+7(3.624E+6)†‡ | 1.133E+7(2.504E+6)†‡ |
| 15 | **2.957E+6(7.776E+4)** | 2.707E+6(1.689E+5)†‡ | 2.352E+6(2.445E+5)†‡ | **4.162E+6(8.932E+4)** | 3.618E+6(3.045E+5)†‡ | 3.222E+6(3.656E+5)†‡ | **1.464E+7(3.626E+5)** | 8.907E+6(2.634E+6)†‡ | 8.21E+6(3.86E+6)†‡ |
| 16 | **2.777E+6(4.51E+4)** | 2.467E+6(1.929E+5)†‡ | 2.016E+6(2.349E+5)†‡ | **5.265E+6(9.786E+4)** | 4.458E+6(3.252E+5)†‡ | 4.077E+6(4.276E+5)†‡ | **1.791E+7(3.522E+5)** | 8.652E+6(5.902E+6)†‡ | 1.025E+7(5.353E+6)†‡ |
| 17 | **3.303E+6(7.914E+4)** | 3.002E+6(1.951E+5)†‡ | 2.647E+6(1.955E+5)†‡ | **4.359E+6(5.197E+4)** | 3.297E+6(4.399E+5)†‡ | 2.967E+6(4.925E+5)†‡ | **2.04E+7(3.353E+5)** | 1.518E+7(1.707E+6)†‡ | 1.55E+7(1.572E+6)†‡ |
| 18 | **2.895E+6(5.186E+4)** | 2.597E+6(1.917E+5)†‡ | 2.13E+6(2.625E+5)†‡ | **4.143E+6(6.805E+4)** | 3.328E+6(4.034E+5)†‡ | 3.088E+6(3.664E+5)†‡ | **1.903E+7(4.362E+5)** | 1.483E+7(1.385E+6)†‡ | 1.404E+7(3.973E+6)†‡ |
| 19 | **3.245E+6(1.151E+5)** | 2.982E+6(2.48E+5)†‡ | 2.495E+6(3.432E+5)†‡ | **4.412E+6(5.293E+4)** | 3.635E+6(3.991E+5)†‡ | 3.501E+6(3.267E+5)†‡ | **2.01E+7(4.918E+5)** | 1.49E+7(3.087E+6)†‡ | 1.553E+7(1.234E+6)†‡ |
| 20 | **3.625E+6(8.149E+4)** | 3.447E+6(1.807E+5)†‡ | 2.963E+6(2.552E+5)†‡ | **4.052E+6(7.587E+4)** | 3.481E+6(3.539E+5)†‡ | 3.061E+6(4.367E+5)†‡ | **1.62E+7(3.337E+5)** | 1.008E+7(2.96E+6)†‡ | 8.736E+6(4.556E+6)†‡ |
| 21 | **3.091E+6(8.58E+4)** | 2.84E+6(2.241E+5)†‡ | 2.416E+6(2.249E+5)†‡ | **5.352E+6(8.781E+4)** | 4.8E+6(3.753E+5)†‡ | 4.343E+6(3.443E+5)†‡ | **1.887E+7(5.301E+5)** | 1.412E+7(2.955E+6)†‡ | 1.47E+7(1.268E+6)†‡ |
| 22 | **3.765E+6(1.152E+5)** | 3.431E+6(1.743E+5)†‡ | 2.889E+6(3.397E+5)†‡ | **4.886E+6(7.978E+4)** | 3.932E+6(3.429E+5)†‡ | 3.652E+6(3.939E+5)†‡ | **1.811E+7(4.289E+5)** | 1.255E+7(2.718E+6)†‡ | 1.209E+7(3.42E+6)†‡ |
| 23 | **2.762E+6(6.143E+4)** | 2.504E+6(2.205E+5)†‡ | 2.051E+6(2.662E+5)†‡ | **5.292E+6(8.216E+4)** | 4.383E+6(3.344E+5)†‡ | 3.877E+6(4.115E+5)†‡ | **2.041E+7(4.622E+5)** | 1.654E+7(1.849E+6)†‡ | 1.646E+7(1.478E+6)†‡ |
| 24 | **3.017E+6(6.519E+4)** | 2.803E+6(2.051E+5)†‡ | 2.385E+6(3.425E+5)†‡ | **4.876E+6(1.129E+5)** | 4.084E+6(3.793E+5)†‡ | 3.632E+6(4.151E+5)†‡ | **1.657E+7(5.379E+5)** | 7.563E+6(4.545E+6)†‡ | 9.229E+6(4.819E+6)†‡ |
| 25 | **3.902E+6(1.18E+5)** | 3.503E+6(1.779E+5)†‡ | 3.156E+6(2.197E+5)†‡ | **4.861E+6(7.388E+4)** | 3.941E+6(3.066E+5)†‡ | 3.803E+6(4.963E+5)†‡ | **2.142E+7(3.758E+5)** | 1.221E+7(6.362E+6)†‡ | 1.563E+7(3.296E+6)†‡ |
| 26 | **3.368E+6(9.574E+4)** | 3.143E+6(1.469E+5)†‡ | 2.752E+6(2.734E+5)†‡ | **5.802E+6(1.6E+5)** | 5.196E+6(3.674E+5)†‡ | 4.847E+6(3.822E+5)†‡ | **1.867E+7(5.404E+5)** | 1.398E+7(1.719E+6)†‡ | 1.417E+7(2.934E+6)†‡ |
| 27 | **3.35E+6(9.777E+4)** | 2.966E+6(2.276E+5)†‡ | 2.525E+6(2.515E+5)†‡ | **4.651E+6(7.539E+4)** | 4.118E+6(3.097E+5)†‡ | 3.886E+6(3.323E+5)†‡ | **1.564E+7(3.204E+5)** | 1.074E+7(2.352E+6)†‡ | 1.141E+7(1.253E+6)†‡ |
| 28 | **2.901E+6(5.882E+4)** | 2.64E+6(1.327E+5)†‡ | 2.283E+6(1.843E+5)†‡ | **4.342E+6(4.325E+4)** | 3.419E+6(3.864E+5)†‡ | 3.227E+6(3.893E+5)†‡ | **1.612E+7(4.121E+5)** | 1.008E+7(3.071E+6)†‡ | 1.076E+7(2.306E+6)†‡ |
| 29 | **2.876E+6(5.751E+4)** | 2.72E+6(1.961E+5)†‡ | 2.32E+6(2.022E+5)†‡ | **4.896E+6(7.313E+4)** | 4.153E+6(3.371E+5)†‡ | 3.905E+6(2.868E+5)†‡ | **1.631E+7(4.129E+5)** | 1.058E+7(4.381E+6)†‡ | 1.104E+7(3.909E+6)†‡ |
| 30 | **2.843E+6(7.653E+4)** | 2.605E+6(1.74E+5)†‡ | 2.228E+6(2.128E+5)†‡ | **4.885E+6(6.509E+4)** | 4.104E+6(4.052E+5)†‡ | 3.822E+6(3.983E+5)†‡ | **2.207E+7(6.177E+5)** | 1.748E+7(1.446E+6)†‡ | 1.726E+7(1.251E+6)†‡ |

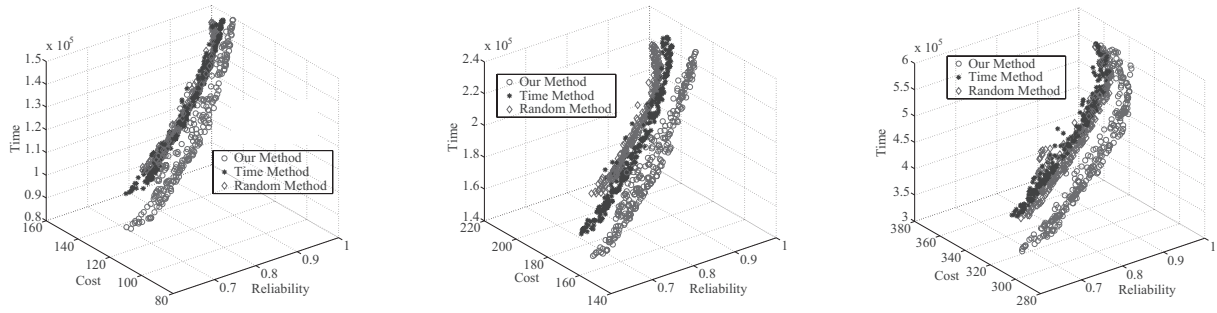
algorithm are dominated by those approximated by the peer algorithm, is adopted as the convergence measure to compare the quality of solutions obtained by different methods. Assume $A$ and $B$ denote the solution sets obtained by the considered algorithm and the competitor, respectively. One solution in $A$ is said to cover another solution in $B$ if the former is not worse than the latter on any objective. $\varepsilon(A, B)$ represents the percentage of $B$ that is covered by solutions in $A$. $\varepsilon(A, B) > \varepsilon(B, A)$ indicates a win for the considered algorithm against the peer algorithm. Then, a series of such wins demonstrate that the considered algorithm is statistically significantly better than the competitor [31]. Notice that for every instance, we removed the duplicate solutions and computed CV on the basis of each combined solution set according to the suggestions in [30].

Apart from quality indicators for general MOPs such as HV and CV, problem-specific indicators which can accommodate the user's explicit/implicit preferences have been highly recommended to evaluate solution sets [32]. Here, we considered the capacity (CP) metric [33]. CP counts the number of satisfactory non-dominated solutions whose reliability is not lower than the given reliability constraint $\mathcal{R}^*$. Similarly, for each run, we removed the duplicate solutions from the solution set before the calculation of CP. In this work, we used $|\cdot|$ to denote the CP values of different solution sets obtained by the compared methods.

Each instance was repeated for 30 independent runs with different random seeds on a PC with Intel(R) Core(TM) i7-3537U 3rd generation CPU @ 2.00GHz, 10 GB of DDR3L RAM @ 1600MHz, and Windows 10 operating system.

### *B. Research Question 1: Comparing Different Constraint Handling Techniques*

In the first experiment, we will compare and evaluate the state-of-the-art constraint handling techniques designed specifically for the MOTRAP. To analyze the effectiveness and understand the mechanism of the proposed ECHTs, we used the constrained NSGA-II [9] as the "backplane" algorithm. This is because we want to show whether the proposed ECHTs works even working with the most basic optimizer. On the other hand, we want to reduce the effect of MOEAs. That is, we want to show that the outperformance is not from the state-of-the-art MOEAs, but our constraint solving approaches. We compared the proposed ECHTs (henceforth called Our Method) with a random-reduction based constraint handling technique (henceforth called Random Method) [4], [12] and a time-satisfied constraint handling technique (henceforth called Time Method) [11]. Note that Algorithms 1-4 in the proposed ECHTs are an organic whole of four, without one of which the others will lose their effects for the reason that each individual satisfying $\mathcal{T}^*$ is a necessary condition for the repairs in crossover and mutation to be effective. Accordingly, in this experiment, we evaluated the whole ECHTs rather than each single technique. Furthermore, the time complexity of Our Method is mainly determined by the employed NSGA-II. Assume that $N$ is the population size and $M$ is the number of objectives. The worst case time complexity of the individual initialization, simulated binary crossover, or polynomial mutation of NSGA-II is $O(ND)$, while the worst case time complexity of environmental selection of NSGA-II is $O(MN^2)$ [9]. Clearly, Our Method has the same worst case time complexity as the employed MOEAs, but it is




(a) Complex System (b) Large System (c) Larger System

Fig. 4. Non-dominated solutions obtained by NSGA-II with different constraint solving methods on the three employed software systems (one case study).

TABLE IV
CP AND CV RESULTS OF NSGA-II UNDER $\mathcal{R}^* = 0.65$ AND DIFFERENT CONSTRAINT SOLVING METHODS, SYSTEMS, AND INSTANCES. $A$, $B$, AND $C$ DENOTE THE FINAL SOLUTION SETS OBTAINED BY OUR METHOD, TIME METHOD, AND RANDOM METHOD, RESPECTIVELY. THE BETTER RESULTS FOR EACH INSTANCE IS HIGHLIGHTED IN BOLDFACE

| Instance | Complex System | | | | | | | Large System | | | | | | | Larger System | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\|A\|$ | $\|B\|$ | $\|C\|$ | $\varepsilon(A,B)$ | $\varepsilon(B,A)$ | $\varepsilon(A,C)$ | $\varepsilon(C,A)$ | $\|A\|$ | $\|B\|$ | $\|C\|$ | $\varepsilon(A,B)$ | $\varepsilon(B,A)$ | $\varepsilon(A,C)$ | $\varepsilon(C,A)$ | $\|A\|$ | $\|B\|$ | $\|C\|$ | $\varepsilon(A,B)$ | $\varepsilon(B,A)$ | $\varepsilon(A,C)$ | $\varepsilon(C,A)$ |
| 1 | 7499 | 7498 | **7500** | **97.8%** | 26.02% | **97.12%** | 8.32% | 7500 | 7500 | 7500 | **99.75%** | 5.64% | **97.03%** | 18.03% | 7500 | 7500 | 7500 | **99.91%** | 7.28% | **100%** | 8.29% |
| 2 | 7498 | **7500** | 7496 | **97.37%** | 29.49% | **96.52%** | 6.75% | 7499 | 7500 | 7500 | **96.81%** | 20.31% | **95.27%** | 9.52% | **7500** | 4750 | 5000 | **100%** | 6.72% | **100%** | 6.89% |
| 3 | 7499 | 7499 | 7499 | **94.16%** | 40.83% | **93.93%** | 15.35% | 7500 | 7500 | 7500 | **99.93%** | 5.48% | **99.48%** | 4.08% | 7500 | 7500 | 7500 | **99.99%** | 0.68% | **99.93%** | 5.23% |
| 4 | 7499 | **7500** | 7498 | **88.24%** | 68.88% | **94.44%** | 11.69% | 7499 | 7500 | 7500 | **98.69%** | 5.24% | **97.45%** | 1.83% | **7500** | 6750 | 6250 | **100%** | 0.75% | **100%** | 2.37% |
| 5 | 7500 | 7500 | 7498 | **97.37%** | 23.08% | **97.32%** | 5.47% | 7500 | 7500 | 7500 | **99.89%** | 6.04% | **97.69%** | 7.77% | **7500** | 7250 | 7250 | **99.38%** | 18.63% | **99.88%** | 14.47% |
| 6 | **7500** | 7499 | 7498 | **90.84%** | 51.55% | **97.32%** | 27.72% | 7500 | 7500 | 7500 | **98.61%** | 22.07% | **99.51%** | 10.2% | **7499** | 7000 | 6750 | **100%** | 1.43% | **100%** | 1.6% |
| 7 | 7499 | 7500 | 7500 | **93.99%** | 43.18% | **95.53%** | 26.75% | 7500 | 7500 | 7500 | **97.35%** | 24.33% | **93.61%** | 24.4% | 7500 | 7000 | 7500 | **100%** | 1.59% | **100%** | 3.53% |
| 8 | 7499 | 7499 | **7500** | **97.53%** | 33% | **97.69%** | 14.19% | 7500 | 7500 | 7500 | **98.39%** | 16.01% | **94.87%** | 17.4% | 7500 | 7500 | 7500 | **100%** | 6.31% | **100%** | 2.36% |
| 9 | 7497 | 7500 | 7500 | **97.21%** | 20.89% | **96.03%** | 12.27% | 7499 | 7500 | 7500 | **99.57%** | 14.96% | **95.63%** | 20.52% | 7500 | 7500 | 7250 | **100%** | 2.33% | **100%** | 1.45% |
| 10 | 7499 | 7499 | 7497 | **94.99%** | 41.66% | **95.82%** | 24.56% | 7499 | 7499 | **7500** | **93.61%** | 61.34% | **96.03%** | 22.42% | 7500 | 7250 | 7500 | **99.61%** | 13% | **99.88%** | 9.29% |
| 11 | 7498 | 7499 | **7500** | **97.76%** | 23.63% | **97.73%** | 9.44% | 7500 | 7499 | 7500 | **98.65%** | 15.71% | **97.72%** | 7.69% | **7500** | 7000 | 6750 | **100%** | 2.91% | **100%** | 8.2% |
| 12 | 7499 | 7496 | **7500** | **86.46%** | 69.25% | **94.29%** | 43.39% | 7500 | 7499 | 7500 | **95.77%** | 22.35% | **95.59%** | 5.91% | 7500 | 7500 | 7250 | **100%** | 4.77% | **99.96%** | 9.91% |
| 13 | 7495 | 7499 | **7500** | **92.77%** | 45.07% | **97.88%** | 20.67% | 7500 | 7500 | 7500 | **97.8%** | 19.68% | **97.27%** | 7.95% | **7500** | 5250 | 6000 | **100%** | 0.67% | **100%** | 1.07% |
| 14 | 7497 | 7499 | 7499 | **89.92%** | 49.58% | **94.83%** | 25.13% | 7500 | 7500 | 7500 | **99.49%** | 1.76% | **96.09%** | 8.44% | **7500** | 6750 | 7250 | **100%** | 0.21% | **100%** | 0.63% |
| 15 | 7498 | **7499** | 7498 | **96.03%** | 28.01% | **96.79%** | 14.04% | 7500 | 7500 | 7499 | **94.05%** | 56.48% | **95.72%** | 9.71% | **7500** | 7000 | 6250 | **100%** | 0% | **100%** | 0.89% |
| 16 | 7499 | 7499 | 7498 | **93.67%** | 33.22% | **94.73%** | 6.33% | 7500 | 7500 | 7500 | **97.76%** | 24.51% | **94.63%** | 20.65% | **7500** | 5250 | 6000 | **100%** | 0.05% | **100%** | 0.97% |
| 17 | **7500** | 7497 | 7498 | **95.46%** | 30.53% | **95.68%** | 24.24% | 7500 | 7500 | 7499 | **99.75%** | 2.12% | **99.72%** | 0.23% | 7500 | 7500 | 7500 | **99.99%** | 2% | **100%** | 1.95% |
| 18 | 7499 | 7499 | 7495 | **96.29%** | 28.4% | **99.2%** | 7.65% | **7500** | 7498 | 7499 | **97.31%** | 13.63% | **97.41%** | 6.37% | 7500 | 7500 | 7000 | **100%** | 2.09% | **100%** | 2.93% |
| 19 | 7497 | 7497 | **7498** | **83.61%** | 78.87% | **92.48%** | 51.49% | 7500 | 7500 | 7500 | **98.65%** | 10.41% | **95.32%** | 11.93% | 7500 | 7250 | 7500 | **99.99%** | 2.39% | **100%** | 2.71% |
| 20 | **7499** | 7498 | 7498 | **83.74%** | 59.41% | **97.27%** | 16.47% | 7499 | **7500** | 7499 | **88.72%** | 54.19% | **92.72%** | 15.58% | **7500** | 7000 | 6000 | **100%** | 0.12% | **100%** | 1.13% |
| 21 | 7498 | 7497 | **7500** | **89.92%** | 48.52% | **98.84%** | 13.47% | 7500 | 7500 | 7500 | **89.56%** | 55.71% | **92.8%** | 21.97% | 7500 | 7250 | 7500 | **100%** | 4.19% | **100%** | 5.04% |
| 22 | 7499 | 7498 | 7499 | **96.91%** | 30.72% | **98.83%** | 13.23% | 7499 | 7500 | 7500 | **99.33%** | 8.75% | **96.21%** | 16.24% | **7500** | 7250 | 7000 | **99.99%** | 0.81% | **100%** | 1.92% |
| 23 | 7500 | 7500 | 7498 | **84.36%** | 57.83% | **92.16%** | 13.21% | 7500 | 7500 | 7500 | **98.65%** | 12.09% | **97.96%** | 7.19% | 7500 | 7500 | 7500 | **99.01%** | 22.35% | **99.33%** | 15.33% |
| 24 | 7497 | 7498 | **7500** | **91.72%** | 41.76% | **93.64%** | 27.85% | 7499 | 7500 | 7500 | **99.96%** | 12.45% | **99.23%** | 6.8% | **7500** | 5750 | 6000 | **100%** | 0.07% | **100%** | 0.81% |
| 25 | 7499 | 7499 | 7499 | **98.07%** | 28.35% | **98.97%** | 11% | 7500 | 7499 | 7500 | **99.17%** | 10.61% | **97.33%** | 11.4% | **7500** | 6000 | 7250 | **100%** | 0.36% | **100%** | 2.96% |
| 26 | 7499 | 7499 | 7499 | **91.95%** | 45.11% | **95.87%** | 33.02% | 7500 | 7500 | 7499 | **98.52%** | 39.45% | **97.83%** | 24.4% | 7500 | 7500 | 7250 | **99.83%** | 7.32% | **100%** | 2.52% |
| 27 | **7500** | 7499 | 7498 | **97.55%** | 34.67% | **97.47%** | 11.67% | 7500 | 7500 | 7500 | **94.77%** | 35.28% | **94.32%** | 16.51% | 7500 | 7250 | 7500 | **100%** | 0.47% | **99.87%** | 10.04% |
| 28 | 7497 | **7500** | 7499 | **95.69%** | 25.09% | **95.69%** | 13.49% | 7500 | 7500 | 7500 | **99.99%** | 1.52% | **99.05%** | 3.96% | **7500** | 7000 | 7250 | **100%** | 0.03% | **100%** | 2.13% |
| 29 | 7499 | 7499 | 7499 | **88.67%** | 45.81% | **96.32%** | 12.48% | 7500 | 7500 | 7499 | **97.32%** | 27.95% | **94.64%** | 22.45% | **7500** | 6500 | 6750 | **100%** | 2.11% | **100%** | 3.4% |
| 30 | 7498 | 7497 | **7499** | **92.37%** | 42.9% | **94.77%** | 24.59% | 7500 | 7500 | 7500 | **97.13%** | 20.81% | **95.92%** | 10.43% | 7500 | 7500 | 7500 | **99.67%** | 4.33% | **100%** | 1.33% |

slightly slower than the employed MOEAs since it corrects constraint violations in individual initialization, simulated binary crossover, and polynomial mutation.

To make a fair comparison, we used the common algorithmic parameters for all the tests: the population size is 250; the maximum iteration number is 500; the crossover rate is 0.9; and the mutation rate is $1.0/D$. Additionally, for each system in Table I, we generated 30 different instances randomly from a normal distribution on the basis of the pre-defined intervals in Table II.

To measure the overall performance, Table III shows the results of the three methods with $\mathcal{R}^* = 0.65$ regarding the mean and standard deviation values of HV under different systems and instances. To have statistically sound conclusions, the Wilcoxon rank-sum test [34] at a 0.05 significance level was adopted to test the significance of the differences between the results obtained by the peer method and Our Method. Additionally, to further check whether the differences have practical significance, the popular Vargha and Delaneys' $\hat{A}_{12}$ effect size statistics were also computed [35]. The $\hat{A}_{12}$ effect size indicates the probability that the considered method yields higher values of the measurement taken than running the peer method. As can be seen from the table, Our Method achieved a better HV value in all the 90 instances. Also, the results have both statistical and practical significance on each instance. Hence, we have evidence that different methods achieve significantly different HV values. Moreover, an interesting phenomenon in Table III is that Time Method outperforms Random Method on the complex and large systems, but is inferior to Random Method on the larger system. Obviously, Time Method is not stable. A possible reason for this could be that Time Method pays too much attention to the time constraint but ignores the reliability constraint. On the other side, our method is able to keep good performance

with the increase of the complexity of software systems. The above results indicate that under the reliability constraint, Our Method significantly improves the HV metric and can achieve a better balance between convergence and diversity than Random Method and Time Method.

For a visual understanding of the distribution of solutions as well as of what a higher HV value means, in Fig. 4 we plot the final nondominated solutions of a particular run where the obtained HV result is the closest to the mean value on the basis of Instance 5 on the complex system, Instance 17 on the large system, and Instance 2 on the larger system. These particular instances are associated with the results which have the maximal differences of HV values between the compared three methods. As can be seen in the figure, the solutions obtained by Our Method spread wider in the objective space and provide more diversity than those obtained by Time Method and Random Method. Additionally, Our Method appears to have better convergence than its competitors, with a significant number of solutions being superior to those obtained by Time Method and Random Method, which can be easily observed from the plots in Fig. 4.

To measure the convergence ability, Table IV illustrates the results of the three methods with $\mathcal{R}^* = 0.65$ regarding CP and CV under different systems and instances. As can be seen from the table, Our Method is very robust and is able to find a large amount of feasible solutions at each run on all the 90 instances. Time Method and Random Method are workable on both the complex and the large systems but often cannot find any feasible solution on the larger system. The above results imply that Our Method is more stable and reliable than Time Method and Random Method. On the other hand, the CV values of Our Method versus Time Method and Our Method versus Random Method are greater than those of Time Method versus Our Method and Random Method versus Our Method on all the 90 instances. More precisely, Our Method covers, on average, 93.08% and 96.17% of the solutions obtained by Time Method and Random Method on the complex system, respectively; on the large system, Our Method covers, on average, 97.5% and 96.47% of the solutions obtained by Time Method and Random Method, respectively; on the larger system, Our Method covers, on average, 99.91% and 99.96% of the solutions obtained by Time Method and Random Method, respectively. On the contrary, Time Method and Random Method cover only few solutions attained by Our Method for every instance, especially on the larger system. The above results suggest that under the given reliability constraint, Our Method can obtain more and higher quality solutions than Time Method and Random Method. This is because both Random Method and Time Method only focus on time, all the repairs are designed to handle the violation on $\mathcal{T}^*$, and there is no repair corresponding to the lower bounds. That is, Random Method and Time Method can only repair and make infeasible individuals satisfy $\mathcal{T}^*$ rather than $\mathcal{R}^*$. Differently, Our Method not only utilizes the new lower bounds to allocate the limited testing time to the most time-efficient modules, but also maintains the evolutionary tendency of the amended individuals using proportional reduction. In other words, Our Method is driven by both reliability and time.

### C. Research Question 2: Comparing Different Constrained MOEAs

To verify whether a new algorithm needs to be developed to solve the reliability-constrained MOTRAP, in this subsection we compared the proposed ECHTs with two general constrained MOEAs for solving CMOPs with very small feasible regions: MOEA/D-IEpsilon [25] and CCMO [27]. Particularly, as shown in Fig. 2, the greater value of the reliability constraint $\mathcal{R}^*$ is, the smaller the feasible solution region of the MOTRAP is, and the higher exploration ability of the MOEA is required. Consequently, we evaluate the performance of Our Method, CCMO, and MOEA/D-IEpsilon under different values of $\mathcal{R}^*$. Specifically, Our Method is still combined with NSGA-II. Following the practice in [27], CCMO is also embedded with NSGA-II; the truncation strategy in the improved strength Pareto evolutionary algorithm [36] is adopted in the environmental selection instead of crowding distance; the helper problem is set to the MOTRAP with no constraint. In MOEA/D-IEpsilon [25], an external archive is iteratively updated on the basis of the non-dominated sorting strategy in NSGA-II. It should be noted that both CCMO and MOEA/D-IEpsilon can not find any feasible solution due to the great difference between the value range of $\mathcal{R}^*$ and $\mathcal{T}^*$. Therefore, in this experiment, the normalized constraint violation is used in CCMO and MOEA/D-IEpsilon.

The basic algorithmic parameters were recalled from [25], [27] for fair comparisons. In common, the maximal number of evaluations is 150,000, the crossover rate is 0.9, and the mutation rate is $1.0/D$. In Our Method, the population size is 300. In CCMO, the sizes of the two coevolutionary populations are both 300. In MOEA/D-IEpsilon, both the population size and the archive size are 300; the neighbour size is 30; the probability that parent solutions are selected from neighbourhood is 0.9; the maximal number of solutions replaced by each child solution is 2; the scaling factor in DE crossover is 0.5; the Tchebycheff decomposition method and the weight vectors defined in the file "W3D_300.dat" in the jMetalCpp project [37] are used.

Table V depicts the results of the three constrained MOEAs regarding the mean and standard deviation values of HV under different systems and $\mathcal{R}^*$. As can be seen from the table, for each system, with the increase of $\mathcal{R}^*$, the HV values obtained by the three algorithms decrease gradually. An explanation is that with the increase of $\mathcal{R}^*$, the feasible solution region of the MOTRAP becomes smaller, leading to less diversity of the solution set. Fortunately, in such situations, Our Method always outperforms CCMO and MOEA/D-IEpsilon. Our Method achieved a better HV value on each instance with both statistical and practical significance. Additionally, with the increase of the system size or $\mathcal{R}^*$, the difference of HV values among the three algorithms becomes larger. In particular, CCMO found few feasible solutions on the large system when $\mathcal{R}^* = 0.95$ and failed in finding a feasible solution on the larger system. The possible reason is that in such a harsh situation, a candidate solution is easily infeasible for the sake of constraint violations and the role of infeasible solutions derived from the helper problem is very weak. It

TABLE V
HV (MEAN AND STANDARD DEVIATION) RESULTS OF THE THREE CONSTRAINED MOEAS UNDER DIFFERENT SYSTEMS AND $\mathcal{R}^*$. THE BETTER RESULTS REGARDING THE MEAN FOR EACH INSTANCE IS HIGHLIGHTED IN BOLDFACE. "†" AND "‡" INDICATE THAT THE DIFFERENCE IS STATISTICALLY AND PRACTICAL SIGNIFICANT, RESPECTIVELY

| $\mathcal{R}^*$ | Complex System | | | Large System | | | Larger System | | |
|---|---|---|---|---|---|---|---|---|---|
| | Our Method | CCMO | MOEA/D-IEpsilon | Our Method | CCMO | MOEA/D-IEpsilon | Our Method | CCMO | MOEA/D-IEpsilon |
| 0.5 | **5.504E+6(1.309E+5)** | 3.639E+6(2.678E+5)†‡ | 4.401E+6(2.608E+5)†‡ | **7.319E+6(1.33E+5)** | 2.572E+6(5.591E+5)†‡ | 5.29E+6(5.989E+5)†‡ | **3.017E+7(6.576E+5)** | 0(0)†‡ | 1.79E+7(1.362E+6)†‡ |
| 0.55 | **4.597E+6(6.905E+4)** | 2.966E+6(2.464E+5)†‡ | 3.542E+6(2.266E+5)†‡ | **6.215E+6(1.235E+5)** | 2.082E+6(4.205E+5)†‡ | 4.294E+6(5.12E+5)†‡ | **2.416E+7(3.946E+5)** | 0(0)†‡ | 1.338E+7(1.15E+6)†‡ |
| 0.6 | **3.767E+6(7.539E+4)** | 2.283E+6(2.633E+5)†‡ | 2.843E+6(2.495E+5)†‡ | **5.395E+6(1.134E+5)** | 1.621E+6(4.456E+5)†‡ | 3.673E+6(4.336E+5)†‡ | **2.321E+7(6.515E+5)** | 0(0)†‡ | 1.274E+7(1.217E+6)†‡ |
| 0.65 | **3.286E+6(9.53E+4)** | 1.95E+6(2.926E+5)†‡ | 2.4E+6(2.379E+5)†‡ | **4.545E+6(9.503E+4)** | 1.153E+6(3.718E+5)†‡ | 3.004E+6(2.856E+5)†‡ | **1.663E+7(3.839E+5)** | 0(0)†‡ | 8.647E+6(9.634E+5)†‡ |
| 0.7 | **2.876E+6(1.029E+5)** | 1.7E+6(1.914E+5)†‡ | 2.124E+6(2.014E+5)†‡ | **4.075E+6(4.568E+4)** | 1.024E+6(2.624E+5)†‡ | 2.443E+6(3.839E+5)†‡ | **1.612E+7(3.94E+5)** | 0(0)†‡ | 8.13E+6(1.173E+6)†‡ |
| 0.75 | **2.64E+6(6.664E+4)** | 1.444E+6(2.714E+5)†‡ | 1.927E+6(2.089E+5)†‡ | **3.303E+6(4.638E+4)** | 5.93E+5(2.478E+5)†‡ | 2.017E+6(3.471E+5)†‡ | **1.469E+7(3.538E+5)** | 0(0)†‡ | 7.466E+6(9.604E+5)†‡ |
| 0.8 | **1.501E+6(3.215E+4)** | 7.919E+5(1.285E+5)†‡ | 8.924E+5(1.516E+5)†‡ | **2.278E+6(1.267E+4)** | 3.641E+5(1.696E+5)†‡ | 1.216E+6(2.427E+5)†‡ | **1.244E+7(3.837E+5)** | 0(0)†‡ | 6.071E+6(7.211E+5)†‡ |
| 0.85 | **1.057E+6(1.273E+4)** | 4.801E+5(9.736E+4)†‡ | 6.096E+5(1.513E+5)†‡ | **1.767E+6(2.175E+4)** | 2.081E+5(1.275E+5)†‡ | 8.628E+5(1.887E+5)†‡ | **7.96E+6(2.621E+5)** | 0(0)†‡ | 3.173E+6(6.211E+5)†‡ |
| 0.9 | **6.328E+5(3973)** | 2.549E+5(8.709E+4)†‡ | 2.794E+5(1.042E+5)†‡ | **7.641E+5(5756)** | 3.519E+4(4.126E+4)†‡ | 2.2E+5(1.105E+5)†‡ | **5.667E+6(2.236E+5)** | 0(0)†‡ | 1.992E+6(5.256E+5)†‡ |
| 0.95 | **2.512E+5(1461)** | 5.281E+4(4.962E+4)†‡ | 5.609E+4(5.064E+4)†‡ | **2.861E+5(2051)** | 432.6(2369)†‡ | 2.535E+4(3.957E+4)†‡ | **2.876E+6(1.013E+5)** | 0(0)†‡ | 5.526E+5(3.997E+5)†‡ |

TABLE VI
CP AND CV RESULTS OF THE THREE CONSTRAINED MOEAS UNDER DIFFERENT SYSTEMS AND $\mathcal{R}^*$. $A$, $B$, AND $C$ DENOTE THE FINAL SOLUTION SETS OBTAINED BY OUR METHOD, CCMO, AND MOEA/D-IEPSILON, RESPECTIVELY. THE BETTER RESULTS FOR EACH INSTANCE IS HIGHLIGHTED IN BOLDFACE

| $\mathcal{R}^*$ | Complex System | | | | | | | Large System | | | | | | | Larger System | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\|A\|$ | $\|B\|$ | $\|C\|$ | $\varepsilon(A,B)$ | $\varepsilon(B,A)$ | $\varepsilon(A,C)$ | $\varepsilon(C,A)$ | $\|A\|$ | $\|B\|$ | $\|C\|$ | $\varepsilon(A,B)$ | $\varepsilon(B,A)$ | $\varepsilon(A,C)$ | $\varepsilon(C,A)$ | $\|A\|$ | $\|B\|$ | $\|C\|$ | $\varepsilon(A,B)$ | $\varepsilon(B,A)$ | $\varepsilon(A,C)$ | $\varepsilon(C,A)$ |
| 0.5 | **8990** | 8512 | 8948 | **99.8%** | 0.22% | **99.72%** | 0.18% | **8983** | 8455 | 8627 | **100%** | 0% | **99.34%** | 0.21% | **8989** | 0 | 8779 | **100%** | 0% | **100%** | 0% |
| 0.55 | **8979** | 8208 | 8952 | **99.98%** | 0.68% | **99.72%** | 0.19% | **8988** | 8444 | 8629 | **100%** | 0% | **99.37%** | 0.03% | **8980** | 0 | 8457 | **100%** | 0% | **100%** | 0% |
| 0.6 | 8979 | **9000** | 8940 | **99.98%** | 0.82% | **99.73%** | 0.12% | **8985** | 8384 | 8896 | **100%** | 0% | **99.94%** | 0% | **8986** | 0 | 8671 | **100%** | 0% | **100%** | 0% |
| 0.65 | **8984** | 8815 | 8951 | **100%** | 0.96% | **99.6%** | 0.22% | **8982** | 8456 | 8628 | **100%** | 0% | **100%** | 0% | **8986** | 0 | 8482 | **100%** | 0% | **100%** | 0% |
| 0.7 | 8988 | **9000** | 8949 | **100%** | 0.08% | **100%** | 0.16% | **8992** | 8397 | 8934 | **100%** | 0% | **100%** | 0% | **8977** | 0 | 8338 | **100%** | 0% | **100%** | 0% |
| 0.75 | 8989 | **9000** | 8953 | **100%** | 0.1% | **99.81%** | 0.12% | **8986** | 8103 | 8925 | **100%** | 0% | **99.81%** | 0% | **8984** | 0 | 8356 | **100%** | 0% | **100%** | 0% |
| 0.8 | 8984 | **9000** | 8933 | **100%** | 0% | **100%** | 0% | **8989** | 7759 | 8915 | **100%** | 0% | **99.44%** | 0.02% | **8981** | 0 | 7997 | **100%** | 0% | **100%** | 0% |
| 0.85 | 8992 | **9000** | 8945 | **100%** | 0.08% | **99.81%** | 0.04% | **8986** | 6910 | 8840 | **100%** | 0% | **99.93%** | 0% | **8984** | 0 | 7798 | **100%** | 0% | **100%** | 0% |
| 0.9 | 8990 | **9000** | 8944 | **99.96%** | 0.1% | **100%** | 0% | **8986** | 6222 | 8887 | **100%** | 0% | **100%** | 0% | **8981** | 0 | 6716 | **100%** | 0% | **100%** | 0% |
| 0.95 | **8992** | 8770 | 8938 | **100%** | 0% | **100%** | 0% | **8994** | 26 | 8859 | **100%** | 0% | **100%** | 0% | **8985** | 0 | 5961 | **100%** | 0% | **100%** | 0% |

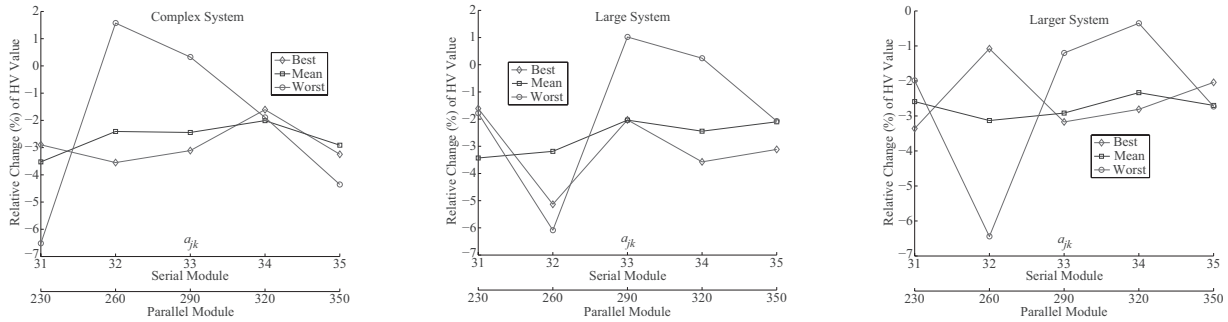
is difficult for those infeasible solutions to provide useful information for population evolution. On the other side, Our Method significantly improved the HV metric and achieved a better balance between convergence and diversity than CCMO and MOEA/D-IEpsilon, especially when the system size or $\mathcal{R}^*$ is large.

Table VI shows the results of the three constrained MOEAs regarding CP and CV under different systems and $\mathcal{R}^*$. It can be observed that Our Method and MOEA/D-IEpsilon are stable and can find diverse feasible solutions at each run on all the 30 instances. CCMO is workable on the complex and the large systems but failed in finding a feasible solution on the larger system, even if the value of $\mathcal{R}^*$ is small. On the other hand, the CV values of Our Method versus CCMO and Our Method versus MOEA/D-IEpsilon are far greater than those of CCMO versus Our Method and MOEA/D-IEpsilon versus Our Method on each instance. Particularly, Our Method can cover almost all the solutions obtained by CCMO and MOEA/D-IEpsilon on each system, especially under high reliability constraints. The above results imply that Our Method is more effective on harsh constraints in the MOTRAP than CCMO and MOEA/D-IEpsilon. The reason is that both CCMO and MOEA/D-IEpsilon need to consider the possible constraint violations on $\mathcal{T}^*$ and $\mathcal{R}^*$, in which the constraint $\mathcal{T}^*$ is very easily violated when the system size and $\mathcal{R}^*$ is large. On the contrary, Our Method only pays attention to the constraint $\mathcal{R}^* \in [0,1]$ due to the fact that each repaired solution obeys $\mathcal{T}^*$ and at the same time, is very close to the required $\mathcal{R}^*$. Consequently, under the same number of evaluations, Our Method can find more and better feasible solutions. The above results also suggest that leveraging the derived problem's knowledge can speed up the convergence rate.

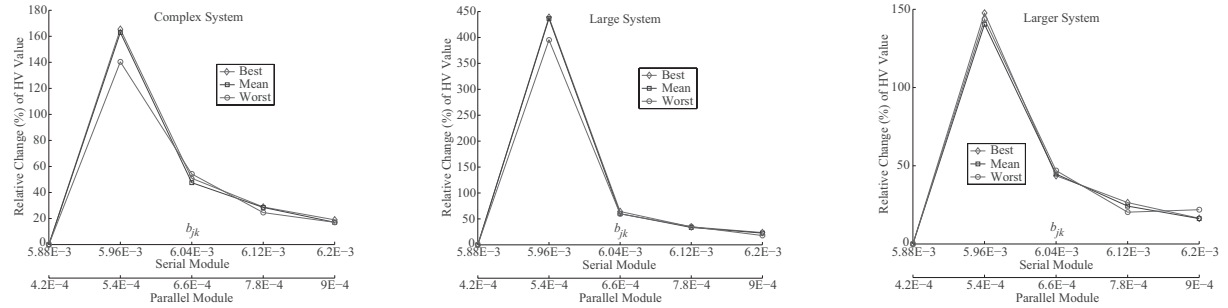
### D. Research Question 3: Sensitivity Analysis

As shown in Table II, there are total five modular parameters in the MOTRAP whose values are estimated by collected software failure times and maximum likelihood estimation. Since it is inevitable that estimation errors may be incurred in practice, it is necessary to conduct sensitivity analysis in terms of the five modular parameters to further evaluate whether Our Method is robust with the modular parameters. As mentioned earlier, the HV is a very good and popular metric to evaluate the overall quality of the obtained solution set. Consequently, we examine how much the HV values fluctuate with the increase of the value of the modular parameters. Fig. 5 illustrates the results of sensitivity analysis for Our Method combined with NSGA-II with respect to the increases of $a_{jk}$, $b_{jk}$, $c_1^{jk}$, $c_2^{jk}$, and $c_3^{jk}$, respectively, in which "Serial Module" means that the associated subsystem has only one module and "Parallel Module" indicates that the connected subsystem has more than one module. Note that the parameter value range is different between a serial module and a parallel module, as shown in Table II. Consequently, two x-axes were used to show the changes of the parameter values of the serial and the parallel module, respectively.

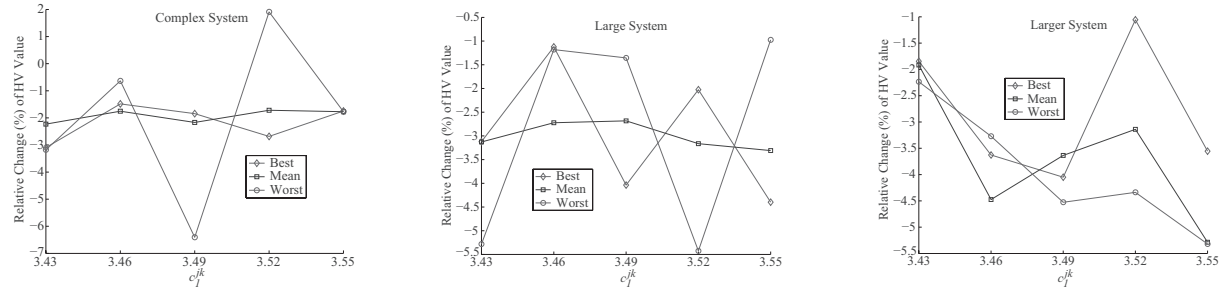
In Fig. 5(a), as can be seen, with the increase of the value of $a_{jk}$, the best and worst value of the HV changes at most by about -6.52%, which occurs on the complex system. Additionally, the average HV value changes little and is in a downward trend. The maximum value of relative change of the average HV value is about -3.52% on the complex



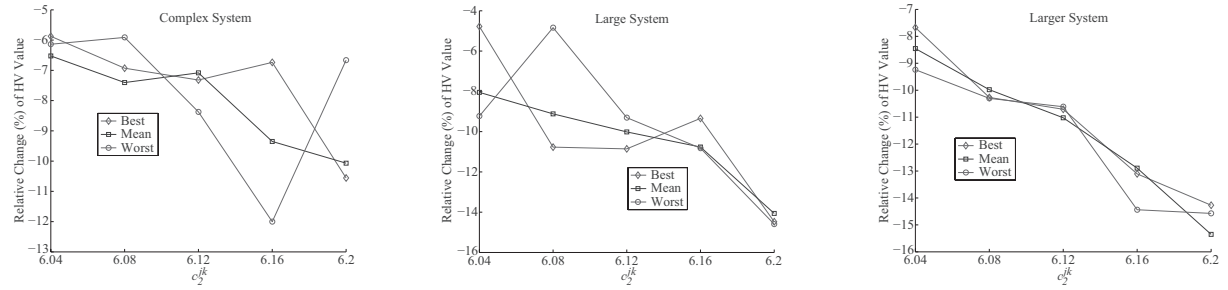

(a) Change of $a_{jk}$



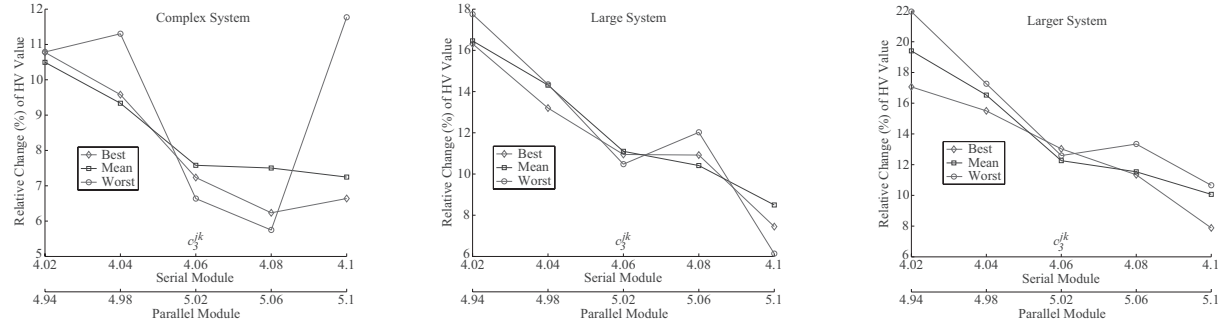

(b) Change of $b_{jk}$




(c) Change of $c_1^{jk}$




(d) Change of $c_2^{jk}$




(e) Change of $c_3^{jk}$

Fig. 5. Sensitivity analysis for Our Method in terms of the changes of the five modular parameters.

system. The reason is that with the increase of the value of $a_{jk}$, reliability becomes increasingly lower. This implies that the obtained solutions under bigger values of $a_{jk}$ can hardly dominate the solutions under smaller values of $a_{jk}$, meaning that the convergence of the solution set decreases gradually.

In Fig. 5(b), with the increase of the value of $b_{jk}$, the best, mean, and worst value of the HV changes at most by about 438.89%, which occurs on the large system. The particular fluctuations of the HV values seem high. However, when $b_{jk}$ increases to a certain value, there is a rapid decline in the range of fluctuations of the HV values. Besides, the average HV value is in an upward trend. This is because with the increase of the value of $b_{jk}$, reliability increases faster than cost. It means that the obtained solutions under bigger values of $b_{jk}$ can easily dominate the solutions under smaller values of $b_{jk}$, which enhances the convergence of the solution set. Hence, the fluctuations in Fig. 5(b) are very natural in terms of the characteristic of parameter $b_{jk}$.

In Fig. 5(c) and Fig. 5(d), with the increase of the value of $c_1^{jk}$ or $c_2^{jk}$, the best, mean, and worst value of the HV changes at most by about -6.4% and -15.35%, respectively. The former occurs on the complex system and the latter occurs on the larger system. In addition, the average HV value is in a downward trend for both $c_1^{jk}$ and $c_2^{jk}$. The reason is that with the increase of the value of $c_1^{jk}$ or $c_2^{jk}$, cost becomes increasingly higher under the same reliability and time. This means that it is difficult for the obtained solutions under bigger values of $c_1^{jk}$ or $c_2^{jk}$ to dominate the solutions under smaller values of $c_1^{jk}$ or $c_2^{jk}$. Hence, the convergence of the solution set becomes increasingly worse.

In Fig. 5(e), with the increase of the value of $c_3^{jk}$, the best, mean, and worst value of the HV changes at most by about 21.96%, which occurs on the larger system. Besides, the average HV value is in an upward trend on the whole. This is because with the increase of the value of $c_3^{jk}$, cost becomes increasingly lower under the same reliability and time, and thus the obtained solutions under bigger values of $c_3^{jk}$ can easily dominate the solutions under smaller values of $c_3^{jk}$. Accordingly, the convergence of the solution set becomes increasingly better. However, the range of the relative change of the average HV value in Fig. 5(e) is much lower than that in Fig. 5(b). This suggests that the effect of $b_{jk}$ on the quality of the solution set is greater than that of $c_3^{jk}$, which is in line with the natural characteristic of these two parameters.

In general, for $a_{jk}$, $c_1^{jk}$, $c_2^{jk}$, and $c_3^{jk}$, the curves for the HV value fluctuate a little between some ranges. For $b_{jk}$, the overall fluctuation of the HV value is not great and the particular fluctuations are attributed to its natural characteristic. The changes are foreseeable and the sensitivities are at acceptable levels. Particularly, Our Method combined with MOEAs can well perceive the influence of the change of the parameter value on the solution quality. Accordingly, Our Method is robust and adaptive on the five modular parameters.

## VII. Conclusion

This paper investigated the problem of multi-objective testing resource allocation, named MOTRAP, to achieve high reliability, low cost and time at the same time. We noted that reliability is important: the software project manager is very interested in allocation schemes that can achieve the desired reliability for user satisfaction. As a result, we first presented an MOTRAP model with the pre-specified reliability. Then, we theoretically deduced the new lower bounds on the time invested in different modules on the basis of the necessary condition for the achievement of the desired reliability. Importantly, these new bounds can tell us which modules are time-efficient or time-consuming to achieve the pre-specified reliability. Moreover, to leverage the derived problem's knowledge, we developed several enhanced constraint handling techniques (ECHTs) for individual initialization, simulated binary crossover, and polynomial mutation, which can be directly combined with multi-objective evolutionary algorithms (MOEAs). Finally, we evaluated the proposed ECHTs through extensive experiments which were carried out on 195 instances that are randomly generated from a normal distribution for three software systems with large sizes and gradual complexities. The experiments show several positive results. 1) The proposed ECHTs outperform the state-of-the-art constraint solving methods designed specifically for the MOTRAP. 2) The proposed ECHTs (combined with NSGA-II) are superior to the general constrained MOEAs especially under different reliability constraints. 3) The proposed ECHTs are robust with the modular parameters from the perspective of sensitivity analysis.

This paper can be seen as the first step toward a reasonable guide to solving the MOTRAP with the reliability constraint, which should be helpful for the software project manager. There are a number of directions that may be further pursued in future work. First, the strengths and weaknesses of the proposed ECHTs will be further studied on the basis of alternative MOEAs. Second, the new lower bounds on time deduced in this paper are soft, and can we obtain the exact lower bounds by mathematical optimisation methods and use these exact bounds as the reference point to guide the search approximating the Pareto front? Third, the testing cost may be concerned by the software project manager, so can we deduce the new upper bounds on time according to the pre-defined cost constraint? Finally, there is value in considering reliability and cost constraints in the MOTRAP with the architecture-based model [3], [12].

## Acknowledgment

The authors would like to thank Prof. K. Deb for the C++ code of the constrained NSGA-II [9], Prof. Z. Fan for the Java code of MOEA/D-IEpsilon [25], and Prof. X. Zhang for the Matlab code of CCMO [27]. The source codes of all the compared methods and all the problem instances can be attained at the link: http://www.cs.bham.ac.uk/~limx/.

## References

[1] 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.

[2] R. Pietrantuono, "On the testing resource allocation problem: Research trends and perspectives," *Journal of Systems and Software*, vol. 161, article no. 110462, 2020.

[3] 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.

[4] 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.

[5] C. A. Coello Coello, G. B. Lamont, D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2nd ed. New York, USA: Springer, 2007.

[6] Y. Guo, J. Cheng, S. Luo, D. Gong, and Y. Xue, "Robust dynamic multi-objective vehicle routing optimization method," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 15, no. 6, pp. 1891–1903, 2018.

[7] J. Ji, Y. Guo, D. Gong, and W. Tang, "MOEA/D-based participant selection method for crowdsensing with social awareness," *Applied Soft Computing*, vol. 87, article no. 105981, 2020.

[8] Y. Guo, X. Zhang, D. Gong, Z. Zhang, and J. Yang, "Novel interactive preference-based multiobjective evolutionary optimization for bolt supporting networks," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 4, pp. 750–764, 2020.

[9] 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.

[10] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2008.

[11] G. Zhang, Z. Su, M. Li, F. Yue, J. Jiang, and X. Yao, "Constraint handling in NSGA-II for solving optimal testing resource allocation problems," *IEEE Transactions on Reliability*, vol. 66, no. 4, pp. 1193–1212, 2017.

[12] 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.

[13] 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.

[14] R. Pietrantuono and S. Russo, "Search-based optimization for the testing resource allocation problem: Research trends and opportunities," in *Proceedings of the IEEE/ACM 11th International Workshop on Search-Based Software Testing*, Gothenburg, Sweden, May 28–29, pp. 6–12, 2018.

[15] J. Petke, S. O. Haraldsson, M. Harman, W. B. Langdon, D. R. White, and J. R. Woodward, "Genetic improvement of software: A comprehensive survey," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 3, pp. 415–432, 2018.

[16] Y. S. Dai, M. Xie, K. L. Poh, and B. Yang, "Optimal testing-resource allocation with genetic algorithm for modular software systems," *Journal of Systems and Software*, vol. 66, no. 1, pp. 47–55, 2003.

[17] P. K. Kapur, A. G. Aggarwal, K. Kapoor, and G. Kaur, "Optimal testing resource allocation for modular software considering cost, testing effort and reliability using genetic algorithm," *International Journal of Reliability, Quality and Safety Engineering*, vol. 16, no. 6, pp. 495–508, 2009.

[18] R. Gao and S. Xiong, "A genetic local search algorithm for optimal testing resource allocation in module software systems," in *Proceedings of the International Conference on Intelligent Computing*, Fuzhou, China, August 20–23, pp. 13–23, 2015.

[19] A. G. Aggarwal, G. Kaur, and P. K. Kapur, "Optimal testing resource allocation for modular software considering imperfect debugging and change point using genetic algorithm," in *Proceedings of the 2nd International Conference on Reliability, Safety and Hazard–Risk-Based Technologies and Physics-of-Failure Methods*, Mumbai, India, December 14–16, pp. 535–541, 2010.

[20] K. Chaudhary, P. Manik, and S. Bali, "Dynamic testing resource allocation of modular software system for SRGM incorporating testing efficiency using differential evolution," in *Proceedings of the 1st International Conference on Soft Computing for Problem Solving*, Roorkee, India, December 20–22, pp. 1011–1023, 2011.

[21] R. Pietrantuono, P. Potena, A. Pecchia, D. Rodriguez, S. Russo, and L. Fernández-Sanz, "Multiobjective testing resource allocation under uncertainty," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 3, pp. 347–362, 2018.

[22] Z. Zhu and L. Jiao, "Improving search-based software testing by constraint-based genetic operators," in *Proceedings of the Genetic and Evolutionary Computation Conference*, Prague, Czech Republic, July 13–17, pp. 1435–1442, 2019.

[23] Z. Liu, Y. Wang, and B. Wang, "Indicator-based constrained multiobjective evolutionary algorithms," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, published online, doi: 10.1109/TSMC.2019.2954491, 2019.

[24] Z. Liu and Y. Wang, "Handling constrained multiobjective optimization problems with constraints in both the decision and objective spaces," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 870–884, 2019.

[25] Z. Fan, W. Li, X. Cai, H. Huang, Y. Fang, Y. You, J. Mo, C. Wei, and E. D. Goodman, "An improved epsilon constraint-handling method in MOEA/D for CMOPs with large infeasible regions," *Soft Computing*, vol. 23, no. 23, pp. 12491–12510, 2019.

[26] Y. Yang, J. Liu, S. Tan, and H. Wang, "A multi-objective differential evolutionary algorithm for constrained multi-objective optimization problems with low feasible ratio," *Applied Soft Computing*, vol. 80, pp. 42–56, 2019.

[27] Y. Tian, T. Zhang, J. Xiao, X. Zhang, and Y. Jin, "A coevolutionary framework for constrained multi-objective optimization problems," *IEEE Transactions on Evolutionary Computation*, published online, doi: 10.1109/TEVC.2020.3004012, 2020.

[28] C. Peng, H. Liu, and E. D. Goodman, "A cooperative evolutionary framework based on an improved version of directed weight vectors for constrained multiobjective optimization with deceptive constraints," *IEEE Transactions on Cybernetics*, published online, doi: 10.1109/TCYB.2020.2998038, 2020.

[29] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms–A comparative case study," in *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, Amsterdam, Netherlands, September 27–30, pp. 292–301, 1998.

[30] M. Li and X. Yao, "Quality evaluation of solution sets in multiobjective optimisation: A survey," *ACM Computing Surveys*, vol. 52, no. 2, article no. 26, 2019.

[31] 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.

[32] M. Li, T. Chen, and X. Yao, "A critical review of "A practical guide to select quality indicators for assessing Pareto-based search algorithms in search-based software engineering": Essay on quality indicator selection for SBSE," in *Proceedings of the 40th International Conference on Software Engineering*, Gothenburg, Sweden, May 27–3 June, pp. 17–20, 2018.

[33] D. A. Van Veldhuizen and G. B. Lamont, "On measuring multiobjective evolutionary algorithm performance," in *Proceedings of the IEEE Congress on Evolutionary Computation*, La Jolla, CA, USA, July 16–19, pp. 204–211, 2000.

[34] E. Zitzler, J. Knowles, and L. Thiele, "Quality assessment of Pareto set approximations," in *Multiobjective Optimization*, J. Branke, K. Deb, K. Miettinen, and R. Slowinski (eds), Springer-Verlag Berlin Heidelberg, 2008.

[35] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in *Proceedings of the 33rd International Conference on Software Engineering*, Honolulu, HI, USA, May 21–28, pp. 1–10, 2011.

[36] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Proceedings of the 5th Conference on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, Athens, Greece, September 19–21, pp. 95–100, 2001.

[37] E. López-Camacho, M. J. G. Godoy, A. J. Nebro, and J. F. Aldana-Montes, "jMetalCpp: optimizing molecular docking problems with a C++ metaheuristic framework," *Bioinformatics*, vol. 30, no. 3, pp. 437-438, 2014.

**Zhaopin Su** received the B.S. and Ph.D. degrees in computer science from Hefei University of Technology, Hefei, China, in 2004 and 2008, respectively.

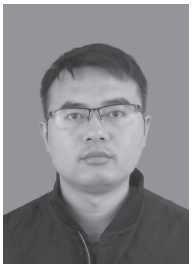
She is currently an Associate Professor of School of Computer Science and Information Engineering at the Hefei University of Technology. Her current research interests covers evolutionary computation, software engineering, and multimedia security.

**Guofu Zhang** 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 a Professor of School of Computer Science and Information Engineering at the Hefei University of Technology. He has published over 50 research papers. He is the Deputy Director of the Anhui Province Key Laboratory of Industry Safety and Emergency Technology. His current research interests include multi-agent systems, evolutionary computation, and search-based software engineering. He was an academic visitor in the School of Computer Science at the University of Birmingham from 2015 to 2016.

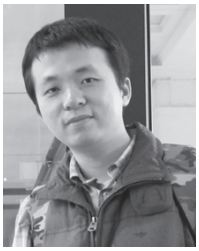
**Feng Yue** received the B.S., M.S., and Ph.D. degrees in computer science from Hefei University of Technology, Hefei, China, in 2004, 2009, and 2015, respectively.

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



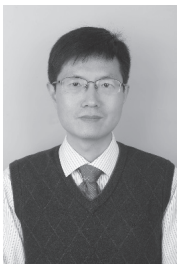
**Dezhi Zhan** received the B.S. degree in electronic and information engineering from the City Institute, Dalian University of Technology, Dalian, China, in 2017.

He is currently pursuing the M.S. degree in information and communication engineering at the School of Computer Science and Information Engineering, Hefei University of Technology. His research interests include multi-objective evolutionary algorithms and search-based software engineering.



**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 Lecturer of School of Computer Science at the University of Birmingham. His research is principally on multi-objective optimisation, where he focuses on developing population-based randomised algorithms (mainly evolutionary algorithms) for both general challenging problems (e.g. many-objective optimisation, constrained optimisation, robust optimisation, expensive optimisation) and specific challenging problems (e.g. those in software engineering, system engineering, product disassembly, post-disaster response, neural architecture search, reinforcement learning for games). He has published over 60 research papers in scientific journals and international conferences. Some of his papers, since published, have been amongst the most cited papers in corresponding journals such as IEEE Transactions on Evolutionary Computation, Artificial Intelligence, ACM Transactions on Software Engineering and Methodology, IEEE Transactions on Parallel and Distribution Systems, and ACM Computing Surveys. His work has received the Best Student Paper Award or Best Paper Award nomination in EC mainstream conferences, CEC, GECCO, and SEAL. He is the Founding Chair of the IEEE CIS' Task Force on Many-Objective Optimisation.



**Bin Li** (M'07) received the B.S. degree from Hefei University of Technology, Hefei, China, in 1992, the M.S. degree from the Institute of Plasma Physics, Chinese Academy of Sciences, Hefei, China, in 1995, and the Ph.D. degree from University of Science and Technology of China (USTC), Hefei, China, in 2001.

He is currently a Professor with the School of Information Science and Technology, USTC. He has authored or co-authored over 60 refereed publications. His current research interests include computation intelligence, artificial intelligence safety, optimal design and human-computer interaction. He is the Founding Chair of the IEEE CIS Hefei Chapter and the Founding Counselor of the IEEE USTC Student Branch.

**Xin Yao** (M'91-SM'96-F'03) received the B.S. degree from University of Science and Technology of China (USTC), Hefei, China, in 1982, the M.S. degree from 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 was 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, the IEEE CIS Evolutionary Computation Pioneer Award in 2013, and the IEEE Frank Rosenblatt Award in 2020. He was the the President (2014-2015) of IEEE CIS, and the Editor-in-Chief (2003-2008) of IEEE Transactions on Evolutionary Computation.